RESEARCH OUTPUTS / RÉSULTATS DE RECHERCHE

SO(2) and O(2) Equivariance in Image Recognition with Bessel-Convolutional Neural Networks

Delchevalerie, Valentin; Mayer, Alexandre; Bibal, Adrien; Frénay, Benoît

Link to publication

Citation for pulished version (HARVARD):
Delchevalerie, V, Mayer, A, Bibal, A & Frénay, B 2023 'SO(2) and O(2) Equivariance in Image Recognition with Bessel-Convolutional Neural Networks'.

SO(2) and O(2) Equivariance in Image Recognition with Bessel-Convolutional Neural Networks

Valentin Delchevalerie   VALENTIN.DELCHEVALERIE@UNAMUR.BE
Faculty of Computer Science
NaDI & naXys institutes
University of Namur
Rue Grandgagnage 21, 5000 Namur, Belgium

Alexandre Mayer   ALEXANDRE.MAYER@UNAMUR.BE
Department of Physics
naXys institute
University of Namur
Rue de Bruxelles 61, 5000 Namur, Belgium

Adrien Bibal   ADRIEN.BIBAL@UCLOUVAIN.BE
CENTAL, UCLouvain   @CUANSCHUTZ.EDU
Place Montesquieu 3, 1348 Louvain-la-Neuve, Belgium
School of Medicine, Univerity of Colorado Anschutz Medical Campus
1890 N Revere Ct, Aurora, CO 80045, USA

Benoît Frénay   BENOIT.FRENAY@UNAMUR.BE
Faculty of Computer Science
NaDI institute
University of Namur
Rue Grandgagnage 21, 5000 Namur, Belgium

Abstract

For many years, it has been shown how much exploiting equivariances can be beneficial when solving image analysis tasks. For example, the superiority of convolutional neural networks (CNNs) compared to dense networks mainly comes from an elegant exploitation of the translation equivariance. Patterns can appear at arbitrary positions and convolutions take this into account to achieve translation invariant operations through weight sharing. Nevertheless, images often involve other symmetries that can also be exploited. It is the case of rotations and reflections that have drawn particular attention and led to the development of multiple equivariant CNN architectures. Among all these methods, Bessel-convolutional neural networks (B-CNNs) exploit a particular decomposition based on Bessel functions to modify the key operation between images and filters and make it by design equivariant to all the continuous set of planar rotations. In this work, the mathematical developments of B-CNNs are presented along with several improvements, including the incorporation of reflection and multi-scale equivariances. Extensive study is carried out to assess the performances of B-CNNs compared to other methods. Finally, we emphasize the theoretical advantages of B-CNNs by giving more insights and in-depth mathematical details.

Keywords: Convolutional neural networks; steerable filters; Bessel functions; SO(2) invariance; O(2) invariance

1 Introduction

For years now, convolutional neural networks (CNNs) are known to be the most powerful tool that we have for image analysis. Their efficiency compared to classic multi-layers perceptrons (MLPs) mainly comes from an elegant exploitation of the translation equivariance involved in image analysis tasks. Indeed, CNNs exploit the fact that patterns can arise at different positions in images by sharing the weights over translations thanks to convolutions. The translation equivariance can be seen as a particular form of prior knowledge and weights can be saved compared to an MLP architecture with similar performances.

By building on the success of exploiting translation equivariance in image analysis, we advocate here that generalizing this to other types of appropriate symmetries can also be useful. For example, in biomedical or satellite imaging, objects of interest can appear at arbitrary positions with arbitrary orientations. To illustrate this, Figure 1 shows four versions of the exact same galaxy that are equally plausible images that could occur in the data set. If the task is to determine the morphology of the galaxy, it is relevant to want these images to be processed in the exact same way. Therefore, introducing rotation equivariance will lead to a more optimal use of the weights and to a better overall efficiency of the models. Being able to guarantee rotation equivariance is also useful to put more trust into models. For instance, experts would be more confident in models that extract the exact same latent features for an object, no matter its particular orientation (of course, depending on the application).
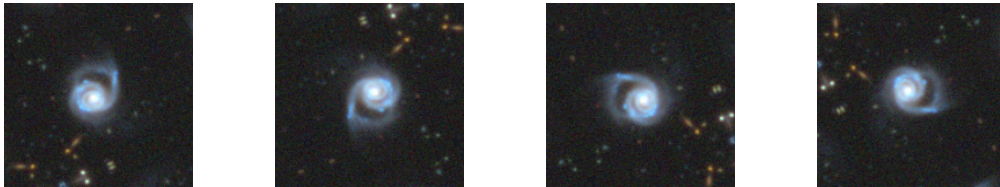

Figure 1: Four rotated versions of the same galaxy image, retrieved from the Galaxy Zoo data set by Willett et al. (2013). This data set contains images of galaxies as well as their morphologies, according to experts. For this application, the orientation is arbitrary and does not contain any information.

Still, introducing other types of equivariance in an efficient way in CNNs is not straightforward. On the one hand, many works propose brute-force solutions like (i) considerably increasing the training set (data augmentation), or (ii) artificially multiplying the number of filters by directly applying the desired symmetries onto them. In practice, these solutions will lead both to an increase of the training time and the size of the models. Furthermore, most of these methods do not provide any mathematical guarantee regarding the equivariance. On the other hand, a few works propose solutions to efficiently bring more general equivariances in CNNs while providing mathematical guarantees. Bessel-convolutional neural networks (B-CNNs) are one of those, and rely on a particular representation of images that is more convenient to deal with rotations and reflections.

In this work, improvements of B-CNNs compared to the initial work of Delchevalerie et al. (2021) are presented, which are mainly an extension to $O(2)$-equivariance and an optimal choice for the initial $\nu_{\max}$ and $j_{\max}$ meta-parameters based on the Nyquist sampling theorem. Also, we present how multi-scale equivariance can easily be achieved in B-CNNs. Finally, a more extensive study is performed to assess the performances of B-CNNs compared to other state-of-the-art methods on different data sets. To do so, we present the full mathematical developments of B-CNNs and we give more detailed explanations. The advantage of using B-CNNs regarding both the size of the model and the training set is highlighted, along with theoretical and experimental evidences of the equivariance. Our implementation is available online at `https://github.com/ValDelch/B_CNNs`.

2 Background and Definition of Invariance and Equivariance

Invariance and equivariance are two different notions that need to be clearly defined for the next sections. Let $\Psi(x, y)$ be an image where x and y represent the pixel coordinates, $K(x, y)$ be an arbitrary filter and $\mathcal{G}$ be a set of transformations that can be applied on the image. The operation defined by $*$ is $\mathcal{G}$-invariant if $(g\Psi)(x, y) * K(x, y) = \Psi(x, y) * K(x, y), \forall g \in \mathcal{G}$, and $\mathcal{G}$-equivariant if $(g\Psi)(x, y) * K(x, y) = g(\Psi * K)(x, y), \forall g \in \mathcal{G}$. In other words, invariance means that the results will be exactly the same for all transformations g of the input image, while equivariance means that the results will be also transformed by the action of g.

Convolutional Neural Networks (CNNs) work by applying a succession of convolutions between an input image $\Psi(x, y)$ and some filters. If $K(x, y)$ is one of those particular filters, convolutions are expressed by

$$\Psi(x, y) * K(x, y) = \int_{-R}^{R} \int_{-R}^{R} \Psi(x - x', y - y') K(x', y') \, dx' dy',$$

where R defines the size of the filter. One can now show that CNNs exhibit a translation equivariance (that is, patterns are detected the same way regardless of their particular positions). Indeed, if $\mathcal{T}_{u,v}$ is a translation operator such that it translates the image by an amount of pixels (u, v)

$$\mathcal{T}_{u,v} \Psi(x, y) = \Psi(x + u, y + v),$$

one can show that

$$
\begin{aligned}
\mathcal{T}_{u,v}(\Psi * K)(x, y) &= \int_{-R}^{R} \int_{-R}^{R} \Psi((x - x') + u, (y - y') + v) K(x', y') \, dx' dy' \\
&= \int_{-R}^{R} \int_{-R}^{R} \Psi((x + u) - x', (y + v) - y') K(x', y') \, dx' dy' \\
&= \int_{-R}^{R} \int_{-R}^{R} \mathcal{T}_{u,v} \Psi(x - x', y - y') K(x', y') \, dx' dy' \\
&= (\mathcal{T}_{u,v} \Psi)(x, y) * K(x, y),
\end{aligned}
$$

which matches the definition of $\mathcal{G}$-equivariance defined earlier, and therefore proves the translation equivariance of CNNs. Figure 2 illustrates this equivariance.
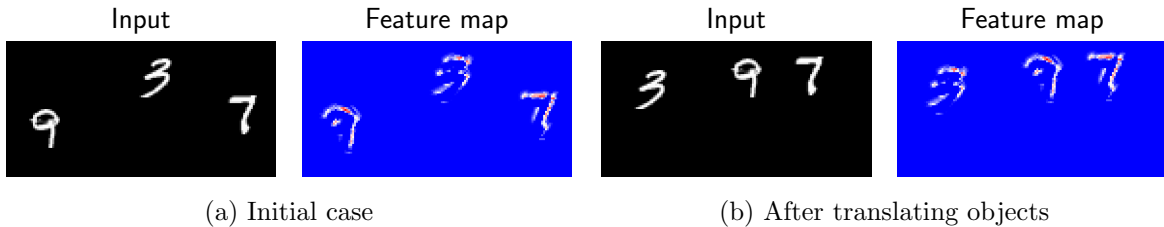
(a) Initial case
(b) After translating objects

Figure 2: Illustration of the translation equivariance in CNNs. Both (a) and (b) are made of the same objects, but at different positions. Nevertheless, CNNs process objects the same way independently of their particular absolute positions.

However, regarding other types of transformations, the equivariance in CNNs is generally not achieved. One can for example consider rotations, by defining R_α as an operator that applies a rotation of an angle α, such that

$$R_\alpha \Psi (x, y) = \Psi (x \cos \alpha - y \sin \alpha, x \sin \alpha + y \cos \alpha).$$

By applying a similar development, it clearly appears that

$$R_\alpha (\Psi * K) (x, y) \neq (R_\alpha \Psi) (x, y) * K (x, y).$$

This is expected as convolution can be seen as element-wise multiplications with a sliding window, and the result of element-wise multiplications depend on the particular orientation of the matrices. This lack of rotation equivariance will be illustrated in Figure 6a.

3 Related Works

Many techniques propose to bring more general equivariance in convolutional neural networks (CNNs). A particular interest was taken in satisfying $SO(2)$ and $O(2)$-equivariance as it is an interesting prior for many applications in image recognition; see for example Chidester et al. (2019) for medical imaging, Dieleman et al. (2015) for astronomical imaging, Li et al. (2020) for satellite imaging and Marcos et al. (2016) for texture recognition. $SO(2)$ is called the special orthogonal group and contains the continuous set of planar rotations, while $O(2)$ is called the orthogonal group and also add all the planar reflections. The different proposed methods can be categorized in different groups: (i) methods that only increase robustness to planar transformations without mathematical guarantees of equivariance, (ii) methods that bring some mathematical guarantees but only for a discrete set of planar transformations (as for example, cyclic C_n and dihedral D_n groups), and (iii) methods that bring mathematical guarantees for the continuous set of transformations.

The most famous technique from the first category is data augmentation (Quiroga et al., 2018). While robustness can be considerably increased with data augmentation, it still requires for the model to learn the equivariance, as it is not used as an explicit constraint. No theoretical guarantees can then be provided, and extracted features will generally not be the same for rotated versions of a particular object. Next to data augmentation, one can also cite spatial transformer networks by Jaderberg et al. (2015), rotation invariant and Fisher discriminative CNNs by Cheng et al. (2016), deformable CNNs by Dai et al.

(2017), and SIFT-CNNs by Kumar et al. (2018). The main drawback of such methods lies in the fact that, as models still learn the equivariance by themselves, many parameters are used to encode redundant information. Therefore, it leads to methods of category (ii) that aim to make model equivariant to discrete groups like C_n or D_n. One can for example cite Group-CNNs by Cohen and Welling (2016), deep symmetry networks by Gens and Domingos (2014), steerable CNNs by Cohen and Welling (2017), steerable filter CNNs by Weiler et al. (2018), dense steerable filter CNNs by Graham et al. (2020), spherical CNNs by Cohen et al. (2018) and Deformation Robust Roto-Scale-Translation Equivariant CNNs by Gao et al. (2021). Compared to category (i), equivariance to a finite number of planar transformations is generally obtained by tying the weights for several transformed versions of the filter. Nevertheless, even if guarantees are now obtained, it is only for a finite set of transformations and it still involves computations with many parameters to encode the equivariance (for example, 5×5 filters in a D_8-invariant convolutional layer will be made of $5 \times 5 \times 8 \times 2 = 400$ parameters[1]). Finally, for the third category (iii), one can cite general $E(2)$-equivariant steerable CNNs ($E(2)$-CNNs) by Weiler and Cesa (2019), where equivariance to continuous groups can be obtained by using a finite number of irreducible representations, harmonic networks (HNets) by Worrall et al. (2017) that use spherical harmonics to achieve a rotational equivariance by maintaining a disentanglement of rotation orders in the network, and Finzi et al. (2020) who generalize equivariance to arbitrary transformations from Lie groups. However, authors of $E(2)$-CNNs highlight that approximating $SO(2)$ (resp, $O(2)$) by using C_n (resp, D_n) groups instead of using a finite number of irreducible representations leads to better results. It follows that $E(2)$-equivariant CNNs are most of the time equivalent to methods of category (ii). Regarding HNets, they are only $SO(2)$-equivariant and involve complex values in the network that are poorly compatible with many already existing tools (for example, activation functions and batch normalization layers should be adapted, saliency maps cannot be easily computed, etc.).

Recently, another type of equivariant CNNs also emerged. While symmetries can be seen as a user constraint for all the previously mentioned techniques, these new equivariant CNNs architectures find by themselves during the training phase the symmetries that should be considered. One can for example cite the work of Dehmamy et al. (2021) in this direction. This is particularly useful when users do not know and have no insight about the symmetries that can be involved in data, or when symmetries are unexpected. However, the aim of such methods differs from the previous ones because symmetries are no longer applied as constraints. Therefore, those methods rely more on the training data, and are useful in a different context of applications. A discussion about the strengths and weaknesses of these methods compared to others is provided at the end of the paper, in Section 8.

Our work is a direct follow-up of the previous work of Delchevalerie et al. (2021), which built on the use of Bessel functions in order to propose a new method that belongs to the third category. Compared to the state of the art, Bessel-convolutional neural networks (B-CNNs) initially proposed a new original technique to bring $SO(2)$ equivariance, while being easy to use with already existing frameworks. In this work, we emphasize the theoretical advantages of B-CNNs by giving more mathematical details. Also, further improvements compared to the prior work of B-CNNs are presented, as for example by making them $O(2)$

---

1. However, note that only $5 \times 5 = 25$ parameters are learnable as the other ones are just transformed versions of the initial filter.

and multi-scale equivariant, and automatically inferring optimal choices for some meta-parameters. Finally, a more extensive comparative study is also carried out to highlight the strengths and weaknesses of different methods.

4 Using Bessel Functions in Image Analysis

In Bessel-convolutional neural networks (B-CNNs), Bessel coefficients are used instead of the raw pixel values conventionally used in vanilla convolutional neural networks (CNNs). This section describes the Bessel functions, and how they can be used to compute these Bessel coefficients. Also, some particular properties of Bessel functions and Bessel coefficients are presented. The aim of this section is to give more insights about the reasons that motivate the use of Bessel functions to achieve different kind of equivariance in CNNs. Compared to the initial work of Delchevalerie et al. (2021), additional mathematical details are provided as well as a discussion on how to perform an optimal choice for the initial meta-parameters $\nu_{\max}$ and $j_{\max}$, and how Bessel coefficients can also be used to express reflections.

4.1 Bessel Functions and Bessel Coefficients

Bessel functions are particular solutions of the differential equation

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

which is known as the Bessel's equation. The solution of this equation can be written as

$$y\left(x\right) = A J_\nu\left(x\right) + B Y_\nu\left(x\right),$$

where A and B are two constants, and $J_\nu\left(x\right)$ and $Y_\nu\left(x\right)$ are called the Bessel functions of the first and second kind, respectively. It has to be noted that these functions are well-defined for orders $\nu \in \mathbb{R}$ in general. In B-CNNs, only the Bessel functions of the first kind are used since $Y_\nu\left(x\right)$ diverges for $x = 0$. Indeed, Bessel functions will be used to express images that can take arbitrary values, including at the origin. Examples of Bessel functions of the first kind for different integer orders ν can be seen in Figure 3a.

From a mathematical point of view, Bessel's equation arises when solving Laplace's or Helmholtz's equation in cylindrical or spherical coordinates. Bessel functions are thus particularly well-known in physics as they appear naturally when solving many important problems, mainly when dealing with wave propagation in cylindrical or spherical coordinates (Riley et al., 2006). Since Bessel functions naturally arise when modeling different problems with circular symmetries in physics, these functions are particularly useful to express more conveniently problems with circular symmetries in other domains. This ascertainment motivated the prior work of Delchevalerie et al. (2021) to express images in a particular basis made of Bessel functions of the first kind.

