Group Equivariant Deep Learning

Lecture 2 - Steerable group convolutions

Lecture 2.1 - Steerable kernels/basis functions

Definition and SO(2) example

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/

Group Equivariant Deep Learning

Lecture 2 - Steerable group convolutions

Lecture 2.1 - Steerable kernels/basis functions

Definition and $SO(2)$ example

Lecture 2.2 - Revisiting regular G-convs with steerable kernels | Template matching viewpoint

Motivating the Fourier transform on and showing we now no longer need a grid on the sub-group H!

Lecture 2.3 - Group Theory | Irreducible representations and Fourier transform

Preliminaries for steerable feature fields and steerable g-conv intuition with a focus on $SO(2)$

Lecture 2.4 - Group Theory | Induced representations and feature fields

Preliminaries (and intuition) for steerable group convolutions

Lecture 2.5 - Steerable group convolutions

And how to use them

Lecture 2.6 - Activation functions for steerable G-CNNs

Examples of which we can and cannot use

Lecture 2.7 - Derivation of Harmonic networks[1] from regular g-convs | Recalling g-convs are all you need!

[1] *Worrall, D. E., Garbin, S. J., Turmukhambetov, D., & Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. CVPR 2017*

Steerable basis

A vector $Y(x) = \begin{pmatrix} \vdots \\ Y_l(x) \\ \vdots \end{pmatrix} \in \mathbb{K}^L$ with (basis) functions $Y_l \in \mathbb{L}_2(X)$ is steerable if

$$\forall_{g \in G}: \quad Y(g\,x) = \rho(g) Y(x),$$

where $g\,x$ denotes the action of G on X and $\rho(g) \in \mathbb{K}^{L \times L}$ is a representation of G.

I.e., we can transform all basis functions simply by taking a linear combination of the original basis functions.

Example: Steerable basis on S^1 (circular harmonics)

Basis functions (for $\mathbb{L}_2(S^1)$): $Y_l(\alpha) = e^{i\,l\,\alpha}$

Are steered by representations: $\rho_l(\theta) = e^{i\,l\,\theta}$

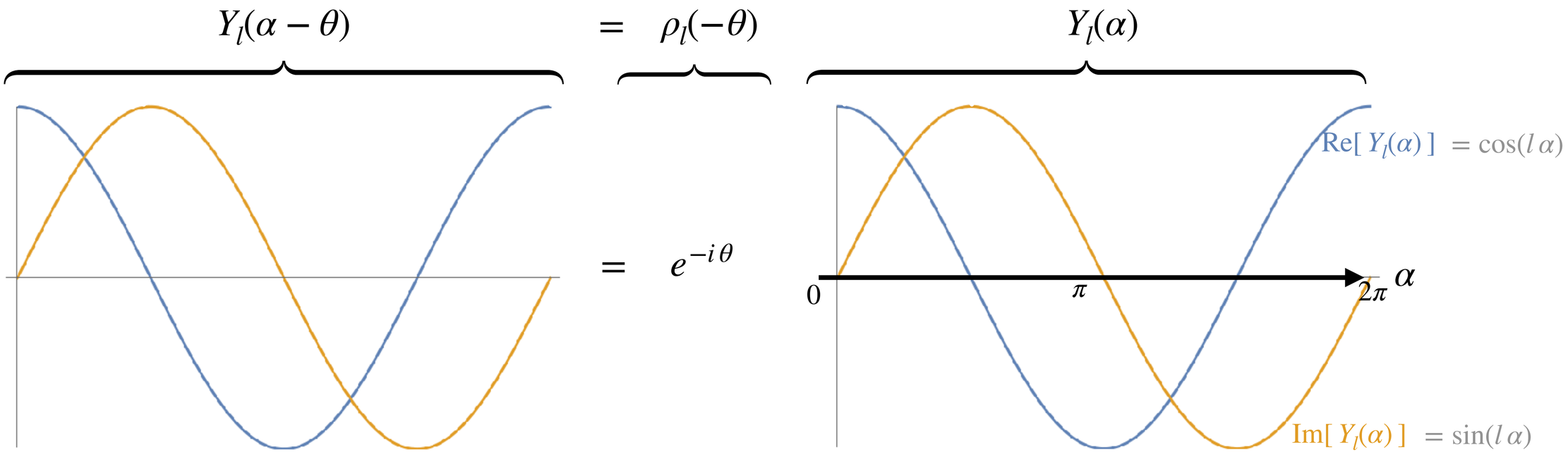
Proof: $Y_l(\alpha - \theta) = e^{i\,l\,(\alpha-\theta)}$
$= e^{-i\,l\,\theta}\, e^{i\,l\,\alpha}$
$= \rho_l(-\theta)\, Y_l(\alpha)$

