Achieving Rotational Invariance with Bessel-Convolutional Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

For many applications in image analysis, learning models that are invariant to translations and rotations is paramount. This is the case, for example, in medical imaging where the objects of interest can appear at arbitrary positions, with arbitrary orientations. As of today, Convolutional Neural Networks (CNN) are one of the most powerful tools for image analysis. They achieve, thanks to convolutions, an invariance with respect to translations. However, even if several works proposed solutions to bring rotational invariance in CNNs, none of them provide a rigorous invariance to the continuous set of all possible rotation angles. In this work, we present a new type of convolutional layer that takes advantage of Bessel functions, well known in physics, to build Bessel-CNNs that are invariant to all possible rotation angles by design.

1 Introduction

Deep learning models, and more particularly Convolutional Neural Networks (CNNs), are known as being among the most powerful tools for image analysis. For this reason, they are still constantly upgraded in order to achieve better performance [1]. One of the main reasons why CNNs are so much used in computer vision lies in the fact that they achieve translation-invariance thanks to convolutions. Filters sweep the image locally and patterns can be recognized regardless of their absolute position in the image. However, some other important types of invariance are more difficult to obtain. It is for example the case for the rotational invariance, which is relevant for many applications. One could for example consider medical imaging where tissues, cells, tumors or other objects of interest have a local, arbitrary, orientation in the images [2]. Another example is satellite imaging of ships, where both the global orientation of the satellite and the local orientation of a ship are arbitrary [3].

Multiple works proposed solutions in order to bring rotational invariance in CNNs. However, many of them (i) only make the model more robust to rotations without providing guarantees regarding the rotational invariance [4, 5, 6, 7, 8], (ii) only provide guarantees for a finite set of rotation angles [9, 10, 11, 12, 13, 14] or (iii) only handle global rotational invariance, while a local one is sometimes more relevant [4, 5, 6]. In this work, we integrate a new kind of convolutional layer in CNNs to build Bessel-Convolutional Neural Networks (B-CNNs). In B-CNNs, Bessel functions from physics are used to build a representation of the images that is more adapted to deal with rotations. This representation is then used to compute feature maps in a rotational equivariant way, which can lead to rotational invariance. To the best of our knowledge, B-CNN is the first method able to achieve rigorous rotational invariance for any possible angle in the continuous set $[0, 2\pi]$ of angles (see Figure 1 for an example).

Section 2 formally defines the problem of rotational invariance in CNNs and provides the necessary notations. Next, Section 3 presents the related works on bringing rotational invariance into CNNs. Our

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

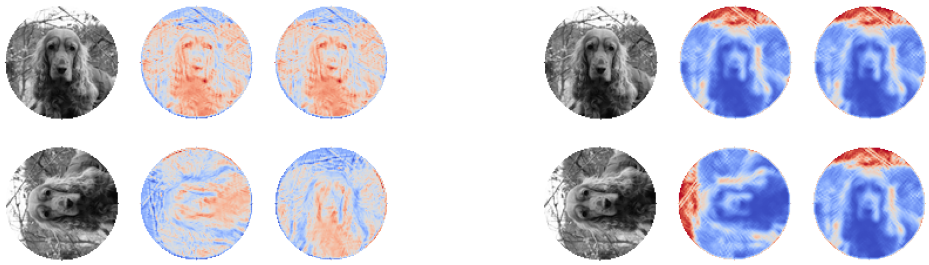


Figure 1: This figure illustrates the lack of rotational equivariance in standard CNNs (left), as opposed to the rotational equivariance of B-CNNs (right). For each triplet of images, the first image is the input image, the second is the feature map obtained, and the last one is the feature map reoriented to make the comparison easier. It can be seen that the last images in the two left triplets for the standard CNN are different. Hence, a different orientation of the input image produces a different feature map. However, feature maps are identical except for a rotation for the B-CNN on the right.

method, B-CNN, is then introduced in Section 4, along with some background on Bessel functions. Experiments are described and discussed in Section 5 before concluding the paper in Section 6.

2 Background and problem definition

For several applications in computer vision, patterns of interest may appear in arbitrary orientations. This typically happens in medical imaging [2], satellite imaging [3], astronomical imaging [15], texture recognition [16], etc. In such cases, the fact that the orientation is meaningless can be used as a prior during learning through a feature extraction that is orientation-agnostic. However, building deep learning models that present guarantees with respect to the rotational invariance is not trivial. Indeed, in the particular case of Convolutional Neural Networks (CNNs), the computation of a feature map is clearly not rotation-invariant since, in general, for a rotation of an angle α, we have

$$\sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} I\left(x-m, y-n\right) K\left(m,n\right) \neq \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} I\left(x'-m, y'-n\right) K\left(m,n\right), \tag{1}$$

where $x' = x\cos\alpha - y\sin\alpha$, $y' = x\sin\alpha + y\cos\alpha$, I is the input of the convolutional layer and K represents a particular filter of size $k_1 \times k_2$. Figure 1 illustrates this lack of invariance.

Two important notions are central in this work: rotational invariance and rotational equivariance. On the one hand, if $f\left(I\right)$ represents the computation of feature maps in a particular layer of a CNN, $f\left(I\right)$ is rotational invariant if $f\left(R\left(\alpha\right)I\right) = f\left(I\right), \forall\alpha \in [0, 2\pi]$, where $R\left(\alpha\right)$ is an operator that rotates I by an angle α. However, if $f\left(R\left(\alpha\right)I\right) = R\left(\alpha\right)f\left(I\right), \forall\alpha \in [0, 2\pi]$, then $f\left(I\right)$ is rotational equivariant since feature maps are identical except for a rotation (see Figure 1). For particular architectures where the feature maps in the final layer have a size of 1×1, this rotational equivariance leads to a rotational invariance. Indeed, if I is one single value, $R\left(\alpha\right)I = I$. Such a strong rotational invariance is something usually not guaranteed, or sometimes only for a finite subset of $[0, 2\pi]$.