Bessel functions of the first kind can be used to build a particular basis

$$\left\{ N_{\nu,j} J_\nu\left(k_{\nu,j}\rho\right) e^{i\nu\theta}, \forall \nu, j \in \mathbb{N} \right\}, \text{ where } N_{\nu,j} = 1 / \sqrt{2\pi \int_0^R \rho J_\nu^2\left(k_{\nu,j}\rho\right) d\rho}, \qquad (4.1)$$

(a) $J_\nu(x)$ for $\nu \in \{0, 1, 2, 3\}$

(b) $\frac{dJ_\nu(x)}{dx} = J_\nu'(x)$ for $\nu \in \{0, 1, 2, 3\}$
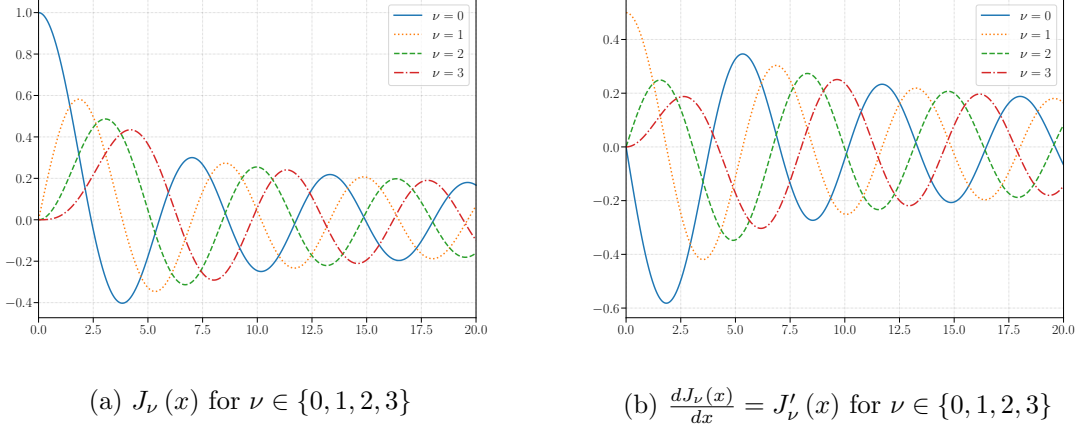
Figure 3: Bessel functions of the first kind are presented along with their derivatives for several integer orders $\nu \in \{0, 1, 2, 3\}$.

for the representation of images defined in a circular domain of radius R, where ρ and θ are the polar coordinates (the Euclidean distance from the origin and the angle with the horizontal axis, respectively). By carefully choosing $k_{\nu,j}$, this basis can be made orthonormal for all squared-integrable functions f such that $f : D^2 \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$ (where the domain D^2 is a disk in $\mathbb{R}^2$). To do so, one can choose $k_{\nu,j}$ such that $J_\nu'(k_{\nu,j}R) = 0$. The proof for the orthonormality of the basis in this case is presented in Appendix A. Another common choice that also leads to orthonormality is to use $J_\nu(k_{\nu,j}R) = 0$. Indeed, these two constraints are suitable since a property of the Bessel functions is that $J_\nu'(x) = \frac{1}{2}(J_{\nu-1}(x) - J_{\nu+1}(x))$. Therefore, applying the constraint on $J_\nu(x)$ or on $J_\nu'(x)$ are both valid solutions that bring orthonormality. However, in our particular case, we choose to apply the constraint on $J_\nu'(x)$ because it makes it more convenient to represent arbitrary functions, as shown by Mayer and Vigneron (1999). The reason is that it exists a solution $k_{\nu,j} = 0$ for $\nu = 0$ such that $J_\nu'(k_{\nu,j}R) = 0$, which would not be the case with the constraint based on $J_\nu(x)$ (see Figure 3). Therefore, the first element in the basis $N_{0,0}J_0(k_{0,0}\rho)e^{i0\theta}$ will be equal to $N_{0,0}$. As the result is constant and does not depend on ρ and θ, this element can be used to describe an arbitrary constant intensity in f. Figure 4 presents some elements of the basis, including the first one. Also, one can point out that when the order ν increases, the angular frequency (the number of zeros along the θ-polar-coordinate) of the basis element increases. On the other side, when the order j increases, the radial frequency (the number of zeros along the ρ-polar-coordinate) increases.

An arbitrary function in polar coordinates $\Psi(\rho, \theta) : D^2 \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$ can be represented in the basis presented in Equation (4.1) as

$$\Psi(\rho, \theta) = \sum_{\nu=-\infty}^{\infty} \sum_{j=0}^{\infty} \varphi_{\nu,j} \ N_{\nu,j} J_\nu(k_{\nu,j}\rho) e^{i\nu\theta}, \tag{4.2}$$
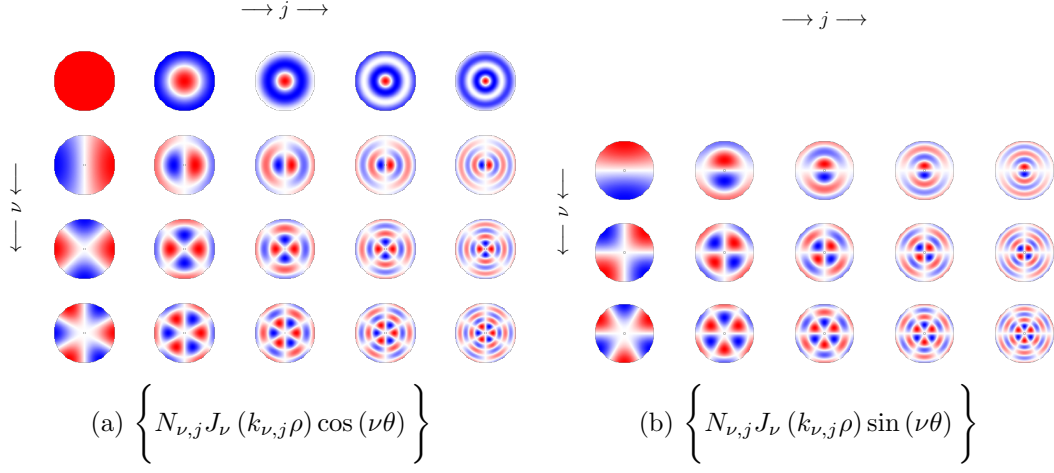
7

(a) $\left\{ N_{\nu,j} J_\nu \left(k_{\nu,j} \rho \right) \cos \left(\nu\theta \right) \right\}$ (b) $\left\{ N_{\nu,j} J_\nu \left(k_{\nu,j} \rho \right) \sin \left(\nu\theta \right) \right\}$

Figure 4: Real (a) and imaginary (b) parts of the basis described by Equation (4.1) for different ν and j. Red and blue correspond to positive and negative values, respectively. One can see that ν is linked to an angular frequency, and j to a radial frequency. Note that for $\nu = 0$, there is no imaginary part.

where $\varphi_{\nu,j} \in \mathbb{C}$ are the Bessel coefficients of $\Psi\left(\rho, \theta\right)$. These $\varphi_{\nu,j}$ are the mathematical projection of $\Psi\left(\rho, \theta\right)$ on the Bessel basis. Therefore, they are obtained by

$$\varphi_{\nu,j} = \int_0^{2\pi} \int_0^R \rho \left[N_{\nu,j} J_\nu \left(k_{\nu,j} \rho \right) e^{i\nu\theta} \right]^* \Psi\left(\rho, \theta\right) d\rho d\theta, \tag{4.3}$$

where the element inside the brackets correspond to the element $\left(\nu, j\right)$ in the Bessel basis. By integrating on D^2, it computes the representation of $\Psi\left(\rho, \theta\right)$ in this basis.

In B-CNNs, images are represented by a set of those Bessel coefficients instead of directly using the raw pixel values. Further motivations about this will be given later. However, one can already point out that Equation (4.2) needs in principle an infinite number of Bessel coefficients in order to faithfully represent the initial function $\Psi\left(\rho, \theta\right)$. From a numerical point of view, these two infinite summations need to be truncated. First of all, one can show that it is not necessary to compute $\varphi_{\nu,j}$ when ν is a negative integer, since $\varphi_{\nu,j}$ and $\varphi_{-\nu,j}$ are not independent. Indeed, if $\nu \in \mathbb{N}$, $J_\nu\left(x\right)$ and $J_{-\nu}\left(x\right)$ are linked by the relation

$$J_{-\nu}\left(x\right) = \left(-1\right)^\nu J_\nu\left(x\right). \tag{4.4}$$

Furthermore, Bessel functions also satisfy

$$J_\nu\left(-x\right) = \left(-1\right)^\nu J_\nu\left(x\right), \tag{4.5}$$

which means that J_ν is an even function if ν is even, and an odd function otherwise. By injecting Equations (4.4) and (4.5) in Equation (4.3), one can show that (proof can be found in Appendix B)

$$\begin{cases} \Re\left(\varphi_{-\nu,j}\right) = \left(-1\right)^\nu \Re\left(\varphi_{\nu,j}\right) \\ \Im\left(\varphi_{-\nu,j}\right) = \left(-1\right)^{\nu+1} \Im\left(\varphi_{\nu,j}\right). \end{cases} \tag{4.6}$$

The infinite summation for ν in Equation (4.2) can be decomposed in two summations, one for $\nu \in \{-\infty, \ldots, -1\}$ and another one for $\nu \in \{0, \ldots, \infty\}$. By exploiting the link between $\varphi_{\nu,j}$ and $\varphi_{-\nu,j}$, the infinite summation for $\nu \in \{-\infty, \ldots, \infty\}$ can then be reduced to a summation for $\nu \in \{0, \ldots, \infty\}$, and it is not necessary to compute Bessel coefficients for negative ν orders. Finally, in order to truncate the infinite summations, two meta-parameters $\nu_{\max}$ and $j_{\max}$ are defined, and the Bessel coefficients are only computed for ν (resp. j) in $\{0, ..., \nu_{\max}$ (resp. $j_{\max})\}$. Nonetheless, it is difficult to make a good choice for these meta-parameters and this may be rather automated by constraining $k_{\nu,j}$ with an upper limit. This is clearly supported by Figure 4, as it shows that high ν (j, respectively) orders correspond to basis elements with an high angular (radial, respectively) frequency. Therefore, as images are sampled on a discrete Cartesian grid, information about frequencies higher than an upper limit cannot be conserved. This upper limit can be determined by the Nyquist frequency, as done by Zhao and Singer (2013), in order to both minimize the aliasing effect and maximize the amount of information preserved by the Bessel coefficients. From now, let us suppose that the radius R of an image is arbitrarily set to 1. If the image is made up of $2n \times 2n$ pixels sampled on a Cartesian grid, it leads to a resolution of $1/n$. Hence, the sampling rate is n and the associated Nyquist frequency (the band-limit) is $n/2$. Therefore, it is optimal to use only the $\varphi_{\nu,j}$ that satisfy the constraint

$$\frac{k_{\nu,j}}{2\pi} \leq \frac{n}{2},$$

because those are the only ones that carry information really contained on the finite Cartesian grid. We then define[2]

$$k_{\max} = \max_{\nu,j} k_{\nu,j} \text{ s.t. } \frac{k_{\nu,j}}{2\pi} \leq \frac{n}{2}. \tag{4.7}$$

One of the consequences of this constraint is that, for larger ν orders, a smaller number of Bessel coefficients will be computed, as $k_{\nu,j}$ will reach $k_{\max}$ more rapidly. Indeed, the zeros of $J_\nu'(x)$ (that are the $k_{\nu,j}$'s if $R=1$) are shifted toward higher x values (see the shifting toward the right for $J_\nu'(x)$ when ν increases in Figure 3b). To conclude this section, the function $\Psi(\rho, \theta)$ will be represented by a matrix with the general form

$$\begin{pmatrix} \varphi_{0,0} & \cdots & \varphi_{0,j} & \cdots & \varphi_{0,j_{\max}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \varphi_{\nu,0} & \cdots & \varphi_{\nu,j} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \varphi_{\nu_{\max},0} & \cdots & 0 & \cdots & 0 \end{pmatrix},$$

where each non-zero element corresponds to values for ν and j that satisfy $k_{\nu,j} \leq k_{\max}$. Figure 5 presents an example where $\Psi(\rho, \theta)$ is an arbitrary image. Bessel coefficients are computed in this particular case with Equation (4.3), and the inverse transformation described by Equation (4.2) is also performed in the middle part of the figure to check how

---

2. It is interesting to mention that this constraint is also a common choice in numerical physics, where $\frac{k}{2\pi} = \frac{1}{\lambda}$, λ being the wavelength. It is meaningless to use larger values for k, as it corresponds to wavelengths smaller than the resolution of space.

much information is preserved by the Bessel coefficients. It also shows how easy it is to apply rotations and reflections with Bessel coefficients as explained below, in Section 4.2 and Section 4.3.

4.2 Effect of Rotations

To understand why using Bessel coefficients is more convenient than using raw pixel values, one can determine the consequence of a rotation of $\Psi(\rho, \theta)$ on $\varphi_{\nu,j}$. Let $\Psi^{\mathrm{rot}}(\rho, \theta)$ be the rotated version of $\Psi(\rho, \theta)$ for an angle $\alpha \in [0, 2\pi[$, that is, $\Psi^{\mathrm{rot}}(\rho, \theta) = \Psi(\rho, \theta - \alpha)$. Its Bessel coefficients are given by

$$\varphi_{\nu,j}^{\mathrm{rot}} = \int_0^{2\pi} \int_0^R \rho \left[N_{\nu,j} J_\nu (k_{\nu,j}\rho) e^{i\nu\theta} \right]^* \Psi^{\mathrm{rot}}(\rho, \theta) \, d\rho d\theta.$$

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

$$\varphi_{\nu,j}^{\mathrm{rot}} = \int_0^{2\pi} \int_0^R \rho \left[N_{\nu,j} J_\nu (k_{\nu,j}\rho) e^{i\nu\theta'} \right]^* \Psi(\rho, \theta') e^{-i\nu\alpha} d\rho d\theta' = \varphi_{\nu,j} e^{-i\nu\alpha}. \qquad (4.8)$$

Therefore, a rotation of an arbitrary function by an angle α only modifies its Bessel coefficients by a multiplication factor $e^{-i\nu\alpha}$. This motivated the development of B-CNNs as it makes rotations conveniently expressed in the Fourier-Bessel transform domain (analogously to how the Fourier transform maps translations to multiplications by complex exponentials). The upper part of Figure 5 illustrates this property (the image is rotated by $\frac{\pi}{2}$ after multiplying its Bessel coefficients by $e^{-i\nu\frac{\pi}{2}}$).

4.3 Effect of Reflections

In addition to rotations, Bessel coefficients are also particularly useful when it comes to express reflections. To check this, let $\Psi^{\mathrm{ref}}(\rho, \theta)$ be the reflected version of $\Psi(\rho, \theta)$ along the vertical axis[3]. The Bessel coefficients of $\Psi^{\mathrm{ref}}(\rho, \theta) = \Psi(\rho, \pi - \theta)$ are given by

$$\varphi_{\nu,j}^{\mathrm{ref}} = \int_0^{2\pi} \int_0^R \rho \left[N_{\nu,j} J_\nu (k_{\nu,j}\rho) e^{i\nu\theta} \right]^* \Psi^{\mathrm{ref}}(\rho, \theta) \, d\rho d\theta.$$

Similarly to what is done for arbitrary rotations, one can define $\theta' = \pi - \theta$. This leads to

$$\varphi_{\nu,j}^{\mathrm{ref}} = -\int_\pi^{-\pi} \int_0^R \rho \left[N_{\nu,j} J_\nu (k_{\nu,j}\rho) e^{i\nu(\pi-\theta')} \right]^* \Psi(\rho, \theta') \, d\rho d\theta'.$$

It is shown in Appendix B that $N_{\nu,j} = N_{-\nu,j}$ and that $k_{-\nu,j} = k_{\nu,j}$. By exploiting this along with Equation (4.4), one can show that

$$\begin{aligned}
\varphi_{\nu,j}^{\mathrm{ref}} &= \int_0^{2\pi} \int_0^R \rho \left[N_{-\nu,j} (-1)^\nu J_{-\nu} (k_{-\nu,j}\rho) e^{-i\nu\theta'} \right]^* e^{-i\nu\pi} \Psi(\rho, \theta') \, d\rho d\theta' \\
&= \int_0^{2\pi} \int_0^R \rho \left[N_{-\nu,j} J_{-\nu} (k_{-\nu,j}\rho) e^{-i\nu\theta'} \right]^* \Psi(\rho, \theta') \, d\rho d\theta' \\
&= \varphi_{-\nu,j}.
\end{aligned} \qquad (4.9)$$

---

3. The reflection along the horizontal axis is not needed, since it can be decomposed as a vertical reflection and a rotation of π radians. By composition, reflection equivariance along the horizontal axis is automatically achieved if the layer is equivariant to rotations and reflections along the vertical axis.
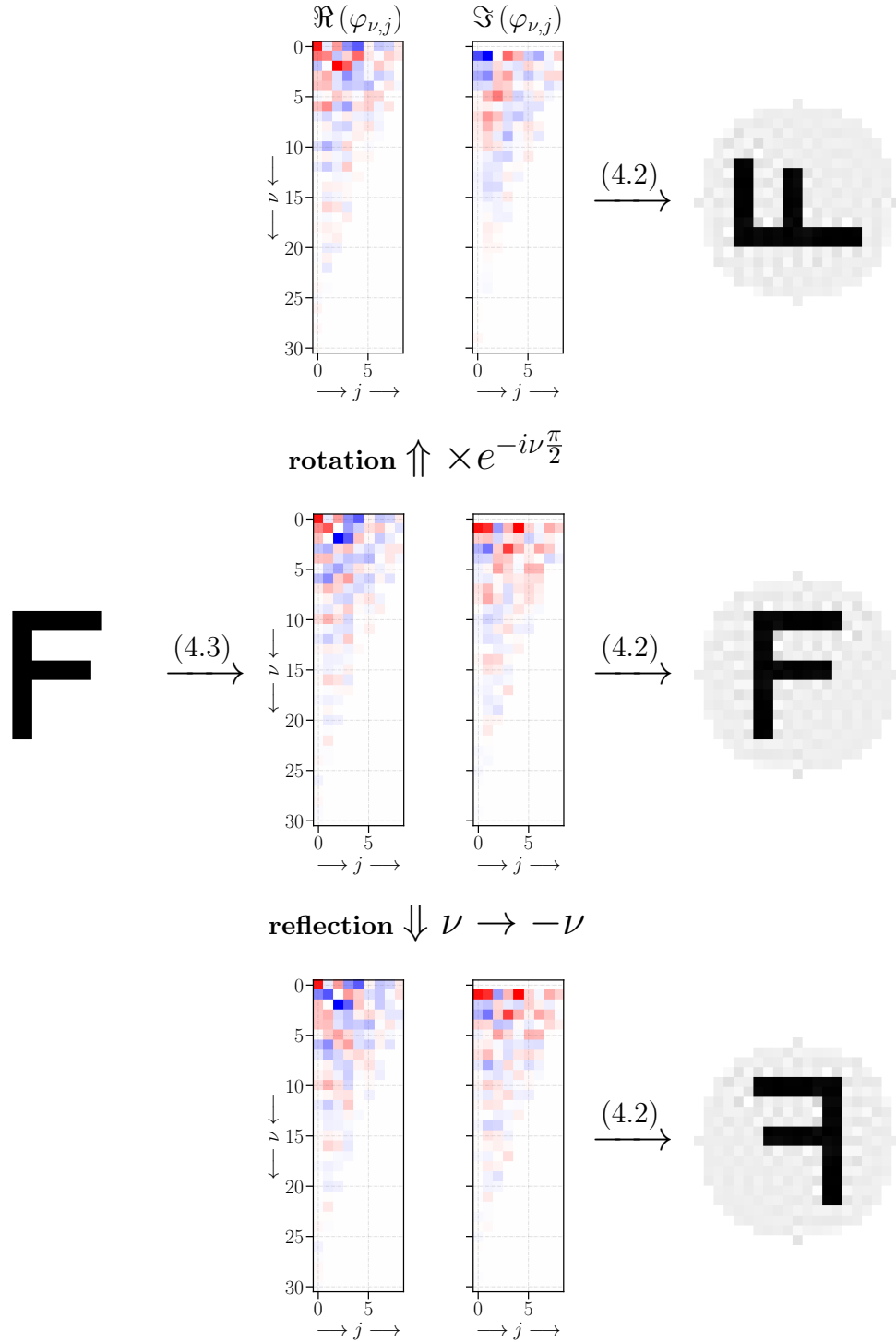
Figure 5: Decomposition of an arbitrary image in some of its Bessel coefficients. Those Bessel coefficients constitute a particular representation of the image and can be used to recover it thanks to Equation (4.2) (middle part). It also illustrates how Bessel coefficients can be conveniently used to apply rotations (upper part) and reflections (lower part).

