

# Group Equivariant Deep Learning

## Lecture 2 - Steerable group convolutions

### Lecture 2.6 - Activation functions for steerable G-CNNs

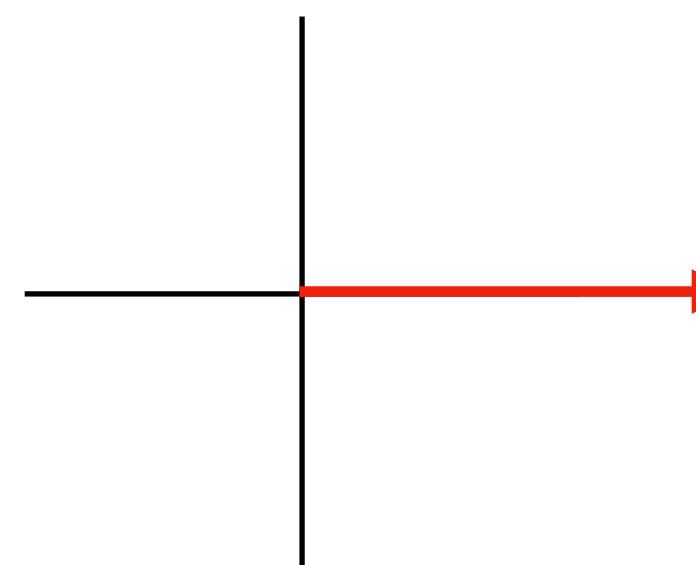
# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Incompatible activation function: **point-wise non-linearity for non-regular types**

$$\rho(\mathbf{R}_\pi) \text{ReLU}\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}\right) = \rho(\mathbf{R}_\pi) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$



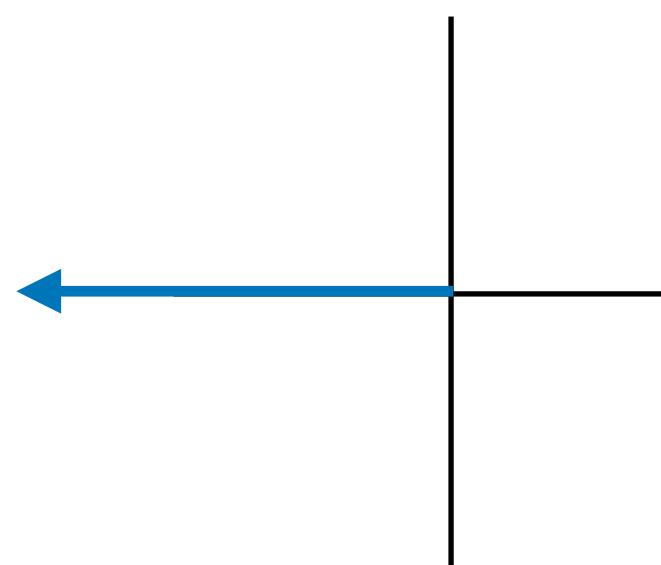
# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Incompatible activation function: **point-wise non-linearity for non-regular types**

$$\rho(\mathbf{R}_\pi) \text{ReLU}\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}\right) = \rho(\mathbf{R}_\pi) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$



# Activation functions

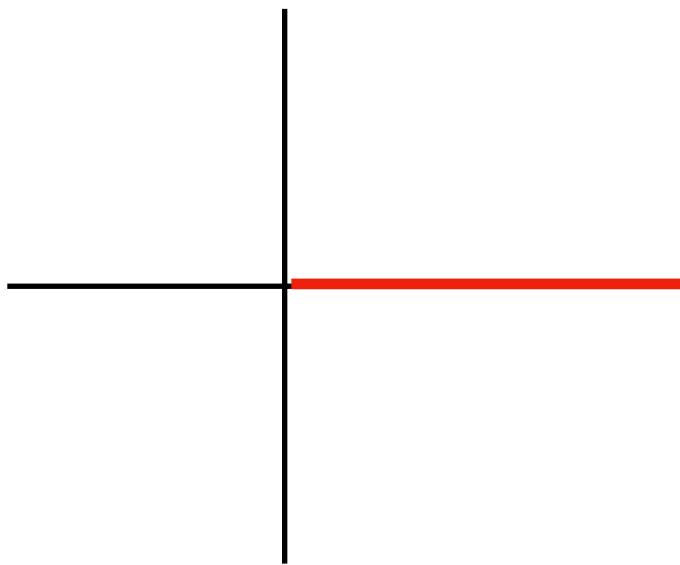
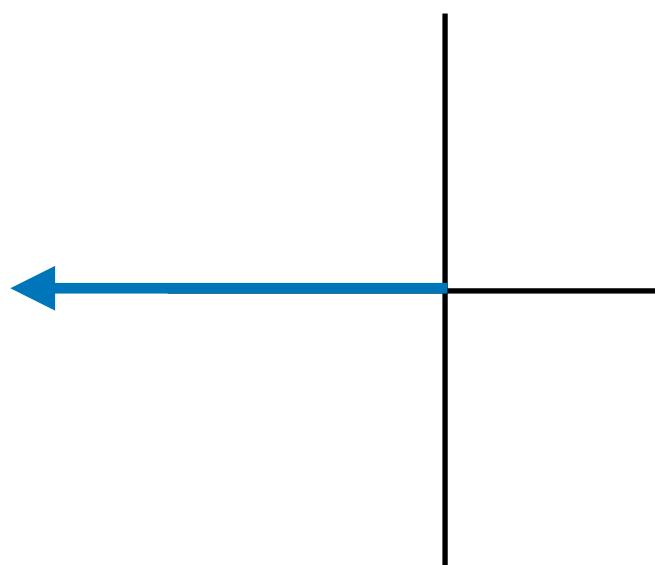
Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Incompatible activation function: **point-wise non-linearity for non-regular types**

$$\rho(\mathbf{R}_\pi) \text{ReLU}\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}\right) = \rho(\mathbf{R}_\pi) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

$$\text{ReLU}\left(\rho(\mathbf{R}_\pi) \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right) = \text{ReLU}\left(\begin{pmatrix} -1 \\ 0 \end{pmatrix}\right) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$



# Activation functions

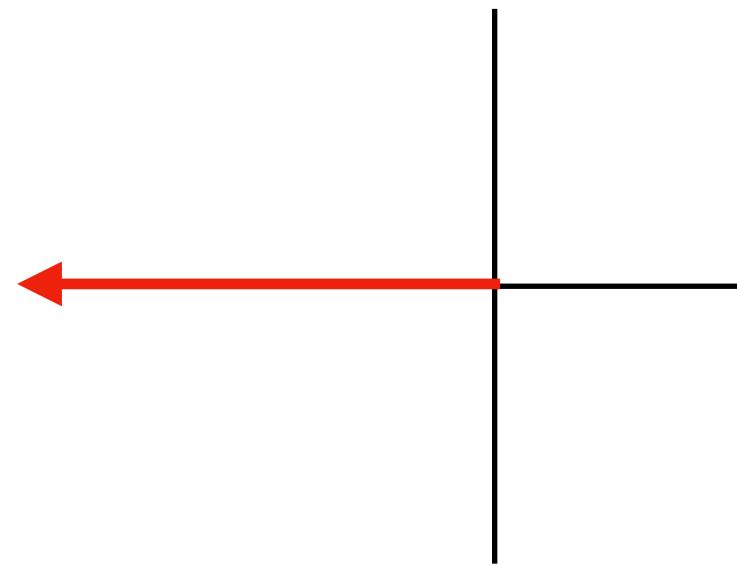
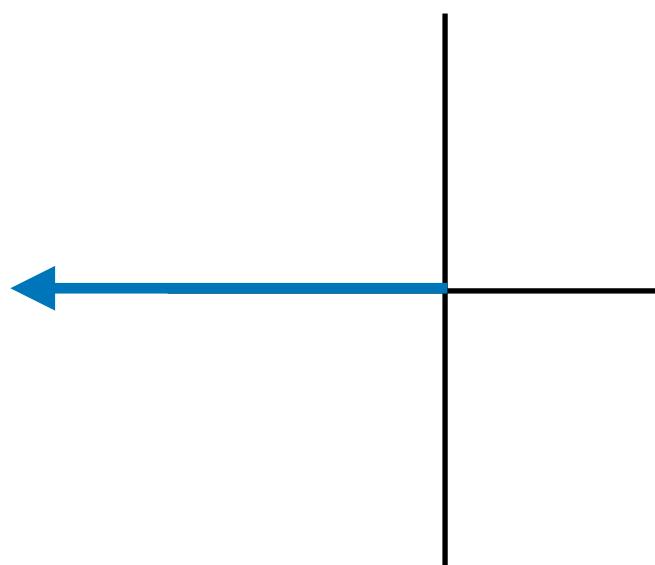
Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Incompatible activation function: **point-wise non-linearity for non-regular types**

$$\rho(\mathbf{R}_\pi) \text{ReLU}\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}\right) = \rho(\mathbf{R}_\pi) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

$$\text{ReLU}\left(\rho(\mathbf{R}_\pi) \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right) = \text{ReLU}\left(\begin{pmatrix} -1 \\ 0 \end{pmatrix}\right) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$