3 Related work on rotational invariance for CNNs

Methods for rotational invariance of CNNs can be categorized in two different groups [4]: the ones (i) transforming the input image or the feature maps, or the ones (ii) modifying the way convolutions are performed. Among all these methods, some provide mathematical guarantees with respect to the rotational invariance, while others only make models more robust.

The first category of methods can be seen as a preprocessing step: the input is prepared to ease the extraction of rotational invariant features. In this way, CNNs are able to learn the equivariance to rotation without any major change in the network. Among the best known techniques in this category, one can cite data augmentation [4], transformation-invariant pooling (TI-POOLING) [9] and spatial transformer networks (STN) [5].

The methods in the second category modify layers in the network to make them invariant to rotation. The most well-known method of this second category is Group CNNs (G-CNNs) [10], which define the transformations for which an invariance is desired using the theory of groups and symmetries. Following this idea, it is then possible to achieve an invariance to rotations if these are among the set of transformations defined by the groups of symmetry used. Other techniques of this category include deep symmetry networks (DSN) [11], rotation-invariant and fisher discriminative CNNs (RFID-CNNs) [6], steerable CNNs [12], deformable CNNs [7], SIFT-CNNs [8], steerable filter CNNs (SFCNNs) [13] and spherical CNNs [14].

While all the methods mentioned above tackle the problem of equivariance or invariance to rotation, some of them only make the model more robust to rotations without providing mathematical guarantees regarding the rotational invariance [4, 5, 6, 7, 8], or only provide such guarantees for a finite set of rotation angles [9, 10, 11, 12, 13, 14]. In the next section, we introduce Bessel-CNNs, based on the use of the Bessel functions well known in physics, to make CNNs invariant to all possible rotation angles in the continuous set $[0, 2\pi]$ by design, which is achieved by modifying the convolution between the image (or feature maps) and the filters. Furthermore, other works only handle global rotational invariance, while a local one is sometimes more relevant [4, 5, 6]. In our case, as the rotational invariance in B-CNNs is achieved when the filters sweep over local parts of the images, we can both achieve global and local (at the scale of the filters size) rotational invariance.

4 Exploiting Bessel functions for CNNs

The mathematical background on Bessel functions and the motivation of their use for rotational invariant image recognition tasks are presented in this section. Some of these developments are inspired from the use of Bessel functions in physics [17]. It is shown how these mathematical developments can lead to a new type of Convolutional Neural Networks (CNN) that we call Bessel-Convolutional Neural Network (B-CNN). Mathematical evidence of a rigorous rotational invariance, which few other methods can provide, are pointed out.

4.1 Bessel functions

Bessel functions are particularly well known in physics because they arise when solving some important problems in polar or cylindrical coordinates. For example, they appear when dealing with wave or heat propagation. Bessel functions may be of different order ν and are defined as a particular solution of the Bessel's differential equation

$$x^2 \frac{d^2 y}{dx^2} + x \frac{dy}{dx} + \left(x^2 - \nu^2\right) y = 0. \tag{2}$$

This equation involves two particular kinds of Bessel functions $J_\nu(x)$ and $Y_\nu(x)$, called the Bessel function of the first and of the second kind, respectively. In this work, ν must be an integer and we only consider $J_\nu(x)$ since $Y_\nu(x)$ diverges for $x = 0$. For $\nu \in \mathbb{N}$, $J_\nu(x)$ and $J_{-\nu}(x)$ are not independent and

$$J_{-\nu}(x) = (-1)^\nu J_\nu(x). \tag{3}$$

Bessel functions also satisfy

$$J_\nu(-x) = (-1)^\nu J_\nu(x), \tag{4}$$

which means that J_ν is an even function if ν is even, and an odd function otherwise. Bessel functions of the first kind can be used to build a particular mathematical basis for the representation in polar coordinates of images defined in a circular domain of radius R, as

$$\left\{ \frac{J_\nu(k_{\nu,j}\rho)\, e^{i\nu\theta}}{A_{\nu,j}} \right\}, \forall \nu, j \in \mathbb{N}, \text{ where } A_{\nu,j} = \sqrt{2\pi \int_0^R \rho J_{\nu'}^2(k_{\nu',j'}\rho)\, d\rho}, \tag{5}$$

forms an orthogonal basis, as we can have

$$\int_0^{2\pi} \int_0^R \rho \left[\frac{J_\nu(k_{\nu,j}\rho)\, e^{i\nu\theta}}{A_{\nu,j}} \right]^* \left[\frac{J_{\nu'}(k_{\nu',j'}\rho)\, e^{i\nu'\theta}}{A_{\nu',j'}} \right] d\theta d\rho = \delta_{\nu\nu'} \delta_{jj'}, \tag{6}$$

if the $k_{\nu,j}$ are solutions of $J'_\nu(k_{\nu,j}R) = 0$ (proven by Watson [18]).

This choice for $k_{\nu,j}$ is not the only one possible (another choice to have an orthogonal basis would be to impose $J_\nu\left(k_{\nu,j}R\right)=0$), but it makes it more convenient to represent arbitrary functions (by the fact that for $\nu=0$ there is a $k_{\nu,j}=0$ if $J'_\nu\left(k_{\nu,j}R\right)=0$, we can use $J_0(0.\rho)e^{i.0.\theta}=1$ to describe an arbitrary constant intensity) [17]. The key advantage of using this basis is that it is more convenient to deal with rotations. Later, this basis will be used to build a new representation for images.