Example: Steerable basis on S^1 (circular harmonics)

Basis functions (for $\mathbb{L}_2(S^1)$): $Y_l(\alpha) = e^{i\,l\,\alpha}$

Are steered by representations: $\rho_l(\theta) = e^{i\,l\,\theta}$

Proof: $Y_l(\alpha - \theta) = e^{i\,l\,(\alpha-\theta)}$
$= e^{-i\,l\,\theta}\, e^{i\,l\,\alpha}$
$= \rho_l(-\theta)\, Y_l(\alpha)$



Example: Steerable basis on S^1 (circular harmonics)

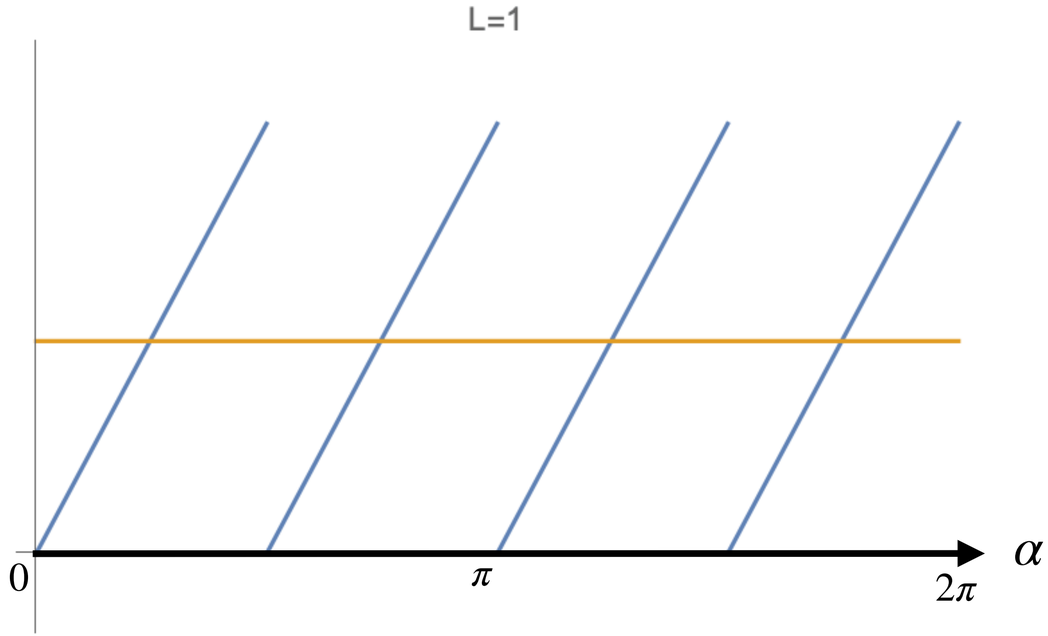
Basis functions (for $\mathbb{L}_2(S^1)$):

$$Y_l(\alpha) = e^{i\,l\,\alpha}$$

Form a complete orthonormal (Fourier) basis:

$$f(\alpha \,|\, \hat{\mathbf{w}}) = \sum_{l=-\infty}^{\infty} \overline{\hat{w}_l}\; Y_l(\alpha)$$

Y_l are given by the irreps of $SO(2)$ and hence form orthogonal basis (Peter-Weyl Theorem)



$$f(\alpha) = \alpha \mod \pi/2$$

$$f(\alpha \,|\, \hat{\mathbf{w}}) = \sum_{l=-L}^{L} \overline{\hat{w}_l}\; Y_l(\alpha)$$