# Activation functions

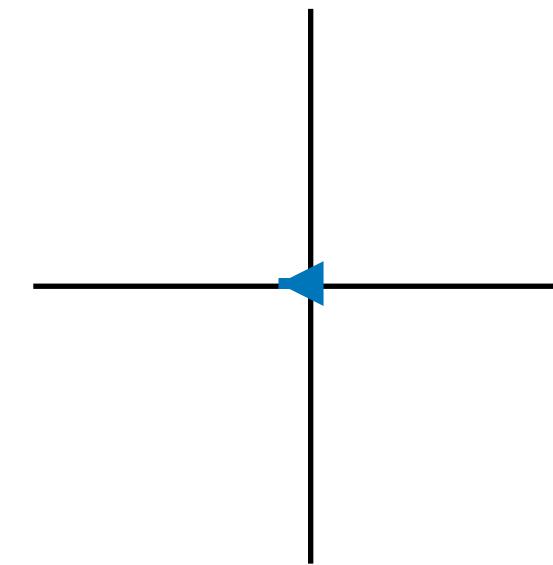
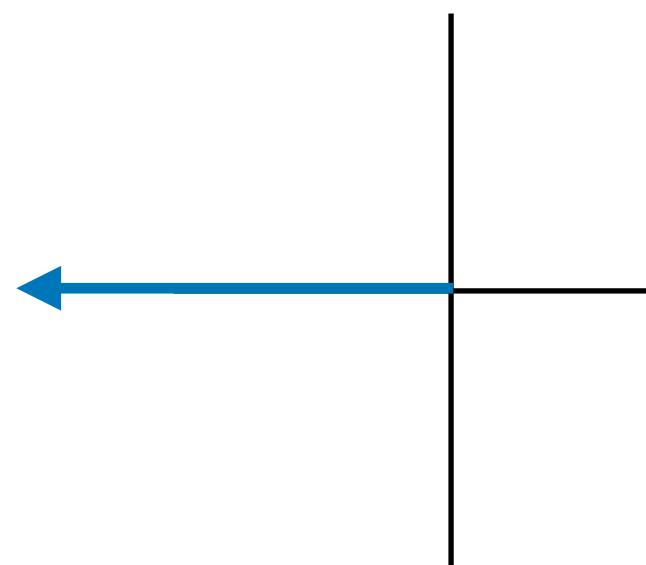
Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Incompatible activation function: **point-wise non-linearity for non-regular types**

$$\rho(\mathbf{R}_\pi) \text{ReLU}\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}\right) = \rho(\mathbf{R}_\pi) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

$$\text{ReLU}\left(\rho(\mathbf{R}_\pi) \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right) = \text{ReLU}\left(\begin{pmatrix} -1 \\ 0 \end{pmatrix}\right) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$



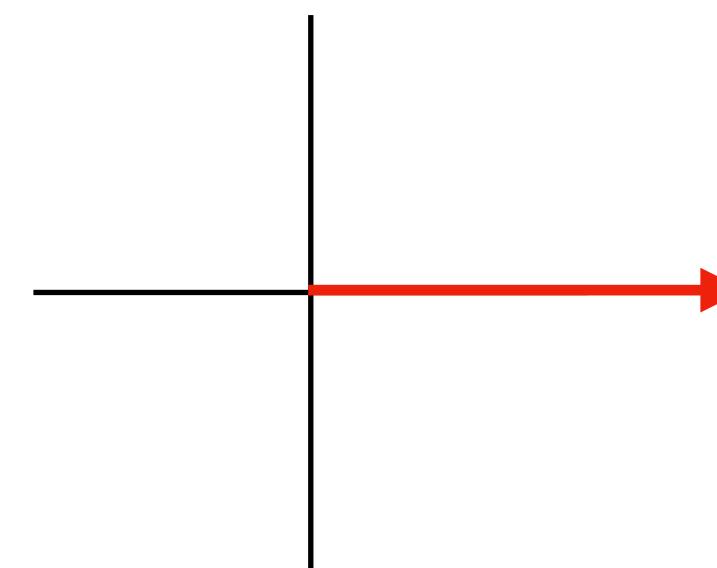
# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Compatible activation function: **norm-based activation functions**

$$\rho(\mathbf{R}_\pi) \sigma(\|\hat{f}(\mathbf{x})\|) \hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$



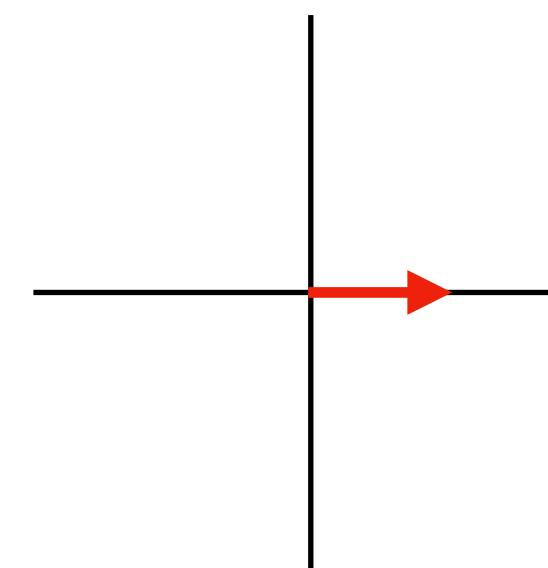
# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Compatible activation function: **norm-based activation functions**

$$\rho(\mathbf{R}_\pi) \sigma(\|\hat{f}(\mathbf{x})\|) \hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$



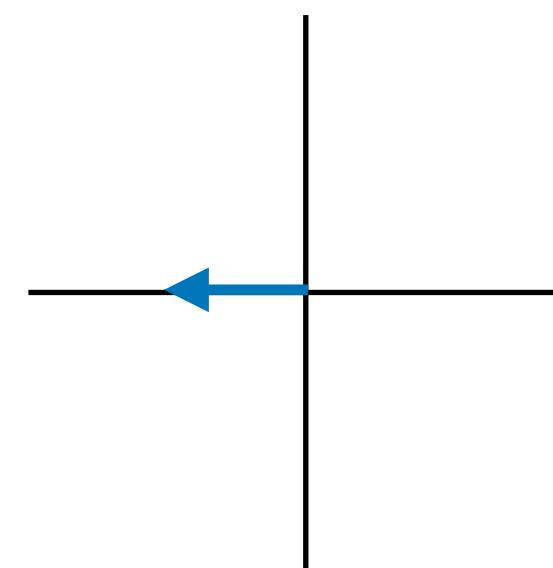
# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Compatible activation function: **norm-based activation functions**

$$\rho(\mathbf{R}_\pi) \sigma(\|\hat{f}(\mathbf{x})\|) \hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$



# Activation functions

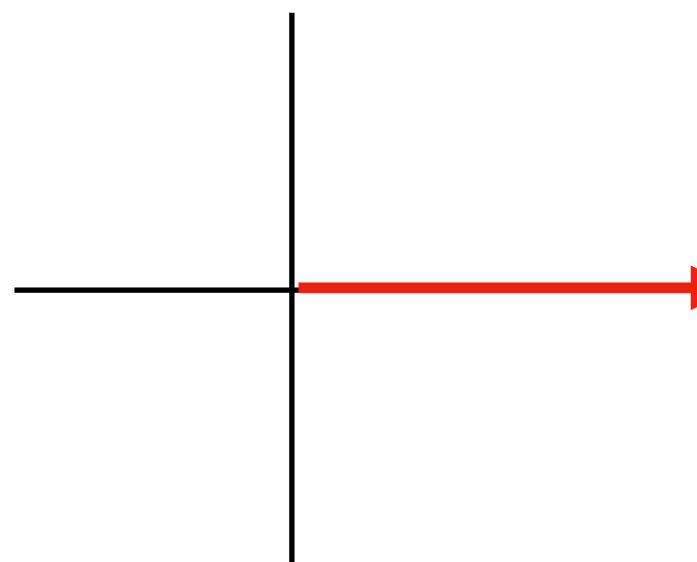
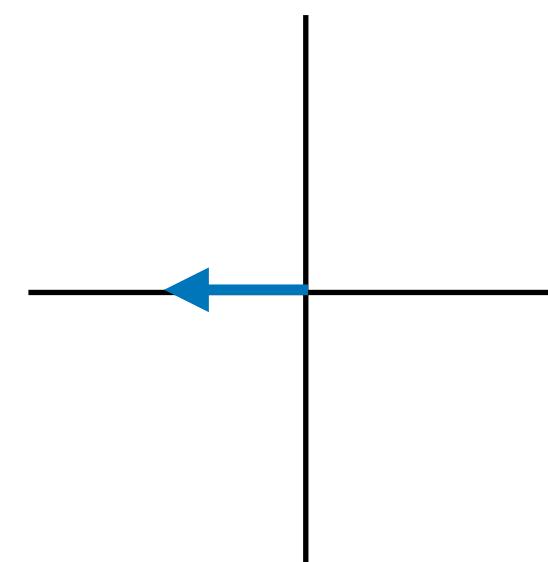
Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Compatible activation function: **norm-based activation functions**

$$\rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

$$\sigma(\|\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x})\|)\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x}) = \rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$