4.2 Bessel coefficients

An arbitrary function $\Psi\left(\rho,\theta\right)\in\mathbb{R}$, where ρ and θ are polar coordinates, can be expressed with the basis in Equation (5) as

$$\Psi\left(\rho,\theta\right)=\sum_{\nu=-\infty}^{\infty}\sum_{j=0}^{\infty}\frac{J_\nu\left(k_{\nu,j}\rho\right)e^{i\nu\theta}}{A_{\nu,j}}\Psi_{\nu,j}, \tag{7}$$

where the Bessel coefficients $\Psi_{\nu,j}$ are obtained by projecting $\Psi\left(\rho,\theta\right)$ on this particular basis

$$\Psi_{\nu,j}=\int_0^{2\pi}\int_0^R\rho\left[\frac{J_\nu\left(k_{\nu,j}\rho\right)e^{i\nu\theta}}{A_{\nu,j}}\right]^*\Psi\left(\rho,\theta\right)d\theta d\rho. \tag{8}$$

In order to be mathematically exact, one should consider all $\nu\in\{-\infty,...,\infty\}$ and all $j\in\{0,...,\infty\}$ to represent $\Psi\left(\rho,\theta\right)$ faithfully. Nevertheless, as only a finite set of coefficients can be considered in practice, values of ν_{max} and j_{max} need to be chosen. By taking into account Properties (3) and (4) when looking at Equation (8) and as $\Psi\left(\rho,\theta\right)\in\mathbb{R}$, we end up with the relations

$$\begin{cases}\Re\left(\Psi_{-\nu,j}\right)=\left(-1\right)^\nu\Re\left(\Psi_{\nu,j}\right)\\ \Im\left(\Psi_{-\nu,j}\right)=\left(-1\right)^{\nu+1}\Im\left(\Psi_{\nu,j}\right).\end{cases} \tag{9}$$

One can then only compute the Bessel coefficients for ν (resp j) in $\{0,...,\nu_{max}$ (resp. $j_{max})\}$ since $\Psi_{-\nu,j}$ and $\Psi_{\nu,j}$ are not independent. Also, note that, when ν or j increases, $k_{\nu,j}$ also increases. As $J_\nu\left(x\right)$ tends to zero when x increases, a large value for ν and j means that $\Psi_{\nu,j}$ will only slightly influence the reconstruction of $\Psi\left(\rho,\theta\right)$ in Equation (7). The values of ν_{max} and j_{max} can be determined through a mathematical insight about the maximal value of $k_{\nu,j}$ to use in a given problem (for example, we could restrict $k_{\nu,j}$ to values s.t. $k_{\nu,j}\leq\frac{2\pi}{\lambda}$, where λ refers to the size of one pixel).

To understand now why working with Bessel coefficients is particularly useful, one can see how an arbitrary rotation of $\Psi\left(\rho,\theta\right)$ by an angle α modifies its Bessel coefficients $\Psi_{\nu,j}$. To do so, let us consider $\Psi^{rot}\left(\rho,\theta\right)=\Psi\left(\rho,\theta-\alpha\right)$ for an angle $\alpha\in[0,2\pi]$. Its Bessel coefficients are given by

$$\Psi^{rot}_{\nu,j}=\int_0^{2\pi}d\theta\int_0^R d\rho\rho\left[\frac{J_\nu\left(k_{\nu,j}\rho\right)e^{i\nu\theta}}{A_{\nu,j}}\right]^*\Psi^{rot}\left(\rho,\theta\right). \tag{10}$$

By defining $\theta'=\theta-\alpha$, this leads to

$$\Psi^{rot}_{\nu,j}=\int_0^{2\pi}d\theta'\int_0^R d\rho\rho\left[\frac{J_\nu\left(k_{\nu,j}\rho\right)e^{i\nu\theta'}}{A_{\nu,j}}\right]^*\Psi\left(\rho,\theta'\right)e^{-i\nu\alpha}=\Psi_{\nu,j}e^{-i\nu\alpha}. \tag{11}$$

This motivates our work, as a rotation of an arbitrary function by an angle α only modifies its Bessel coefficients by a multiplying factor $e^{-i\nu\alpha}$, which really conveniently express rotations. Therefore, if $\Psi\left(\rho,\theta\right)$ represents an image, its Bessel coefficients constitute a more adapted representation to build rotational invariant operations since it is easier to discard the influence of rotations.

4.3 Defining a rotational invariant operation

In order to build a rotational invariant operation $\bullet$ between the Bessel coefficients $\Psi_{\nu,j}$ and a filter made of the same number of complex numbers $K_{\nu,j}$, one can consider

$$\mathbf{K}\bullet\mathbf{\Psi}=\frac{1}{2\pi}\int_0^{2\pi}|\sum_{\nu,j}K^*_{\nu,j}\Psi_{\nu,j}e^{-i\nu\alpha}|^2d\alpha. \tag{12}$$

