UNIVERSITY OF AMSTERDAM

AMLAB Amsterdam Machine Learning Lab

Group Equivariant Deep Learning

Lecture 2 - Steerable group convolutions

Lecture 2.6 - Activation functions for steerable G-CNNs

Erik Bekkers, Amsterdam Machine Learning Lab, University of Amsterdam

This mini-course serves as a module with the UvA Master AI course Deep Learning 2 https://uvadl2c.github.io/

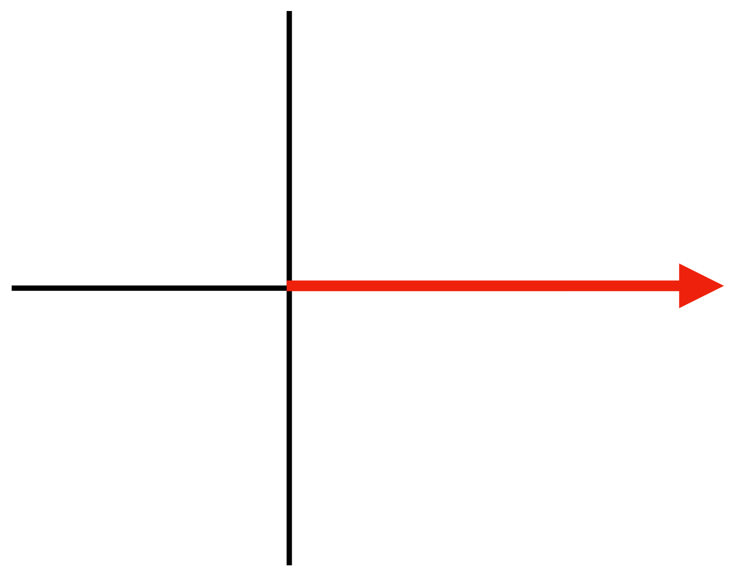
Activation functions

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**

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



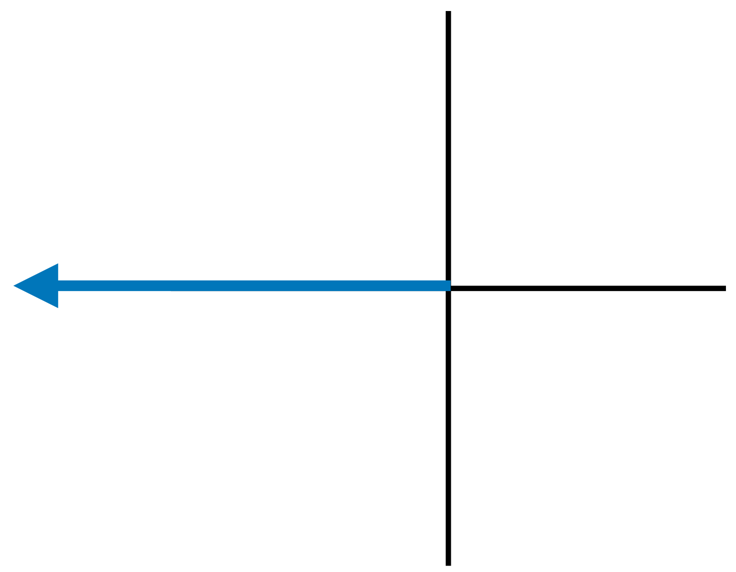
Activation functions

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**

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



Activation functions

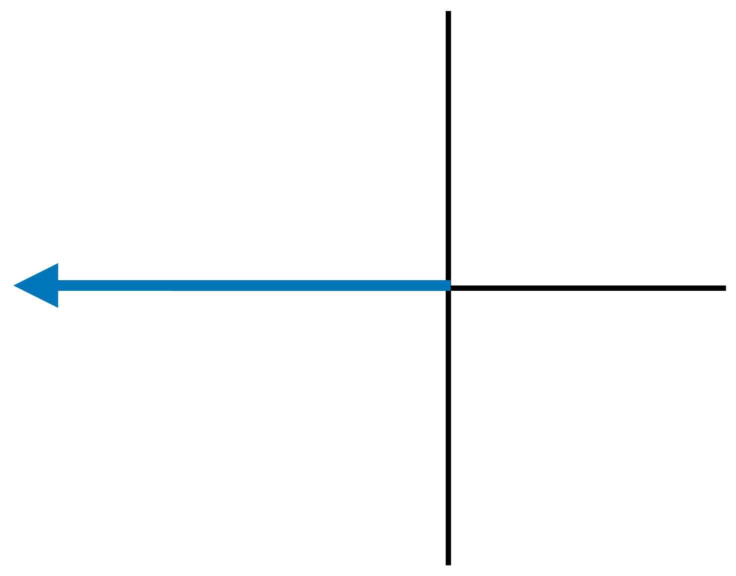
Activation functions should commute with the representation of the fibers

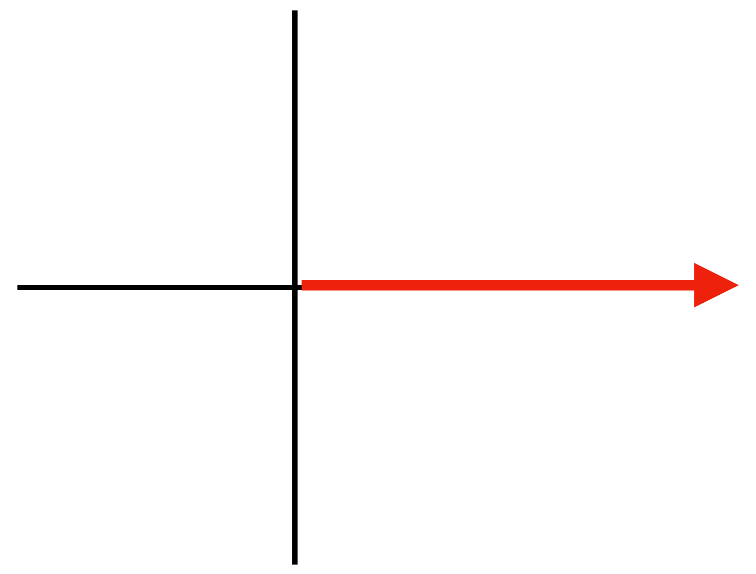
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