Example: Steerable basis on S^1 (circular harmonics)

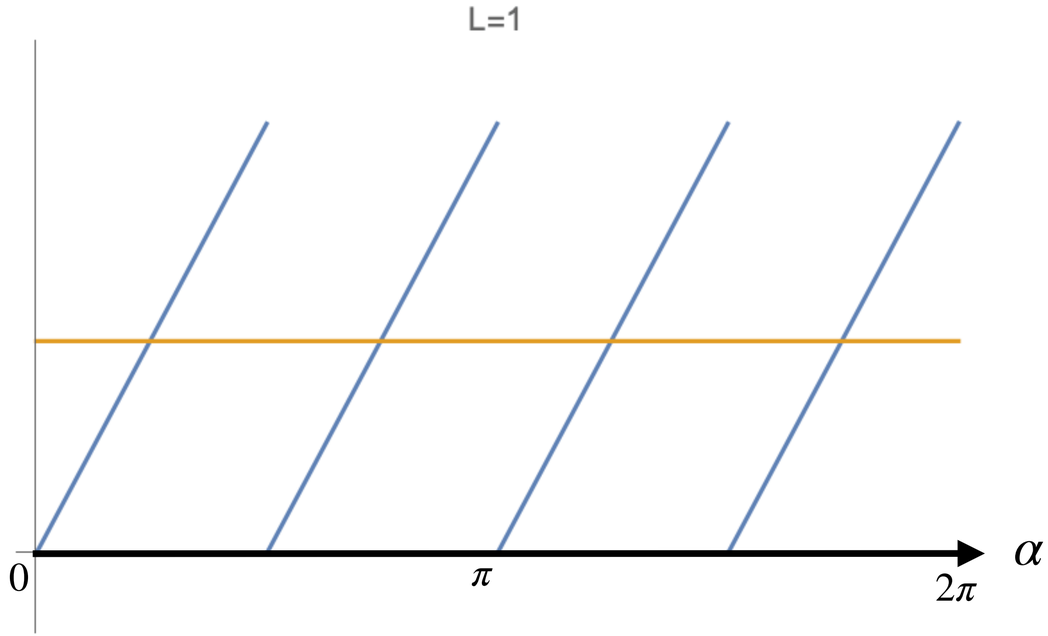
Basis functions (for $\mathbb{L}_2(S^1)$):

$$Y_l(\alpha) = e^{i\,l\,\alpha}$$

Form a complete orthonormal (Fourier) basis:

$$f(\alpha \,|\, \hat{\mathbf{w}}) = \sum_{l=-\infty}^{\infty} \hat{w}_l \, \overline{Y}_l(\alpha)$$

Y_l are given by the irreps of $SO(2)$ and hence form orthogonal basis (Peter-Weyl Theorem)



$$f(\alpha) = \alpha \mod \pi/2$$

$$f(\alpha \,|\, \hat{\mathbf{w}}) = \sum_{l=-L}^{L} \overline{\hat{w}}_l \, Y_l(\alpha)$$

Example: Steerable basis on S^1 (circular harmonics)