This operation is necessarily invariant to rotations as Equation (11) shows that multiplying each $\Psi_{\nu,j}$ by $e^{-i\nu\alpha}$ is equivalent to applying a rotation to the image. By performing the integration for α going

from 0 to 2π, the image achieves a complete rotation around itself. Therefore, $\mathbf{K} \bullet \mathbf{\Psi}$ does not depend anymore on the particular initial orientation of the image, by construction. Now, given that

$$|\sum_{i=1}^{k} \alpha_i z_i|^2 = \sum_{m,j} \Re(\alpha_m) \Re(\alpha_j) |z_m z_j| \cos(\theta_m - \theta_j) + \sum_{m,j} \Im(\alpha_m) \Im(\alpha_j) |z_m z_j| \cos(\theta_m - \theta_j)$$
$$-2 \sum_{m,j} \Re(\alpha_j) \Im(\alpha_m) |z_m z_j| \sin(\theta_m - \theta_j), \tag{13}$$

where α_i and $z_i = |z_i| e^{i\theta_i}$ are complex numbers, the squared modulus in Equation (12) becomes

$$\sum_{\substack{\nu,j \\ \nu',j'}} \Re\left(K^*_{\nu,j}\right) \Re\left(K^*_{\nu',j'}\right) |\Psi_{\nu,j} \Psi_{\nu',j'}| \cos\left(\theta_{\nu,j} - \theta_{\nu',j'} - \alpha(\nu - \nu')\right)$$
$$+ \sum_{\substack{\nu,j \\ \nu',j'}} \Im\left(K^*_{\nu,j}\right) \Im\left(K^*_{\nu',j'}\right) |\Psi_{\nu,j} \Psi_{\nu',j'}| \cos\left(\theta_{\nu,j} - \theta_{\nu',j'} - \alpha(\nu - \nu')\right)$$
$$-2 \sum_{\substack{\nu,j \\ \nu',j'}} \Im\left(K^*_{\nu,j}\right) \Re\left(K^*_{\nu',j'}\right) |\Psi_{\nu,j} \Psi_{\nu',j'}| \sin\left(\theta_{\nu,j} - \theta_{\nu',j'} - \alpha(\nu - \nu')\right), \tag{14}$$

if $\Psi_{\nu,j} = |\Psi_{\nu,j}| e^{i\theta_{\nu,j}}$. The three terms should be integrated over α thanks to Equation (12). However, only the trigonometric functions are α-dependent. A simple integration leads to

$$\int_0^{2\pi} \text{sc}\left(\theta_{\nu,j} - \theta_{\nu',j'} - \alpha(\nu - \nu')\right) d\alpha = \begin{cases} 2\pi \text{ sc}\left(\theta_{\nu,j} - \theta_{\nu',j'}\right) & \textbf{if } \nu = \nu' \\ 0 \text{ otherwise,} \end{cases} \tag{15}$$

where sc can represent the cosine or the sine function. Therefore,

$$\mathbf{K} \bullet \mathbf{\Psi} = \sum_{\nu,j,j'} \Re\left(K^*_{\nu,j}\right) \Re\left(K^*_{\nu,j'}\right) |\Psi_{\nu,j} \Psi_{\nu,j'}| \cos\left(\theta_{\nu,j} - \theta_{\nu,j'}\right)$$
$$+ \sum_{\nu,j,j'} \Im\left(K^*_{\nu,j}\right) \Im\left(K^*_{\nu,j'}\right) |\Psi_{\nu,j} \Psi_{\nu,j'}| \cos\left(\theta_{\nu,j} - \theta_{\nu',j'}\right)$$
$$-2 \sum_{\nu,j,j'} \Im\left(K^*_{\nu,j}\right) \Re\left(K^*_{\nu,j'}\right) |\Psi_{\nu,j} \Psi_{\nu,j'}| \sin\left(\theta_{\nu,j} - \theta_{\nu',j'}\right), \tag{16}$$

and by using once again the result presented in Equation (13), we have

$$\mathbf{K} \bullet \mathbf{\Psi} = \sum_{\nu} |\sum_{j} K^*_{\nu,j} \Psi_{\nu,j}|^2, \tag{17}$$

which is simpler than Equation (12). By construction, the results obtained by Equation (17) do not depend on the particular orientation of the image. Furthermore, $\mathbf{K} \bullet \mathbf{\Psi} \in \mathbb{R}$ (with $K_{\nu,j} \in \mathbb{C}$ and $\Psi_{\nu,j} \in \mathbb{C}$), which is something convenient in order to use classic activation functions.

4.4 Bessel-Convolutional Neural Networks (B-CNNs)

In standard CNNs, a feature map is the result of the convolution between the image and a filter $\mathbf{K}$ of $k_1 \times k_2$ weights, as described in Equation (1). The weights of this filter are real numbers that are tuned during the learning process of the network. In Bessel-Convolutional Neural Networks (B-CNNs), the process to build feature maps is slightly different as the representation of the image based on Bessel coefficients are used. A filter $\mathbf{K}$ made of $(\nu_{max} + 1) \times (j_{max} + 1)$ complex weights $K_{\nu,j}$ is used to sweep the image over sub-regions of size $k \times k$. Note that ν_{max} and j_{max} are generally larger than k_1 and k_2, which results in an increase of the number of trainable parameters in each layer. However, as it is easier for B-CNNs to deal with rotational invariance, we show in Section 5 that architectures with less layers than for other methods still achieve similar or even better performances. For each sub-region of the image centered at position (m, n), its Bessel coefficients $\Psi^{(m,n)}_{\nu,j}$ are computed, and

$$a^{(m,n)} = \sum_{\nu} |\sum_{j} K^*_{\nu,j} \Psi^{(m,n)}_{\nu,j}|^2 \tag{18}$$

is computed, where $a^{(m,n)}$ is a real value that is invariant to rotation, i.e., it does not depend on the particular orientation of the sub-region. All the values obtained for the different sub-regions constitute the feature map, which is then rotation equivariant. This is schematized in Figure 2.