Therefore, performing a reflection of the image only switches the Bessel coefficients $\varphi_{\nu,j}$ to $\varphi_{-\nu,j}$. Thanks to Equations (4.6), it is equivalent to changing the sign of the real (resp. imaginary) part of the Bessel coefficient if ν is odd (resp. even). Therefore, in addition to rotations, Bessel coefficients are also really convenient to express reflections. This is illustrated in the lower part of Figure 5, where the image is reflected vertically after switching each $\varphi_{\nu,j}$ with $\varphi_{-\nu,j}$.

5 Designing Operations with Bessel Coefficients

In CNNs, the main mathematical operation is a convolutional product between the different filters and the image (or feature maps if deeper in the network). Each filter sweeps the image locally and the weights are multiplied with the raw pixel values. However, in B-CNNs, the aim is to use Bessel coefficients instead of raw pixel values to benefit from the properties described in the previous sections. Yet, the key operation between the parameters of the network (filters) and the images needs to be adapted. This section first presents the mathematical operation used to achieve equivariance under rotation. After that, the initial work of Delchevalerie et al. (2021) is extended to also achieve equivariance under reflection.

5.1 A Rotation Equivariant Operation

The convolution performed in CNNs between an arbitrary image $\Psi(x,y)$ and a particular kernel $K(x,y)$ defined for $(x,y) \in [-R,R] \times [-R,R]$ can be written

$$a(x,y) = \Psi(x,y) * K(x,y) = \int_{-R}^{R} \int_{-R}^{R} \Psi(x-x', y-y') K(x', y') \, dx' dy'.$$

By defining $\Psi^{(x,y)}(x',y') = \Psi(x-x', y-y')$ and converting the integration from Cartesian to polar coordinates, it leads to

$$a(x,y) = \int_{0}^{2\pi} \int_{0}^{R} \Psi^{(x,y)}(\rho,\theta) K(\rho,\theta) \rho \, d\rho d\theta. \tag{5.1}$$

Now, in order to obtain a result that is invariant to the particular orientation of $\Psi^{(x,y)}(\rho,\theta)$, one can decompose it in its Bessel coefficients $\varphi_{\nu,j}^{(x,y)}$ and use Equation (4.8) to implement arbitrary rotations. Next, the idea is to combine this with an integration over α in order to equally consider all the possible orientations of the original image while multiplying it with the kernel, resulting in a rotation invariance. We introduce thus a new rotation equivariant convolutional operation described by

$$a(x,y) = \frac{1}{2\pi} \int_{0}^{2\pi} \Big| \int_{0}^{2\pi} \int_{0}^{R} \sum_{\nu,j} \varphi_{\nu,j}^{(x,y)} e^{-i\nu\alpha} N_{\nu,j} J_\nu(k_{\nu,j}\rho) e^{i\nu\theta} K(\rho,\theta) \rho \, d\rho d\theta \Big|^2 d\alpha$$

$$= \frac{1}{2\pi} \int_{0}^{2\pi} \Big| \sum_{\nu,j} \varphi_{\nu,j}^{(x,y)} e^{-i\nu\alpha} \int_{0}^{2\pi} \int_{0}^{R} \left[N_{\nu,j} J_\nu(k_{\nu,j}\rho) e^{-i\nu\theta} \right]^* K(\rho,\theta) \rho \, d\rho d\theta \Big|^2 d\alpha$$

$$= \frac{1}{2\pi} \int_{0}^{2\pi} \Big| \sum_{\nu,j} \varphi_{\nu,j}^{(x,y)} e^{-i\nu\alpha} \kappa_{\nu,j}^* \Big|^2 d\alpha, \tag{5.2}$$

where $\kappa_{\nu,j}$ refers to the Bessel coefficients of the kernel $K(\rho, \theta)$. Thanks to the integration over α from 0 to 2π and the multiplication of $\varphi_{\nu,j}^{(x,y)}$ by $e^{-i\nu\alpha}$ to describe the effect of rotations, the operation with the kernel is performed for all continuous rotations $\Psi^{(x,y)}(\rho, \theta - \alpha)$ of the original image, where $\alpha \in [0, 2\pi[$. Therefore, $a(x,y)$ should not depend on the particular initial orientation anymore. A squared modulus $|\cdot|^2$ is introduced in our operation since, without it, one obtains

$$a(x,y) = \frac{1}{2\pi} \sum_{\nu,j} \varphi_{\nu,j}^{(x,y)} \kappa_{\nu,j}^* \int_0^{2\pi} e^{-i\nu\alpha} d\alpha = \sum_j \varphi_{0,j}^{(x,y)} \kappa_{0,j}^*,$$

and only the subset of coefficients $\{\varphi_{0,j}^{(x,y)}, \forall j\}$ will contribute to $a(x,y)$. This subset alone however does not constitute a faithful representation of the image. The operation without the squared modulus would therefore inevitably lead to an important loss of information. The factor $\frac{1}{2\pi}$ was introduced finally for normalization purpose.

Computing Equation (5.2) seems not straightforward as it requires to perform a numerical integration. However, one can develop it further in order to obtain an analytical solution, which will be much more convenient to implement in practice. To do so, one can first develop the squared modulus given that

$$\begin{aligned}
\Big| \sum_{i=1}^{k} \alpha_i |z_i| e^{i\theta_i} \Big|^2 &= \sum_{m,j} \Re(\alpha_m) \Re(\alpha_j) |z_m z_j| \cos(\theta_m - \theta_j) \\
&+ \sum_{m,j} \Im(\alpha_m) \Im(\alpha_j) |z_m z_j| \cos(\theta_m - \theta_j) \\
&- 2 \sum_{m,j} \Im(\alpha_m) \Re(\alpha_j) |z_m z_j| \sin(\theta_m - \theta_j),
\end{aligned} \tag{5.3}$$

where $\alpha_i \in \mathbb{C}$ and $z_i = |z_i| e^{i\theta_i} \in \mathbb{C}$. By re-writing the complex valued Bessel coefficients $\varphi_{\nu,j}^{(x,y)}$ as $|\varphi_{\nu,j}^{(x,y)}| e^{i\theta_{\nu,j}}$, it leads to

$$\begin{aligned}
a(x,y) =& \frac{1}{2\pi} \int_0^{2\pi} \sum_{\substack{\nu,j \\ \nu',j'}} \Re(\kappa_{\nu,j}^*) \Re(\kappa_{\nu',j'}^*) |\varphi_{\nu,j}^{(x,y)} \varphi_{\nu',j'}^{(x,y)}| \cos(\theta_{\nu,j} - \theta_{\nu',j'} - \alpha(\nu - \nu')) d\alpha \\
&+ \frac{1}{2\pi} \int_0^{2\pi} \sum_{\substack{\nu,j \\ \nu',j'}} \Im(\kappa_{\nu,j}^*) \Im(\kappa_{\nu',j'}^*) |\varphi_{\nu,j}^{(x,y)} \varphi_{\nu',j'}^{(x,y)}| \cos(\theta_{\nu,j} - \theta_{\nu',j'} - \alpha(\nu - \nu')) d\alpha \\
&- \frac{1}{\pi} \int_0^{2\pi} \sum_{\substack{\nu,j \\ \nu',j'}} \Im(\kappa_{\nu,j}^*) \Re(\kappa_{\nu',j'}^*) |\varphi_{\nu,j}^{(x,y)} \varphi_{\nu',j'}^{(x,y)}| \sin(\theta_{\nu,j} - \theta_{\nu',j'} - \alpha(\nu - \nu')) d\alpha.
\end{aligned}$$

In this equation, only the trigonometric functions are α-dependent. Calculating the remaining integrals leads to

$$\int_0^{2\pi} \mathrm{sc}(\theta_{\nu,j} - \theta_{\nu',j'} - \alpha(\nu - \nu')) d\alpha = \begin{cases} 2\pi\,\mathrm{sc}(\theta_{\nu,j} - \theta_{\nu',j'}) & \textbf{if } \nu = \nu' \\ 0 \text{ otherwise,} \end{cases} \tag{5.4}$$

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

$$a\left(x,y\right) = \sum_{\nu}\left[\sum_{j,j'}\Re\left(\kappa_{\nu,j}^{*}\right)\Re\left(\kappa_{\nu,j'}^{*}\right)\left|\varphi_{\nu,j}^{(x,y)}\varphi_{\nu,j'}^{(x,y)}\right|\cos\left(\theta_{\nu,j}-\theta_{\nu,j'}\right)\right.$$

$$+\sum_{j,j'}\Im\left(\kappa_{\nu,j}^{*}\right)\Im\left(\kappa_{\nu,j'}^{*}\right)\left|\varphi_{\nu,j}^{(x,y)}\varphi_{\nu,j'}^{(x,y)}\right|\cos\left(\theta_{\nu,j}-\theta_{\nu,j'}\right)$$

$$\left.-2\sum_{j,j'}\Im\left(\kappa_{\nu,j}^{*}\right)\Re\left(\kappa_{\nu,j'}^{*}\right)\left|\varphi_{\nu,j}^{(x,y)}\varphi_{\nu,j'}^{(x,y)}\right|\sin\left(\theta_{\nu,j}-\theta_{\nu,j'}\right)\right], \tag{5.5}$$

and by using once again Equation (5.3), Equation (5.2) finally leads to

$$a\left(x,y\right) = \sum_{\nu}\left|\sum_{j}\kappa_{\nu,j}^{*}\varphi_{\nu,j}^{(x,y)}\right|^{2}. \tag{5.6}$$

Thanks to the use of Bessel coefficients instead of raw pixel values, the classic convolution has been modified into Equation (5.6) in order to achieve rotation equivariance. The operation is still a convolution-like operation as the filters still sweep the input image to progressively construct the feature maps. Therefore, feature maps will be obtained in both a translation and rotation equivariant way. Feature maps in B-CNNs are equivariant because they are obtained by a succession of local invariances (in other words, the key operation between the image and the filter in Equation 5.6 is rotation invariant, but as it is successively performed for local parts of the input, it leads to a global rotation-equivariance). Nevertheless, by introducing reduction mechanisms in the models to make the feature maps in the final layer of size 1×1 (by using pooling layers or avoiding padding), the global equivariance can lead to global invariance. Figure 6 presents an example where the equivariance of B-CNNs is compared to vanilla CNNs, and it also presents how a succession of equivariant feature maps lead in this case to a global invariance of the model.

Finally, one can point out that $a\left(x,y\right) \in \mathbb{R}$ (even if $\kappa_{\nu,j} \in \mathbb{C}$ and $\varphi_{\nu,j}^{(x,y)} \in \mathbb{C}$). This is an important property as it allows this operation to be compatible with existing deep learning frameworks (for example, classic activation functions and batch normalization can be used), as opposed to the work of Worrall et al. (2017) that uses values in the complex domain. It is also worth to mention that this operation is *pseudo*-injective, meaning that different images will lead to different values of a (*pseudo* makes reference to the exception when an image is compared to a rotated version of itself). The proof for the pseudo-injectivity is presented in Appendix C.

5.2 Adding the Reflection Equivariance

In order to make B-CNNs also equivariant to reflections, and thus $O(2)$ equivariant, one can check how Equation (5.6) behaves for an image and its reflection. To do so, let us compute the quantity

$$\delta = \sum_{\nu}\left|\sum_{j}\kappa_{\nu,j}^{*}\varphi_{\nu,j}\right|^{2} - \sum_{\nu}\left|\sum_{j}\kappa_{\nu,j}^{*}\varphi_{\nu,j}^{\mathrm{ref}}\right|^{2},$$

where $\varphi_{\nu,j}^{\mathrm{ref}}$ are the Bessel coefficients of a reflected version of $\Psi\left(\rho,\theta\right)$. For the operation to be invariant under reflection, δ should therefore be equal to 0. Thanks to Equation (4.9)
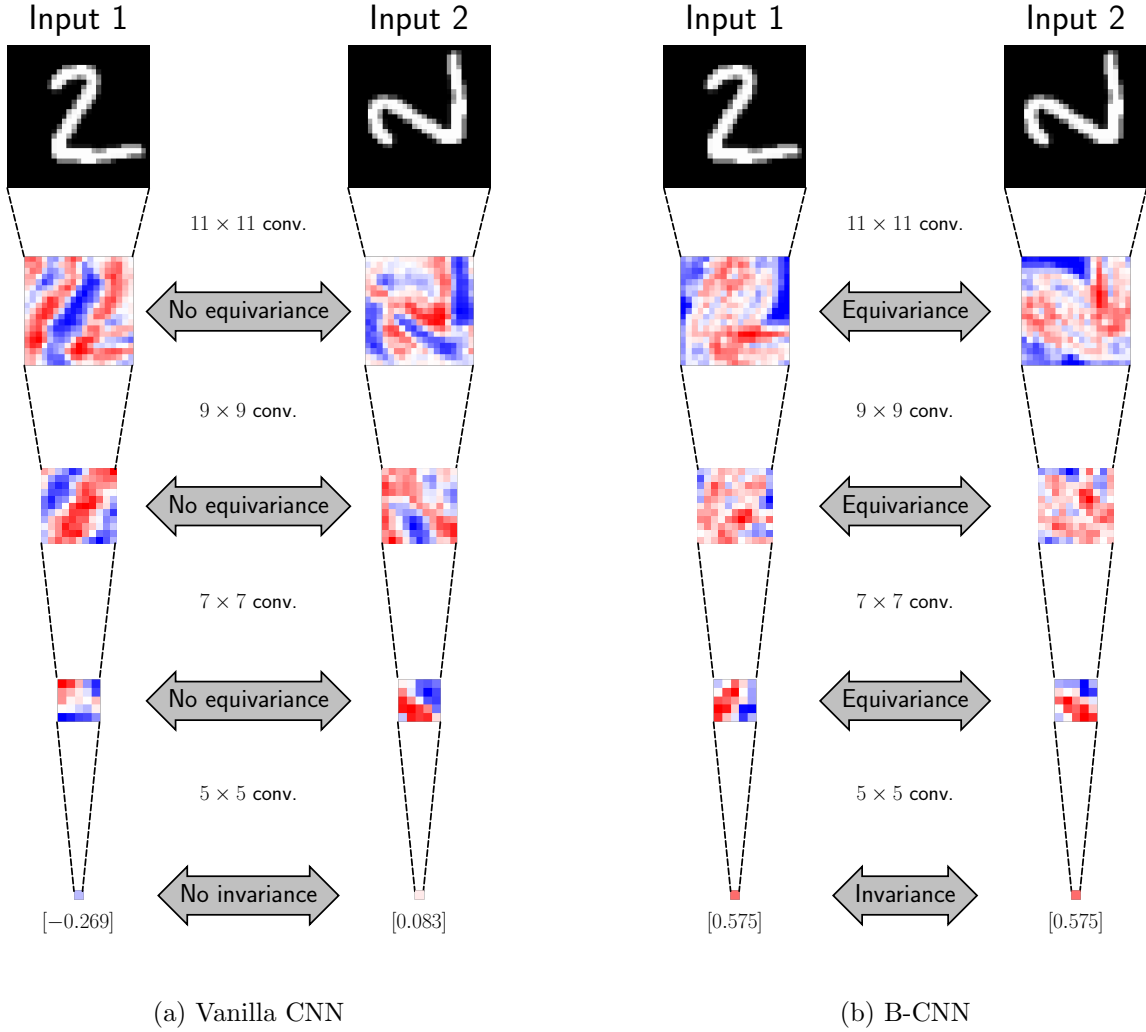
(a) Vanilla CNN   (b) B-CNN

