Learning to Convolve: A Generalized Weight-Tying Approach

Nichita Diaconu [1 *] **Daniel Worrall** [1 *]

Abstract

Recent work (Cohen & Welling, 2016a) has shown that generalizations of convolutions, based on group theory, provide powerful inductive biases for learning. In these generalizations, filters are not only translated but can also be rotated, flipped, etc. However, coming up with exact models of how to rotate a 3×3 filter on a square pixel-grid is difficult. In this paper, we learn how to transform filters for use in the group convolution, focussing on roto-translation. For this, we learn a filter basis and all rotated versions of that filter basis. Filters are then encoded by a set of rotation invariant coefficients. To rotate a filter, we switch the basis. We demonstrate we can produce feature maps with low sensitivity to input rotations, while achieving high performance on MNIST and CIFAR-10.

1. Introduction

Convolutional neural networks (CNNs) are now the model of choice for many perceptual recognition tasks, such as image recognition (Russakovsky et al., 2015) or semantic segmentation (Cordts et al., 2016). Among all the architectural innovations of recent years, a key component has not changed: the convolution. Convolutions constrain a mapping to have translational symmetry (Cohen & Welling, 2016a). This has proven to be a significant inductive bias, and has added bonuses of reducing the number of trainable weights via weight-tying. As such, CNNs have improved sample complexity (Sokolic et al., 2017) and thus better generalization compared to their non-symmetric cousins.

Recently, there has been a flurry of works (Cohen & Welling, 2016a;b; Worrall et al., 2017; Worrall & Brostow, 2018; Bekkers et al., 2018; Weiler et al., 2018b;a; Esteves et al., 2018) on extending the class of symmetries beyond pixelwise translations. Most notably Cohen & Welling (2016a)'s *group convolutions* extend standard translational convolution to the setting where the symmetry is a discrete algebraic group (explained in Section 2.2). In other words, these are convolutions over invertible transformations, so kernels are not only translated but also rotated, flipped, etc.

One of the key assumptions with Cohen & Welling (2016a) and associated approaches is that the set of transformations forms a group. We cannot pick an arbitrary set of transformations. For instance, in Cohen & Welling (2016a) the authors choose the group of pixelwise translations, 90° rotations, and flips, that is the set of all transformations that map the regular square-lattice into itself; and in Hoogeboom et al. (2018) the authors consider the set of all transformations that map the hexagonal lattice into itself. However, in general the set of $\frac{2\pi}{N}$ rotations for integer N and pixelwise translations does not form a group because of pixelwise discretization, yet in Bekkers et al. (2018) and Weiler et al. (2018b), the authors use these sets of transformations. Their line of thought is to model roto-translations in the continuous setting and then discretize *post hoc*. To synthesize rotating a filter, they apply bilinear interpolation (Bekkers et al., 2018) or a Fourier-based interpolation scheme (Weiler et al., 2018b). Both of these methods suffer from the fact that the *post hoc* discretization is not accounted for by the interpolation schemes employed.

Our solution is to learn an interpolation scheme to transform the filters, which accounts for this *post hoc* discretization. Our proposal is to learn a space of filters and transformed versions of this space. A filter at one orientation is defined by its coefficients in a chosen basis. It can then be rotated by changing the basis, while keeping the coefficients unchanged. As a result, this is a generalized version of weight-tying. Since the bases are learned, we do not explicitly define a transformation rule on the filters. This avoids some of the overlooked flaws of handcrafted transformations.

Our contributions are:

- We present an expanded definition of the group convolution, with a new proof of equivariance.
- We pinpoint the spurious non-equivariance properties of existing roto-translational group CNNs down to the non-unitarity of the filter transformations used.
- We present a generalized weight-tying method, where

[*]Equal contribution [1]Philips Lab, University of Amsterdam, Netherlands. Correspondence to: Daniel Worrall <d.e.worrall@uva.nl>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

the convolution is learned.

In Section 2 we cover background material on group convolutions, in Section 3 we explain our method to learn how to rotate filters, and in Section 4 we present experiments on MNIST and CIFAR-10 comparing classification accuracy and robustness under rotations of the input.

2. Background

Here, we introduce discrete group convolutions. We present them from a novel perspective for freshness and to elucidate the leap from handcrafted transformations to learning them.

2.1. Standard convolutions

The standard translational convolution[1] has the form

$$[f \star_{\mathbb{Z}^d} \psi](g) = \sum_{x \in \mathbb{Z}^d} f(x)\psi(x-g), \tag{1}$$

where f is a signal (image) and ψ is a filter, both defined on domain $\mathbb{Z}^d$. It can be interpreted as a collection of inner products of signal f with g-translated versions of filter ψ. To emphasize (and generalize) this translational behavior, we rewrite the filter shift using the translation operator $\mathcal{L}_g$:

$$[f \star_{\mathbb{Z}^d} \psi](g) = \sum_{x \in \mathbb{Z}^d} f(x)\mathcal{L}_g[\psi](x) \tag{2}$$

$$\mathcal{L}_g[\psi](x) = \psi(x-g). \tag{3}$$

In this form, we notice a characteristic property of the convolution. Each of the responses is indexed by the amount of translation g. Notice too that $\mathcal{L}_g$ is indexed by a translation parameter g, so we actually have a set of operators $\{\mathcal{L}_g\}_{g\in G}$, where G is the set of all transformations. Here, the domain of the output response of the convolution is $G = \mathbb{Z}^d$. A fundamental property of the convolution is that it is *equivariant* to translations (Kondor & Trivedi, 2018). This is the behavior that translations of the input f result in translations of the response $[f \star_{\mathbb{Z}^d} \psi]$. We write this as

$$\mathcal{L}_g[f \star_G \psi] = \mathcal{L}_g[f] \star_G \psi, \tag{4}$$

which highlights the commutativity between convolution and translation. So convolving a filter with a transformed image yields the same responses as first convolving the signal and then applying a transformation to that response. It turns out that the convolution (and reparametrizations of it) is the only linear map that commutes with translation.