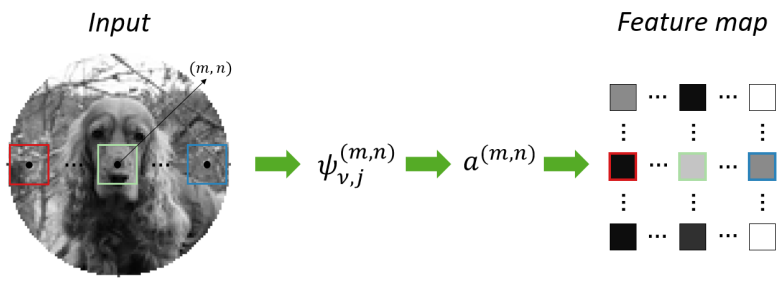


Figure 2: Illustration of how the feature maps are computed in B-CNNs.

4.5 An efficient implementation of B-CNNs

Implementing B-CNNs as described in Section 4.4 can be slow, as it requires to solve Equation (8) for each sub-region of the image. However, developing Equation (18) with Equation (8) gives

$$\begin{aligned} a^{(m,n)} &= \sum_{\nu} |\sum_{j} K_{\nu,j}^{*} \int_{0}^{2\pi} \int_{0}^{R} \frac{J_{\nu}\left(k_{\nu,j}\rho\right) e^{-i\nu\theta}}{A_{\nu,j}} \Psi^{(m,n)}\left(\rho,\theta\right) \rho d\rho d\theta|^{2} \\ &= \sum_{\nu} |\int_{0}^{2\pi} \int_{0}^{R} \Psi^{(m,n)}\left(\rho,\theta\right) \sum_{j} K_{\nu,j}^{*} \frac{J_{\nu}\left(k_{\nu,j}\rho\right) e^{-i\nu\theta}}{A_{\nu,j}} \rho d\rho d\theta|^{2}, \end{aligned} \tag{19}$$

where $\Psi^{(m,n)}\left(\rho,\theta\right)$ represents a particular sub-region of the image centered at (m,n) in polar coordinates. When going back to a Cartesian coordinates system, one can get

$$a^{(m,n)} = \sum_{\nu} |\int_{-\frac{R}{2}}^{\frac{R}{2}} \int_{-\frac{R}{2}}^{\frac{R}{2}} I\left(m-x,n-y\right) \sum_{j} K_{\nu,j}^{*} \frac{\widetilde{J}_{\nu}\left(k_{\nu,j}\sqrt{x^{2}+y^{2}}\right) e^{-i\nu\widetilde{\theta}(x,y)}}{A_{\nu,j}} dxdy|^{2}, \tag{20}$$

where $I\left(x,y\right)$ is the input of the layer, $\widetilde{\theta}\left(x,y\right) = \pi + \arctan\frac{y}{x}$ and $\widetilde{J}_{\nu}\left(k_{\nu,j}\rho\right)$ is defined as

$$\widetilde{J}_{\nu}\left(k_{\nu,j}\rho\right) = \begin{cases} J_{\nu}\left(k_{\nu,j}\rho\right) \text{ if } \rho \leq R \\ 0 \text{ otherwise.} \end{cases} \tag{21}$$

Finally, by defining $T_{\nu,j}\left(x,y\right) = \widetilde{J}_{\nu}\left(k_{\nu,j}\sqrt{x^{2}+y^{2}}\right) e^{-i\nu\widetilde{\theta}(x,y)}/A_{\nu,j}$, Equation (20) leads to

$$\mathbf{a} = \sum_{\nu} |I\left(x,y\right) * \sum_{j} K_{\nu,j}^{*} T_{\nu,j}\left(x,y\right)|^{2} = \sum_{\nu} |I\left(x,y\right) * F_{\nu}\left(x,y\right)|^{2}, \tag{22}$$

where $*$ denotes a convolutional product and $\mathbf{F}_{\nu}\left(x,y\right)$ corresponds to a filter modified by the Bessel functions. Compared to a standard CNN, the only difference is that a layer in B-CNN performs $\nu_{max}+1$ convolutions instead of one only, and it needs to update the filters with the Bessel functions. Nevertheless, $T_{\nu,j}\left(x,y\right)$ is part of the model and only needs to be computed once, as it does not depend on the input of the layer, but only on the choice of ν_{max} and j_{max}.

5 Experiments

This section presents the details of the experiments used to test the rotational invariance of B-CNNs, including the datasets (Section 5.1), the baseline architectures (Section 5.2) and the experimental setup (Section 5.3). Experimental results are presented in Section 5.4 and a discussion on the rotational equivariance of feature maps is proposed in Section 5.5.

5.1 Datasets

In this comparative study, three datasets are used. In all of them, orientations present in the test set are not present in the training set. The first dataset is MNIST [19], a classic baseline of 28×28 images of

handwritten digits, where images in the testing set are randomly rotated by an angle in $[0, 2\pi]$. For all runs, the training set contains $60,000$ images and the test set $10,000$ images. Like Quiroga et al. [4], we chose to not use MNIST-rot [20], since it contains rotations of only 8 different angles, while our work addresses all possible rotations. The second dataset is Outex-TC-00010-r dataset [21], a classic dataset for a more complex classification involving rotations. It contains 128×128 grayscale images of 24 particular textures. The training set contains 480 images with 20 orientations, and the test set is composed of 3840 images with 160 orientations. The third dataset is made of $128 \times 128 \times 3$ brain MRI images [22] that are either cancerous or non-cancerous. Each image is cropped to make the brain fill the entire image. The role of the dataset is to test the local rotational invariance, since tumors are local objects in the image. The training and test sets contain 190 and 63 images, respectively.