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

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

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





Activation functions

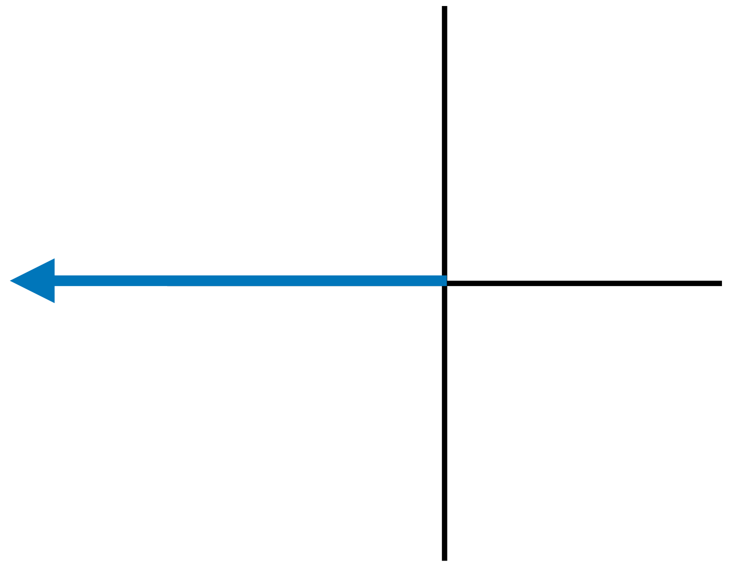
Activation functions should commute with the representation of the fibers

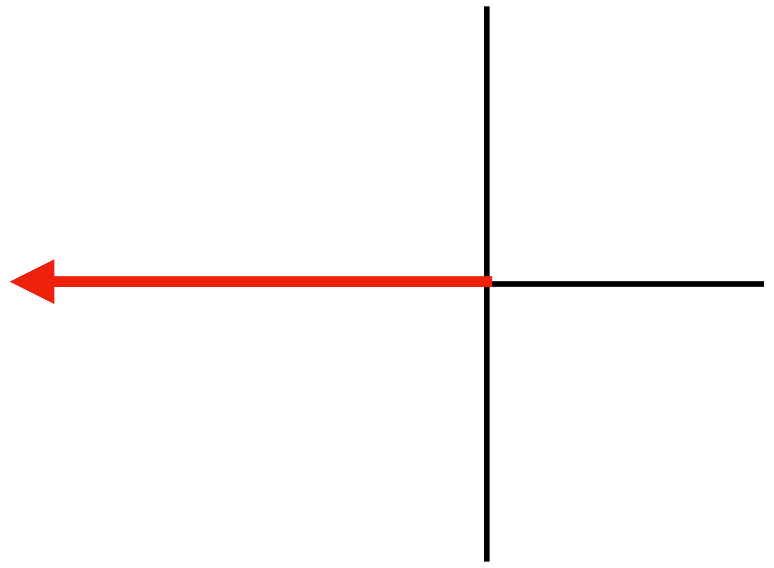
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

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

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

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





Activation functions

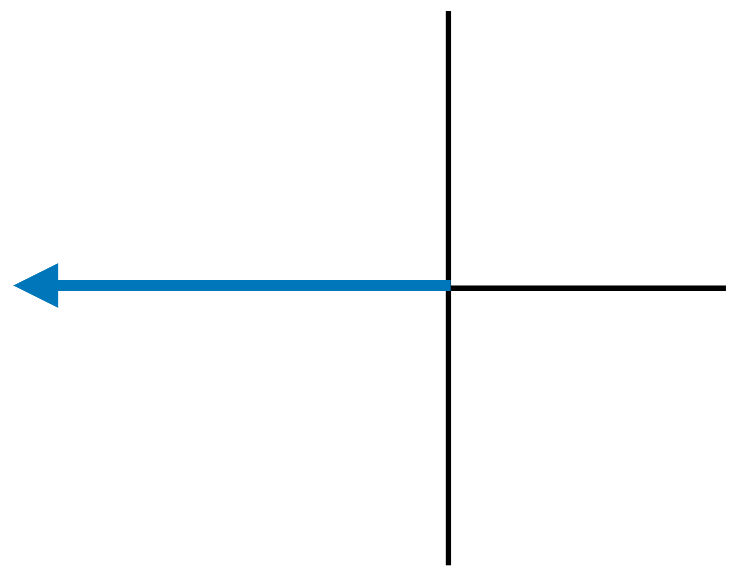
Activation functions should commute with the representation of the fibers

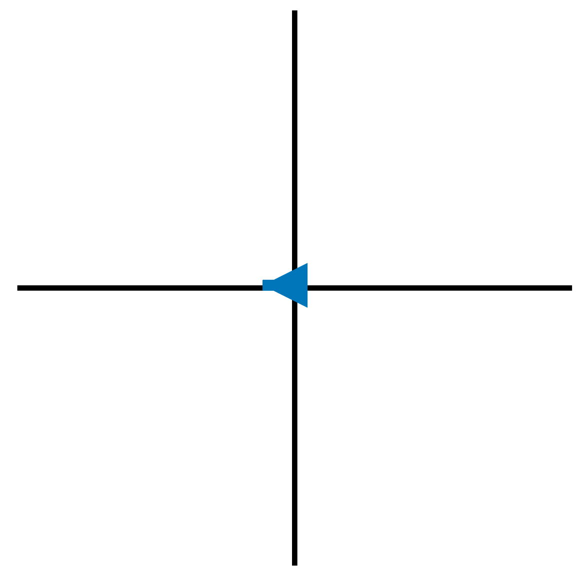
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