2.2. Group convolution

We can extend the standard convolution by swapping the translation operator for another general transformation operator. We shall use the same notation $\mathcal{L}_g$ to denote this more general transformation operator and thus Equation 2 does not change. The basic form of this new convolution can be seen as a collection of inner products between a signal and generically transformed filters, reminiscent of template matching in classical computer vision (Szeliski, 2011).

Group Actions For the generalized convolution to maintain the equivariance property, we have to restrict the class of transformation operators. Following Cohen & Welling (2016a) we demand the following group properties. For all $g, h, k \in G$:

- Closure: $\mathcal{L}_g\mathcal{L}_h = \mathcal{L}_{g\circ h}$, where $\circ : G \times G \to G$ denotes composition.
- Associativity: $\mathcal{L}_g(\mathcal{L}_h\mathcal{L}_k) = (\mathcal{L}_g\mathcal{L}_h)\mathcal{L}_k$.
- Identity: there exists an $e \in G$ such that $\mathcal{L}_e[f] = f$.
- Inverses: for every $\mathcal{L}_g$ there exists an $\mathcal{L}_{g^{-1}}$ such that $\mathcal{L}_g\mathcal{L}_{g^{-1}} = \mathcal{L}_{g^{-1}}\mathcal{L}_g = \mathcal{L}_e$.

If the transformations satisfy these properties then they are called *group actions* and the set G is called a *group*. Typically, we ignore the notation $g \circ h$ and just write gh. Note that we have presented the action as a map from functions to functions, but it can also be applied to points. For instance, we can define an action $\mathcal{R}_R[x] = Rx$, where R is a rotation matrix. We can then define the rotation of a function $f : \mathbb{R}^2 \to \mathbb{R}$ via the rotation of its domain

$$\mathcal{R}_R[f](x) = f(\mathcal{R}_R^{-1}[x]) = f(R^{-1}x). \tag{5}$$

Here we have overloaded the notation $\mathcal{R}_R$ to act on functions and points. It should be clear from context, which is which. Notice how when we bring the action inside the function to act on the domain, we use the inverse of the action on points. This maintains the relation $\mathcal{R}_R\mathcal{R}_S = \mathcal{R}_{RS}$:

$$\mathcal{R}_R[\mathcal{R}_S[f]](x) = \mathcal{R}_S[f](\mathcal{R}_R^{-1}[x]) = f(\mathcal{R}_S^{-1}[\mathcal{R}_R^{-1}[x]]) \tag{6}$$

$$= f(\mathcal{R}_{S^{-1}R^{-1}}[x]]) = f(\mathcal{R}_{(RS)^{-1}}[x]]) = \mathcal{R}_{RS}[f](x). \tag{7}$$

Group convolutions *Group convolutions* $\star_G$ use group actions of the form $\mathcal{L}_g[f](x) = f(\mathcal{L}_g^{-1}x)$, where:

$$[f \star_G \psi](g) = \sum_{x \in X} f(x)\psi(\mathcal{L}_g^{-1}[x]). \tag{8}$$

They differ from Equation 2 in that the domain X of the signal and filter is not necessarily $\mathbb{Z}^d$. Recall, the domain of the response is the group G. Since we wish to concatenate multiple group convolutions to form a CNN, we typically have $X = G$. In Kondor & Trivedi (2018) and Weiler et al. (2018a), the authors call X a *homogeneous space*.

[1] Technically this is a correlation, but we stick to the standard deep learning nomenclature.

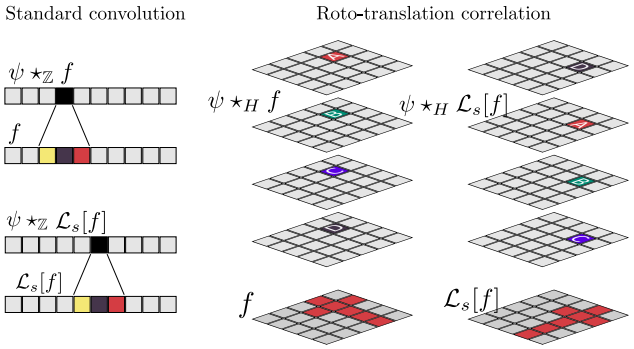

Figure 1. Example of the equivariance properties of a standard convolution versus a roto-translation convolutions. For the standard convolution, pixelwise shifts (left-right translations) of the input induce pixelwise shifts of the response. For the roto-translational group convolution, shifts of the input induce shifts. Rotations of the input induce a rotation of the spatial location of activations within a slice (horizontal plane) at the same time as a rolling (cyclic vertical shift) of the slices. Were we to rotate by ine full turn, the activations would trace a corkscrew path through space.