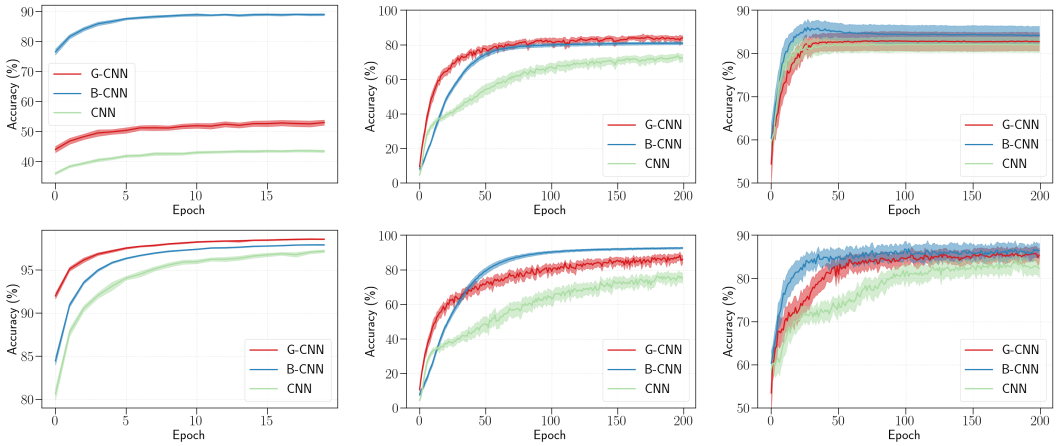
5.2 Baseline architectures

Experiments aim to evaluate how good B-CNNs are for problems where rotational invariance is desired. G-CNNs [10] are among the best performing techniques in the literature to deal with rotational invariance. Therefore, G-CNNs are used, as well as standard CNNs, in our comparative study. For G-CNNs, the group of symmetry C4 is used (rotations of angle $\frac{\pi}{2}$, π, $\frac{3\pi}{2}$ and 2π). For each dataset, one architecture based on CNNs, G-CNNs and B-CNNs has been chosen by picking the one that performs the best on a validation set. Importance is given to the fact that the three models contain roughly the same number of trainable parameters, in order to perform fair comparisons. Attention is also paid to the fact that the feature map in the last convolution layer should be of size 1×1, in order to achieve rotational invariance for G-CNNs and B-CNNs.

The standard CNN architecture for MNIST is made of 7 conv. layers with 32 filters for the first two layers, and 64 filters of size 3×3 for the other layers; one max-pooling layer that operates over 2×2 regions after the second and last layers; and one dense layer with 10 units. The G-CNN architecture is similar, except that the number of filters is divided by 2 in order to keep the same number of parameters. The B-CNN for MNIST is made of 5 Bessel-convolutional layers (one of 16 filters with $\nu_{max} = 11$, $j_{max} = 9$ and $k = 3$; three of 16 filters with $\nu_{max} = 9$, $j_{max} = 7$ and $k = 5$; and one of 32 filters with $\nu_{max} = 7$, $j_{max} = 5$ and $k = 7$), a max-pooling layer over 2×2 regions after the fourth layer and a dense layer with 10 units. Activation functions are relu, except for the B-CNN where tanh leads to better performances on the validation set. Adam optimization with a learning rate of 0.0001 and an exponential decay rate of 0.9 is used for the three models.

The standard CNN for Outex is made of 5 conv. layers (one with 48 filters of size 5×5, one with 64 filters of size 5×5, two with 64 filters of size 3×3 and one with 128 filters of size 3×3), one max-pooling layer over 3×3 regions after the first layer, one over 2×2 regions after the second and third layers, and one dense layer with 24 units. The G-CNN architecture for Outex is similar, but the number of filters is divided by 2 again to have the same number of parameters. The model based on B-CNN is made of 3 Bessel-convolutional layers (one with 16 filters with $\nu_{max} = 15$, $j_{max} = 11$ and $k = 15$; one with 32 filters with $\nu_{max} = 10$, $j_{max} = 9$ and $k = 9$; and one with 32 filters with $\nu_{max} = 7$, $j_{max} = 5$ and $k = 7$), a max-pooling layer over 3×3 regions after the first layer, one over 2×2 regions after the two others, and a dense layer with 24 units. Again, activation functions are relu, except for B-CNN where tanh are used. Adam optimization with a learning rate of 0.0005 and an exponential decay rate of 0.9 is used for the three models.

The standard CNN for brain MRI is made of 5 conv. layers (one with 32 7×7 filters of size 7×7, one with 32 filters of size 5×5, two with 64 filters of size 5×5 and one with 64 filters of size 3×3), one max-pooling layer over 3×3 regions after the first layer, one over 2×2 regions after the second and third layers, and one dense layer with 2 units. The G-CNN architecture is similar but the number of filters is once again divided by 2. The model based on B-CNN is made of 4 Bessel-convolutional layers (one with 32 filters with $\nu_{max} = 11$, $j_{max} = 11$ and $k = 9$; one with 16 filters with $\nu_{max} = 9$, $j_{max} = 9$ and $k = 7$; one with 16 filters with $\nu_{max} = 7$, $j_{max} = 7$ and $k = 5$; and one with 32 filters with $\nu_{max} = 7$, $j_{max} = 7$ and $k = 5$), a max-pooling layer over 3×3 regions after the first layer, one over 2×2 regions after the three other layers, and a dense layer with 2 units. Again, activation functions are relu, except for B-CNN where tanh are used. Adam optimization with a learning rate of 0.001 and a exponential decay rate of 0.9 is used for the three models.