# Activation functions

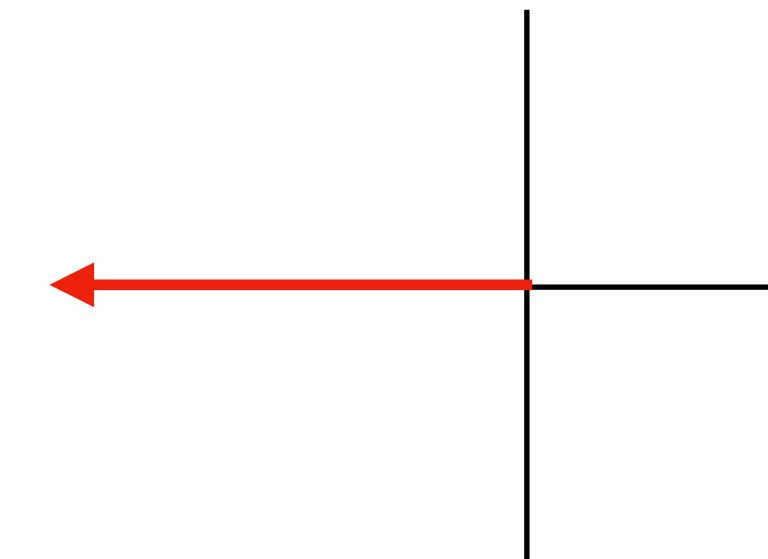
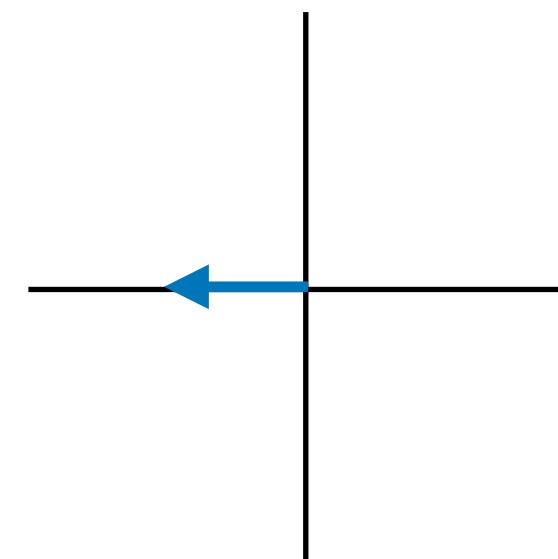
Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Compatible activation function: **norm-based activation functions**

$$\rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

$$\sigma(\|\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x})\|)\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x}) = \rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$



# Activation functions

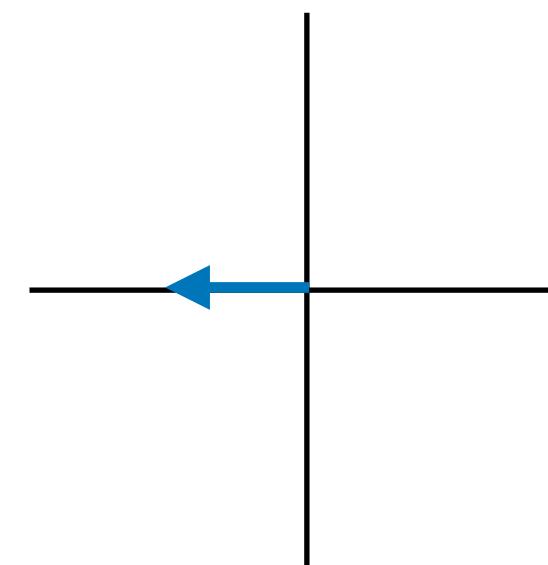
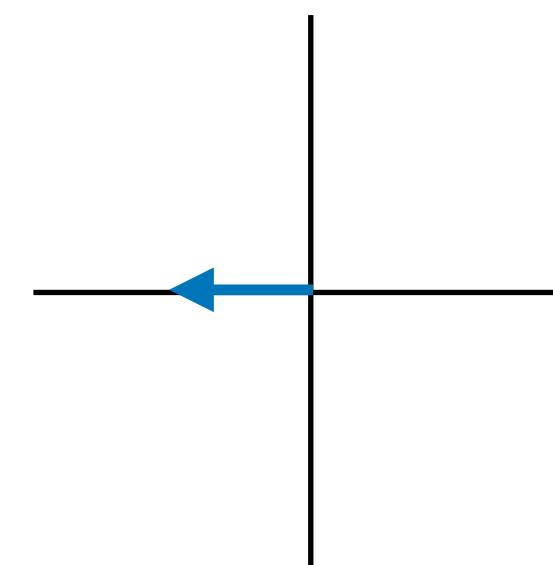
Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Compatible activation function: **norm-based activation functions**

$$\rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

$$\sigma(\|\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x})\|)\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x}) = \rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$



# Activation functions

Activation functions should commute with the representation of

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

## Compatible activation function: norm-based activation functions

$$\rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

$$\sigma(\|\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x})\|)\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x})$$

## Harmonic Networks: Deep Translation and Rotation Equivariance

Daniel E. Worrall, Stephan J. Garbin, Dariyar Turmukhambetov and Gabriel J. Brostow

{d.worrall, s.garbin, d.turmukhambetov, g.brostow}@cs.ucl.ac.uk

University College London\*

### Abstract

Translating or rotating an input image should not affect the results of many computer vision tasks. Convolutional neural networks (CNNs) are already translation equivariant: input image translations produce proportionate feature map translations. This is not the case for rotations. Global rotation equivariance is typically sought through data augmentation, but patch-wise equivariance is more difficult. We present Harmonic Networks or H-Nets, a CNN exhibiting equivariance to patch-wise translation and 360°-rotation. We achieve this by replacing regular CNN filters with circular harmonics, returning a maximal response and orientation for every receptive field patch.

H-Nets use a rich, parameter-efficient and fixed computational complexity representation, and we show that deep feature maps within the network encode complicated rotational invariants. We demonstrate that our layers are general enough to be used in conjunction with the latest architectures and techniques, such as deep supervision and batch normalization. We also achieve state-of-the-art classification on rotated-MNIST, and competitive results on other benchmark challenges.

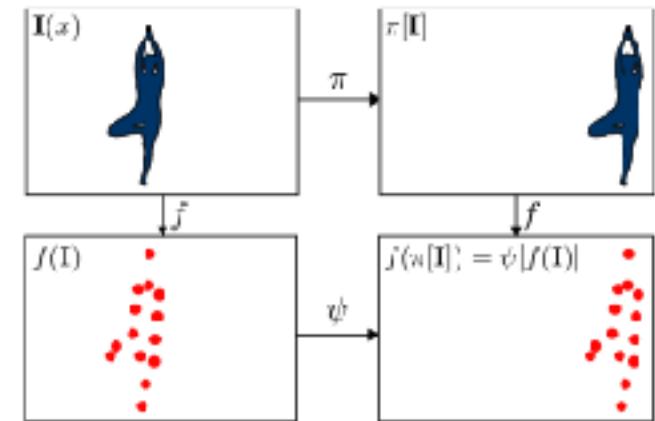


Figure 1. Patch-wise translation equivariance in CNNs arises from translational weight tying, so that a translation  $\pi$  of the input image  $\mathbf{I}$  leads to a corresponding translation  $\psi$  of the feature maps  $f(\mathbf{I})$ , where  $\pi \neq \psi$  in general, due to pooling effects. However, for rotations, CNNs do not yet have a feature space transformation  $\psi$  ‘hard-baked’ into their structure, and it is complicated to discover what  $\psi$  may be, if it exists at all. Harmonic Networks have a hard-baked representation, which allows for easier interpretation of feature maps—see Figure 3.