Figure 6: Feature maps obtained with 4 vanilla (a) and Bessel (b) convolutional layers on a sample drown from the MNIST data set (LeCun et al., 1998). Input 2 is the exact same image as Input 1, except for a $\frac{\pi}{2}$ rotation. Inputs are of size 29×29, and no padding is used such that the size of feature maps is progressively reduced until it reaches the final size of 1×1. One can observe that no equivariance is obtained in the feature maps for the vanilla CNN, leading to different final results. However, B-CNN provides equivariance for the feature maps, leading to a final result that is invariant to the orientation of the image. Indeed, for B-CNN, feature maps for Input 2 are rigorously identical up to a $\frac{\pi}{2}$ rotation, which is not the case for vanilla CNN. It has to be noted that in this example, equivariance/invariance are rigorously achieved because $\frac{\pi}{2}$ rotations are well-defined on Cartesian grids.

and Equation (4.6), we can write

$$\delta = \sum_{\nu} |\sum_{j} \kappa_{\nu,j}^* \varphi_{\nu,j}|^2 - \sum_{\nu} |\sum_{j} \kappa_{\nu,j}^* \left((-1)^{\nu} \Re(\varphi_{\nu,j}) + i\,(-1)^{\nu+1}\,\Im(\varphi_{\nu,j})\right)|^2$$

$$= \sum_{\nu} |\sum_{j} \kappa_{\nu,j}^* \varphi_{\nu,j}|^2 - \sum_{\nu} |(-1)^{\nu} \sum_{j} \kappa_{\nu,j}^* \varphi_{\nu,j}^*|^2$$

$$= \sum_{\nu} \left[|\sum_{j} \kappa_{\nu,j}^* \varphi_{\nu,j}|^2 - |\sum_{j} \kappa_{\nu,j}^* \varphi_{\nu,j}^*|^2 \right].$$

By using again the development that led to Equation (5.5), one can show that

$$\delta = -4 \sum_{\nu,j,j'} \Im\left(\kappa_{\nu,j}^*\right) \Re\left(\kappa_{\nu,j'}^*\right) |\varphi_{\nu,j}\varphi_{\nu,j'}| \sin\left(\theta_{\nu,j} - \theta_{\nu,j'}\right).$$

It means that δ may be different from 0 and an $O(2)$ equivariance will in general not be achieved. The objective is now to slightly modify Equation (5.6) in order to obtain $\delta = 0$. To do so, one can see that the terms that do not vanish are those that involve $\Im\left(\kappa_{\nu,j}^*\right) \Re\left(\kappa_{\nu,j'}^*\right)$. By avoiding such crossed terms between the real and imaginary parts of $\kappa_{\nu,j}$, one can obtain $\delta = 0$ and therefore a reflection equivariance, while still keeping the rotation equivariance. This can be achieved by using

$$a\,(x,y) = \sum_{\nu} |\sum_{j} \Re\left(\kappa_{\nu,j}^*\right) \varphi_{\nu,j}^{(x,y)}|^2 + |\sum_{j} \Im\left(\kappa_{\nu,j}^*\right) \varphi_{\nu,j}^{(x,y)}|^2. \tag{5.7}$$

To conclude, B-CNNs can be made $SO(2)$ equivariant (that is, equivariant to all the continuous planar rotations) by using Equation (5.6) as operation between the filters and the images, or $O(2)$ equivariant (that is, equivariant to all the continuous planar rotations and reflections) by using Equation (5.7) instead. Users can decide, based on the application, which equivariance is required.

6 Bessel-Convolutional Neural Networks

This section constitutes a sum up and gives more intuition about the global working of B-CNNs. It also presents an efficient way for implementing the previous developments in convolutional neural networks architectures. It is finally shown how bringing multi-scale equivariance is straightforward with this implementation. Multi-scale equivariance means that patterns can be detected even if they appear at slightly different scales in the images. Developments in this section are presented in the particular case of $SO(2)$ equivariance. We will hence consider Equation (5.6) instead of Equation (5.7). However, developments can easily be adapted for this second case.

6.1 B-CNNs From a Practical Point of View

The key modification in B-CNNs compared to vanilla CNNs is to replace the element-wise multiplication between raw pixel values and the filters by the mathematical operation described by Equation (5.6). Filters, which are described by their Bessel coefficients $\{\kappa_{\nu,j}\}$,

sweep locally the image and the Bessel coefficients for the sub-region of the image $\{\varphi_{\nu,j}^{(x,y)}\}$ are computed. Equation (5.6) is then used to progressively build feature maps. This process is summarized in Figure 7. However, implementing B-CNNs by using this straightforward strategy requires to perform many Bessel coefficients decompositions, which are really expensive.
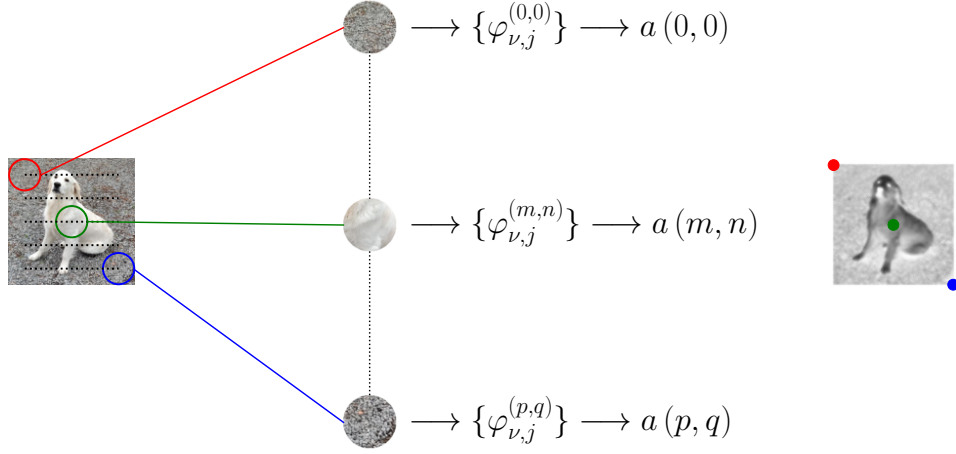

Figure 7: Illustration of how feature maps are obtained in B-CNNs. One can see that B-CNNs still work in a convolutional fashion, but the key operation between the filters and the image is modified.

A more efficient implementation can be obtained by developing Equation (5.6) with Equation (4.3). Indeed, it gives

$$a(x,y) = \sum_\nu |\sum_j \kappa_{\nu,j}^* \int_0^{2\pi} \int_0^R \rho \left[N_{\nu,j} J_\nu(k_{\nu,j}\rho) e^{i\nu\theta} \right]^* \Psi^{(x,y)}(\rho,\theta)\, d\rho d\theta|^2$$

$$= \sum_\nu |\int_0^{2\pi} \int_0^R \Psi^{(x,y)}(\rho,\theta) \sum_j \rho \left[N_{\nu,j} J_\nu(k_{\nu,j}\rho) e^{i\nu\theta} \right]^* \kappa_{\nu,j}^*\, d\rho d\theta|^2,$$

and, by converting to Cartesian coordinates thanks to $\widetilde{\theta} \equiv \widetilde{\theta}(x,y) = \arctan\frac{y}{x}$ and $\widetilde{\rho} \equiv \widetilde{\rho}(x,y) = \sqrt{x^2 + y^2}$, it leads to

$$a(x,y) = \sum_\nu |\int_{-R}^R \int_{-R}^R \Psi^{(x,y)}(x',y') \sum_j \left[N_{\nu,j} \widetilde{J}_\nu\left(k_{\nu,j}\widetilde{\rho}\right) e^{i\nu\widetilde{\theta}} \right]^* \kappa_{\nu,j}^*\, dx' dy'|^2, \qquad (6.1)$$

where $\widetilde{J}_\nu(k_{\nu,j}\rho)$ is defined as

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

17

This definition of $\widetilde{J}_\nu$ is required to compensate the fact that we are now integrating over the square domain $[-R, R] \times [-R, R]$ instead of the circular domain D^2 of radius R. By defining

$$T_{\nu,j}(x, y) = N_{\nu,j} \widetilde{J}_\nu \left(k_{\nu,j} \widetilde{\rho} \right) e^{-i\nu \widetilde{\theta}}, \tag{6.2}$$

one can finally obtain

$$a(x, y) = \sum_\nu \left| \Psi(x, y) * \sum_j T_{\nu,j}(x, y) \kappa_{\nu,j}^* \right|^2$$

$$= \sum_\nu \left| \Psi(x, y) * F_\nu(x, y) \right|^2, \tag{6.3}$$

where

$$F_\nu(x, y) = \sum_j T_{\nu,j}(x, y) \kappa_{\nu,j}^*. \tag{6.4}$$

From a numerical point of view, there are two main advantages in using Equation (6.3) instead of directly implementing Equation (5.6) as presented in Figure 7. Firstly, this equation directly involves the input $\Psi(x, y)$ instead of its Bessel coefficients. Secondly, $T_{\nu,j}(x, y)$ does not depend on the input or the weights of the model. Therefore, it can be computed only once at the initialization of the model. After discretizing space, $T_{\nu,j}(x, y)$ can be seen as a transformation matrix that maps the weights of the model from the Fourier-Bessel transform domain (the Bessel coefficients of the filters $\{\kappa_{\nu,j}\}$) to a set of filters in the direct space $\{F_\nu(x, y)\}$. The feature maps can then be obtained by applying classic convolutions between the input and these filters. Also note that the $\nu_{\max}$ convolutions that need to be performed can be wrapped with the output channel dimension to only perform one call to the convolution function. Algorithm 1 presents how to efficiently implement a B-CNN layer in practice, including the initialization step and the forward propagation.

6.2 Numerical Complexity of B-CNNs

Regarding the computational complexity, if the input of a vanilla CNN layer is of size $[W \times H \times C_{in}]$ and if it implements C_{out} filters of size $[2n \times 2n]$, the number of mathematical operations to perform for a forward pass (assuming that padding is used along with unitary strides) is

$$N_{op} = WH \left[4n^2 C_{in} + \left(4n^2 C_{in} - 1 \right) \right] C_{out}$$

$$= 8WHn^2 C_{in} C_{out} - WHC_{out}.$$

Indeed, C_{out} filters made of $2n \times 2n \times C_{in}$ parameters will sweep WH local parts of the input image. For each local part, $4n^2 C_{in}$ multiplications are then performed as well as $4n^2 C_{in} - 1$ additions. Therefore, it leads to a computational complexity of $\mathcal{O}\left(WHn^2 C_{in} C_{out} \right)$. Compared to vanilla CNNs, B-CNNs need to perform more operations as it is required (i) to compute $F_\nu(x, y)$, (ii) to perform $2\nu_{\max}$ times more convolutions and (iii) to compute squared modulus and a sum over ν. Step (i) consists of $\nu_{\max}$ matrix multiplications, that involve for each element in the final matrix $j_{\max}$ scalar multiplications and $j_{\max} - 1$ additions. Step (ii) is the same that for vanilla CNNs, except that one should perform this for each ν and

---

Algorithm 1: Implementation of a B-CNN layer

/* Initialization (performed only one time)   */

Input: The size of the filters $(2n + 1)$; the number of output channels C_{out}

Compute $k_{\max} = \frac{2n+1}{2}\pi$, $\nu_{\max}$ and $j_{\max}$ (Equation 4.7)

Let **K** be a randomly-initialized trainable tensor with shape $[\nu_{\max}, j_{\max}, C_{in} \times C_{out}]$
forall (ν, j) *s.t.* $k_{\nu,j} > k_{\max}$ **do**
   /* Parameters s.t. $k_{\nu,j} > k_{\max}$ are set to 0   */
   $\mathbf{K}[\nu, j, :] = 0$

Let **T** be a zero-initialized tensor with shape $[\nu_{\max}, 2n + 1, 2n + 1, j_{\max}]$
forall (ν, j) *s.t.* $k_{\nu,j} \leq k_{\max}$ **do**
   forall $(x, y) \in \{-1, -1 + \frac{1}{n}, \ldots, 1 - \frac{1}{n}, 1\} \times \{-1, -1 + \frac{1}{n}, \ldots, 1 - \frac{1}{n}, 1\}$ **do**
      $\mathbf{T}[\nu, x, y, j] = N_{\nu,j}\widetilde{J}_\nu\left(k_{\nu,j}\sqrt{x^2 + y^2}\right)e^{-i\nu\left(\arctan\frac{y}{x}\right)}$ (Equation 6.2)

Reshape **T** into $[\nu_{\max}, (2n + 1) \times (2n + 1), j_{\max}]$

/* Forward propagation   */

Input : A tensor **I** of N images of size $[N, W, H, C_{in}]$
Output: The tensor **A** with the corresponding feature maps

Let **F** be a tensor with shape $[\nu_{\max}, (2n + 1) \times (2n + 1), C_{in} \times C_{out}]$
for *($\nu = 0$; $\nu < \nu_{\max}$; $\nu = \nu + 1$)* **do**
   /* **F** can be computed thanks to $\nu_{\max}$ matrix multiplications   */
   $\mathbf{F}[\nu, :, :] = \mathrm{matmul}\left(\mathbf{T}[\nu, :, :], \mathbf{K}[\nu, :, :]\right)$ (Equation 6.4)
Reshape **F** into $[(2n + 1), (2n + 1), C_{in}, \nu_{\max}C_{out}]$

/* W_{out} and H_{out} depend on the padding and the stride   */
$\mathbf{Z} = \mathrm{conv2d}\left(\mathbf{I}, \mathbf{F}\right)$
Reshape **Z** into $[N, W_{out}, H_{out}, C_{out}, \nu_{\max}]$

Let **A** be a zero-initialized tensor with shape $[N, W_{out}, H_{out}, C_{out}]$
for *($\nu = 0$; $\nu < \nu_{\max}$; $\nu = \nu + 1$)* **do**
   $\mathbf{A} \mathrel{+}= \left|\mathbf{Z}[:, :, :, :, \nu]\right|^2$ (Equation 6.3)

return A

---

both for the real and imaginary parts of $F_\nu(x, y)$. Finally, the squared modulus in step (iii) involves $2WHC_{out}$ multiplications and WHC_{out} additions, and the final summation over ν involves $\nu_{\max} - 1$ additions. At the end, the final numbers of operations to perform

for each step are

$$N_{op}^{(i)} = \nu_{\max} \left[j_{\max} + (j_{\max} - 1) \right] 4n^2 C_{in} C_{out},$$
$$N_{op}^{(ii)} = WH \left[4n^2 C_{in} + \left(4n^2 C_{in} - 1 \right) \right] 2C_{out} \nu_{\max},$$
$$N_{op}^{(iii)} = 3WHC_{out}\nu_{\max} + (\nu_{\max} - 1) WHC_{out}.$$

However, by looking at Figure 8, one can see that both $\nu_{\max}$ and $j_{\max}$ scale linearly with n, thanks to the constraint expressed by Equation (4.7). It follows that

$$N_{op} \propto n^4 C_{in} C_{out} + WHn^3 C_{in} C_{out} + WHnC_{out},$$

resulting in a computational complexity of $\mathcal{O}\left(WHn^3 C_{in} C_{out}\right)$ for a forward pass in a B-CNN layer. Since generally $n \ll \min\left(W, H, C_{in}, C_{out}\right)$, the increase in computational time compared to vanilla CNNs is reasonable with respect to the gain in expressiveness. Furthermore, the computational complexity of $E(2)$-equivariant models (Weiler and Cesa, 2019) for a symmetry group $\mathcal{G}$ is $\mathcal{O}\left(WHn^2 C_{in} C_{out}|\mathcal{G}|\right)$ as C_{out} is artificially increased by the number of discrete operations in $\mathcal{G}$. Therefore, if $n < |\mathcal{G}|$ (which is generally the case as $n = 5, 7$ or 9, and $\mathcal{G} = C_8, C_{16}, D_8$ or D_{16} leading to $|\mathcal{G}| = 8, 16, 16$ or 32, respectively), B-CNNs are therefore more efficient from a computational point of view.
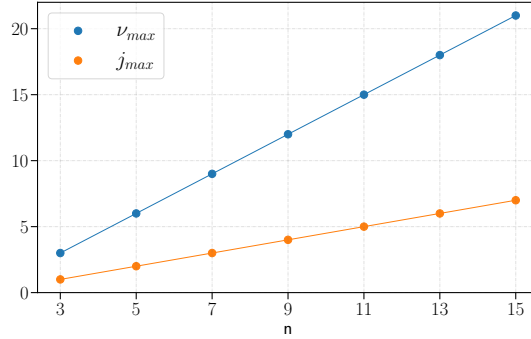

Figure 8: Relation between n and $\nu_{\max}/j_{\max}$ thanks to Equation (4.7).

6.3 Rotation Equivariance From a Numerical Point of View

As opposed to most of the state-of-the-art methods, B-CNNs do not rely on a particular discretization of the continuous $(S\text{-})O(2)$ group. The equivariance is automatically guaranteed by processing the input image thanks to an $(S\text{-})O(2)$ equivariant mathematical operation, which replaces the simple convolution in the direct space. It follows that B-CNNs directly provide theoretical guarantees regarding the equivariance to the continuous set of rotation angles $[0, 2\pi[$. Indeed, as the Bessel coefficients of the filter $\{\kappa_{\nu,j}\}$ are not computed but defined as the learnable parameters of the model, it does not involve any numerical error. Furthermore, Equation (5.6) is rotation invariant and this independently of the number of Bessel coefficients used (that is, independently of $k_{\max}$). However, one should mention that, from a numerical point of view, exact $(S\text{-})O(2)$ equivariance is rarely possible due to the

discrete nature of numerical images. Indeed, $\Psi(x, y)$ is only known on a finite Cartesian grid, and rotations of angles in $[0, 2\pi[\setminus \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$ are not well defined, and will result in numerical errors. The only source of errors in B-CNNs regarding the $(S-)O(2)$ equivariance lies in the discretization of $\Psi^{(x,y)}(x', y')$ on an $[2n \times 2n]$ Cartesian grid, which is involved by Equation (6.1). Therefore, numerical errors may be reduced by increasing n, that is, the size of the filters.