The roto-translation group In this paper, we focus primarily on the roto-translation group; although, our method is applicable to other group-structured transformations. The roto-translation group[2] is the group of all proper rigid-body transformations. Transforming a point x is performed as

$$\mathcal{L}_{R,z}[x] = Rx + z, \tag{9}$$

where R is a rotation matrix and z is the translation vector. We treat the tuple $g = (R, z)$ as a single element in the group G. We can also treat the roto-translation action as a composition of two actions: the rotation action $\mathcal{R}_R[x] = Rx$ followed by the translation action $\mathcal{T}_z[x] = x + z$, so

$$\mathcal{L}_{R,z}[x] = \mathcal{T}_z[\mathcal{R}_R[x]] = \mathcal{R}_R[x] + z = Rx + z. \tag{10}$$

A handy representation of elements (R, z) is the homogeneous representation shown Equation 11, where 0 is an appropriately sized vector of zeros. In this representation, composition of transformations is matrix multiplication. Thus for group elements (R, z) and (S, x) we find the useful expressions

$$\underbrace{\begin{bmatrix} R & z \\ 0^\top & 1 \end{bmatrix}}_{(R,z)} \underbrace{\begin{bmatrix} S & x \\ 0^\top & 1 \end{bmatrix}}_{(S,x)} = \underbrace{\begin{bmatrix} RS & Rx+z \\ 0^\top & 1 \end{bmatrix}}_{(RS,Rx+z)} \tag{11}$$

$$\underbrace{\begin{bmatrix} R & z \\ 0^\top & 1 \end{bmatrix}^{-1}}_{(R,z)^{-1}} = \underbrace{\begin{bmatrix} R^{-1} & -R^{-1}z \\ 0^\top & 1 \end{bmatrix}}_{(R^{-1},-R^{-1}z)}. \tag{12}$$

[2] Also known as the Euclidean group

We can now use this knowledge to write down an explicit group convolution for roto-translation. In both convolutions we will explicitly separate out the filter rotation component, which we later intend to learn. There is one expression for the network input, where $X = \mathbb{Z}^d$, and another for intermediate activations, where $X = G$. For the input:

$$[f \star_G \psi](R, z) = \sum_{x \in \mathbb{Z}^d} f(x)\psi(R^{-1}(x - z)) \tag{13}$$

$$= [f \star_{\mathbb{Z}^d} \mathcal{R}_R[\psi]](z). \tag{14}$$

In the first line, we have written the group convolution of Equation 8, substituting in the roto-translation action of Equation 9. In the second line, we have rewritten this as a standard translational convolution with different rotated versions of the filters ψ. For the intermediate convolutions, the behavior is more sophisticated. Since the input activations live on the group, the filters also must live on the group and thus have a rotation and a translation 'axis' and are written $\psi(S, x)$. In the following, we shall refer to $\psi_S(\cdot) = \psi(S, \cdot)$ as a *slice* (see Figure 1):

$$[f \star_G \psi](R,z) = \sum_{(S,x) \in G} f(S,x)\psi(R^{-1}S, R^{-1}(x-z)) \tag{15}$$

$$= \sum_S [f_S \star_{\mathbb{Z}^d} \mathcal{R}_R[\psi_{R^{-1}S}]](z) \tag{16}$$

Again in the first line we wrote out the convolution using the substitution of Equation 9 in 8. In the second line we notice that this convolution has two components i) for fixed S, we perform a standard translational convolution with feature map slice f_S and an R-rotated filter slice $\psi_{R^{-1}S}$, ii) we then sum over all responses from each slice. Notice that on top of the filter rotation we also permute the slices according to $R^{-1}S$, which we call a *roll* (see Figure 1), since in 2D this corresponds to a cyclic permutation of the slices. It is more complicated in higher dimensions, as shown in Worrall & Brostow (2018).

The equivariance properties of this convolution are interesting. Input shifts induce shifts of the response. Rotations induce a more complex transformation. We visualize this transformation in Figure 1. First of all, responses are rotated spatially within a slice by the same degree as the input rotation, and at the same time there is a rolling of the slices over the 'rotation-axis' of the response tensor. After one full turn the activations return to where they begun, tracing a corkscrew-like path through space.

Our presentation of the group convolution differs from previous works in that we present the group action as $\mathcal{L}_g[f](x)$ rather than $f(\mathcal{L}_g^{-1}[x])$. The difference between these two expressions is that in the first, we can transform both the pixel locations and values of a signal, but in the second we can only permute their locations. To demonstrate that this is

a more accurate model of signal transformations consider the case where x is a point in the lattice $\mathbb{Z}^d$. The transformation $\mathcal{L}_g^{-1}[x]$ may not actually exist for general g—an example is where g represents a 45° rotation. Here $\mathcal{L}_g^{-1}[x]$ maps x to a point off the lattice and therefore cannot exist. To remedy this, we have two options: i) in Cohen & Welling (2016a) the authors opt not to allow transformations where $g^{-1}x$ do not exist and focus on 90° rotations, ii) in Bekkers et al. (2018) and Weiler et al. (2018b) the authors instead choose to use an interpolation function to perform filter rotation. This second method is the only satisfactory method to achieve equivariance at non-90° rotations, but this is implementing $\mathcal{L}_g[f]$, not $f(\mathcal{L}_g^{-1}[x])$.

3. Method

In the following, we present a modified and expanded definition of the discrete group convolution, based of actions of the form $\mathcal{L}_g[f]$ rather than actions $f(\mathcal{L}_g^{-1}[x])$. We then set out the condition needed for this expanded definition to satisfy equivariance, namely that the action has to be unitary under the inner product of the convolution. We then explore what happens if we replace the group for an arbitrary non-group-structured set of transformations.