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

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

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





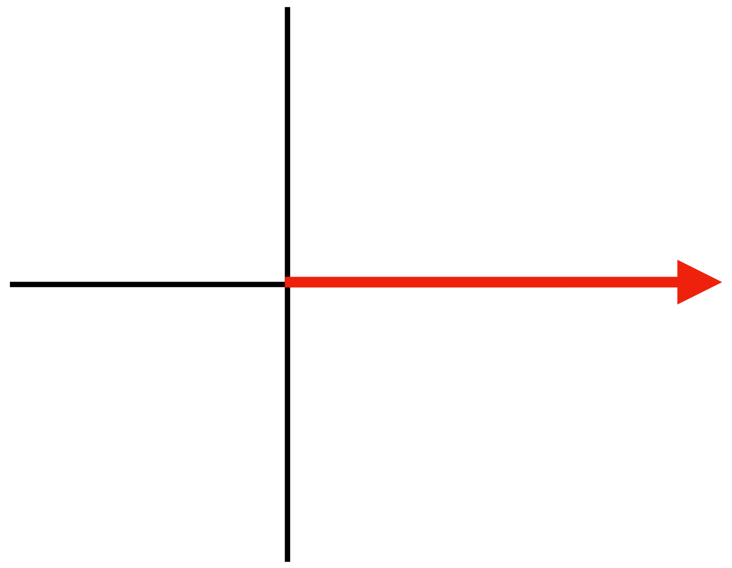
Activation functions

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**

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



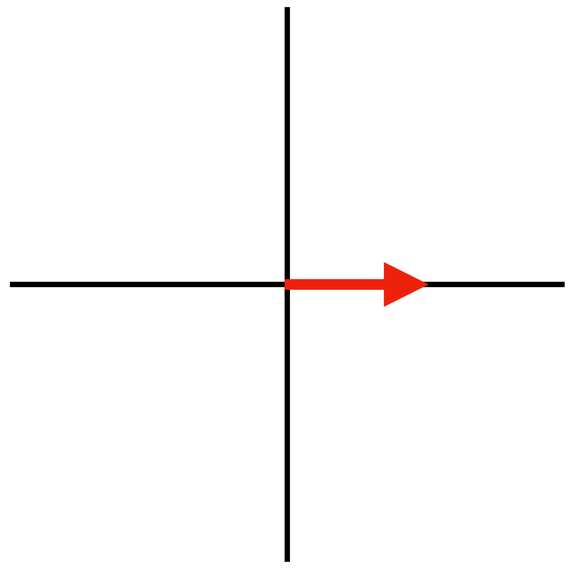
Activation functions

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**

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



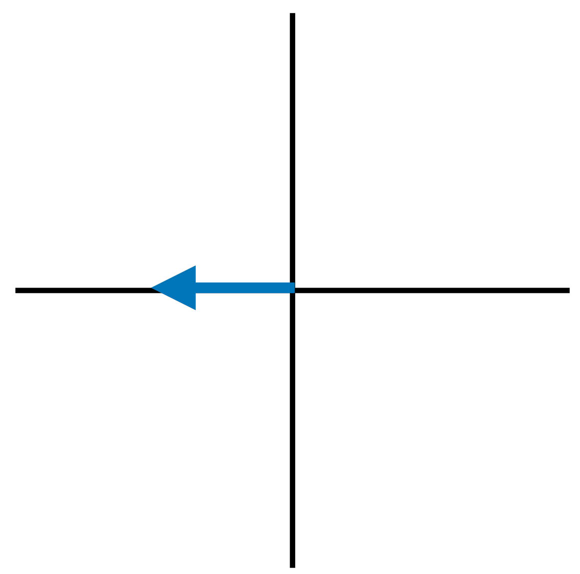
Activation functions

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**

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



Activation functions

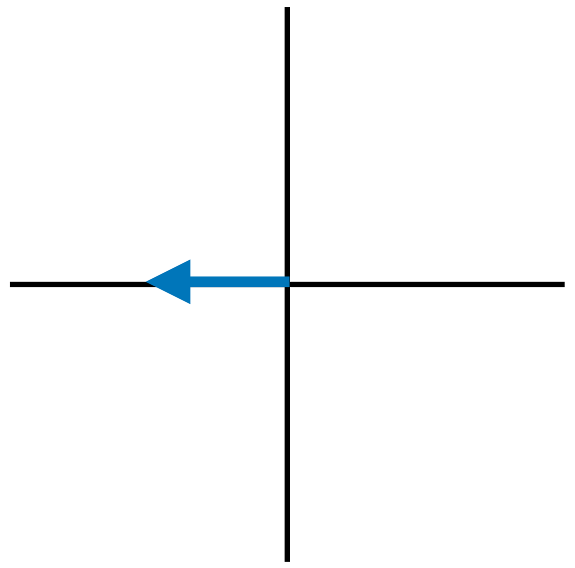
Activation functions should commute with the representation of the fibers

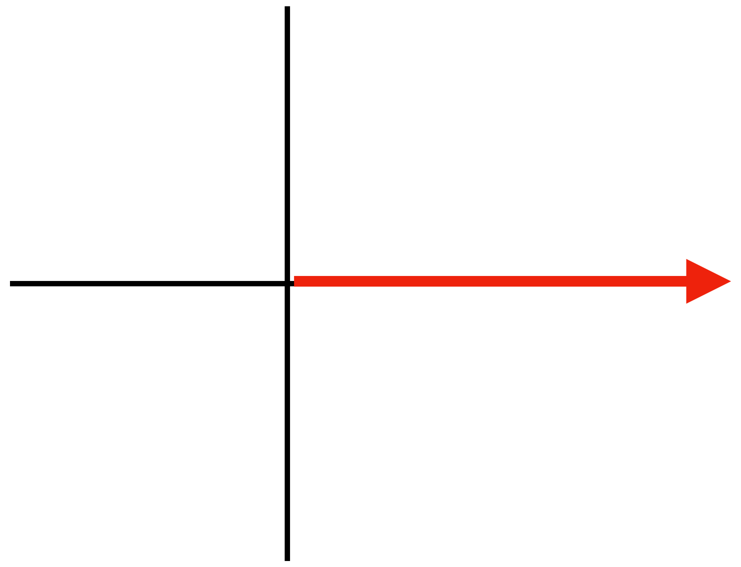
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

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

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

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





Activation functions

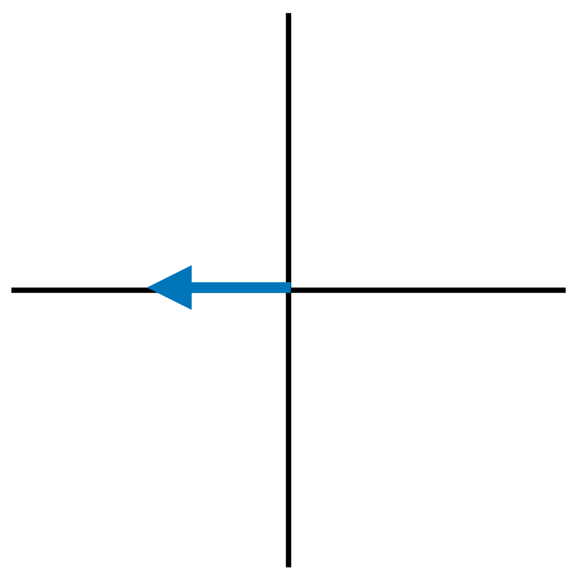
Activation functions should commute with the representation of the fibers