6.4 Adding a Multi-Scale Equivariance

Previous sections focus on achieving $SO(2)$ and $O(2)$ equivariance. However, for particular applications, patterns of interest may also vary in scale. To illustrate this, see for example biomedical applications where tumors may be of different sizes. Prior works (Xu et al., 2014; Li et al., 2019; Ghosh and Gupta, 2019) already showed that a multi-scale equivariance can be incorporated into CNNs, leading to better performances for such applications. The aim of this section is to present how these prior works can be easily transposed to the particular case of B-CNNs.

As the size of the filter in the direct space is determined by the discretization of $T_{\nu,j}(x, y)$, it is easy in B-CNNs to implement already-existing scaling invariance techniques. To do so, we only need to pre-compute multiple versions of $T_{\nu,j}(x, y)$ for different kernel sizes, and only keep the one with the highest response. More formally, the idea is to define multiple transformation matrices $T_{\nu,j}^n(x, y)$ that act on circular domains of different size n. Those matrices can be pre-computed at initialization. They can then be used to project the filters in the Fourier-Bessel transform domain to filters of different sizes in the direct space. One can then consider keeping only the most active feature maps. The process is summarized in Figure 9.

7 Experiments

This section presents the details of all the experiments performed to assess and to compare the equivariance obtained with B-CNNs with other state-of-the-art methods. The data sets used are presented, as well as the experimental setup. After that, quantitative results are presented for each data set.

7.1 Data sets

Three data sets are used to assess the performances in different practical situations:

- The MNIST (LeCun et al., 1998) data set is a classical baseline for image classification. This data set is made of 28×28 grayscale images of handwritten digits that belong therefore to one out of 10 different classes. More precisely, four variants of this data set are considered: (i) MNIST, (ii) MNIST-rot, (iii) MNIST-back and (iv) MNIST-rot-back[4]. In the *rot* variants, images are randomly rotated by an angle $\alpha \in [0, 2\pi[$. In the *back* variants, a patch from a black and white image was used as the background for the digit image. This adds useless information that can be disturbing for some architectures. All these MNIST data sets are perfectly balanced.

---

4. All these variants are generated from the initial MNIST data set, and can be found at `https://sites.google.com/a/lisa.iro.umontreal.ca/public_static_twiki/variations-on-the-mnist-digits`
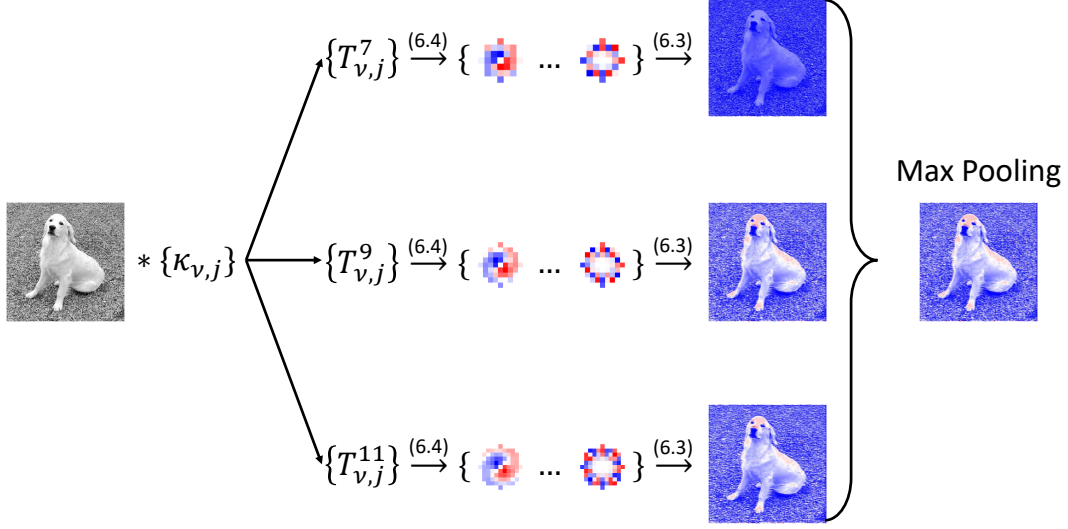
Figure 9: This figure presents how B-CNNs can handle multi-scale equivariance. In this case, a single filter represented by its Bessel coefficients $\{\kappa_{\nu,j}\}$ is projected in the direct space thanks to different transformation matrices. The filter is mapped to ν filters of size 7×7, 9×9 and 11×11. Max pooling is then used to only keep the most responding feature maps. Note that only the real parts of the projected filters are represented here, for convenience.

- The Galaxy10 DECals data set is a subset of the original Galaxy Zoo data set (Willett et al., 2013). This data set is initially made of 256×256 RGB images of galaxies that belong to one out of 10 roughly balanced classes, representing different possible morphologies according to experts. Images are resized to 128×128 in our work for computational resources purpose.

- The Malaria (Yu et al., 2020) data set is made of 64×64 RGB microscope images of blood films. Those images belong to two perfectly balanced classes highlighting the presence or not of the parasites responsible for Malaria.

An overview for all those three data sets is presented in Table 1, along with visual examples.

7.2 Experimental Setup

In order to perform this empirical study, (i) $E(2)$-equivariant CNNs from Weiler and Cesa (2019) ($E(2)$-CNNs), (ii) Harmonic Networks from Worrall et al. (2017) (HNets) as well as (iii) vanilla CNNs are considered along with our method (B-CNNs). This choice is motivated by the fact that $E(2)$-CNN and HNets constitute the state of the art for constraining CNNs with known symmetry groups. Each technique is tested in different setups (mainly, for different symmetry groups or different representations of the same group). The different setups for each method are described below:
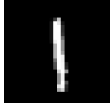
Data set	N	C	Task	Resolution	Examples
MNIST -rot -back -rot-back	$62,000$	10	Multi-class classification	$28 \times 28 \times 1$	
Galaxy10 DECals	$17,736$	10	Multi-class classification	$128 \times 128 \times 3$	
Malaria	$27,558$	2	Binary classification	$64 \times 64 \times 3$	

Table 1: Overview of the data sets. N is the total number of available data, and C is the number of classes.

- For $E(2)$-CNNs, we consider the discrete C_4 ($\{n\frac{\pi}{2}\}_{n=1}^4$ rotations) and C_8 ($\{n\frac{\pi}{4}\}_{n=1}^8$ rotations) symmetry groups using a regular representation, as well as the continuous one, $SO(2)$ (all the continuous rotations) and $O(2)$ (all the continuous rotations and the reflections along vertical and horizontal axes), using irreducible representations. Those setups are a subset of all the setups tested by the authors of $E(2)$-CNNs. More details about this and how $E(2)$-CNNs work can be found in the work of Weiler and Cesa (2019). Furthermore, the authors provide an implementation for $E(2)$-CNN that has been used in this work.

- For HNets, similarly to what the authors did in their work, two different setups to achieve $SO(2)$ invariance are tested using an approximation to the first and second order. In this work, we use again the implementation provided by the authors of $E(2)$-CNNs, who re-implement HNets in their own framework, for convenience.

- Regarding our B-CNNs, four setups are considered to achieve $SO(2)$ or $O(2)$, with or without scale invariance (denoted by the presence or not of "+" in our tables and figures), with the computation of $k_{\max}$ as described by Equation (4.7). Another setup for $SO(2)$ invariance with a stronger cutoff frequency, that corresponds to half

the initial $k_{\max}$, is also considered. This last setup is motivated by the empirical observation that it often leads to better performances.

- Finally, a vanilla CNN with the same architecture than for the other methods, as well as a ResNet-18 (He et al., 2016) are also trained for reference.

The architectures are inspired from the work of Weiler and Cesa (2019) and are presented in a generic fashion in Table 2. Note that the size of the filters is larger than conventional sizes in CNNs. The reason why it is preferable to increase the size of the filters in those cases is explained in Section 6.3. The same template architecture is used for all the methods (except for the ResNet-18 architecture that is kept unmodified) and data sets. Nonetheless, minor modifications are sometimes performed. Firstly, the number of filters in each convolutional layer should be adapted from one method to another, in order to keep the same number of trainable parameters. To do so, a parameter λ is introduced to manually scale the number of filters and guarantee the same number of trainable parameters for all the methods. To give an idea, λ is arbitrarily set to 1 for B-CNNs with the soft cutoff frequency policy, and the corresponding number of trainable parameters is close to $115,000$. Secondly, $E(2)$-CNNs require a particular operation called *invariant projection* before applying the dense layer. This specific operation is not performed for the other methods. Thirdly, each convolutional layer is followed by a batch-normalization and a *ReLU* activation function, except for B-CNNs. Indeed, we empirically observed that both the batch-normalization and the *ReLU* activation function generally decrease convergence for B-CNNs, while this is not the case for the other methods. Therefore, we use another type of batch-normalization layer that is introduced by Li et al. (2021) as well as *softsign* activation functions, which seems to perform better in our case. Note that we are still not able to really understand why classic batch-normalization and *ReLU* activation functions reduce the performances of B-CNNs. Finally, the padding and the final layer are adapted according to the considered data set, as they involve different tasks and image sizes.

As it is expected that constraining CNNs with symmetry groups becomes more useful when less data are available (as CNNs should no more learn the invariances by themselves), experiments are performed in (i) High, (ii) Intermediate and (iii) Low data settings for each data set, with different data augmentation strategies. The different data settings correspond to different sizes for the training sets. Attention is paid to keep the same percentage of samples of each target class, in order to avoid biases.

For the MNIST data sets, those settings correspond to the use of (i) 20%, (ii) 2% and (iii) 0.2% of the total number of available data for training. On top of this, three different data augmentation policies are tested. Firstly, models are trained on MNIST-rot(-back) using online data augmentation (by performing random rotation before being given as input). Secondly, models are still trained on MNIST-rot(-back) but without further data augmentation (the same image is always seen by the model in the same orientation). Thirdly, models are trained on MNIST(-back) while being tested on random rotated versions of the test images. Those setups allow us to see how the amount of data impact the performance of the models, and how much the models still rely on the training phase to achieve the desired invariances.

For the other data sets, the different data settings correspond to the use of (i) 80%, (ii) 8% and (iii) 0.8% of the total number of available training data, respectively. As for

Layer	# C	MNIST(-rot)(-back)	Galaxy10 DECals	Malaria
Conv layer 9×9	8λ	pad 4	pad 0	pad 4
Conv layer 7×7	16λ	pad 3	pad 0	pad 3
Av. pool. 2×2	-	pad 0	pad 0	pad 0
Conv layer 7×7	24λ	pad 3	pad 0	pad 3
Conv layer 7×7	24λ	pad 3	pad 0	pad 0
Av. pool. 2×2	-	pad 0	pad 0	pad 0
Conv layer 7×7	32λ	pad 3	pad 0	pad 0
Conv layer 7×7	40λ	pad 0	pad 0	pad 0
(*Inv. projection*)	-	-	-	-
Global av. pool.	-	-	-	-
Dense layer	$\rightarrow$	10, *softmax*	10, *softmax*	2, *softmax*

Table 2: Generic architecture used for the different data sets. *Conv layer* can either be vanilla-conv, B-conv, $E(2)$-conv or HNet-conv. After each *Conv layer*, a batch-normalization as well as an activation function is applied (*softsign* for B-conv, *ReLU* for others). The *Invariant projection* is only required in $E(2)$-CNNs. As the number of parameters in a filter may differ between the different methods, a parameter λ is introduced. This parameter is fixed for each methods in order to tweak the number of filters (# C) so that the total numbers of trainable parameters are as close as possible to each others.

the MNIST data sets, models are again trained with and without using data augmentation. However, in this case, the data augmentation also performs random planar reflections (not pertinent for MNIST). Also note that only two data augmentation policies are possible, because non-rotated version of images for Galaxy10 DECals and Malaria is meaningless (as opposed to MNIST where digits have a well-defined orientation, a priori).

For the High and Intermediate data setting experiments, models are trained using the Adam optimizer through 50 epochs. A warm-up cosine decay scheduler that progressively increase the learning rate from 0 to 0.001 during the first 10 epochs before slowly decreasing it to 0 following a cosine function during the remaining epochs is used. For the low data setting experiments, 150 epochs are performed, with the warm-up phase during the first 30 epochs. Each experiment is performed on 3 independent runs.

7.3 Results on MNIST(-rot)

Table 3 presents the results obtained on the MNIST(-rot) data sets. For the sake of completeness, Figure 10 also presents all the corresponding training curves.

From a general point of view, one can observe that a proper use of $E(2)$-CNNs, HNets and B-CNNs can lead to better performances than vanilla CNNs, even if the number of parameters is much smaller in the case of equivariant models ($\pm 115,000$ parameters against $\pm 11,000,000$ for ResNet-18). Vanilla CNNs techniques are only able to compete with equivariant models in high data settings, and when performing data augmentation (first

Method	Group	MNIST-rot — With data aug.			MNIST-rot — Without data aug.			MNIST — No rotation during training			Reference
		High	Inter.	Low	High	Inter.	Low	High	Inter.	Low	
Vanilla CNN	{e}	98.16±0.03	93.95±0.20	41.64±17.87	76.96±0.84	—	22.27±3.52	42.82±0.23	36.15±1.42	22.84±2.76	–
ResNet-18	{e}	98.42±0.10	92.42±0.37	73.18±1.01	95.51±0.07	59.52±9.66	26.43±2.20	41.05±0.21	30.78±2.01	20.34±1.29	He et al. (2016)
E(2)-CNN / regular	C_4	**99.03±0.03**	**96.55±0.23**	**86.91±0.40**	93.91±0.33	—	69.41±2.20	68.06±1.94	65.97±0.88	55.45±0.22	
E(2)-CNN / regular	C_8	**99.11±0.05**	**96.89±0.26**	**85.24±1.21**	94.75±0.32	—	73.66±1.15	68.81±0.44	67.08±0.76	58.73±1.06	Weiler and Cesa (2019)
E(2)-CNN / irreps ≤ 1	$SO(2)$	98.77±0.01	95.57±0.09	79.50±0.81	95.18±0.28	—	**79.94±1.63**	**97.11±0.20**	**91.45±1.45**	**69.24±4.93**	
E(2)-CNN / irreps ≤ 1	$O(2)$	96.85±0.04	89.66±0.60	67.79±1.23	88.80±0.45	—	68.45±2.39	92.26±0.07	82.73±0.40	56.57±3.87	
E(2)-CNN / irreps ≤ 3	$SO(2)$	98.52±0.05	93.30±0.33	77.57±1.50	90.95±0.43	—	74.02±0.72	95.75±0.36	86.90±0.83	64.32±5.48	
E(2)-CNN / irreps ≤ 3	$O(2)$	96.57±0.14	89.32±0.30	65.38±0.13	85.86±0.34	—	65.24±2.24	91.49±0.63	79.75±2.01	54.67±2.84	
HNets / 1st order	$SO(2)$	98.82±0.03	95.86±0.11	81.08±1.16	**98.73±0.04**	**95.77±0.30**	**80.99±0.49**	**96.91±0.28**	—	**71.20±6.88**	Worrall et al. (2017)
HNets / 2nd order	$SO(2)$	98.78±0.00	95.33±0.20	81.65±0.80	**98.72±0.06**	**94.25±0.37**	76.94±1.29	96.90±0.16	—	64.20±9.09	
B-CNNs / k_{max}	$O(2)$	98.92±0.03	95.98±0.23	80.67±1.71	95.06±0.27	—	76.40±1.32	95.22±0.86	90.61±1.27	65.76±1.82	Our work
B-CNNs / k_{max}	$SO(2)+$	97.76±0.07	93.45±0.06	69.73±5.20	91.92±0.52	—	68.83±2.00	92.07±0.23	86.40±1.56	68.60±3.95	
B-CNNs / k_{max}	$O(2)+$	98.95±0.05	95.82±0.09	80.30±2.94	95.23±0.31	—	78.00±1.51	95.85±0.73	**92.02±1.24**	67.81±5.51	
B-CNNs / k_{max}	$SO(2)$	97.87±0.01	93.14±0.37	72.75±1.22	92.01±0.19	—	72.48±2.08	92.15±0.72	80.90±8.40	55.02±17.45	
B-CNNs / $k_{max}/2$	$SO(2)$	**99.00±0.03**	**96.56±0.35**	**82.70±2.08**	**96.33±0.27**	—	**82.05±0.26**	**97.34±0.15**	**94.41±0.52**	**79.67±2.57**	