3.1. The Unitary Group Convolution

Our expanded definition of the group convolution uses function transformations of the form $\mathcal{L}_g[f]$ so that we have

$$[f \star_G \psi](g) = \sum_{x \in X} f(x)\mathcal{L}_g[\psi](x). \tag{17}$$

This may look like a small change from Equation 8, but it heralds a shift in the nature of the group convolution. It turns out the proof of equivariance for the group convolution of Cohen & Welling (2016a) (see Supplementary material) no longer holds. We cannot prove equivariance of this new expression without an extra property. This property is unitarity of $\mathcal{L}_g$ with respect to the inner product; that is

$$\sum_{x \in X} \mathcal{L}_g[f](x)\mathcal{L}_g[\psi](x) = \sum_{x \in X} f(x)\psi(x). \tag{18}$$

This leads to the proof of equivariance:

$$\begin{aligned}
[\mathcal{L}_t[f] \star_G \psi](g) &= \sum_{x \in G} \mathcal{L}_t[f](x)\mathcal{L}_g[\psi](x) && (19)\\
&= \sum_{x \in G} f(x)\mathcal{L}_{t^{-1}g}[\psi](x) && (20)\\
&= [f \star_G \psi](t^{-1}g) && (21)\\
&= \mathcal{L}_t[f \star_G \psi](g). && (22)
\end{aligned}$$

From the first to the second lines, we have used the unitarity property; from the second to third lines we have used the definition of the group convolution; and from the third to fourth lines we have used the fact that the group action on functions on the group is defined as $\mathcal{L}_t[f](g) = f(t^{-1}g)$. We call this version of the group convolution the *unitary group convolution*, to differentiate it from the regular group convolution of Equation 8.

Unitarity imposes a very intuitive constraint on the form of the group convolution. This constraint is that the response value at g matters only on the relative transformation between f and ψ, and is independent of their absolute transformations. Linking back to the group convolution of Cohen & Welling (2016a), the inner product is automatically unitary if we choose the group action to be of the form

$$\mathcal{L}_g[f](x) = f(\mathcal{L}_g^{-1}[x]), \tag{23}$$

so this is a strict generalization of that work. By demanding unitarity, we see that we cannot use any interpolation method to impose filter transformations. Neither the bilinear interpolation of Bekkers et al. (2018) nor the Fourier-based interpolation of Weiler et al. (2018b) satisfy unitarity.

3.2. Representing Filter Spaces

We want to build a convolution without using a proper group, but an arbitrary discretization of a continuous group, which we denote as $\tilde{G}$. Thus we wish to approximate the equivariance properties of the unitary group convolution. This boils down to us having to find an approximate action $\mathcal{L}$.

We choose to learn this action based on examples of images and their rotated versions. Furthermore, since we require a technique to rotate all filters independent of their particular values, what we are actually searching for is a technique to rotate an entire filter space. We choose to model filters ψ as living in a linear vector space spanned by a basis $\{e^i(x)\}_{i=1}^N$, where $x \in \tilde{G}$. Then each filter can be formed as a linear combination of the bases:

$$\psi(x) = \sum_{i=1}^{N} \hat{\psi}_i e^i(x) \tag{24}$$

where $\{\hat{\psi}_i\}_{i=1}^N$ are the filter coefficients. As a shorthand, we will stack the basis into a vector $e = [e^1(x), ..., e^N(x)]^\top$ and the coefficients into a vector $\hat{\psi} = [\hat{\psi}_1, ..., \hat{\psi}_N]^\top$, so that we can write $\psi(x) = \hat{\psi}^\top e(x)$. Now if we apply an action to the filter, we have

$$\mathcal{L}_g[\psi](x) = \mathcal{L}_g\left[\hat{\psi}^\top e\right](x) = \hat{\psi}^\top \mathcal{L}_g[e](x) \tag{25}$$

where in the second equality we have imposed that the action be linear with respect to the input function. In fact linearity is not an extra condition, since unitary operators are always linear. So to transform a filter, we have to find the filter coefficients $\hat{\psi}$ in the original basis $e(x)$ and change

Algorithm 1 Task-specific training using pre-trained basis

Load basis: e
Initialize coefficients: $\hat{\psi}$
for *minibatch in dataset* **do**
  $\psi \leftarrow \hat{\psi}^\top e$
  Forward prop and compute loss
  $\hat{\psi} \leftarrow$ backprop update$(\frac{\partial L}{\partial \hat{\psi}})$
end

Algorithm 2 The basis is pretrained offline

Initialize basis: e
Set sample $angles = [0, \frac{\pi}{8}, \frac{2\pi}{8}, ... \frac{7\pi}{8}]$
for *minibatch in dataset* **do**
  Draw $S, R \sim$ Uniform$(angles)$
  Compute $L_{\text{equiv}}, L_{\text{rec}}, L_{\text{orth}}$
  $L \leftarrow L_{\text{equiv}} + L_{\text{rec}} + L_{\text{orth}}$
  $e \leftarrow$ backprop update$(\frac{\partial L}{\partial e})$
end
Save basis e

the basis for its transformed version $\mathcal{L}_g[e](x)$. If we can achieve this, then the filter coefficients $\hat{\psi}$ are invariant to the transformation. This is a generalized form of weight-tying.

The main problem now is to find a transformed basis $\mathcal{L}_g[e](x)$. Instead of learning how to transform the basis, we choose to learn a separate basis for each transformation g. We denote this learned basis as $e_g(x)$, so

$$\mathcal{L}_g[\hat{\psi}^\top e](x) = \hat{\psi}^\top e_g(x) \tag{26}$$

for all $\hat{\psi}$ and g. Algorithmically this basis is used at training time as shown in Algorithm 1. Looking at Equation 15, we see that the roto-translation convolution contains two parts. One is a spatial rotation, the other is an axis roll. We know exactly how to perform an axis roll, but not how to perform the rotation. We opt to learn the rotation aspect only and hard-code the roll ourselves.