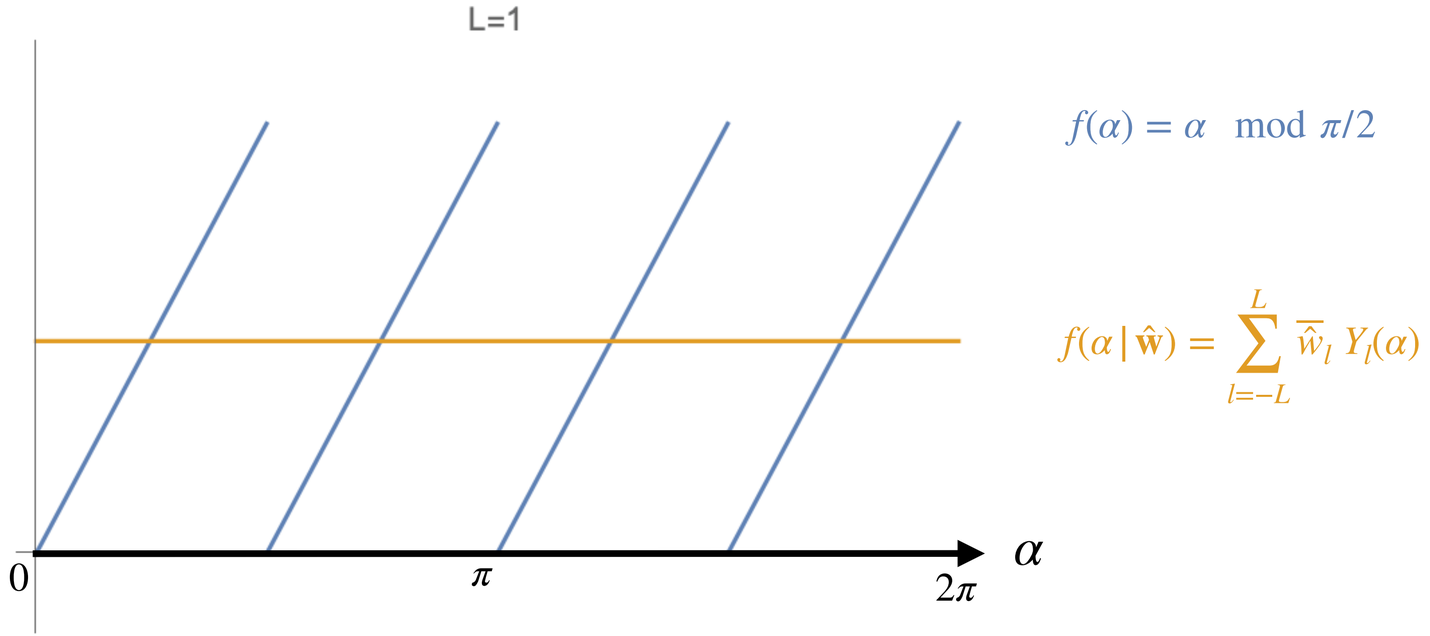
Basis functions (for $\mathbb{L}_2(S^1)$):

$$Y_l(\alpha) = e^{i\,l\,\alpha}$$

Form a complete orthonormal (Fourier) basis:

$$f(\alpha \,|\, \hat{\mathbf{w}}) = \sum_{l=-\infty}^{\infty} \hat{w}_l \; Y_l(-\alpha)$$

Y_l are given by the irreps of $SO(2)$ and hence form orthogonal basis (Peter-Weyl Theorem)



$$f(\alpha) = \alpha \mod \pi/2$$

$$f(\alpha \,|\, \hat{\mathbf{w}}) = \sum_{l=-L}^{L} \overline{\hat{w}}_l \; Y_l(\alpha)$$

Example: Steerable basis on S^1 (circular harmonics)

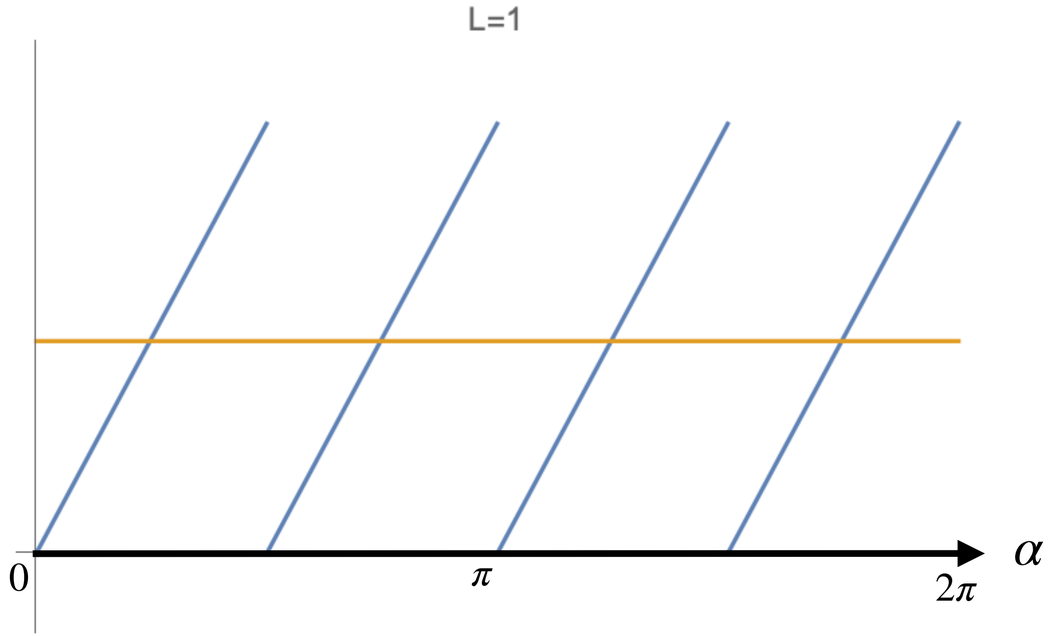
Basis functions (for $\mathbb{L}_2(S^1)$):