### 1. Introduction

We tackle the challenge of representing 360°-rotations in convolutional neural networks (CNNs) [19]. Currently, convolutional layers are constrained by design to map an image to a feature vector and *translated* versions of the image map to proportionally-translated versions of the same feature vector [21] (ignoring edge effects)—see Figure 1. However, until now, if one *rotates* the CNN input, then the feature vectors do not necessarily rotate in a meaningful or easy to predict manner. The sought-after property, directly relating input transformations to feature vector transformations, is called *equivariance*.

A special case of equivariance is invariance, where feature vectors remain constant under all transformations of the input. This can be a desirable property globally for a model, such as a classifier, but we should be careful not to restrict all intermediate levels of processing to be transformation invariant. For example,

\*<http://visual.cs.ucl.ac.uk/pubs/harmonicNets/>

consider detecting a deformable object, such as a butterfly. The pose of the wings is limited in range, and so there are only certain poses our detector should normally see. A transformation invariant detector, good at detecting wings, would detect them whether they were bigger, further apart, rotated, etc., and it would encode all these cases with the same representation. It would fail to notice nonsense situations, however, such as a butterfly with wings rotated past the usual range, because it has thrown that extra pose information away. An equivariant detector, on the other hand, does not dispose of local pose information, and so it ends on a richer and more useful representation to downstream processes. Equivariance conveys more information about an input to downstream processes: it also constrains the space of possible learned models to those that are valid under the rules of natural image formation [30]. This makes learning more reliable and helps with generalization. For instance, consider CNNs. The key insight is that the statistics of natural images, embodied in the correlations between pixels, are a) invariant to translation, and b) highly localized. Thus features at every layer in a CNN are computed on local receptive fields, where weights are shared



# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Compatible activation function: **norm-based activation functions**

$$\rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

$$\sigma(\|\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x})\|)\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x}) = \rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

Compatible activation function: **gated non-linearities**

$$\rho(\mathbf{R}_\pi)\sigma(f_0(\mathbf{x}))\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

$$\sigma(\rho_0(\mathbf{R}_\pi)f_0(\mathbf{x}))\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x}) = \rho(\mathbf{R}_\pi)\sigma(f_0(\mathbf{x}))\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

*using predicted scalar fields (type-0)*

# Activation functions

Activation functions should commute with the representation change

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Compatible activation function: **norm-based activation functions**

$$\rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

$$\sigma(\|\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x})\|)\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

Compatible activation function: **gated non-linearities**

$$\rho(\mathbf{R}_\pi)\sigma(f_0(\mathbf{x}))\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

$$\sigma(\rho_0(\mathbf{R}_\pi)f_0(\mathbf{x}))\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

using predicted scalar fields (type-0)

## 3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data

Maurice Weiler\*  
University of Amsterdam  
m.weiler@uva.nl

Mario Geiger\*  
EPFL  
mario.geiger@epfl.ch

Max Welling  
University of Amsterdam, CIFAR,  
Qualcomm AI Research  
m.welling@uva.nl

Wouter Boomsma  
University of Copenhagen  
wb@di.ku.dk

Taco Cohen  
Qualcomm AI Research  
taco.cohen@gmail.com

### Abstract

We present a convolutional network that is equivariant to rigid body motions. The model uses scalar-, vector-, and tensor fields over 3D Euclidean space to represent data, and equivariant convolutions to map between such representations. These SE(3)-equivariant convolutions utilize kernels which are parameterized as a linear combination of a complete steerable kernel basis, which is derived analytically in this paper. We prove that equivariant convolutions are the most general equivariant linear maps between fields over  $\mathbb{R}^3$ . Our experimental results confirm the effectiveness of 3D Steerable CNNs for the problem of amino acid propensity prediction and protein structure classification, both of which have inherent SE(3) symmetry.

### 1 Introduction

Increasingly, machine learning techniques are being applied in the natural sciences. Many problems in this domain, such as the analysis of protein structure, exhibit exact or approximate symmetries. It has long been understood that the equations that define a model or natural law should respect the symmetries of the system under study, and that knowledge of symmetries provides a powerful constraint on the space of admissible models. Indeed, in theoretical physics, this idea is enshrined as a fundamental principle, known as Einstein's principle of general covariance. Machine learning, which is, like physics, concerned with the induction of predictive models, is no different: our models must respect known symmetries in order to predict physically meaningful results.

A lot of recent work, reviewed in Sec. 2, has focused on the problem of developing equivariant networks, which respect some known symmetry. In this paper, we develop the theory of SE(3)-equivariant networks. This is far from trivial, because  $SE(3)$  is both non-commutative and non-compact. Nevertheless, all that is required to make a 3D convolution equivariant using our method, is to parameterize the convolution kernel as a linear combination of pre-computed steerable basis kernels. Hence, the 3D Steerable CNN incorporates equivariance to symmetry transformations without deviating far from current engineering best practices.

The architectures presented here fall within the framework of Steerable G-CNNs [8, 10, 41, 45], which represent their input as fields over a homogeneous space ( $\mathbb{R}^3$  in this case), and use steerable

\* Equal Contribution. MG initiated the project, derived the kernel space constraint, wrote the first network implementation and ran the Shrec17 experiment. MW solved the kernel constraint analytically, designed the anti-aliased kernel sampling in discrete space and coded / ran many of the CATH experiments.

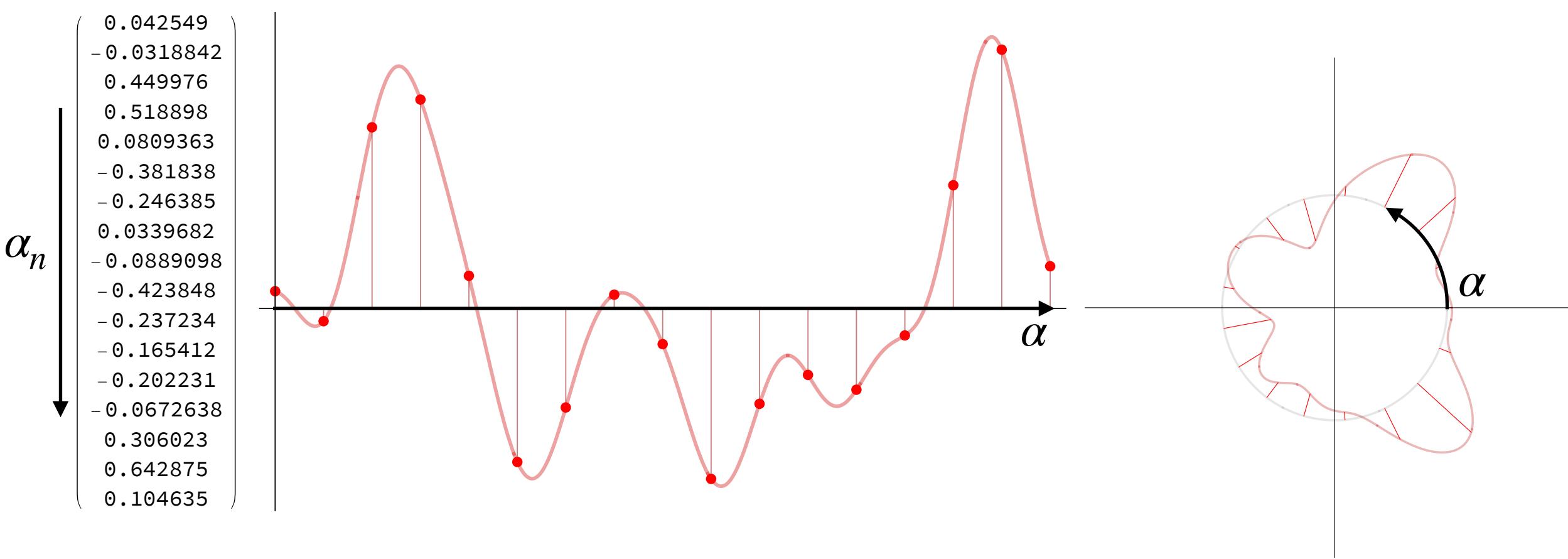
Source code is available at <https://github.com/mariogeiger/se3cnn>

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$



Compatible activation function: any **element-wise activation for regular representations or scalar fields**