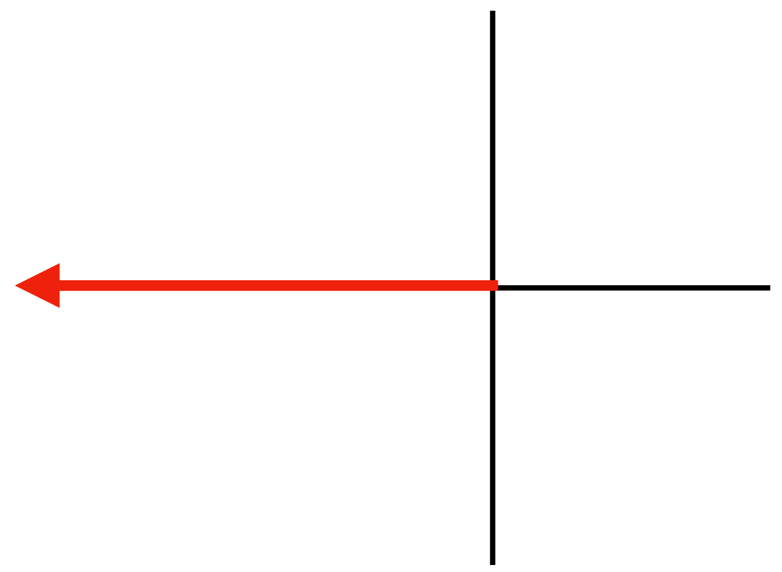
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

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

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

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





Activation functions

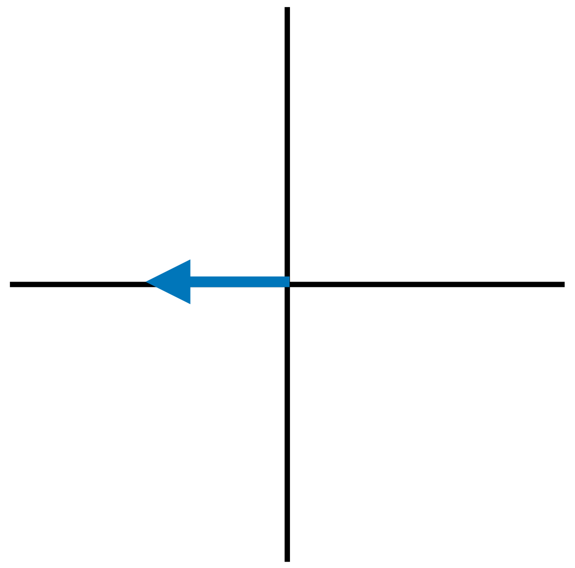
Activation functions should commute with the representation of the fibers

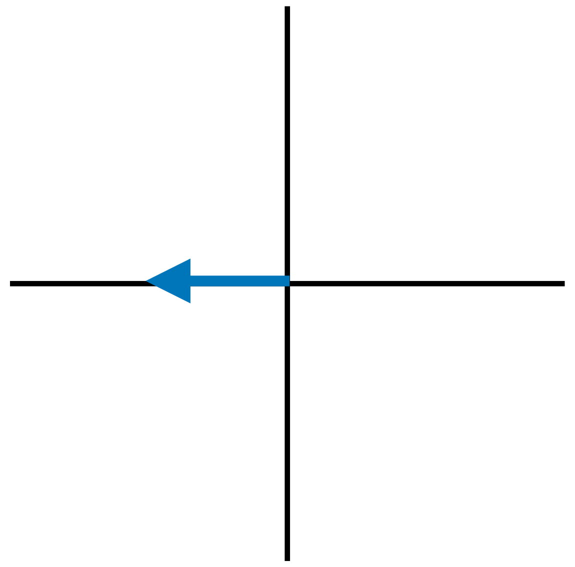
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

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

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

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





Activation functi

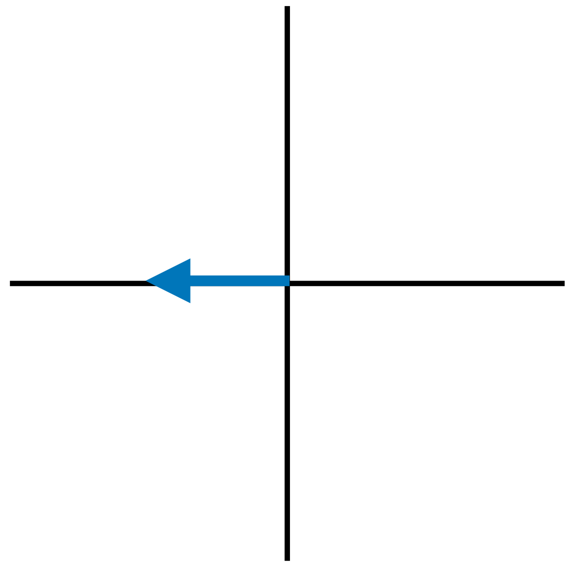
Activation functions should commute with the representation of

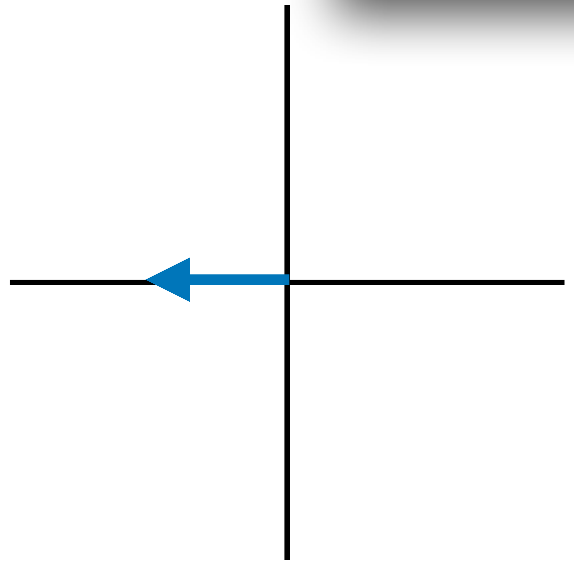
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