3.3. Learning To Rotate

Recall that the roto-translation convolution at the input and intermediate layers can be written

$$[f \star_G \psi](R, z) = [f \star_{\mathbb{Z}^d} \mathcal{R}_R[\psi]](z) \tag{27}$$

$$[f \star_G \psi](R, z) = \sum_S [f_S \star_{\mathbb{Z}^d} \mathcal{R}_R[\psi_{R^{-1}S}]](z). \tag{28}$$

From these expressions, we see that the convolution is composed of translations, rotations and rolls. We know how to perform translations and rolls exactly, but not rotations. Thus we only need to learn how to rotate our bases, which we do by minimizing a sum of three losses (explained below),

$$L = L_{\text{equiv}} + L_{\text{orth}} + L_{\text{rec}}. \tag{29}$$

Algorithmically this process is shown in Algorithm 2.

Orthogonality loss To encourage the learned basis to span the space of filters, we add an orthogonality loss

$$L_{\text{orth}} = \sum_R \left\| e_R e_R{}^\top - I \right\|_1 \tag{30}$$

for each R-rotated version of the basis, where I is an identity matrix of size $N \times N$, and N is number of basis rotations.

Equivariance loss We make use of the following fact

$$\mathcal{R}_S[f] \star_{\mathbb{Z}^d} \mathcal{R}_R[\psi] = \mathcal{R}_S[f \star_{\mathbb{Z}^d} \mathcal{R}_{S^{-1}R}[\psi]] \tag{31}$$

which is proven in the *Supplementary material*. This relationship shows that translationally convolving an S-rotated image $\mathcal{R}_S[f]$ with an R-rotated filter is the same as convolving the unrotated image f with an $S^{-1}R$-rotated filter and then rotating the responses by S. The usefulness of Equation 31 is that it connects filters at rotation R with filters at rotation $S^{-1}R$. Thus we can use it to learn rotated versions of filters. Since this expression is correct for any ψ, it must also be correct for any basis element e^i. To learn a base element and its rotated versions we minimize an $L1$-loss

$$L_{\text{equiv}} = \sum_{S,R,i} L_{\text{equiv}}^{S,R,i} \tag{32}$$

$$L_{\text{equiv}}^{S,R,i} = \sum_{x \in \mathbb{Z}^d} \left\| [\mathcal{R}_S[f] \star_{\mathbb{Z}^d} e_R^i] - \mathcal{R}_S[f \star_{\mathbb{Z}^d} e_{S^{-1}R}^i] \right\|_1 \tag{33}$$

for all rotations S and R, all basis elements $\{e^i\}_{i=1}^N$, and all f in our training set. $\| \cdot \|_1$ is the $L1$-norm. In practice, we sample angles S and R at random from the set of rotations (in our case $S, R \in \{\frac{\pi n}{4} | n \in \{0, ..., 7\}\}$.

Reconstruction loss Finally, we find that an extra reconstruction loss helps in learning a basis. If we translationally convolve a signal f with a base element e^i, we reconstruct the original signal by translationally convolving the response with the transposed filter (flipped in the x and y axes) (Dumoulin & Visin, 2016), which we denote as $\bar{e}^i$. The reconstruction does not work for any e^i, but only for a subset of filters, known as *self-adjoint*. These are in fact unitary. The transpose convolution is denoted as: $\star_{\mathbb{Z}^d} \bar{e}$, where we only change the notation for the basis, because when the convolution is viewed as a matrix multiplication, the transpose convolution operation, given the same filters, is equivalent to transposing the matrix of the filters (Dumoulin & Visin, 2016). If the reconstruction task works then e^i convolved with its transpose $\bar{e}^i$ should result in a Dirac delta,

so we have that $e^i \star_{\mathbb{Z}^d} \bar{e}^i = \delta(0)$. This allows us to find the following expression

$$\mathcal{R}_S[f] = \mathcal{R}_S[f] \star_{\mathbb{Z}^d} [e^i_R \star_{\mathbb{Z}^d} \bar{e}^i_R] \tag{34}$$

$$= \mathcal{R}_S[f \star_{\mathbb{Z}^d} e^i_{RS^{-1}}] \star_{\mathbb{Z}^d} \bar{e}^i_R \tag{35}$$

For the reconstruction loss, we found a sum over the different basis elements leads to stabler optimization:

$$L_{\text{rec}} = \sum_{x \in \mathbb{Z}^d} \left\| \mathcal{R}_S[f](x) - \sum_i [\mathcal{R}_S[f \star_{\mathbb{Z}^d} e^i_{RS^{-1}}] \star_{\mathbb{Z}^d} \bar{e}^i_R](x) \right\|_1 \tag{36}$$

3.4. Implementation details

Imposing Extra Constraints We do not need to learn how to rotate at every angle, since we can rotate by $90°$ perfectly due to the square symmetry of the grid. We thus only learn to rotate in the range $[0°, 90°)$ and populate all other filters, by exactly rotating our learned filters afterwards.

Convolving using a basis Once we have learned a basis, we implement the unitary group-like convolution as

$$[f \star_G \psi](R, z) = [f \star_{\mathbb{Z}^d} \hat{\psi}^\top e_R](z) \tag{37}$$

$$[f \star_G \psi](R, z) = \sum_S [f_S \star_{\mathbb{Z}^d} \hat{\psi}^\top_{R^{-1}S} e_R](z). \tag{38}$$

Rotating images To train the equivariance loss and the reconstruction loss, we still have to perform a spatial rotation of the feature maps. We do this using a Gaussian interpolator with width $\sigma = 0.5$ and a kernel of size 3. Due to the fact that, when rotating an image, we have to interpolate values outside of the image grid, when we compute either the equivariance or the reconstruction, we crop $\frac{1}{4}$ of the feature maps on all sides.