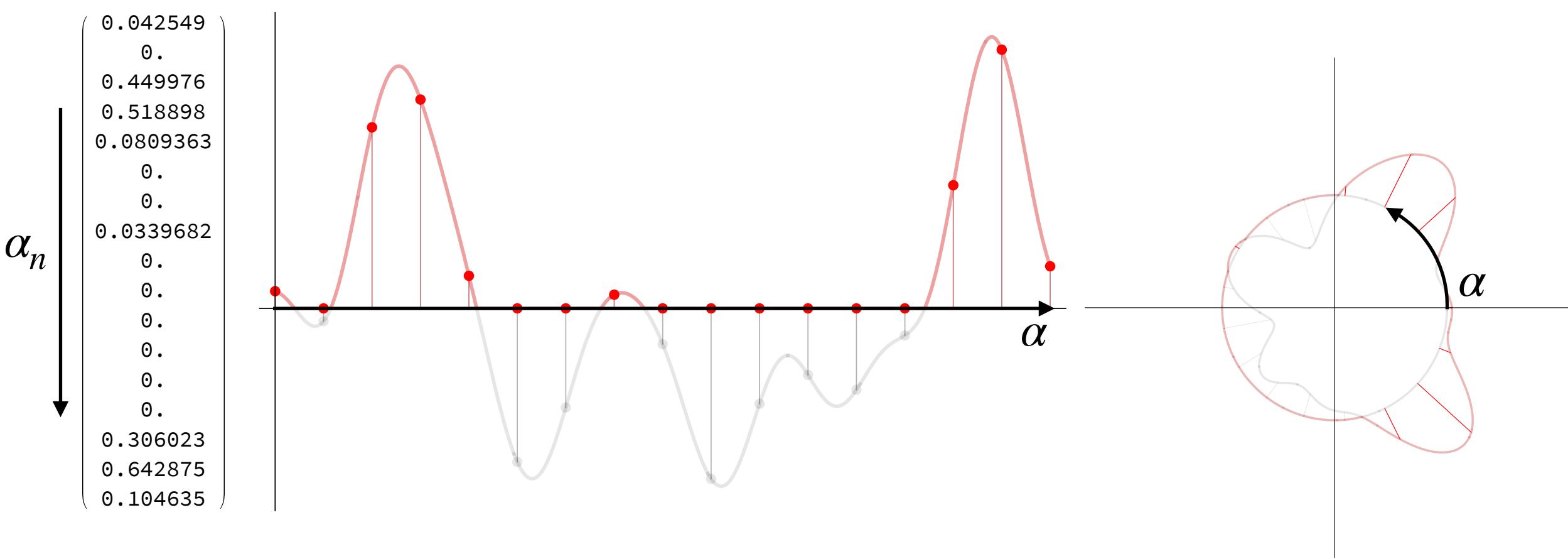
$$\mathcal{L}_\theta \sigma(f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

*element-wise activations commute with permutations*

# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$



Compatible activation function: any **element-wise activation for regular representations or scalar fields**

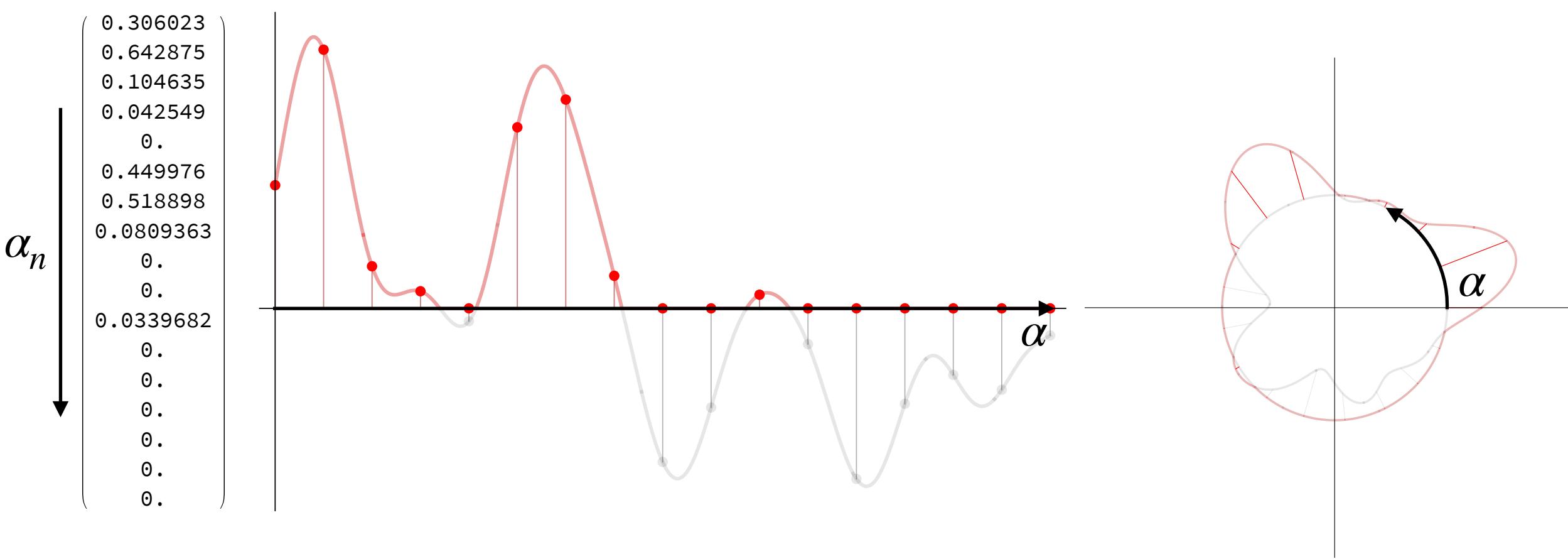
$$\mathcal{L}_\theta \sigma(f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

*element-wise activations commute with permutations*

# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$



Compatible activation function: any **element-wise activation for regular representations or scalar fields**

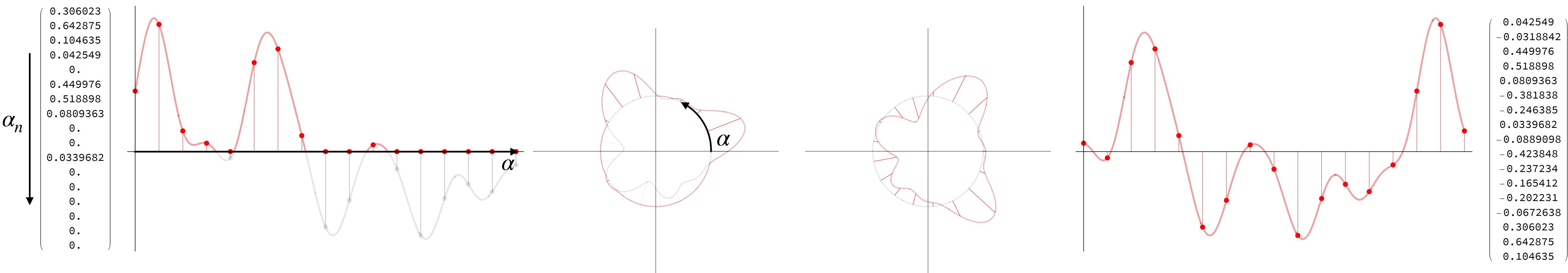
$$\mathcal{L}_\theta \sigma(f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

*element-wise activations commute with permutations*

# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$



Compatible activation function: any **element-wise activation for regular representations or scalar fields**

$$\mathcal{L}_\theta \sigma(f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

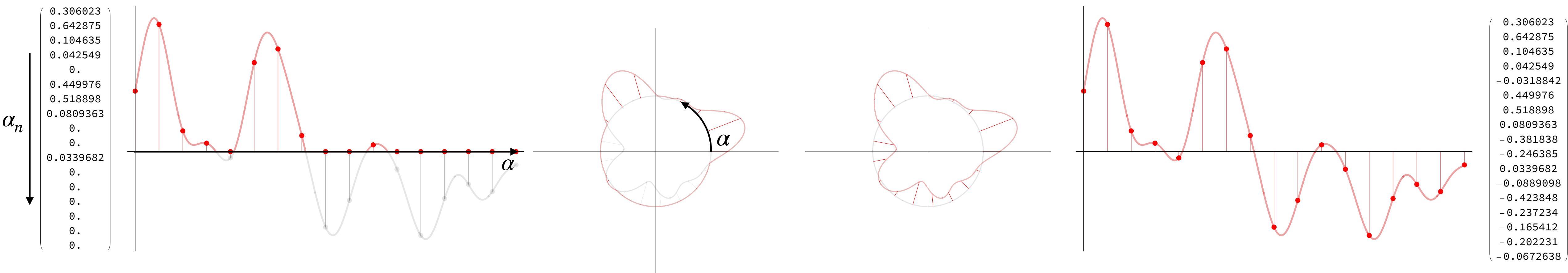
$$\sigma(\mathcal{L}_\theta f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

*element-wise activations commute with permutations*

# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$



Compatible activation function: any **element-wise activation for regular representations or scalar fields**

$$\mathcal{L}_\theta \sigma(f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