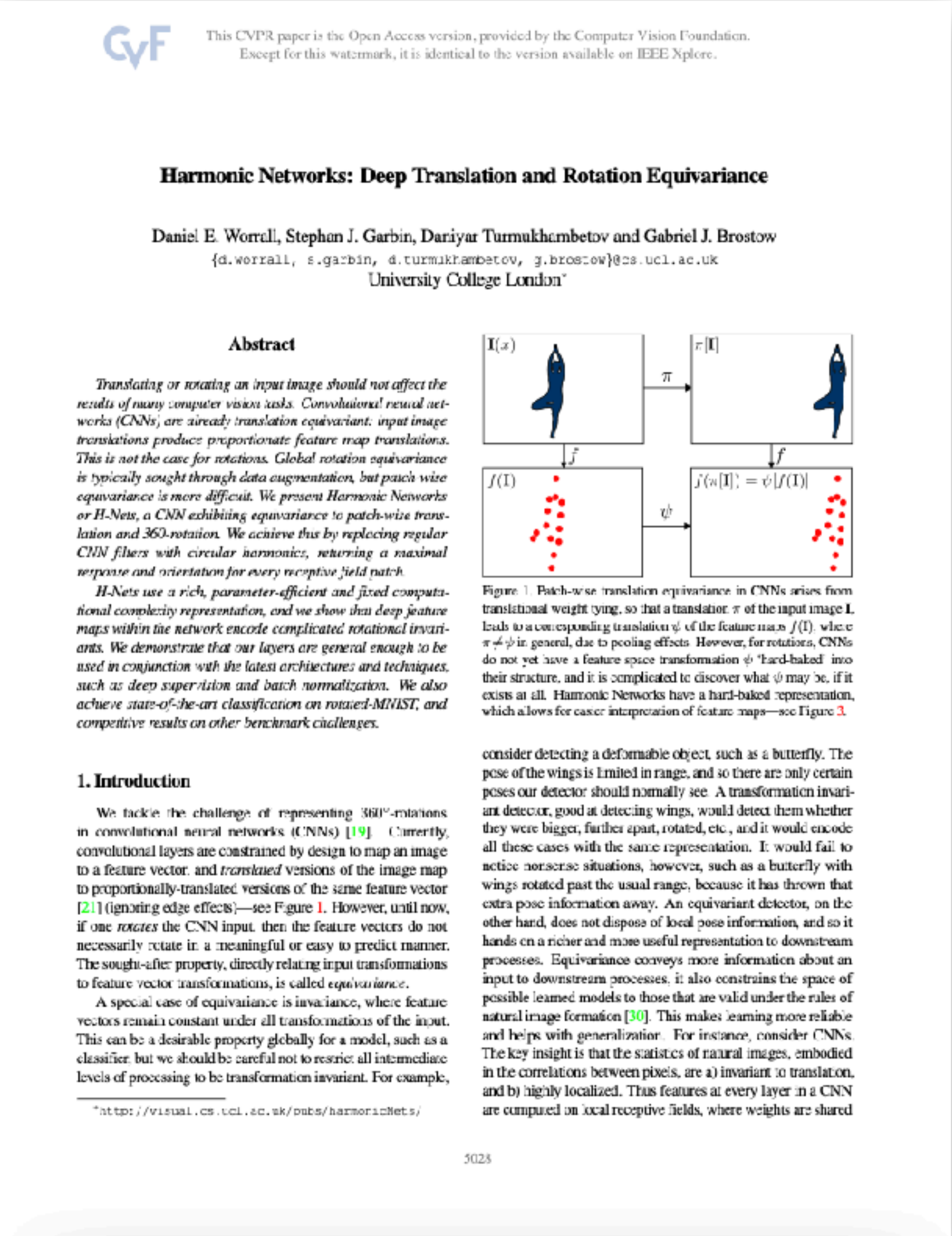
Compatible activation function: **norm-based activation functions**

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

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







Harmonic Networks: Deep Translation and Rotation Equivariance

Daniel E. Worrall, Stephan J. Garbin, Daniyar 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.

$\mathbf{I}(x)$ $\xrightarrow{\pi}$ $\pi[\mathbf{I}]$; f; $f(\mathbf{I})$ $\xrightarrow{\psi}$ $f(\pi[\mathbf{I}]) = \psi[f(\mathbf{I})]$

Figure 1. Patch-wise translation equivariance in CNNs arises from translational weight tying, so that a translation π of the input image $\mathbf{I}$, leads to a corresponding translation ψ 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 ψ "hard-baked" into their structure, and it is complicated to discover what ψ 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, 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 hands 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

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

5028

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**

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

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

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

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

using predicted scalar fields (type-0)

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

Activation functi

Activation functions should commute with the representation c

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

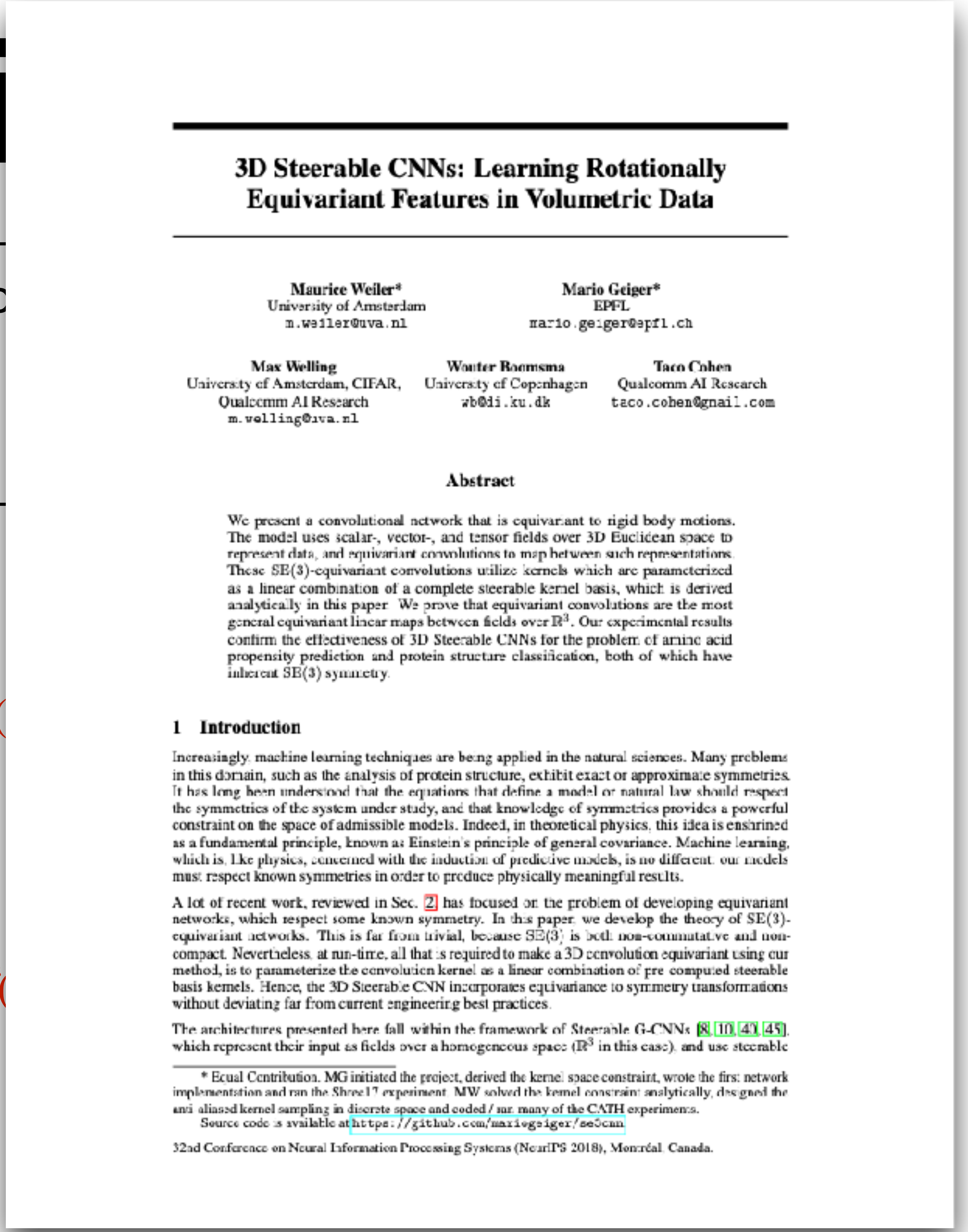
Compatible activation function: **norm-based activation functions**