(a) Performance on MNIST (b) Performance on Outex (c) Performance on brain MRI

Figure 3: Mean test accuracy over 40 runs with 99% confidence intervals for B-CNN, G-CNN and standard CNN on MNIST, Outex and brain MRI. In the first row, the training sets contain input images as they are originally in the dataset, whereas data augmentation is used in the second row.

5.3 Experimental setup

Each experiment consists of 40 independent runs, and mean accuracy scores with 99% confidence intervals are reported. In order to isolate intrinsic rotational invariance, each model is trained for each dataset in two different settings. In the first setting, images are not rotated at all during training. In the second setting, input images are rotated by a random angle during training (this can be seen as a form of data augmentation). In order to avoid border effects due to the appearance of black corners when images are rotated for the Outex dataset, images are cropped from an initial size of 128×128 to 88×88. This is not performed for MNIST and brain MRI, since image corners are already black.

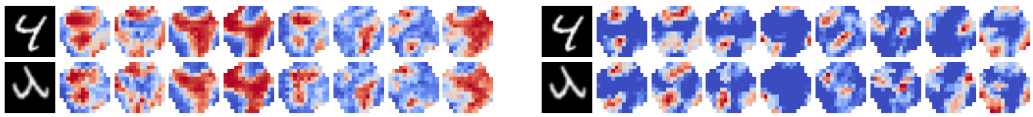
5.4 Experimental results

Figure 3 shows the mean test accuracy of B-CNNs, G-CNNs and standard CNNs on MNIST, Outex and brain MRI. The first observation is that standard CNNs have poor performance in almost all cases, which justifies the need for better methods. A second observation is that using data augmentation (second row), as opposed to using the input images as they originally are in the dataset (first row), increases the performance of all methods in all cases. Furthermore, in the setup without data augmentation, it can be observed that B-CNNs largely outperform G-CNNs and standard CNNs on MNIST. This indicates that G-CNNs need data augmentation in addition to the groups of symmetry to learn rotational invariance, while Bessel convolutions intrinsically impose the invariance.

The results are interestingly reversed for Outex, where B-CNNs perform better than G-CNNs with data augmentation, while being roughly equivalent without data augmentation. The good performance of G-CNNs without data augmentation on Outex may be explained by the fact that the texture images contain an intrinsic data augmentation (a rotation of an image can be present in the original dataset). This is not the case of MNIST, where all images in the original dataset have the same orientation.

It can also be observed that the variance of G-CNNs is generally higher than B-CNNs, due to the fact that a discrete set of rotations is encoded into the groups of the G-CNNs. Because of that, a mismatch can happen between the rotations learned with the randomly rotated input images and the groups on the one hand, and the rotation of the images in the test set on the other hand. This mismatch is minimized for B-CNNs, as the continuous set of rotation angles is imposed by design. Thanks to that, the predictions of B-CNNs are more stable than the one of state-of-the-art G-CNNs.

The generalized instability in performance for all methods on brain MRI (see Figure 3c) can be explained by the small number of instances used at each epoch (190 for training and 63 for testing). This characteristic of the dataset may also explain that B-CNN achieves better performance more



(a) Some feature maps of a B-CNN (b) Some feature maps of a G-CNN

Figure 4: The first row corresponds to random feature maps for a non-rotated 4 from MNIST. The second row corresponds to the same feature maps, but for a 4 rotated by 60°. To ease the comparison with the first row, these feature maps have been reoriented like in Figure 1. The corner pixels of the images of the second row are hidden as they correspond to pixels that are out of the non-rotated image. For the sake of comparison, the same pixels have been hidden in the images of the first row.

quickly, as the rotational invariance is present by design (i.e., invariance is set as prior knowledge), by opposition to G-CNN that more heavily relies on the training instances to learn this invariance.

5.5 On the equivariance of feature maps

In addition to their performance stability, B-CNNs also offer a certain stability from the equivariance of their feature maps. Unlike standard CNNs in Figure 1, the feature maps of B-CNNs extract the exact same latent features, no matter the orientation of the input image. This characteristic of B-CNNs can help in providing trust to its users. For instance, medical experts would see that the model extract the same latent features for tumorous brain images, and this no matter the orientation of the tumor.

Figure 4 shows the equivariance in B-CNNs and G-CNNs. The feature maps in the figure are randomly taken from the last convolution layer before the one that leads to 1×1 feature maps. It can be observed that the feature maps in the first row (non-rotated 4 from MNIST) for B-CNNs are roughly the same as those of the second row (corresponding to the same 4, but rotated of 60°, see Figure 4a). However, when training a G-CNN with the same number of layers and the same filter sizes, its feature maps change more significantly after rotating the input image (see Figure 4b).

In addition to Figure 4, we also performed a quantitative study in order to assess the rotational equivariance of B-CNNs, G-CNNs and CNNs. To do so, we performed the same experiment as the previous one, but for the whole testing set and for random rotational angles. For each sample, the 8 feature maps that are the most active are min-max scaled between 0 and 1 (this step is needed since feature maps of different architectures may have absolute values at a different scale). Then, the difference between the non-rotated and the rotated feature maps is computed (the difference between the first and second row in Figure 4). The bigger this value is, the less equivariant the method is. We obtained a mean value of $7.58\% \pm 2.65\%$ for the B-CNN, $11.36\% \pm 5.18\%$ for the G-CNN and $13.32\% \pm 7.29\%$ for the CNN. B-CNNs therefore achieve a better rotational equivariance. The still relative high value for B-CNNs may be explained by the low resolution of the images. When rotating the input images, pixels may be off-grid and, therefore, the images are altered.