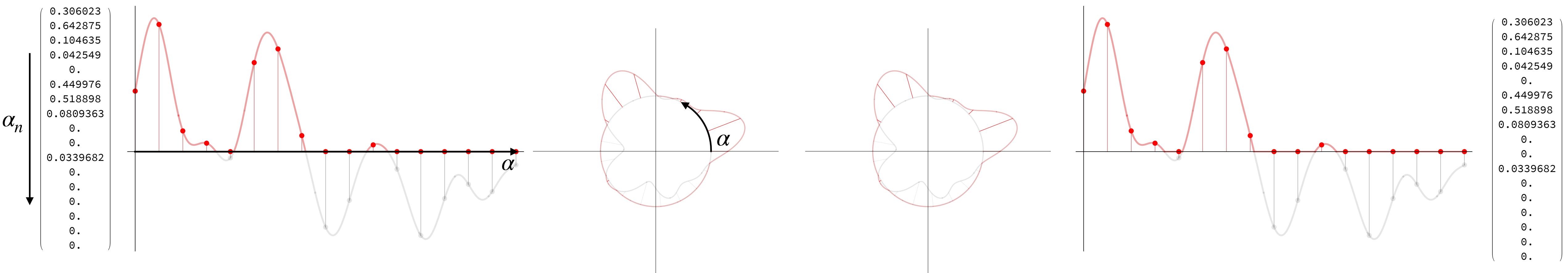
$$\sigma(\mathcal{L}_\theta f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

*element-wise activations commute with permutations*

# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$



Compatible activation function: any **element-wise activation for regular representations or scalar fields**

$$\mathcal{L}_\theta \sigma(f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

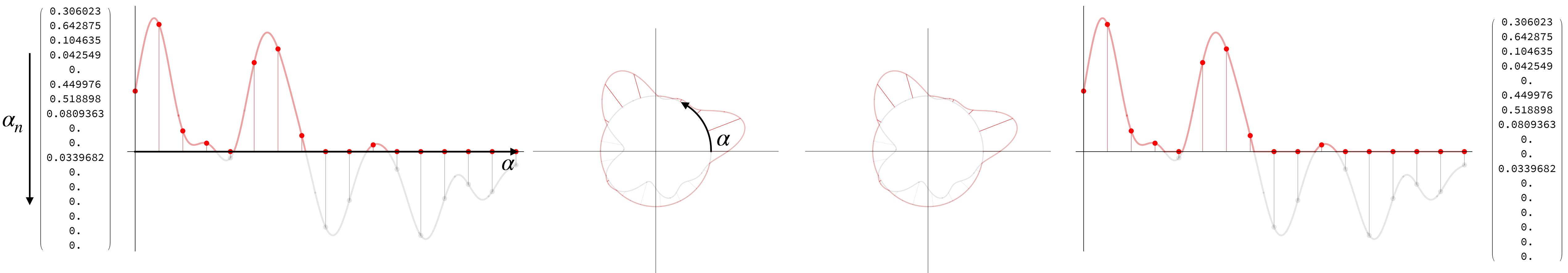
$$\sigma(\mathcal{L}_\theta f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

*element-wise activations commute with permutations*

# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$



Compatible activation function: any **element-wise activation** for **regular representations or scalar fields** **Fourier-based** ( $\mathcal{F}_H \sigma(\mathcal{F}_H^{-1} \hat{f})$ )

$$\mathcal{L}_\theta \sigma(f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

$$\sigma(\mathcal{L}_\theta f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

*element-wise activations commute with permutations*

# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Compatible activation function: **norm-based activation functions**

$$\rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

$$\sigma(\|\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x})\|)\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x}) = \rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

Compatible activation function: **gated non-linearities**

$$\rho(\mathbf{R}_\pi)\sigma(f_0(\mathbf{x}))\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

$$\sigma(\rho_0(\mathbf{R}_\pi)f_0(\mathbf{x}))\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x}) = \rho(\mathbf{R}_\pi)\sigma(f_0(\mathbf{x}))\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

*using predicted scalar fields (type-0)*

Compatible activation function: any **element-wise activation for regular representations or scalar fields** **Fourier-based** ( $\mathcal{F}_H\sigma(\mathcal{F}_H^{-1}\hat{f})$ )

$$\mathcal{L}_\theta\sigma(f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

$$\sigma(\mathcal{L}_\theta f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

Compatible activation function: **tensor product activations** (equivariant polynomials)

# *N*-BODY NETWORKS: A COVARIANT HIERARCHICAL NEURAL NETWORK ARCHITECTURE FOR LEARNING ATOMIC POTENTIALS<sup>1</sup>

Risi Kondor  
Department of Computer Science & Statistics  
The University of Chicago  
rk18@cs.uchicago.edu

## ABSTRACT

We describe *N*-body networks, a neural network architecture for learning the behavior and properties of complex many body physical systems. Our specific application is to learn atomic potential energy surfaces for use in molecular dynamics simulations. Our architecture is novel in that (a) it is based on a hierarchical decomposition of the many body system into subsystems (b) the activations of the network correspond to covariants of the system. The covariants are constant under rotations and are realized as linear combinations of the network's weights. The covariants are covariant under the action of the group of rigid motions of the system.

## 1. INTRODUCTION

In principle, quantum mechanics provides a complete description of atomic systems. However, for a few dozen atoms, the computational cost of a feasible propagation is prohibitive. Therefore, approximate methods are used.

Consequently, the field of molecular dynamics has developed a variety of approximate, and often empirical, methods. One of the most common approaches is the so-called (effective) pair potential method, which consists of a sum of pair potentials, with  $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$  being the distance between atoms  $i$  and  $j$ , and  $\phi(r_{ij})$  being the pair potential. The total potential is given by  $V(\mathbf{r}) = \sum_{i,j} \phi(r_{ij})$ . The pair potential is often approximated by a Lennard-Jones potential,  $\phi(r) = 4\epsilon \left( \left( \frac{\sigma}{r} \right)^{12} - \left( \frac{\sigma}{r} \right)^6 \right)$ , where  $\epsilon$  and  $\sigma$  are constants. The potential is zero for  $r \gg \sigma$  and increases rapidly as  $r \rightarrow 0$ .

Empirical potential functions have been used for many years, but the field has only recently entered this field. The aggregate function is a function that takes a small number of inputs and produces a veritable explosion of outputs. For example, the aggregate function for a molecular dynamics simulation is a function that takes a small number of inputs and produces a veritable explosion of outputs.

<sup>1</sup>This note describes a neural network architecture for learning atomic potentials. It was presented at the International Conference on Machine Learning (ICML) in 2018. The code is available at <https://github.com/rkondor/nbodynet>.

and  $C_{\ell_1, \ell_2, \ell}$  is the part of  $C_{\ell_1, \ell_2}$  matrix corresponding to the  $\ell$  in  $\text{V}(\mathbf{r})$ . Thus, in this case the operator  $T_1^\ell$  just corresponds to multiplying the tensor product by  $C_{\ell_1, \ell_2, \ell}$ . By linearity, the above relationship also extends to non-irreducible vectors. If  $\psi_1$  is of type  $\tau_1$  and  $\psi_2$  is of type  $\tau_2$ , then

$$\psi_1 \otimes \psi_2 = \bigoplus_{\ell} \bigoplus_{m=1}^{\kappa_{\tau_1, \tau_2}(\ell)} \bar{\psi}_m^\ell$$

where

$$\kappa_{\tau_1, \tau_2}(\ell) = \sum_{\ell_1} \sum_{\ell_2} [\tau_1]_{\ell_1} \cdot [\tau_2]_{\ell_2} \cdot \mathbb{I}[\ell_1 - \ell_2 \leq \ell \leq \ell_1 + \ell_2],$$

and  $\mathbb{I}[\cdot]$  is the indicator function. Once again, the actual  $\bar{\psi}_m^\ell$  fragments are computed by applying the appropriate  $C_{\ell_1, \ell_2, \ell}$  matrix to the appropriate combination of irreducible fragments of  $\psi_1$  and  $\psi_2$ . It is also clear that the by applying the Clebsch–Gordan decomposition recursively, we can decompose a tensor product of any order, e.g.,

$$\psi_1 \otimes \psi_2 \otimes \psi_3 \otimes \dots \otimes \psi_k = ((\psi_1 \otimes \psi_2) \otimes \psi_3) \otimes \dots \otimes \psi_k.$$

In an actual computation of such higher order products, however, a considerable amount of thought might have to go into optimizing the order of operations and reusing potential intermediate results to minimize computational cost.