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

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

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

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 produce 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, at run-time, 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, 40, 45], which represent their input as fields over a homogeneous space ($\mathbb{R}^3$ in this case), and use steerable

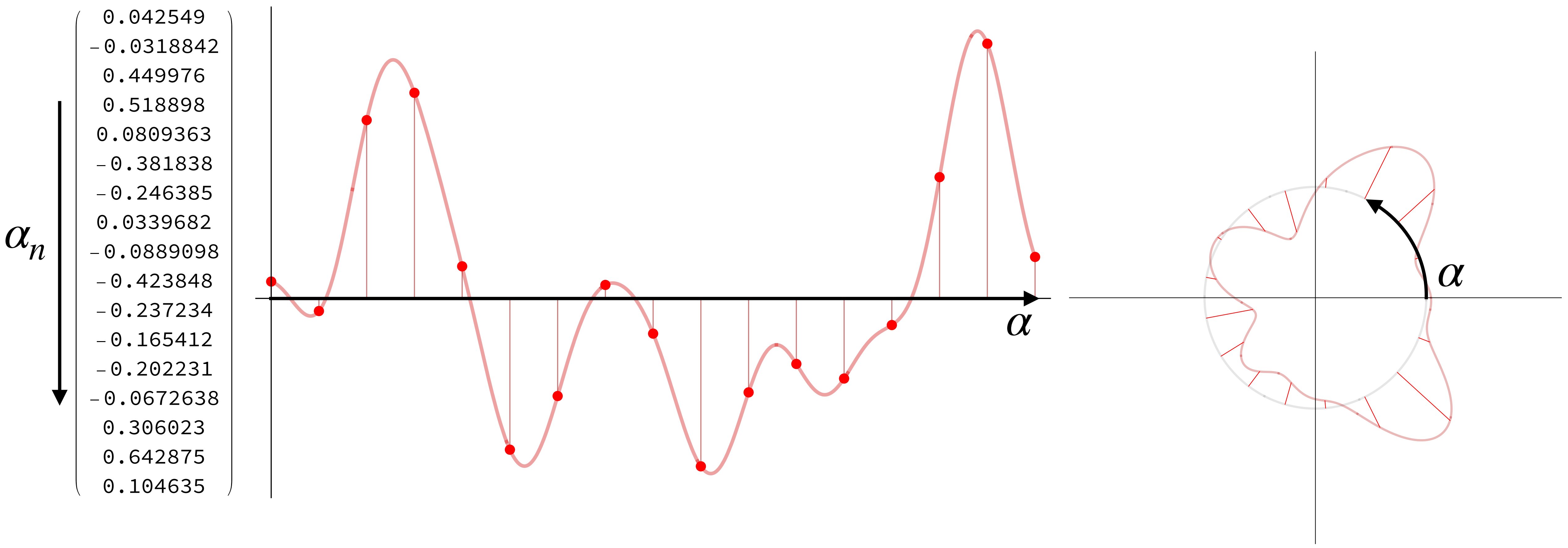
* Equal Contribution. MG initiated the project, derived the kernel space constraint, wrote the first network implementation and ran the Shrec17 experiment. MW solved the kernel constraint analytically, designed the anti-aliased kernel sampling in discrete space and coded / ran many of the CATH experiments.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