6 Conclusion

This paper proposes a new kind of CNNs, called Bessel CNNs (or B-CNNs), that are intrinsically invariant to the rotation of input images. In order to achieve this, a new convolutional layer has been designed, based on Bessel functions from physics. To the best of our knowledge, this work is the first to introduce Bessel functions in machine learning, as well as to propose a rigorous invariance to the continuous set of rotation angles in CNNs. Experiments show that B-CNNs do not need any preprocessing of the input images to be rotational invariant. Furthermore, the full rotational invariance brought by the Bessel convolutions, by opposition to the classic discrete set of orientations, lead to more stable results. We conclude that Bessel convolutions should be used in any application involving global and/or local rotations of images. Among future works, a deeper analysis will be performed to automatically infer ν_{max} and j_{max} from Section 4.2. Furthermore, the potential benefit of combining the Bessel convolutions with group symmetries will be investigated.

References

[1] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29:1–98, 2017.

[2] Benjamin Chidester, Tianming Zhou, Minh N Do, and Jian Ma. Rotation equivariant and invariant neural networks for microscopy image analysis. *Bioinformatics*, 35(14):i530–i537, 2019.

[3] Linhao Li, Zhiqiang Zhou, Bo Wang, Lingjuan Miao, and Hua Zong. A novel CNN-based method for accurate ship detection in HR optical remote sensing images via rotated bounding box. *IEEE Transactions on Geoscience and Remote Sensing*, 59(1):686–699, 2020.

[4] Facundo Quiroga, Franco Ronchetti, Laura Lanzarini, and Aurelio F Bariviera. Revisiting data augmentation for rotational invariance in convolutional neural networks. In *International Conference on Modelling and Simulation in Management Sciences*, pages 127–141, 2018.

[5] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2017–2025, 2015.

[6] Gong Cheng, Peicheng Zhou, and Junwei Han. RIFD-CNN: Rotation-invariant and fisher discriminative convolutional neural networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2884–2893, 2016.

[7] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 764–773, 2017.

[8] Abhay Kumar, Nishant Jain, Chirag Singh, and Suraj Tripathi. Exploiting sift descriptor for rotation invariant convolutional neural network. In *IEEE India Council International Conference (INDICON)*, pages 1–5, 2018.

[9] Dmitry Laptev, Nikolay Savinov, Joachim M Buhmann, and Marc Pollefeys. TI-POOLING: transformation-invariant pooling for feature learning in convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 289–297, 2016.

[10] Taco S Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*, pages 2990–2999, 2016.

[11] Robert Gens and Pedro M Domingos. Deep symmetry networks. *Advances in Neural Information Processing Systems (NIPS)*, 27:2537–2545, 2014.

[12] Taco S Cohen and Max Welling. Steerable CNNs. In *International Conference on Learning Representations (ICLR)*, 2017.

[13] Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant CNNs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 849–858, 2018.

[14] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. In *International Conference on Learning Representations (ICLR)*, 2018.

[15] Sander Dieleman, Kyle W. Willett, and Joni Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 450(2):1441–1459, 2015.

[16] Diego Marcos, Michele Volpi, and Devis Tuia. Learning rotation invariant convolutional filters for texture classification. In *International Conference on Pattern Recognition (ICPR)*, pages 2012–2017, 2016.

[17] Alexandre Mayer and Jean-Paul Vigneron. Transfer matrices combined with Green's functions for the multiple-scattering simulation of electronic projection imaging. *Physical Review B*, 60(4):2875–2882, 1999.

[18] G. N. Watson. *A Treatise on the Theory of Bessel Functions*. Cambridge University Press, 2nd edition edition, 1995.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[20] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 473–480, 2007.

[21] Timo Ojala, Topi Maenpaa, Matti Pietikainen, Jaakko Viertola, Juha Kyllonen, and Sami Huovinen. Outex-new framework for empirical evaluation of texture analysis algorithms. In *Object Recognition Supported by User Interaction for Service Robots*, volume 1, pages 701–706, 2002.

[22] Navoneel Chakrabarty. Brain MRI images for brain tumor detection, version 1. Retrieved May 9, 2021 from `https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection`.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The claim is that we achieve rigorous rotational invariance for $\alpha \in [0, 2\pi]$, which is theoretically proved in Section 4 and tested in Section 5.
   (b) Did you describe the limitations of your work? [Yes] In Section 4.4, we discuss the fact that using B-CNNs generally leads to an increase in the number of parameters in each layer. In Section 6, we point out that a deeper analysis needs to be performed to more easily chose the meta-parameters of B-CNNs.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Except for Equation (6) because it would be too long, but a reference to the proof is provided [18].
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Our implementation of B-CNNs is provided in the supplemental material. All the instructions and codes to reproduce the results are provided, and all the data used are available following the references.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] In our experiments, 99% confidence intervals are provided (obtained over 40 independent runs). Standard deviations are also given for experiments in Section 5.5
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] The training for all the architectures is performed using GPUs. However, the total amount of compute and the type of GPUs is not provided as our implementation of B-CNNs is not, for the moment, as well optimized as the two other methods we tested (G-CNNs and CNNs). The comparison would therefore not make much sense.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] For data, references are given, and for the use of Group-CNNs, authors are cited.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our B-CNNs implementation is provided in the supplemental material
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]