## Point-wise activation for regular representations or scalar fields

$$\sigma_\theta \sigma(\mathcal{J}(\mathbf{x}, \alpha)) = \sigma(\mathcal{J}(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

$$\sigma(\mathcal{L}_\theta f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

## Fourier-based

ions should commute with the representation of the fibers

$$\sigma(\rho(g)\hat{f}(\mathbf{x})) = \rho'(g)\sigma(\hat{f}(\mathbf{x}))$$

## Compatible activation function: tensor product activations (equivariant polynomials)

# *N*-BODY NETWORKS: A COVARIANT HIERARCHICAL NEURAL NETWORK ARCHITECTURE FOR LEARNING ATOMIC POTENTIALS<sup>1</sup>

Risi Kondor  
Department of Computer Science & Statistics  
The University of Chicago  
[rk12@cs.uchicago.edu](mailto:rk12@cs.uchicago.edu)

## ABSTRACT

We describe *N*-body networks, a neural network architecture for learning the behavior and properties of complex many body physical systems. Our specific application is to learn atomic potential energy surfaces for use in molecular dynamics simulations. Our architecture is novel in that (a) it is based on a hierarchical decomposition of the many body system into subsystems (b) the activations of the network correspond to covariants of the system, and (c) the network is covariant with respect to rotations. We also show that the network is realizable as a fully connected feed-forward network. The activations of the network are covariant with respect to rotations, and the network is realizable as a fully connected feed-forward network. The activations of the network are covariant with respect to rotations, and the network is realizable as a fully connected feed-forward network.

## 1. INTRODUCTION

In principle, quantum mechanics provides a complete description of atomic systems. However, for a few dozen atoms, the computational cost of a feasible propagation is prohibitive. Consequently, the theory is often approximated, and the resulting approximation, called (effective) potential energy, is used to approximate the system. Consequently, the theory is often approximated, and the resulting approximation, called (effective) potential energy, is used to approximate the system.

Consequently, the theory is often approximated, and the resulting approximation, called (effective) potential energy, is used to approximate the system. Consequently, the theory is often approximated, and the resulting approximation, called (effective) potential energy, is used to approximate the system. Consequently, the theory is often approximated, and the resulting approximation, called (effective) potential energy, is used to approximate the system.

Consequently, the theory is often approximated, and the resulting approximation, called (effective) potential energy, is used to approximate the system. Consequently, the theory is often approximated, and the resulting approximation, called (effective) potential energy, is used to approximate the system.

and  $C_{\ell_1, \ell_2, \ell}$  is the part of  $C_{\ell_1, \ell_2}$  matrix corresponding to the  $\ell$  in  $\ell_1 \ell_2 \ell$ . Thus, in this case the operator  $T_1^\ell$  just corresponds to multiplying the tensor product by  $C_{\ell_1, \ell_2, \ell}$ . By linearity, the above relationship also extends to non-irreducible vectors. If  $\psi_1$  is of type  $\tau_1$  and  $\psi_2$  is of type  $\tau_2$ , then

$$\psi_1 \otimes \psi_2 = \bigoplus_{\ell} \bigoplus_{m=1}^{\kappa_{\tau_1, \tau_2}(\ell)} \bar{\psi}_m^\ell$$

where

$$\kappa_{\tau_1, \tau_2}(\ell) = \sum_{\ell_1} \sum_{\ell_2} [\tau_1]_{\ell_1} \cdot [\tau_2]_{\ell_2} \cdot \mathbb{I}[\ell_1 - \ell_2 \leq \ell \leq \ell_1 + \ell_2],$$

and  $\mathbb{I}[\cdot]$  is the indicator function. Once again, the actual  $\bar{\psi}_m^\ell$  fragments are computed by applying the appropriate  $C_{\ell_1, \ell_2, \ell}$  matrix to the appropriate combination of irreducible fragments of  $\psi_1$  and  $\psi_2$ . It is also clear that the by applying the Clebsch–Gordan decomposition recursively, we can decompose a tensor product of any order, e.g.,

$$\psi_1 \otimes \psi_2 \otimes \psi_3 \otimes \dots \otimes \psi_k = ((\psi_1 \otimes \psi_2) \otimes \psi_3) \otimes \dots \otimes \psi_k.$$

In an actual computation of such higher order products, however, a considerable amount of thought might have to go into optimizing the order of operations and reusing potential intermediate results to minimize computational cost.

<sup>1</sup>This note describes work done while the author was visiting the Institute for Molecular Engineering at the University of Chicago, Chicago, IL, USA, and while the author was visiting the University of Chicago, Chicago, IL, USA, and while the author was visiting the University of Chicago, Chicago, IL, USA.

# Clebsch–Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network

Risi Kondor<sup>1\*</sup> Zhen Lin<sup>1\*</sup> Shubhendu Trivedi<sup>2\*</sup>  
1The University of Chicago 2Toyota Technological Institute  
[rk12, z.lin@uchicago.edu](mailto:{rk12, z.lin}@uchicago.edu), [shubhendu@ttic.edu](mailto:shubhendu@ttic.edu)

## Abstract

Recent work by Cohen *et al.* [1] has achieved state-of-the-art results for learning spherical images in a rotation invariant way by using ideas from group representation theory and non-commutative harmonic analysis. In this paper we propose a spherical convolutional neural network that generally exhibits improved performance, but with a much simpler and more intuitive view. An unusual feature of this network is that it uses the Clebsch–Gordan transform as its basic building block, thus avoiding repeated forward and backward Fourier transforms. The ideas of the paper generalize to constructing neural networks that are invariant under the action of other compact groups.

In deep learning, we still do not have a satisfactory understanding of how to achieve such spectacular performance on a wide range of tasks. One clear, however, is that certain architectures pick up on natural components of their success. The classic example is of course the convolutional neural network (CNN) for image classification [2]. Recall that, fundamentally, each layer consists of two operations: a linear one consisting of convolving the previous layer with a small learnable filter, and a nonlinear but pointwise one, such as ReLU. It is sufficient to guarantee *translation equivariance*, meaning that if we apply a translation  $\mathbf{t}$  to the input image, then the activation pattern in each higher layer changes by the same amount. Equivariance is crucial to image recognition for two reasons: (a) it guarantees that exactly the same filters are applied to each part of the image. (b) Assuming that finally, at the very top of the network, we have learned an invariant feature, the entire network will be invariant, ensuring that it can recognize objects regardless of its location.

We have shown that examining equivariance from the theoretical point of view, that is, the natural way to generalize convolutional networks through generalizing the notion of equivariance itself to other groups. Letting  $f^g$  denote the activations of the neurons in layer  $g$  of a convolutional neural network, mathematically, equivariance to a group  $G$  means that the activations  $f^g$  are transformed by some transformation  $g \in G$ . That is, the fixed set of linear transformations  $\{T_g\}_{g \in G}$ , so that  $f^g \mapsto T_g f^g$ . Note that in this case, the difference between the two words being only one of interpretation.

multiple channels, and correspondingly multiple filters per layer, but it does not affect the network's invariance properties.

# ions

## of the fibers

$$\mathbf{x} = \rho(\mathbf{R}_\pi) \sigma(\|\hat{f}(\mathbf{x})\|) \hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

$$\mathbf{x} = \rho(\mathbf{R}_\pi) \sigma(f_0(\mathbf{x})) \hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

## s or scalar fields

$$\mathbf{x} = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

## Fourier-based

$$\mathcal{L}_\theta \sigma(J(\mathbf{x}, \alpha)) = \sigma(J(\mathbf{x}, \alpha - \theta)) = J(\mathbf{x}, \alpha - \theta)$$

Compatible activation function: **tensor product activations** (equivariant polynomials)

# $N$ -BODY NETWORKS: A COVARIANT HIERARCHICAL NEURAL NETWORK ARCHITECTURE FOR LEARNING ATOMIC POTENTIALS<sup>1</sup>

Risi Kondor  
Department of Computer Science & Statistics  
The University of Chicago  
[risi@cs.uchicago.edu](mailto:risi@cs.uchicago.edu)

## ABSTRACT

We describe  $N$ -body networks, a neural network architecture for learning the behavior and properties of complex many body physical systems. Our specific application is to learn atomic potential energy surfaces for use in molecular dynamics simulations. Our architecture is novel in that (a) it is based on a hierarchical decomposition of the many body system into subsystems (b) the activations of the network correspond to rotations and are realizable as a sum of covariant parts of the weights, covariant

## 1. INTRODUCTION

In principle, quantum mechanics provides a theory of atomic systems for any number of atoms, but for a few dozen atoms the computational cost is prohibitive. Feasible propositions are based on various approximations.

Consequently, the theory is often used explicitly, and for approximation, various so-called (effective) potentials are used. For example, the potential  $V_{ij} = -\nabla r_{ij} \cdot \phi(r_{ij})$  is a closed form form of the potential of its  $j$ th neighbor. It is also clear that the by applying the Clebsch–Gordan decomposition recursively, we can decompose a tensor product of any order, e.g.,

$$\psi_1 \otimes \psi_2 \otimes \psi_3 \otimes \dots \otimes \psi_k = ((\psi_1 \otimes \psi_2) \otimes \psi_3) \otimes \dots \otimes \psi_k.$$

In an actual computation of such higher order products, however, a considerable amount of thought might have to go into optimizing the order of operations and reusing potential intermediate results to minimize computational cost.