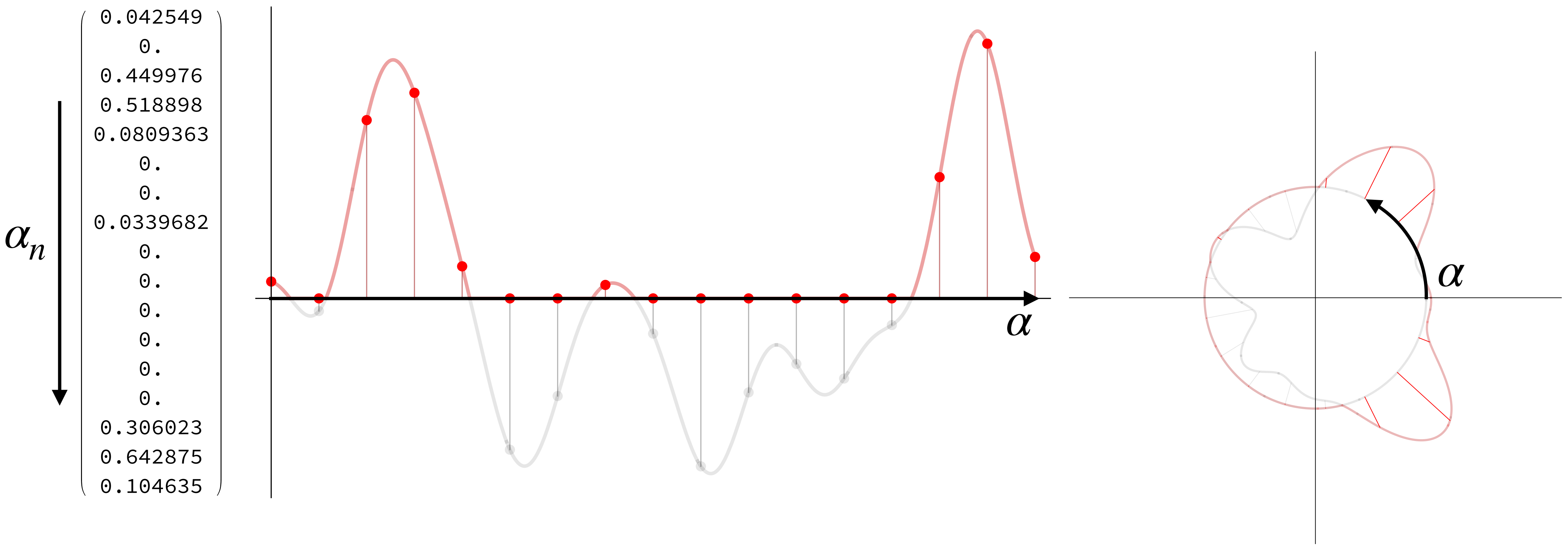
$$\mathscr{L}_\theta\,\sigma(f(\mathbf{x},\alpha)) \quad = \quad \sigma(f(\mathbf{x},\alpha-\theta)) \quad = \quad f'(\mathbf{x},\alpha-\theta)$$

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**

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

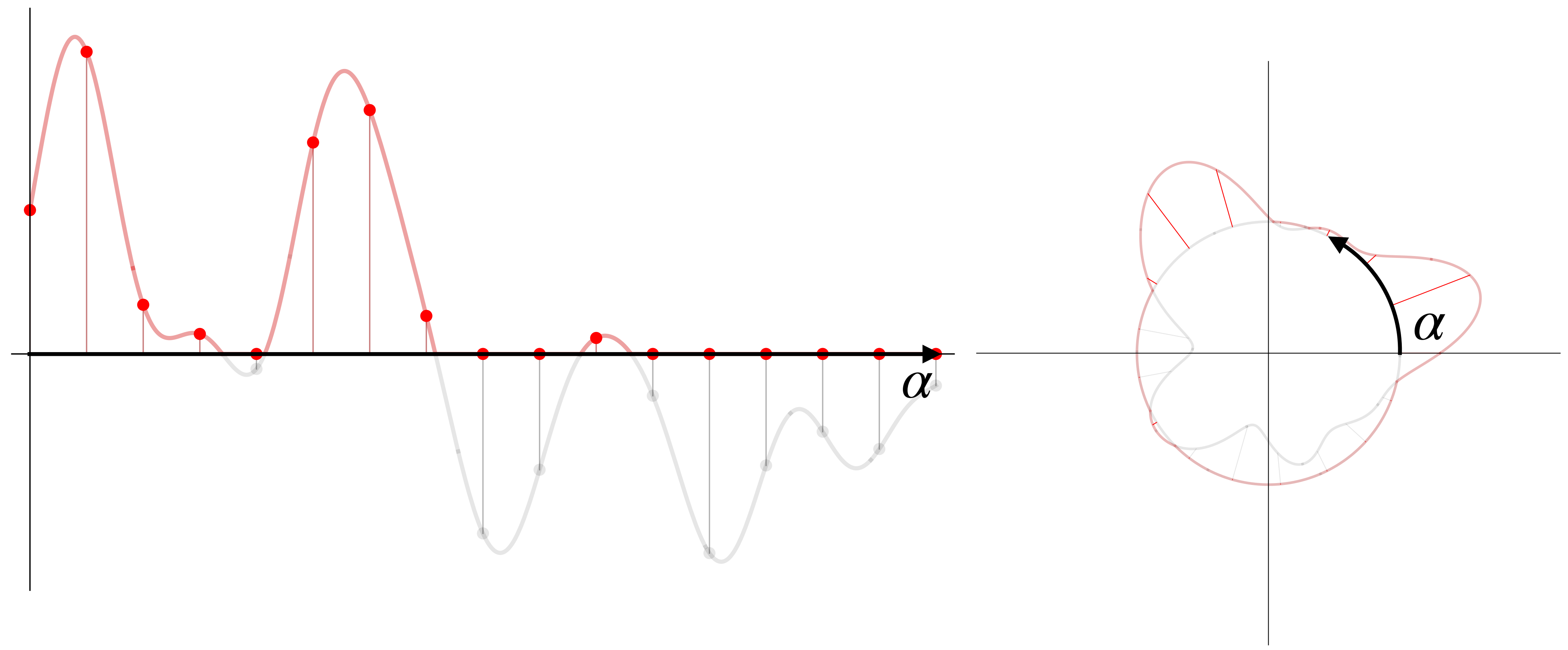
element-wise activations commute with permutations

Activation functions

Activation functions should commute with the representation of the fibers

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

$$\alpha_n \downarrow \begin{pmatrix} 0.306023 \\ 0.642875 \\ 0.104635 \\ 0.042549 \\ 0. \\ 0.449976 \\ 0.518898 \\ 0.0809363 \\ 0. \\ 0. \\ 0.0339682 \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \end{pmatrix}$$



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

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

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**

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

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

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**

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

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**

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

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** $(\mathscr{F}_H\,\sigma(\mathscr{F}_H^{-1}\hat{f}))$

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

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**

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

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

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

using predicted scalar fields (type-0)

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

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

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

vation functions

ions should commute with the representation of the fibers

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

ons

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

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

nt-wise activation for regular representations or scalar fields **Fourier-based**

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

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

N–BODY NETWORKS: A COVARIANT HIERARCHICAL NEURAL NETWORK ARCHITECTURE FOR LEARNING ATOMIC POTENTIALS[1]

Risi Kondor
Departments of Computer Science & Statistics
The University of Chicago
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 correspo ... are cons ... to rotatic ... are reali ... part of t ... weights ... covarian

1. INTRODUC

In principle, qua ... of atomic system ... few dozen atoms ... feasible proposit ... used approximati

Consequently, th ... explicitly, and f ... approximation, w ... called (effective) ... with $\hat{r}_j = r_{p_j}$ ... of its j'th neigh ... $F_i = -\nabla r_i \phi_i(\hat{r}$ ... closed form form ... potentials (empi ... others.