4. Experiments and Results

We present our results on some basic benchmarks. We demonstrate competitive classification performance on the CIFAR-10 image recognition dataset, and compare the equivariance properties of activations from different models.

4.1. CIFAR-10

We run a simple test on the CIFAR-10 classification benchmark (Krizhevsky, 2009) to check that our network is able to produce comparable results to standard baselines. We compare against a standard CNN (CONV), a group CNN with a random basis (RANDOM), a group CNN with a Fourier-basis as per Weiler et al. (2018b) (WEILER), a group CNN with a Gaussian-interpolated basis (GAUSSIAN), and a group CNN with a bilinear-interpolated basis (BILINEAR). For the last two interpolated basis methods, we also need a 'zero-orientation' basis to rotate, so we use one of our learned bases. We also compare three versions of our learned basis: a basis learned at all angles (FULL), a basis learned in the range $0° - 90°$ and manually rotated to the other angles (PARTIAL), and an overcomplete basis (OVERCOMPLETE) learned at all angles with 3 times as many base elements.

Table 1. Results on the CIFAR-10 benchmark, with no augmentation left and with full rotation augmentation (right). The methods are split into no roto-translation equivariance, handcrafted equivariance, and learned equivariance (ours).

METHOD	TEST ACC	TEST ACC (AUG.)
All-CNN-C-like	92.23	84.14
Random	90.79	80.86
Weiler et al. (2018b)	90.2	89.48
Bilinear	90.63	89.41
Gaussian	90.72	89.45
Full (Ours)	**92.75**	89.84
Partial (Ours)	90.93	89.60
Overcomplete (Ours)	91.16	**89.93**

Our model, detailed in the Appendix, is an All-CNN-C-like (Springenberg et al., 2014) architecture. For all our experiments we used $|G| = 8$, meaning we have 8 orientations of the basis, at multiples of 45 degrees. We used the AMSGrad variant of Adam (Reddi et al., 2018) as the optimizer and we trained for 100 epochs at learning rate 10^{-3} and weight decay 10^{-6}. We use a minibatch size of 100. For data augmentation, we use random flips, color normalization and random translations of at most 4 pixels.

Accuracy: CIFAR-10 The accuracy results can be seen in Table 1. When using no augmentation, we can see that interestingly, the random baseline performs very well. Among interpolation methods, ours performs best. Moreover, the difference between translation and roto-translational model increases as we add rotation augmentation to the dataset. The key message we learn from this is that our learned filter basis is competitive with other baseline methods. We can see, by comparison, that there is an overall decrease in performance from the column with no augmentation to the column with full data augmentation. We hypothesize that CIFAR-10 contains an orientation bias (e.g. the sky is always up). As a result, the non roto-equivariant models need extra capacity to learn object classifications at all angles, when the roto-equivariant models have this already built in.

4.2. Testing equivariance

Test error fluctuations One test of equivariance is to monitor how test error fluctuates as a function of input rotation. For this we followed a similar procedure to Weiler

et al. (2018b) and trained all models on the MNIST and CIFAR-10 classification benchmarks. We also trained the PARTIAL model using various amounts of data augmentation: rotations at multiples of $90°, 45°$ or rotations at all angles in $[0°, 360°)$, which we call 'full augmentation'. We plot the test error against input image rotation of the test set in Figures 2 and 3. Our models are shown in solid lines (pink/brown). The dashed lines show translational models and the dotted-and-dashed lines show competing group convolutional models using handcrafted interpolation methods.

In Figures 2 we see that our PARTIAL method performs best over all runs. At its worst it has a $\sim 2\%$ error increase over its best error on MNIST and 20% on CIFAR-10. By comparison, the other interpolation methods suffer from a $7 - 10\%$ error increase on MNIST and $30 - 40\%$ error increase on CIFAR-10. Interestingly, the fully learned basis does not perform well. We posit that small asymmetries in the fully learned basis may be exploited during training time to overfit on a single input orientation. Further evidence to support this is shown in Figure 4, where we demonstrate that the fully learned basis has poor equivariance error compared to models with exact equivariance to $90°$ rotations, such as the PARTIAL model. In Figure 3 we see that the more augmentation we add into the training set, the more equivariant our model becomes, while also increasing in performance at intermediate angles, between multiples of $45°$. The graph also shows that, for MNIST, the accuracy on the unrotated validation set increases with data augmentation, while on CIFAR-10, the highest accuracy does not change. An exception to this is when adding only $90°$ augmentation, which results in no improvement since PARTIAL is already equivariant to $90°$ rotations.

Activation robustness Another way to test for equivariance is to measure fluctuations in the activations under rotation of the input. For this, we took trained networks and fed in pairs of test images. Each pair of test images $(\mathcal{R}_R[f], \mathcal{R}_S[f])$ consists of the same image but at different angles defined by R and S. To measure the equivariance error, we then have to 'rectify' the activations by transforming one of them, say the second image's activations a_S, by $\mathcal{R}_{RS^{-1}}$. We use is the squared normalized L_2-error metric

$$L(a_R, a_S)^2 = \sum_k \sum_{x \in \mathbb{Z}^2} \frac{\left\| a_R^k(x) - \mathcal{R}_{RS^{-1}}[a_S^k](x) \right\|_2^2}{\|a_R^k(x)\|_2 \|\mathcal{R}_{RS^{-1}}[a_S^k](x)\|_2}, \tag{39}$$

where k is the channel number. We set $R = 0$ and average this loss over N images and all angles S in the set of rotations, where N_R is number of rotations, so

$$L_{\text{equivariance}} = \frac{1}{N N_R} \sum_a \sum_R L(a_0, a_R)^2. \tag{40}$$