Table 3: Classification accuracy obtained for the different methods on the MNIST-rot and MNIST data sets, with and without data augmentation. The *High*, *Inter.* and *Low* data regimes correspond to 12,000, 1,200 and 120 images for the training set (same percentage of samples of each target class), respectively. For each column, bold is used to highlight the top-3 performing models. Accuracy and standard deviation are assessed using 3 independent runs. The corresponding training curves are presented in Figure 10.
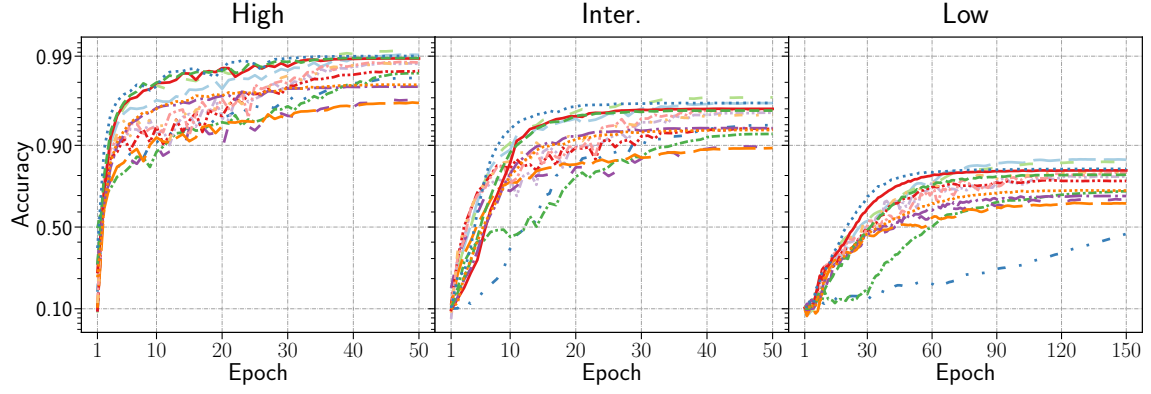
(a) MNIST-rot with augmentation
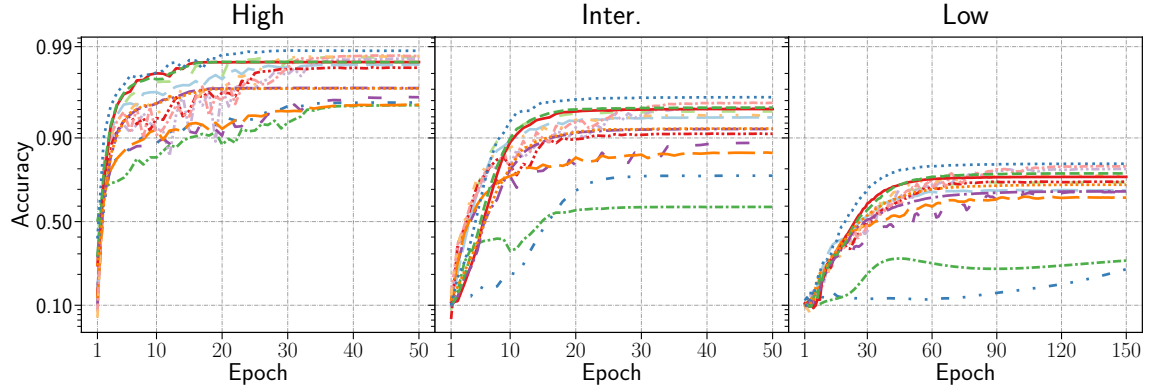
(b) MNIST-rot without augmentation
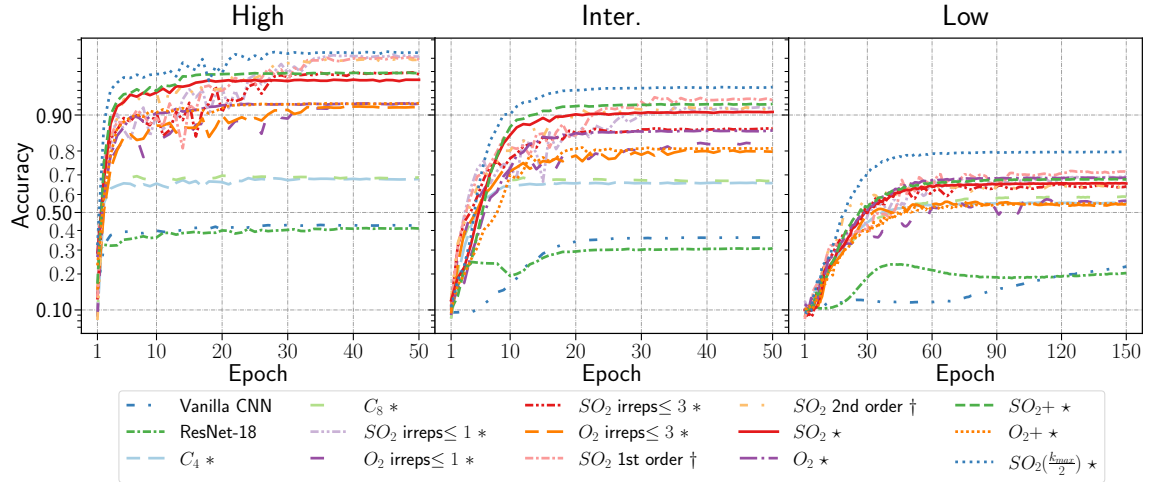
(c) MNIST

Figure 10: Learning curves obtained for the different methods on the MNIST-rot and MNIST data sets. Those learning curves are averaged over 3 independent runs. The legend is the same for all the graphs. The symbols $*$, $\dagger$ and $\star$ refer to the use of $E(2)$-CNNs, HNets and B-CNNs, respectively.

27

column). This clearly highlights the fact that vanilla CNNs are sensitive to quality and the amount of data in order to learn the invariances. Furthermore, even in the most favorable situation for vanilla CNNs, convergence is much slower than for equivariant models.

Next, by taking a closer look at the equivariant models, it appears that the $E(2)$-CNNs that use the straightforward discrete groups C_4 and C_8 perform quite well, and are even the best performing models when used along with data augmentation (first 3 columns). However, performances fall a little on the MNIST-rot data set without data augmentation (middle 3 columns), and becomes really bad compared to the $(S-)O(2)$ equivariant models when trained on the MNIST data set, when they cannot see rotated versions of the digits (last 3 columns). Even if those models are better than vanilla CNNs, it also appears that they still rely on training to learn really continuous rotation invariance, which was something expected. It is also interesting to mention that using a symmetry group that is not appropriate may be worse than not using any symmetry group at all. For example, in high data setting with data augmentation, vanilla CNNs perform better than $O(2)$-based models.

Finally, almost all the $(S-)O(2)$ equivariant models seem to achieve very similar performances on MNIST-rot, with and without data augmentation. Nonetheless, one can observe that the $SO(2)$ B-CNNs with the strong cutoff policy is always in the top-3 performing models, and achieve significantly better results than all the other models when only trained on MNIST (last 3 columns). This highlights the fact that the $SO(2)$ invariance achieved by design in the B-CNNs is stronger than the one achieved by other models, allowing generalization to rotated versions of digits, even if none of those are observed during training.

7.4 Results on MNIST(-rot)-back

Table 4 presents the results obtained on the MNIST(-rot)-back data sets, which are variants of the MNIST data set with randomly rotated digits and black and white images as background. For the sake of completeness, Figure 11 also presents all the corresponding training curves.

For vanilla CNNs, the conclusions are the same as for MNIST(-rot). Performances quickly drop when using a smaller amount of data, or when data augmentation is not performed properly.

Now, by opposition to the observation for the MNIST(-rot) data set, it appears that B-CNNs are from a general point of view significantly better than other equivariant models, in each setup. For the high data setting on MNIST-back (without seeing rotated images during training), Figure 11 clearly reveals several groups of plateau corresponding to vanilla CNNs, discrete $E(2)$-CNNs, $O(2)$ $E(2)$-CNNs, $SO(2)$ $E(2)$-CNNs and HNets, and finally all the B-CNNs models with the $SO(2)$ ones being in top of them. Again, in addition to the better performances, one should also highlight a faster convergence for the B-CNNs.

7.5 Results on Galaxy10 DECals

Table 5 presents the results obtained on the Galaxy10 DECals data set. For the sake of completeness, Figure 12 also presents all the corresponding training curves.

| Method	Group	MNIST-rot-back With data aug.			MNIST-rot-back Without data aug.			MNIST-back No rotation during training			Reference
		High	Inter.	Low	High	Inter.	Low	High	Inter.	Low	
Vanilla CNN	{e}	88.43 ± 0.24	70.49 ± 0.35	23.31 ± 2.71	76.33 ± 0.65	32.39 ± 0.85	14.77 ± 0.57	34.59 ± 0.24	25.73 ± 0.45	14.58 ± 1.53	-
ResNet-18	{e}	87.34 ± 0.09	60.59 ± 0.77	30.32 ± 0.48	68.51 ± 0.31	33.80 ± 0.64	14.36 ± 0.37	32.56 ± 0.07	26.03 ± 0.42	13.65 ± 1.30	He et al. (2016)
E(2)-CNN / regular	C_4	89.52 ± 0.32	75.62 ± 0.27	31.33 ± 3.03	84.46 ± 0.52	51.61 ± 1.36	24.17 ± 1.07	45.37 ± 0.65	38.46 ± 0.99	21.03 ± 1.42	
	C_8	**89.93 ± 0.23**	**77.87 ± 1.27**	**34.68 ± 5.13**	**86.59 ± 0.42**	56.00 ± 5.02	23.08 ± 1.16	45.80 ± 1.78	39.47 ± 1.42	24.35 ± 2.39	
E(2)-CNN / irreps ≤ 1	$SO(2)$	81.97 ± 0.07	60.06 ± 0.81	26.43 ± 2.33	79.15 ± 1.41	57.02 ± 0.83	27.04 ± 1.49	56.17 ± 2.40	45.80 ± 1.42	25.31 ± 2.18	Weiler and Cesa (2019)
	$O(2)$	74.71 ± 0.48	48.66 ± 2.25	23.78 ± 1.62	69.75 ± 0.31	44.30 ± 1.21	20.88 ± 0.78	42.18 ± 1.00	39.47 ± 1.00	21.02 ± 0.48	
E(2)-CNN / irreps ≤ 3	$SO(2)$	82.17 ± 0.86	61.16 ± 1.08	32.21 ± 0.85	79.14 ± 1.31	52.18 ± 1.04	25.44 ± 2.25	52.49 ± 2.50	42.18 ± 0.78	25.02 ± 2.59	
	$O(2)$	76.99 ± 0.43	54.38 ± 1.92	25.55 ± 1.85	70.74 ± 1.29	43.20 ± 0.95	20.99 ± 1.25	38.69 ± 0.83	25.31 ± 0.59	20.39 ± 1.46	
HNets / 1st order	$SO(2)$	82.24 ± 0.29	60.75 ± 0.88	26.72 ± 1.63	81.93 ± 0.65	57.36 ± 1.13	24.24 ± 1.00	53.40 ± 3.22	42.18 ± 1.00	21.78 ± 2.25	Worrall et al (2017)
HNets / 2nd order	$SO(2)$	83.11 ± 0.37	62.52 ± 0.26	29.53 ± 3.26	81.91 ± 0.48	54.68 ± 1.11	26.17 ± 1.24	52.81 ± 2.11	42.49 ± 2.50	23.52 ± 0.75	
B-CNNs / k_{max}	$SO(2)$	**90.74 ± 0.30**	**80.44 ± 0.37**	**37.84 ± 5.11**	**89.28 ± 0.38**	77.45 ± 0.75	31.69 ± 2.23	**84.43 ± 0.39**	**74.72 ± 0.24**	**35.82 ± 4.86**	Our work
	$O(2)$	88.22 ± 0.17	76.14 ± 0.64	32.63 ± 5.99	86.40 ± 0.42	72.79 ± 0.99	25.99 ± 3.21	79.71 ± 0.23	70.69 ± 2.07	**39.25 ± 2.97**	
	$SO(2)+$	**90.82 ± 0.18**	**80.29 ± 1.21**	**36.54 ± 2.59**	**88.93 ± 0.13**	**77.51 ± 0.57**	**35.17 ± 2.35**	**84.59 ± 0.60**	**74.34 ± 0.65**	34.10 ± 2.13	
	$O(2)+$	88.11 ± 0.19	74.49 ± 1.15	29.94 ± 2.22	86.19 ± 0.34	72.75 ± 0.27	**31.95 ± 1.86**	79.37 ± 0.54	68.60 ± 1.70	33.95 ± 3.73	
B-CNNs / $k_{max}/2$	$SO(2)$	87.35 ± 0.16	76.48 ± 0.20	30.72 ± 2.06	86.36 ± 0.05	**75.35 ± 0.18**	**36.39 ± 1.10**	**84.05 ± 0.14**	**75.33 ± 0.77**	25.65 ± 2.98	

Table 4: Classification accuracy obtained for the different methods on the MNIST-rot-back and MNIST-back data sets, with and without using data augmentation. The *High*, *Inter.* and *Low* data regimes correspond to $12,000$, $1,200$ and 120 images for the training set (same percentage of samples of each target class), respectively. For each column, bold is used to highlight the top-3 performing models. Accuracy and standard deviation are assessed using 3 independent runs. The corresponding training curves are presented in Figure 11.
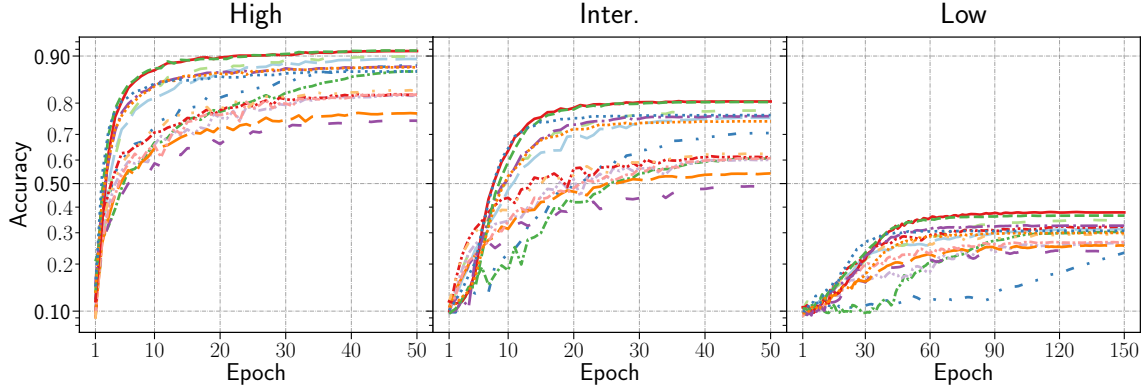
(a) MNIST-rot-back with augmentation
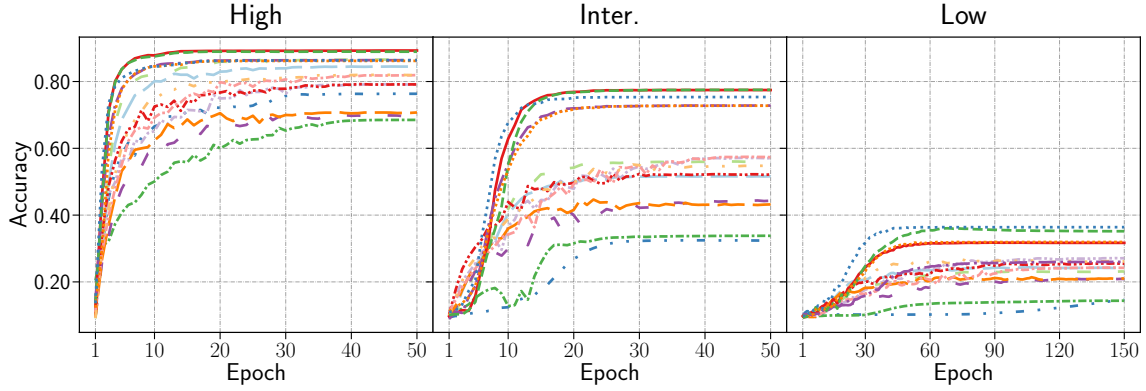
(b) MNIST-rot-back without augmentation
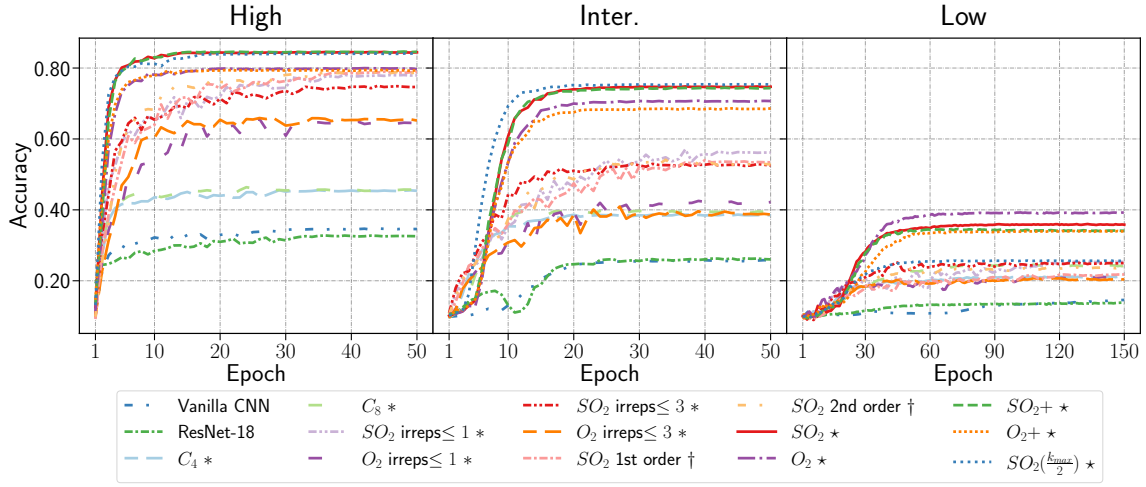
(c) MNIST-back

Figure 11: Learning curves obtained for the different methods on the MNIST-back and MNIST-rot-back data sets. Those learning curves are averaged over 3 independent runs. The legend is the same for all the graphs. The symbols $*$, $\dagger$ and $\star$ refer to the use of $E(2)$-CNNs, HNets and B-CNNs, respectively.