Empirical potent ... atoms, limiting t ... entered this field ... the aggregate fo ... small number of ... a veritable explo ... molecular dynam

This note desc ... ing for Molecules ... Beach, CA) on December 8, 2017.

and $C_{\ell_1,\ell_2,\ell}$ is the part of C_{ℓ_1,ℓ_2} matrix corresponding to the ℓ'th "block". Thus, in this case the operator T_1^ℓ just corresponds to mutiplying the tensor product by $C_{\ell_1,\ell_2,\ell}$. By linearity, the above relationship also extends to non-irreducible vectors. If ψ_1 is of type τ_1 and ψ_2 is of type τ_2, then

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

where

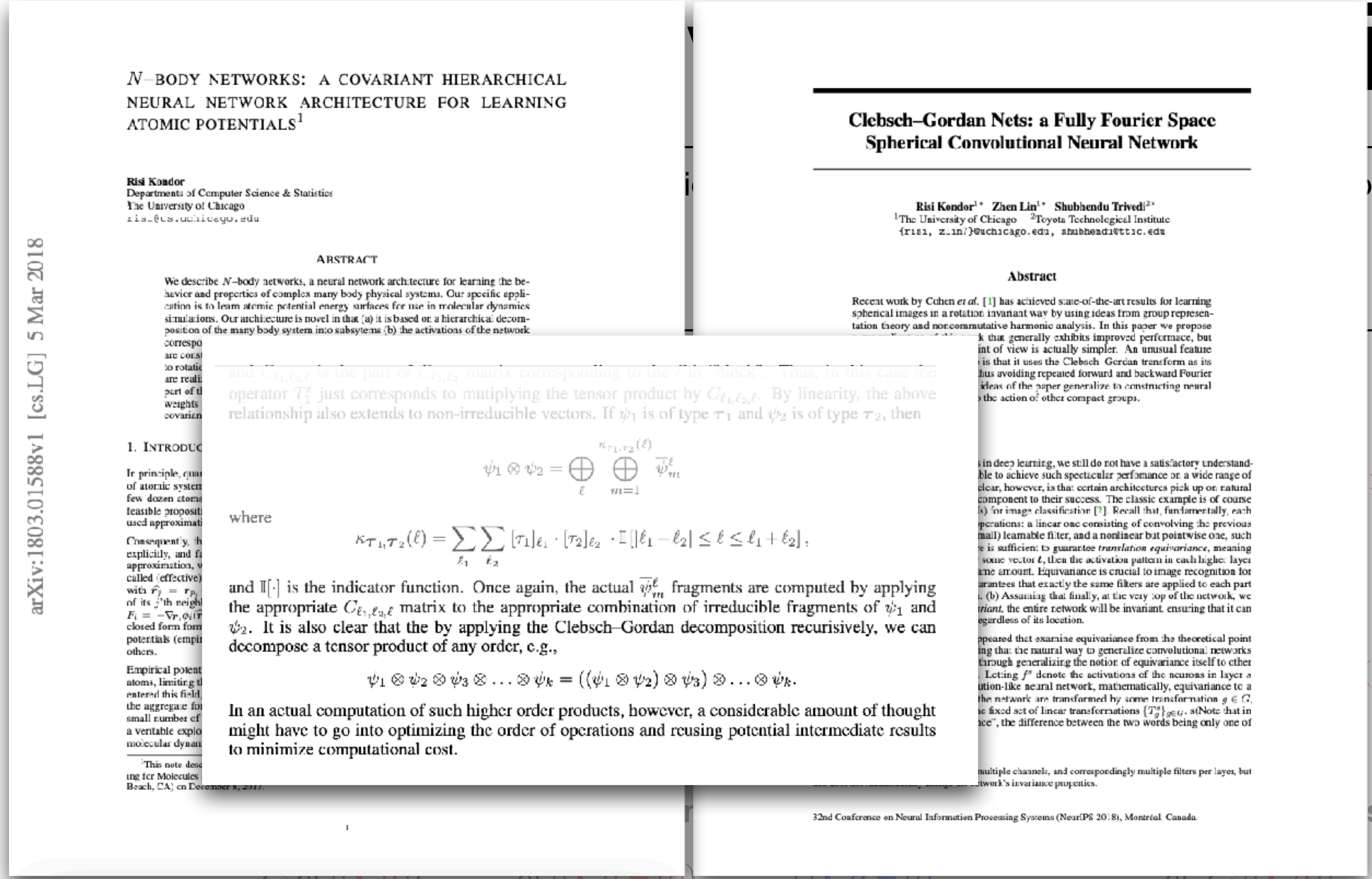
$$\kappa_{\tau_1,\tau_2}(\ell) = \sum_{\ell_1}\sum_{\ell_2} [\tau_1]_{\ell_1}\cdot[\tau_2]_{\ell_2}\cdot \mathbb{I}\left[|\ell_1-\ell_2|\le\ell\le\ell_1+\ell_2\right],$$

and $\mathbb{I}[\cdot]$ is the indicator function. Once again, the actual $\overline{\psi}_m^\ell$ fragments are computed by applying the appropriate $C_{\ell_1,\ell_2,\ell}$ matrix to the appropriate combination of irreducible fragments of ψ_1 and ψ_2. It is also clear that the by applying the Clebsch–Gordan decomposition recurisively, we can decompose a tensor product of any order, e.g.,

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

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.

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



N–BODY NETWORKS: A COVARIANT HIERARCHICAL NEURAL NETWORK ARCHITECTURE FOR LEARNING ATOMIC POTENTIALS[1]

Risi Kondor
Department of Computer Science & Statistics
The University of Chicago
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

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

Risi Kondor[1*] Zhen Lin[1*] Shubhendu Trivedi[2*]
[1]The University of Chicago [2]Toyota Technological Institute
{risi, zlin7}@uchicago.edu, 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 noncommutative harmonic analysis. In this paper we propose

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

operator T_τ^ℓ 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 ψ_1 is of type τ_1 and ψ_2 is of type τ_2, then

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

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 $\widetilde{\psi}_m^\ell$ fragments are computed by applying the appropriate $C_{\ell_1,\ell_2,\ell}$ matrix to the appropriate combination of irreducible fragments of ψ_1 and ψ_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.,

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

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.

ions

of the fibers

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

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

s or scalar fields

Fourier-based

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

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

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

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**

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

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

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

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

using predicted scalar fields (type-0)

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

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

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

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

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