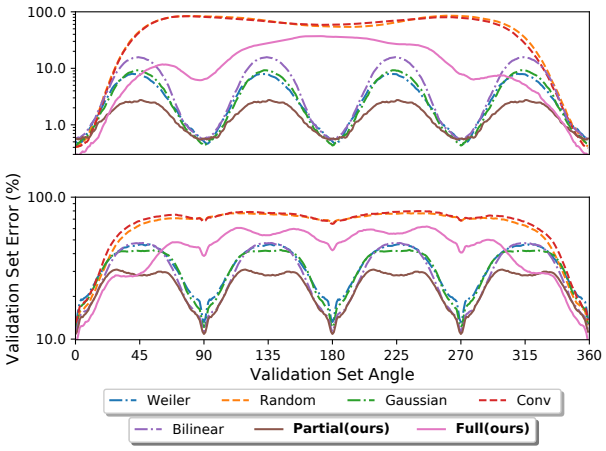

Figure 2. Validation error of a set of models on MNIST (top) and CIFAR-10 (bottom) when we rotate the validation set at different angles. We see similar behaviors in both datasets. Notably, the PARTIAL method performs best.

We show the errors per layer in Figure 4. We see that the PARTIAL method performs best. We also see that the FULL method is similar to the PARTIAL method in terms of equivariance error for the first two layers. We suspect small asymmetries in the FULL basis make the errors build up over depth, so that at the final layer the equivariance loss reaches the same values as the BILINEAR or GAUSSIAN methods. Our experiments demonstrate that with a finite number of rotations we can achieve good equivariance properties.

5. Related Work

There are a number of recent works on group convolutions, these are: continuous roto-translation in 2D (Worrall et al., 2017) and 3D (Weiler et al., 2018a; Kondor et al., 2018; Thomas et al., 2018) and discrete roto-translations in 2D (Cohen & Welling, 2016a; Weiler et al., 2018b; Bekkers et al., 2018; Hoogeboom et al., 2018) and 3D (Worrall et al., 2017), continuous rotations on the sphere (Esteves et al., 2018; Cohen et al., 2018), in-plane reflections (Cohen & Welling, 2016a). All of these works use handcrafted filter transformations. The closest to ours are Bekkers et al. (2018) and Weiler et al. (2018b) who handcraft interpolation methods to deal with *post hoc* discretization of the roto-translation group. Sabour et al. (2017) and Hinton et al. (2018) are the only other known works by the authors, where equivariance is learned as part of the objective function.

Jacobsen et al. (2016) and Weiler et al. (2018b) both represent filters using smooth bases to rotate filters by discrete angles; whereas Worrall et al. (2017), Weiler et al. (2018a), Kondor & Trivedi (2018), and Thomas et al. (2018)

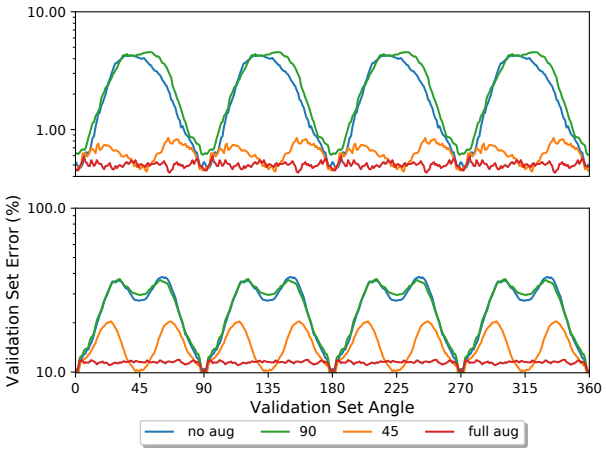

Figure 3. Validation error of the PARTIAL model on MNIST (top) and CIFAR-10 (bottom) when we rotate the validation set at different angles for various degrees of rotational data augmentation during training.

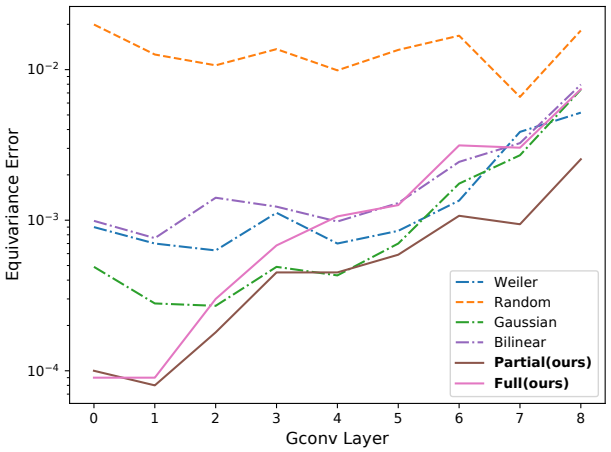

Figure 4. Per layer equivariance loss of the models in the baseline. We see that as the network depth increases all of the equivariance losses also increase. The handcrafted interpolation methods have higher error than the learned ones and this is most pronounced at the input to the network. Among the learned methods, the PARTIAL one performs per than the FULL method. We suspect that small asymmetries in the fully learned basis are exploited at training time to overfit to an orientation bias in the data.

use the steerability properties of harmonic bases to achieve continuous equivariance. The original work on steerable bases was performed by Freeman & Adelson (1991) and later formalized by Teo & Hel-Or (1997). In a different line of work Stanley et al. (2019) propose the HyperNEAT framework, which is an evolutionary algorithm for learning structured network weights. HyperNEAT has the capacity for learning invariances, but it could possibly achieve this without the need for weight-tying. Nonetheless, there is no explicit mechanism encouraging the framework to learn equi/invariance to symmetries in the input.