Method	Group	Galaxy10 DECals With data aug.			Without data aug.			Reference
		High	Inter.	Low	High	Inter.	Low	
Vanilla CNN	{e}	83.15 ± 0.38	72.32 ± 0.45	46.38 ± 4.35	77.28 ± 0.52	66.59 ± 0.20	40.56 ± 1.00	-
ResNet-18	{e}	85.66 ± 0.67	72.81 ± 0.60	43.20 ± 0.48	71.06 ± 0.31	38.75 ± 0.91	33.21 ± 0.58	He et al. (2016)
E(2)-CNN / regular	C_4	**87.08 ± 0.50**	**80.36 ± 0.57**	**53.69 ± 0.34**	83.92 ± 0.19	76.35 ± 0.41	51.29 ± 1.97	
	C_8	**87.70 ± 0.43**	**80.90 ± 0.33**	**57.33 ± 0.77**	**84.60 ± 0.34**	**78.76 ± 0.38**	**57.92 ± 0.80**	
E(2)-CNN / irreps ≤ 1	$SO(2)$	85.28 ± 0.42	75.42 ± 0.41	48.33 ± 2.62	82.98 ± 0.24	70.79 ± 0.43	40.64 ± 0.83	Weiler and Cesa (2019)
	$O(2)$	83.80 ± 0.67	74.12 ± 0.80	**53.91 ± 0.48**	83.02 ± 0.73	69.76 ± 0.83	45.14 ± 2.96	
E(2)-CNN / irreps ≤ 3	$SO(2)$	84.99 ± 0.18	74.23 ± 0.17	45.85 ± 0.46	81.97 ± 0.37	67.64 ± 0.27	26.18 ± 2.29	
	$O(2)$	84.10 ± 0.41	73.42 ± 0.69	50.55 ± 2.13	81.74 ± 0.29	65.25 ± 2.36	25.07 ± 1.75	
HNets / 1st order	$SO(2)$	**85.77 ± 0.81**	75.37 ± 0.72	48.25 ± 0.63	83.55 ± 0.26	72.00 ± 0.27	42.98 ± 0.92	Worrall et al. (2017)
HNets / 2nd order	$SO(2)$	85.36 ± 0.01	73.70 ± 0.49	48.65 ± 0.72	82.23 ± 0.55	70.25 ± 0.75	32.48 ± 1.87	
B-CNNs / $k_{\max}$	$SO(2)$	85.08 ± 0.38	76.95 ± 0.68	51.33 ± 2.46	83.99 ± 0.40	76.28 ± 0.50	**51.99 ± 3.17**	Our work
	$O(2)$	84.65 ± 0.55	77.33 ± 0.56	50.60 ± 1.17	**84.82 ± 0.30**	**76.64 ± 0.98**	49.31 ± 1.46	
	$SO(2)+$	85.11 ± 0.19	77.53 ± 0.15	47.75 ± 2.30	83.80 ± 0.19	75.33 ± 1.02	51.36 ± 3.41	
	$O(2)+$	84.70 ± 0.11	76.95 ± 1.34	51.12 ± 3.78	84.40 ± 0.12	76.60 ± 0.41	49.04 ± 3.14	
B-CNNs / $k_{\max}/2$	$SO(2)$	85.25 ± 0.48	**78.25 ± 0.32**	51.92 ± 1.84	84.67 ± 0.78	**77.87 ± 0.20**	**54.84 ± 3.21**	

Table 5: Classification accuracy obtained for the different methods on the Galaxy10 DECals data set, with and without using data augmentation. The *High*, *Inter.* and *Low* data regimes correspond to 14,188, 1,418 and 141 images for the training set (same percentage of samples of each target class), respectively. For each column, bold is used to highlight the top-3 performing models. Accuracy and standard deviation are assessed using 3 independent runs. The corresponding training curves are presented in Figure 12.
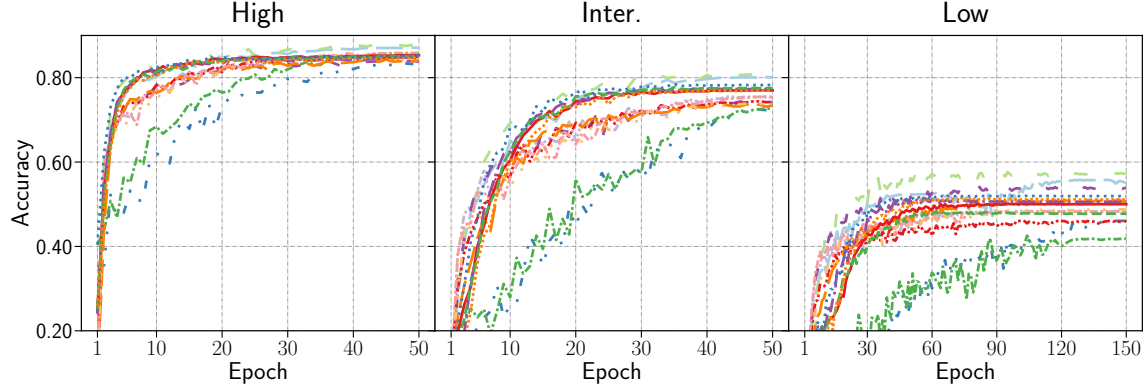
(a) Galaxy10 DECals with augmentation
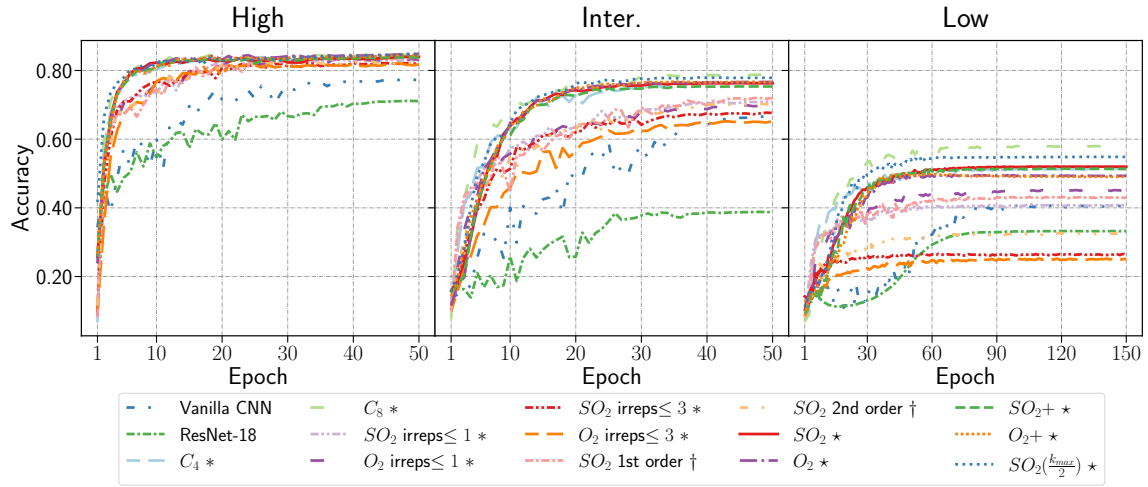
(b) Galaxy10 DECals without augmentation

Figure 12: Learning curves obtained for the different methods on the Galaxy10 DECals data set. Those learning curves are averaged over 3 independent runs. The legend is the same for all the graphs. The symbols $*$, $\dagger$ and $\star$ refer to the use of $E(2)$-CNNs, HNets and B-CNNs, respectively.

From those results, one can see again that using the symmetry group C_4 and C_8 already allow $E(2)$-CNNs to achieve very good results. For Galaxy10 DECals, this observation stands for each setup, even without using data augmentation.

$SO(2)$-based B-CNNs with the strong cutoff policy is again one of the best performing model when used without data augmentation.

It is interesting to see that using the $O(2)$ group does not always lead to better performances compared to results obtained using $SO(2)$, despite the fact planar reflections are meaningful for this application.

7.6 Results on Malaria

Table 6 presents the results obtained on the Malaria data set. For the sake of completeness, Figure 13 also presents all the corresponding training curves. Interestingly, for this data set, B-CNNs seem to perform slightly worse than other methods. However, one can see that the ResNet-18 is among the best performing models in high data setting with data augmentation and remains competitive in other settings. This highlights the fact that $(S-)O(2)$ invariance may be less useful than for the other tested applications. Still, performances are most of the time very close to each others.

8 Discussion

This section first provides a global discussion regarding the different experiments and results presented in the previous section. Then, we discuss the choice of using models with automatic symmetry discovery, or models as B-CNNs based on applying (strong) constraints to guarantee user-defined symmetry.

8.1 Global Discussion of the Results

From the experiments and the preliminary discussions in the previous section, several insights can be retrieved.

Equivariant models vs. vanilla CNNs In the particular case where many data are available, vanilla CNNs do a very decent job by being only marginally below the top-accuracy. Thanks to data augmentation, those models seem to be able to learn meaningful invariances. However, by taking a look at the training curves, it appears clearly that convergence is much slower. This is easily explained by the fact that vanilla CNNs should learn the invariances, while it is not the case for equivariant models. Therefore, computation time and energy may be saved by using instead an equivariant model with training through much less epochs. Furthermore, this drawback is emphasized when vanilla CNNs should work with less data and/or without data augmentation, up to leading to very poor performances in those cases.

Discrete vs. continuous groups As already spotted by Weiler and Cesa (2019), using discrete groups already largely improve performances, and even sometimes constitute the best performing models. Nonetheless, experiments also highlight that it does not guarantee equivariance and often still requires a larger amount of data as well as data augmentation.

| | | Malaria | | | | | | |
| | | With data aug. | | | Without data aug. | | | |
Method	Group	High	Inter.	Low	High	Inter.	Low	Reference
Vanilla CNN	{e}	97.27 ± 0.11	96.00 ± 0.13	92.50 ± 1.27	95.89 ± 0.15	94.20 ± 0.27	70.23 ± 2.86	–
ResNet-18	{e}	**97.30 ± 0.11**	95.81 ± 0.17	91.29 ± 1.65	96.10 ± 0.14	93.47 ± 0.37	63.03 ± 1.12	He et al. (2016)
E(2)-CNN / regular	C_4	97.18 ± 0.16	95.98 ± 0.19	91.37 ± 0.26	96.45 ± 0.17	94.94 ± 0.26	80.37 ± 2.46	Weiler and Cesa (2019)
E(2)-CNN / regular	C_8^t	97.12 ± 0.31	**96.10 ± 0.12**	92.62 ± 0.57	96.48 ± 0.10	94.90 ± 0.34	88.41 ± 0.69	
E(2)-CNN / regular	$SO(2)$	95.98 ± 0.19	95.80 ± 0.52	**92.96 ± 1.27**	**96.80 ± 0.12**	**95.90 ± 0.23**	**94.31 ± 0.22**	
E(2)-CNN / irreps ≤ 1	$O(2)$	97.00 ± 0.44	**96.18 ± 0.27**	**93.16 ± 1.29**	96.54 ± 0.22	**95.79 ± 0.17**	**93.88 ± 0.62**	
E(2)-CNN / irreps ≤ 1	$SO(2)$	**97.33 ± 0.21**	95.54 ± 0.67	91.89 ± 1.82	96.66 ± 0.24	95.64 ± 0.15	91.09 ± 2.46	
E(2)-CNN / irreps ≤ 3	$O(2)$	96.89 ± 0.47	95.32 ± 0.89	92.68 ± 1.54	96.64 ± 0.19	95.53 ± 0.23	91.24 ± 1.95	
HNets / 1st order	$SO(2)$	**97.43 ± 0.17**	**96.24 ± 0.32**	**93.58 ± 1.15**	**96.88 ± 0.24**	**96.15 ± 0.13**	**94.31 ± 0.34**	Worrall et al. (2017)
HNets / 2nd order	$SO(2)$	97.15 ± 0.32	95.71 ± 0.61	92.06 ± 2.08	**96.92 ± 0.13**	95.80 ± 0.06	90.49 ± 1.70	
B-CNNs / k_{max}	$O(2)$	96.76 ± 0.27	95.53 ± 0.12	92.33 ± 1.71	96.29 ± 0.23	95.21 ± 0.12	82.35 ± 7.14	Our work
B-CNNs / k_{max}	$SO(2)$+	96.89 ± 0.11	95.50 ± 0.18	87.99 ± 7.37	96.35 ± 0.08	95.44 ± 0.25	90.62 ± 4.30	
B-CNNs / k_{max}	$O(2)$+	96.81 ± 0.08	95.36 ± 0.12	92.33 ± 0.76	96.18 ± 0.21	95.04 ± 0.06	89.34 ± 4.34	
B-CNNs / $k_{max}/2$	$SO(2)$	96.97 ± 0.14	95.42 ± 0.10	92.26 ± 0.92	96.61 ± 0.12	95.37 ± 0.30	91.31 ± 2.00	

Table 6: Classification accuracy obtained for the different methods on the Malaria data set, with and without using data augmentation. The *High*, *Inter.* and *Low* data regimes correspond to 22, 046, 2, 204 and 220 images for the training set (same percentage of samples of each target class), respectively. For each column, bold is used to highlight the top-3 performing models. Accuracy and standard deviation are assessed using 3 independent runs. The corresponding training curves are presented in Figure 13.
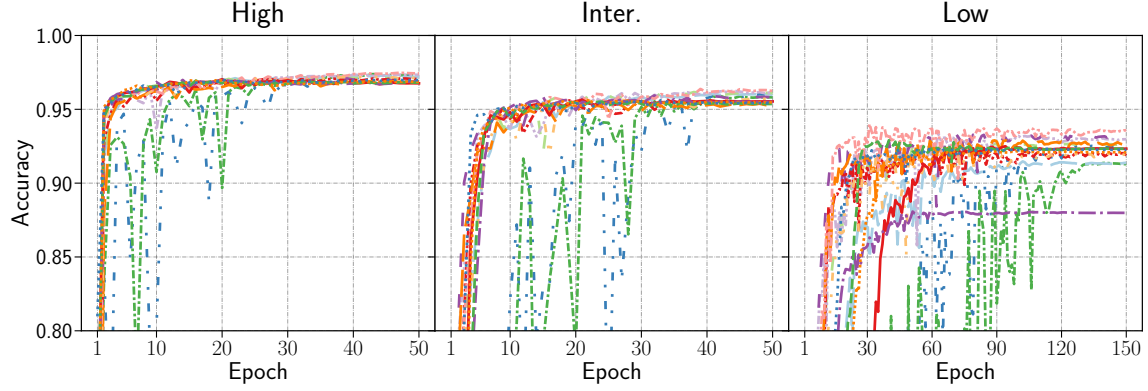
(a) Malaria with augmentation
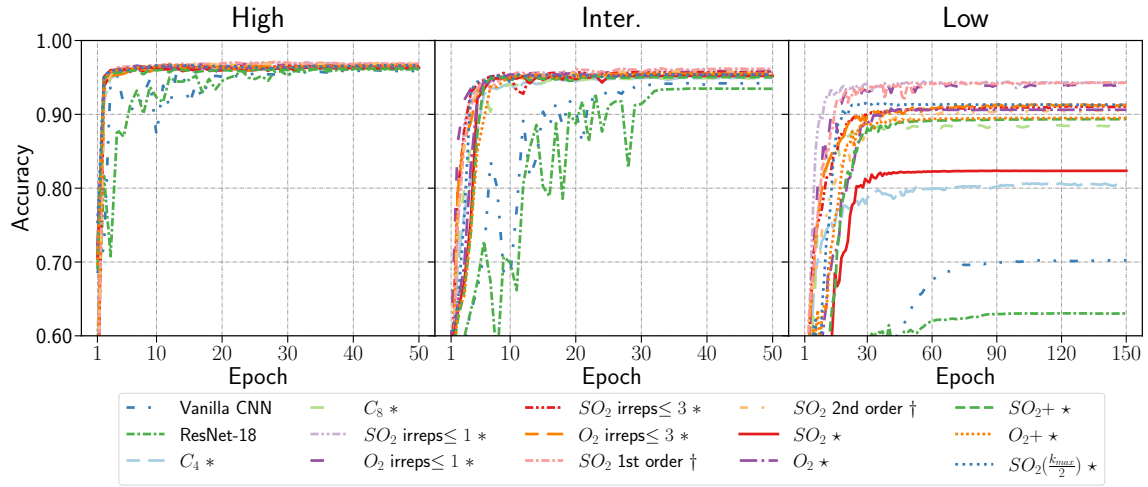
(b) Malaria without augmentation

Figure 13: Learning curves obtained for the different methods on the Malaria data set. Those learning curves are averaged over 3 independent runs. The legend is the same for all the graphs. The symbols $*$, $\dagger$ and $\star$ refer to the use of $E(2)$-CNNs, HNets and B-CNNs, respectively.

B-CNNs vs. other equivariant models In our experiments, B-CNNs are most of the time at least able to achieve state-of-the-art performances. In low data settings, they are often the best performing models. In particular, B-CNNs with strong cutoff policies seem very efficient. They achieve top-1 accuracy in 8 setups and top-3 accuracy in 17 setups, among a total of 30 different setups (all the columns for all the data sets). Now, by considering all the B-CNNs model at the same time, they achieve together top-1 accuracy in 16 setups and top-3 accuracy in 22 setups. For comparison, it is better than $E(2)$-CNNs and HNets, which achieve top-1 accuracy in 9 and 5 setups, and top-3 accuracy in 22 and 13 setups, respectively. Also, from the experiments on MNIST and MNIST-back (no rotation during training), one can see that the invariance achieved by design in B-CNNs is better than for other methods as performances for example for the strong cutoff policy are significantly above. Finally, we can observe that B-CNNs are able to achieve good performances even without data augmentation, which is not/less the case for other methods. As data augmentation increases the computational cost of training (because of the increase of the number of training data and/or the number of training epochs), B-CNNs may therefore be a more favorable approach.

8.2 Automatic Symmetry Discovery vs. Constraints-based Models

This work focuses on constraint-based models that assume that users know *a priori* the appropriate symmetry group(s) for the application at hand. However, it can sometimes be hard to obtain this prior knowledge as it requires good understanding of the data/application. Methods like the one proposed by Dehmamy et al. (2021) (L-CNNs) therefore attempt to infer the invariance(s) that need to be enforced automatically during training. Here, we advocate that the constraint-based approach remains relevant and often competitive.

Firstly, B-CNNs and other constraint-based methods can be adapted to handle symmetries in a data-driven fashion. For example, one can consider a method similar to the one in Section 6.4 to let the model choose meaningful symmetries. By designing the architecture with multiple networks in parallel that provide different invariances (one could simultaneously consider $SO(2)$, $O(2)$, $SO(2)+$ and $O(2)+$), the model can benefit from multiple views of the problem and use the features that are the most relevant. In a data-driven fashion, the relevant part(s) of the network will be retained so as to enforce appropriate invariance(s). This approach does not rely on the hypothetical ability of vanilla CNNs to learn specific types of invariance, but rather builds on models that are designed for that.