<sup>1</sup>This note describes work done while the author was at the University of Chicago. It was presented at the 2017 NIPS workshop on “Learning Molecules and Materials” in San Diego, CA, on December 5, 2017.

$$\mathcal{L}_\theta \sigma(J(\mathbf{x}, \alpha)) = \sigma(J(\mathbf{x}, \alpha - \theta)) = J(\mathbf{x}, \alpha - \theta)$$

$$\sigma(\mathcal{L}_\theta J(\mathbf{x}, \alpha)) = \sigma(J(\mathbf{x}, \alpha - \theta)) = J(\mathbf{x}, \alpha - \theta)$$

Compatible activation function: **tensor product activations** (equivariant polynomials)

# Clebsch–Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network

Risi Kondor<sup>1\*</sup> Zhen Lin<sup>1\*</sup> Shubhendu Trivedi<sup>2\*</sup>  
1The University of Chicago 2Toyota Technological Institute  
[{risi, z.lin}@uchicago.edu](mailto:{risi, z.lin}@uchicago.edu), [shubhendu@ttic.edu](mailto:shubhendu@ttic.edu)

## Abstract

Recent work by Cohen *et al.* [1] has achieved state-of-the-art results for learning spherical images in a rotation invariant way by using ideas from group representation theory and non-commutative harmonic analysis. In this paper we propose a spherical convolutional neural network that generally exhibits improved performance, but is at the same time much simpler. An unusual feature of this network is that it uses the Clebsch–Gordan transform as its building block, thus avoiding repeated forward and backward Fourier transforms. The ideas of the paper generalize to constructing neural networks that are invariant under the action of other compact groups.

$$\psi_1 \otimes \psi_2 = \bigoplus_{\ell} \bigoplus_{m=1}^{\kappa_{\tau_1, \tau_2}(\ell)} \bar{\psi}_m^\ell$$

where

$$\kappa_{\tau_1, \tau_2}(\ell) = \sum_{\ell_1} \sum_{\ell_2} [\tau_1]_{\ell_1} \cdot [\tau_2]_{\ell_2} \cdot \mathbb{I}[\ell_1 - \ell_2 \leq \ell \leq \ell_1 + \ell_2],$$

and  $\mathbb{I}[\cdot]$  is the indicator function. Once again, the actual  $\bar{\psi}_m^\ell$  fragments are computed by applying the appropriate  $C_{\ell_1, \ell_2, \ell}$  matrix to the appropriate combination of irreducible fragments of  $\psi_1$  and  $\psi_2$ . It is also clear that the by applying the Clebsch–Gordan decomposition recursively, we can decompose a tensor product of any order, e.g.,

In deep learning, we still do not have a satisfactory understanding of how to achieve such spectacular performance on a wide range of tasks. One reason, however, is that certain architectures pick up on natural symmetries to their success. The classic example is of course the AlexNet for image classification [2]. Recall that, fundamentally, each layer consists of two operations: a linear one consisting of convolving the previous (small) learnable filter, and a nonlinear but pointwise one, such as ReLU. It is sufficient to guarantee *translation equivariance*, meaning that if we apply a translation  $t$  to the input, then the activation pattern in each higher layer remains the same. Equivariance is crucial to image recognition for two reasons: (a) it guarantees that exactly the same filters are applied to each part of the image. (b) Assuming that finally, at the very top of the network, we have learned an invariant feature, the entire network will be invariant, ensuring that it can recognize objects regardless of their location.

We have also examined equivariance from the theoretical point of view: the natural way to generalize convolutional networks through generalizing the notion of equivariance itself to other groups. Letting  $f^g$  denote the activations of the neurons in layer  $g$  of a convolutional neural network, mathematically, equivariance to a group  $G$  means that the activations  $f^g$  of the neurons in layer  $g$  of the network are transformed by some transformation  $g \in G$  as if the network were transformed by some fixed set of linear transformations  $\{T_g\}_{g \in G}$ . Note that in this case, the difference between the two words being only one of

multiple channels, and correspondingly multiple filters per layer, but it does not affect the network's invariance properties.

# General Nonlinearities in $SO(2)$ -Equivariant CNNs

Daniel Franzen  
Institute of Computer Science  
Johannes Gutenberg University Mainz  
Staudingerweg 9,  
55122 Mainz, Germany  
[dfranz@uni-mainz.de](mailto:dfranz@uni-mainz.de)

Michael Wand  
Institute of Computer Science  
Johannes Gutenberg University Mainz  
Staudingerweg 9,  
55122 Mainz, Germany  
[wardm@uni-mainz.de](mailto:wardm@uni-mainz.de)

## Abstract

Invariance under symmetry is an important problem in machine learning. Our paper looks specifically at equivariant neural networks where transformations of inputs yield homomorphic transformations of outputs. Here, steerable CNNs have emerged as the standard solution. An inherent problem of steerable representations is that general nonlinear layers break equivariance, thus restricting architectural choices. Our paper applies harmonic distortion analysis to illuminate the effect of nonlinearities on Fourier representations of  $SO(2)$ . We develop a novel FFT-based algorithm for computing representations of non-linearly transformed activations while maintaining band-limitation. It yields exact equivariance for polynomial (approximations of) nonlinearities, as well as approximate solutions with tunable accuracy for general functions. We apply the approach to build a fully  $E(3)$ -equivariant network for sampled 3D surface data. In experiments with 2D and 3D data, we obtain results that compare favorably to the state-of-the-art in terms of accuracy while permitting continuous symmetry and exact equivariance.

## 1 Introduction

Modeling of symmetry in data, i.e., the invariance of properties under classes of transformations, is a cornerstone of machine learning: Invariance of statistical properties over samples is the basis of any form of generalization, and the prior knowledge of additional symmetries can be leveraged for performance gains. Aside from data efficiency prospects, some applications require exact symmetry. For example, in computational physics, symmetry of potentials and force fields is directly linked to conservation laws, and is therefore important for the stability of simulations.

In deep neural networks, (discrete) translational symmetry over space and/or time is exploited in many architectures and is the defining feature of convolutional neural networks (CNNs) and their successors. In most applications, we are typically interested in invariance (e.g., classification remains unchanged) or co-variance (e.g., predicted geometry is transformed along with the input). Formally, this goal is captured under the more general umbrella of equivariance [6]:

Let  $f : X \rightarrow Y$  be a function (e.g., a network layer) that maps between vector spaces  $X, Y$  (e.g., feature maps in a CNN). Let  $G$  be a group and let (in slight abuse of notation)  $g \circ v$  denote the application of the action of group element  $g$  on a vector  $v$ .  $f$  is called *equivariant*, iff:

$$\forall g \in G : f(g \circ v) = h(g) \circ f(v), \quad (1)$$

where  $h : G \rightarrow G'$  is a group homomorphism mapping into a suitable group  $G'$ . Informally speaking, the effect of a transformation on the input should have an effect on the output that has (at least) the same algebraic structure. Invariance ( $h \equiv 1_{G'}$ ) and covariance ( $h = id_{G \rightarrow G'}$ ) are special cases, along with contra-variance and any other isomorphisms of subgroups of  $G$ .

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

# Activation functions

Activation functions should commute with the representation of the fibers

$$\sigma\left(\rho(g)\hat{f}(\mathbf{x})\right) = \rho'(g)\sigma\left(\hat{f}(\mathbf{x})\right)$$

Compatible activation function: **norm-based activation functions**

$$\rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x}) \quad \sigma(\|\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x})\|)\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x}) = \rho(\mathbf{R}_\pi)\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

Compatible activation function: **gated non-linearities**

$$\rho(\mathbf{R}_\pi)\sigma(f_0(\mathbf{x}))\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x}) \quad \sigma(\rho_0(\mathbf{R}_\pi)f_0(\mathbf{x}))\rho(\mathbf{R}_\pi)\hat{f}(\mathbf{x}) = \rho(\mathbf{R}_\pi)\sigma(f_0(\mathbf{x}))\hat{f}(\mathbf{x}) = \hat{f}'(\mathbf{x})$$

*using predicted scalar fields (type-0)*

Compatible activation function: any **element-wise activation for regular representations or scalar fields** **Fourier-based** ( $\mathcal{F}_H\sigma(\mathcal{F}_H^{-1}\hat{f})$ )

$$\mathcal{L}_\theta\sigma(f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta) \quad \sigma(\mathcal{L}_\theta f(\mathbf{x}, \alpha)) = \sigma(f(\mathbf{x}, \alpha - \theta)) = f'(\mathbf{x}, \alpha - \theta)$$

Compatible activation function: **tensor product activations** (equivariant polynomials)