6. Discussion and Future Works

We have shown that *post hoc* discretization introduces asymmetries in group CNNs and we have tackled this by learning how to rotate filters. We showed that this give us comparable if not better performance on standard benchmarks and that this greatly improves robustness properties of group CNNs to input transformations. At the moment, the pipeline we have constructed is not an end-to-end learning system. Instead, we have to learn the bases offline, and then learn the filter coefficient afterwards. It would be most desirable for us to be able to construct a framework where we could learn both jointly, perhaps in the same vein as capsule networks (Sabour et al., 2017). It would also be interesting to see whether we could learn other kinds of transformation, such as scalings or affine warps, under *post hoc* discretization of non-compact groups, which to date have not be researched in great depth. Ideally, in the future, we should not even have to specify the exact mechanism of transformation but it would be desirable to learn the symmetries automatically.

To date, the authors are not aware of any literature, where a CNN learns symmetries in the task from an equivariance standpoint.

7. Conclusion

In this work we have made two main contributions. The first was a generalization of the discrete group convolution of Cohen & Welling (2016a) to allow for convolutions over groups, where the transformation operator applied to the kernel is allowed to be more complicated than a simple permutation of the pixel locations. This new model allows for pixel interpolation and is thus suited for non-90° filter rotations. For this generalized group convolution to work, we noted that the transformations have to be unitary under the inner product of the convolution. This indicated that prior works where filters are rotated using bilinear/Fourier-based interpolation do not satisfy the condition of equivariance.

Our second contribution was to introduce a method to learn how to rotate filters, to deal with *post hoc* discretization artifect. This method represents filters as linear combinations of basis filters. To rotate a filter, we swap the basis for a learned, rotated version of the basis. As such, this corresponds to a generalized version of weight-tying. We showed that this method produces competitive accuracy on the CIFAR-10 image recognition benchmark against matched architectures. We also showed that this method outperforms all other methods for representation stability.

References

Bekkers, E. J., Lafarge, M. W., Veta, M., Eppenhof, K. A. J., Pluim, J. P. W., and Duits, R. Roto-translation covariant convolutional networks for medical image analysis. In *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part I*, pp. 440–448, 2018. doi: 10.1007/978-3-030-00928-1_50.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 2990–2999, 2016a.

Cohen, T. S. and Welling, M. Steerable cnns. *CoRR*, abs/1612.08498, 2016b.

Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. Spherical cnns. *CoRR*, abs/1801.10130, 2018.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. The cityscapes dataset for semantic urban scene understanding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 3213–3223, 2016. doi: 10.1109/CVPR.2016.350.

Dumoulin, V. and Visin, F. A guide to convolution arithmetic for deep learning. *arXiv e-prints*, art. arXiv:1603.07285, March 2016.

Esteves, C., Allen-Blanchette, C., Makadia, A., and Daniilidis, K. Learning SO(3) equivariant representations with spherical cnns. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, pp. 54–70, 2018. doi: 10.1007/978-3-030-01261-8_4.

Freeman, W. T. and Adelson, E. H. The design and use of steerable filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(9):891–906, 1991. doi: 10.1109/34.93808.

Hinton, G., Sabour, S., and Frosst, N. Matrix capsules with em routing. 2018. URL https://openreview.net/pdf?id=HJWLfGWRb.

Hoogeboom, E., Peters, J. W. T., Cohen, T. S., and Welling, M. Hexaconv. *CoRR*, abs/1803.02108, 2018.

Jacobsen, J., van Gemert, J. C., Lou, Z., and Smeulders, A. W. M. Structured receptive fields in cnns. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 2610–2619, 2016. doi: 10.1109/CVPR.2016.286.

Kondor, R. and Trivedi, S. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 2752–2760, 2018.

Kondor, R., Lin, Z., and Trivedi, S. Clebsch-gordan nets: a fully fourier space spherical convolutional neural network. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pp. 10138–10147, 2018.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, F. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 3859–3869, 2017.

Sokolic, J., Giryes, R., Sapiro, G., and Rodrigues, M. R. D. Generalization error of invariant classifiers. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pp. 1094–1103, 2017.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. A. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.

Stanley, K. O., Clune, J., Lehman, J., and Miikkulainen, R. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35, 2019.

Szeliski, R. *Computer Vision - Algorithms and Applications*. Texts in Computer Science. Springer, 2011. ISBN 978-1-84882-934-3. doi: 10.1007/978-1-84882-935-0.

Teo, P. C. and Hel-Or, Y. A computational approach to steerable functions. In *1997 Conference on Computer Vision and Pattern Recognition (CVPR '97), June 17-19, 1997, San Juan, Puerto Rico*, pp. 313–318, 1997. doi: 10.1109/CVPR.1997.609341.

Thomas, N., Smidt, T., Kearnes, S. M., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks:

Rotation- and translation-equivariant neural networks for 3d point clouds. *CoRR*, abs/1802.08219, 2018.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pp. 10402–10413, 2018a.

Weiler, M., Hamprecht, F. A., and Storath, M. Learning steerable filters for rotation equivariant cnns. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 849–858, 2018b.

Worrall, D. E. and Brostow, G. J. Cubenet: Equivariance to 3d rotation and translation. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part V*, pp. 585–602, 2018. doi: 10.1007/978-3-030-01228-1_35.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 7168–7177, 2017. doi: 10.1109/CVPR.2017.758.