Secondly, B-CNNs and other constraint-based methods have the advantage to guarantee specific invariances. Instead of using data augmentation and relying on a proper learning of the invariances, mathematically sound mechanisms are used, such as Bessel coefficients for B-CNNs. However, these mechanisms can only deal with an invariance that can be described with reasonable mathematical complexity. Yet, handling symmetries in a data-driven fashion (discovering useful symmetries during training, with vanilla CNNs or other more adapted methods like L-CNNs) is not a one-fits-all solution and some invariances may be impossible to learn without additional mechanisms.

Thirdly, the way constraints are enforced in B-CNNs allows them to exhibit an invariance that can even not be present in the training data set. Hence, as shown in the above experiments, B-CNNs do not rely on data augmentation that increases the computational

cost[5], nor do they require to see training data with different orientations for the rotation invariance. This is a consequence of the mathematical soundness of our approach.

To conclude, automatic symmetry discovery and constraints-based models are two paradigms that should be used in different situations. While automatic symmetry discovery is useful when no prior knowledge is available and the invariance may be complex to describe mathematically, it relies on a appropriate learning of the invariances that may fail. On the other side, constraints-based models like our work require a prior knowledge of the invariances involved in the applications, but can provide strong guarantees.

9 Conclusion and Future Work

This work provides a comprehensive explanation of B-CNNs, including their mathematical foundations and key findings. Improvements are presented and compared to the initial work of Delchevalerie et al. (2021), including making B-CNNs also equivariant to reflections and multi-scale. Furthermore, the previous troublesome meta-parameters m_{max} and j_{max} that were hard to fine-tune have been replaced with a single meta-parameter k_{max} for which an optimal choice can be computed using the Nyquist frequency.

An extensive empirical study has been conducted to assess the performance of B-CNNs compared to already existing techniques. One can conclude that B-CNNs have, most of the time, better performances than the other state-of-the-art methods, and achieve in the worst cases roughly the same performances. In low data settings, they actually outperform other models most of the time. This is mainly due to the B-CNNs ability to maintain robust invariances without resorting to data augmentation techniques, which is often not the case for other models. Finally, B-CNNs do not involve particular, more exotic (such as complex-valued feature maps), representations for feature maps and are therefore highly compatible with already existing deep learning techniques and frameworks.

Regarding future work, it could be interesting to tailor B-CNNs for segmentation tasks, given their relevance in fields such as biomedical and satellite imaging. Such domains benefit greatly from rotation and reflection equivariant models, making B-CNNs a promising candidate for these tasks. Finally, a major actual concern in deep learning is the robustness regarding adversarial attacks or, more generally, small perturbations in the image. It could be interesting to evaluate if the use of Bessel coefficients and the $(S-)O(2)$ equivariant constraint make B-CNNs more robust to those specific perturbations or not.

---

5. Data augmentation is considered as costly because it leads to (i) an increase of the number of training data, and/or (ii) an increase of the required number of training epochs.

Appendix A.

In this Appendix we prove that the Bessel basis described in Section 4.1 can be used as an orthonormal basis by carefully choosing $k_{\nu,j}$.

Theorem 1 *Let D^2 be a circular domain of radius R in $\mathbb{R}^2$. Let $J_\nu(x)$ be the Bessel function of the first kind of order ν, and let $k_{\nu,j}$ be defined such that $J'_\nu(k_{\nu,j}R) = 0$, $\forall \nu, j \in \mathbb{N}$. Then*

$$\left\{ N_{\nu,j} J_\nu(k_{\nu,j}\rho) e^{i\nu\theta} \right\}, \ \text{where } N_{\nu,j} = 1/\sqrt{2\pi \int_0^R \rho J_\nu^2(k_{\nu,j}\rho) \, d\rho}$$

is an orthonormal basis well-defined to express any squared-integrable functions f such that $f : D^2 \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$.

Proof To prove this, we will use the fact that

$$\int_0^{2\pi} e^{i\theta(\nu'-\nu)} d\theta = \int_0^{2\pi} \cos\left(\theta\left(\nu'-\nu\right)\right) + i \int_0^{2\pi} \sin\left(\theta\left(\nu'-\nu\right)\right)$$
$$= 2\pi\delta_{\nu,\nu'}, \tag{A.1}$$

since $\nu' - \nu$ is always an integer in our use of Bessel functions. We use also Lommel's integrals, which are in our particular case

$$\int_0^R \rho J_\nu(k_{\nu,j}\rho) J_\nu(k_{\nu,j'}\rho) \, d\rho =$$
$$\begin{cases} \frac{1}{k_{\nu,j'}^2 - k_{\nu,j}^2} \left[\rho\left(k_{\nu,j} J'_\nu(k_{\nu,j}\rho) J_\nu(k_{\nu,j'}\rho) - k_{\nu,j'} J'_\nu(k_{\nu,j'}\rho) J_\nu(k_{\nu,j}\rho)\right) \right]_0^R & \text{if } k_{\nu,j} \neq k_{\nu,j'}, \\ \left[\frac{\rho^2}{2} \left[J'^2_\nu(k_{\nu,j}\rho) + \left(1 - \frac{\nu^2}{k_{\nu,j}^2\rho^2}\right) J_\nu(k_{\nu,j}\rho) \right] \right]_0^R & \text{otherwise.} \end{cases}$$
$$\tag{A.2}$$

By taking into account that $J'_\nu(k_{\nu,j}R) = 0$, Lommel's integrals lead to

$$\int_0^R \rho J_\nu(k_{\nu,j}\rho) J_\nu(k_{\nu,j'}\rho) \, d\rho = \begin{cases} 0 \text{ if } k_{\nu,j} \neq k_{\nu,j'} \\ \left(\frac{R^2}{2} - \frac{\nu^2}{2k_{\nu,j}^2} \right) J_\nu(k_{\nu,j}R) \text{ otherwise.} \end{cases}$$
$$= \left(\frac{R^2}{2} - \frac{\nu^2}{2k_{\nu,j}^2} \right) J_\nu(k_{\nu,j}R) \delta_{j,j'}. \tag{A.3}$$

Now, by using Equation (A.1)

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

which, by using Equation (A.3), leads to

$$2\pi\delta_{\nu,\nu'}\int_0^R \rho N_{\nu,j}J_\nu\left(k_{\nu,j}\rho\right)N_{\nu,j'}J_\nu\left(k_{\nu,j'}\rho\right)d\rho$$

$$=2\pi N_{\nu,j}^2\left(\frac{R^2}{2}-\frac{\nu^2}{2k_{\nu,j}^2}\right)J_\nu\left(k_{\nu,j}R\right)\delta_{\nu,\nu'}\delta_{j,j'}.$$

To conclude this proof, one can show by using Equation (A.3) again that

$$N_{\nu,j}^2 = \frac{1}{2\pi\int_0^R \rho J_\nu^2\left(k_{\nu,j}\rho\right)d\rho}$$

$$= \frac{1}{2\pi\left(\frac{R^2}{2}-\frac{\nu^2}{2k_{\nu,j}^2}\right)J_\nu\left(k_{\nu,j}R\right)},$$

and then finally,

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

which is the definition of an orthonormal basis[6]. $\blacksquare$

Appendix B.

In this Appendix we prove the properties that link $\varphi_{\nu,j}$ and $\varphi_{-\nu,j}$.

Theorem 2 *Let $\varphi_{\nu,j}, \forall \nu, j \in \mathbb{N}$ be the Bessel coefficients of a particular function $\Psi\left(\rho,\theta\right):$ $D^2 \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$ defined on a circular domain of radius R, that is,*

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

Then, these coefficients are not all independent. They are linked by the relations

$$\varphi_{-\nu,j} = (-1)^\nu\varphi_{\nu,j}^* \iff \begin{cases}\Re\left(\varphi_{-\nu,j}\right) = (-1)^\nu\Re\left(\varphi_{\nu,j}\right)\\ \Im\left(\varphi_{-\nu,j}\right) = (-1)^{\nu+1}\Im\left(\varphi_{\nu,j}\right).\end{cases}$$

Proof To prove this, we will use different properties of the Bessel functions. Firstly,

$$J_{-\nu}\left(x\right) = (-1)^\nu J_\nu\left(x\right),$$

and secondly,

$$J_\nu'\left(x\right) = \frac{1}{2}\left(J_{\nu-1}\left(x\right)-J_{\nu+1}\left(x\right)\right).$$

---

6. Note that the proof for $k_{\nu,j}$ defined by $J_\nu\left(k_{\nu,j}R\right) = 0$ is now straightforward since it only sweeps the non-zero term in Equation (A.2). The following remains the same.

Then, by using these two relations, one can show that

$$
\begin{aligned}
J'_{-\nu}(x) &= \frac{1}{2}\left(J_{-\nu-1}(x) - J_{-\nu+1}(x)\right) \\
&= \frac{(-1)^{\nu+1}}{2}\left(J_{\nu+1}(x) - J_{\nu-1}(x)\right) \\
&= (-1)^{\nu} J'_{\nu}(x)
\end{aligned} \tag{B.1}
$$

However, if $k_{-\nu,j}$ is such that $J'_{-\nu}(k_{-\nu,j}R) = 0$, it also leads thanks to Equation (B.1) to $J'_{\nu}(k_{-\nu,j}R) = 0$. And then, the only possibility is that $k_{-\nu,j} = k_{\nu,j}$ (because we still have $J'_{\nu}(k_{\nu,j}R) = 0$).

Now, regarding the normalization factor,

$$
\begin{aligned}
N^{-1}_{-\nu,j} &= \sqrt{2\pi \int_0^R \rho J^2_{-\nu}(k_{-\nu,j}\rho)\,d\rho} \\
&= \sqrt{2\pi \int_0^R \rho J^2_{\nu}(k_{\nu,j}\rho)(-1)^{2\nu}\,d\rho} \\
&= N^{-1}_{\nu,j}.
\end{aligned}
$$

One can now put all this together to show that

$$
\begin{aligned}
\varphi_{-\nu,j} &= \int_0^{2\pi}\int_0^R \rho\left[N_{-\nu,j}J_{-\nu}(k_{-\nu,j}\rho)e^{-i\nu\theta}\right]^* \Psi(\rho,\theta)\,d\theta d\rho \\
&= (-1)^{\nu}\int_0^{2\pi}\int_0^R \rho\left[N_{\nu,j}J_{\nu}(k_{\nu,j}\rho)e^{-i\nu\theta}\right]^* \Psi(\rho,\theta)\,d\theta d\rho \\
&= (-1)^{\nu}\varphi^*_{\nu,j},
\end{aligned}
$$

which leads to the end of the proof. ∎

Appendix C.

In this Appendix, we prove that the rotation invariant operation described in Section 5.1 is pseudo-injective. It means that results will be different if images are different. Pseudo makes reference to the exception when an image is compared to a rotated version of itself.

Theorem 3 *Let $\{\varphi_{\nu,j}\}$ be the Bessel coefficients of a particular function $\Psi(\rho,\theta) : D^2 \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$ defined on a circular domain of radius R. Let $\{\varphi'_{\nu,j}\}$ be the Bessel coefficients of another particular function $\Psi'(\rho,\theta)$ defined on the same domain D^2. Finally, let $\{\kappa_{\nu,j}\}$ be some arbitrary complex numbers. Then,*

$$
\sum_{\nu}\left|\sum_{j}\kappa^*_{\nu,j}\varphi_{\nu,j}\right|^2 = \sum_{\nu}\left|\sum_{j}\kappa^*_{\nu,j}\varphi'_{\nu,j}\right|^2 \Rightarrow \exists\alpha : \Psi(\rho,\theta) = \Psi'(\rho,\theta-\alpha), \forall\rho, \forall\theta.
$$

41

Proof   To make developments easier, one can use the bra-ket notation commonly used in quantum mechanics to denote quantum states. In this notation, $|v\rangle$ is called a *ket* and denotes a vector in an abstract complex vector space, and $\langle v|$ is called a *bra* and corresponds to the same vector but in the dual vector space. It follows that $|v\rangle^{\dagger} = \langle v|$, and the inner-product between two vectors is conveniently expressed by $\langle v|f\rangle$, and the outer-product by $|f\rangle\langle v|$.

By using the fact that $|z|^2 = zz^*$ and the bra-ket notation,

$$\sum_{\nu} |\sum_{j} \kappa_{\nu,j}^* \varphi_{\nu,j}|^2 = \sum_{\nu} |\sum_{j} \kappa_{\nu,j}^* \varphi'_{\nu,j}|^2$$

leads to

$$\sum_{\nu} \langle \kappa_{\nu}|\varphi_{\nu}\rangle \langle \kappa_{\nu}|\varphi_{\nu}\rangle^* = \sum_{\nu} \langle \kappa_{\nu}|\varphi'_{\nu}\rangle \langle \kappa_{\nu}|\varphi'_{\nu}\rangle^*,$$

where κ_{ν} (resp., φ_{ν}) is a vector that contains all the different values $\kappa_{\nu,j}$ (resp., $\varphi_{\nu,j}$) for this particular ν. This Equation can further be written

$$\sum_{\nu} \langle \kappa_{\nu}|\varphi_{\nu}\rangle \langle \varphi_{\nu}|\kappa_{\nu}\rangle = \sum_{\nu} \langle \kappa_{\nu}|\varphi'_{\nu}\rangle \langle \varphi'_{\nu}|\kappa_{\nu}\rangle. \tag{C.1}$$

However, since the $\kappa_{\nu,j}$'s are totally arbitrary, the only possibility to satisfy Equation (C.1) is that

$$\sum_{\nu} |\varphi_{\nu}\rangle\langle\varphi_{\nu}| = \sum_{\nu} |\varphi'_{\nu}\rangle\langle\varphi'_{\nu}|.$$

In quantum mechanics, $|\varphi_{\nu}\rangle\langle\varphi_{\nu}|$ is called the density matrix of φ_{ν}, and it is known that the only way to achieve identical density matrices for different states φ_{ν} and φ'_{ν} is that they should only differ by a phase factor[7]. ∎

References

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.

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.

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

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

---

7. Indeed, if $\varphi_{\nu} = \varphi'_{\nu}e^{i\nu\alpha}$, then $|\varphi'_{\nu}\rangle\langle\varphi'_{\nu}| = |e^{i\nu\alpha}\varphi_{\nu}\rangle\langle e^{i\nu\alpha}\varphi_{\nu}| = e^{i\nu\alpha}e^{-i\nu\alpha}|\varphi_{\nu}\rangle\langle\varphi_{\nu}| = |\varphi_{\nu}\rangle\langle\varphi_{\nu}|$

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

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.

Nima Dehmamy, Robin Walters, Yanchen Liu, Dashun Wang, and Rose Yu. Automatic symmetry discovery with lie algebra convolutional network. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2503–2515, 2021.

Valentin Delchevalerie, Adrien Bibal, Benoît Frénay, and Alexandre Mayer. Achieving rotational invariance with bessel-convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 28772–28783, 2021.

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.

Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3165–3176, 2020.

Liyao Gao, Guang Lin, and Wei Zhu. Deformation robust roto-scale-translation equivariant CNNs. *arXiv:2111.10978*, 2021.

Robert Gens and Pedro M Domingos. Deep symmetry networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2537–2545, 2014.

Rohan Ghosh and Anupam K Gupta. Scale steerable filters for locally scale-invariant convolutional neural networks. *arXiv:1906.03861*, 2019.

Simon Graham, David Epstein, and Nasir Rajpoot. Dense steerable filter CNNs for exploiting rotational symmetry in histology images. *IEEE Transactions on Medical Imaging*, 39 (12):4124–4136, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

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

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.

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.

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.

Shaohua Li, Yong Liu, Xiuchao Sui, Cheng Chen, Gabriel Tjio, Daniel Shu Wei Ting, and Rick Siow Mong Goh. Multi-instance multi-scale cnn for medical image classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 531–539, 2019.

Xilai Li, Wei Sun, and Tianfu Wu. Attentive Normalization, March 2021.

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.

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.

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.

K. F. Riley, M. P. Hobson, and S. J. Bence. *Mathematical Methods for Physics and Engineering: A Comprehensive Guide*. Cambridge University Press, 3 edition, 2006.

Maurice Weiler and Gabriele Cesa. General E(2)-equivariant steerable CNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 14334–14345, 2019.

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.

Kyle W. Willett, Chris J. Lintott, Steven P. Bamford, Karen L. Masters, Brooke D. Simmons, Kevin R. V. Casteels, Edward M. Edmondson, Lucy F. Fortson, Sugata Kaviraj, William C. Keel, Thomas Melvin, Robert C. Nichol, M. Jordan Raddick, Kevin Schawinski, Robert J. Simpson, Ramin A. Skibba, Arfon M. Smith, and Daniel Thomas. Galaxy Zoo 2: detailed morphological classifications for 304 122 galaxies from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society*, 435(4):2835–2860, 2013.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5028–5037, 2017.

Yichong Xu, Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, and Zheng Zhang. Scale-invariant convolutional neural networks. *arXiv:1411.6369*, 2014.

Hang Yu, Feng Yang, Sivaramakrishnan Rajaraman, Ilker Ersoy, Golnaz Moallem, Mahdieh Poostchi, Kannappan Palaniappan, Sameer Antani, Richard J. Maude, and Stefan Jaeger. Malaria Screener: a smartphone application for automated malaria screening. *BMC Infectious Diseases*, 20(1):825, November 2020. doi: 10.1186/s12879-020-05453-1.

Zhizhen Zhao and Amit Singer. Fourier–Bessel rotational invariant eigenimages. *Journal of the Optical Society of America A*, 30(5):871, 2013. ISSN 1084-7529, 1520-8532.