$$Y_l(\alpha) = e^{i\,l\,\alpha}$$

Form a complete orthonormal (Fourier) basis:

$$f(\alpha \,|\, \hat{\mathbf{w}}) = \sum_{l=-\infty}^{\infty} \overline{\hat{w}_l}\; Y_l(\alpha)$$

Y_l are given by the irreps of $SO(2)$ and hence form orthogonal basis (Peter-Weyl Theorem)



$$f(\alpha) = \alpha \mod \pi/2$$

$$f(\alpha \,|\, \hat{\mathbf{w}}) = \sum_{l=-L}^{L} \overline{\hat{w}_l}\; Y_l(\alpha)$$

Example: Steerable basis on S^1 (circular harmonics)

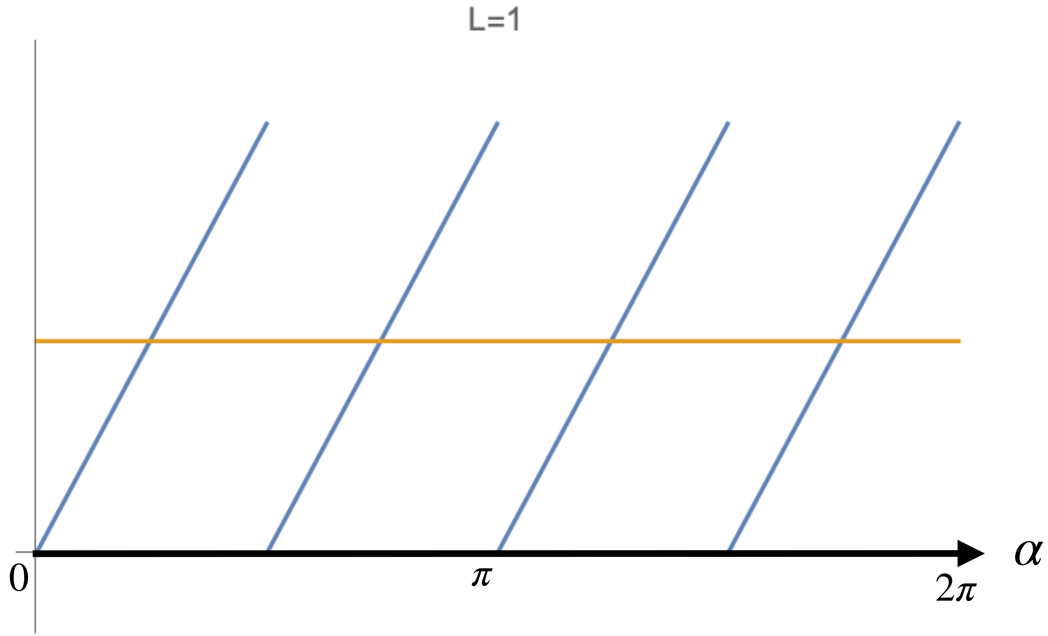
Basis functions (for $\mathbb{L}_2(S^1)$):

$$Y_l(\alpha) = e^{i\,l\,\alpha}$$

Form a complete orthonormal (Fourier) basis:

$$f(\alpha \,|\, \hat{\mathbf{w}}) = \sum_{l=-\infty}^{\infty} \overline{\hat{w}_l}\; Y_l(\alpha)$$

Y_l are given by the irreps of $SO(2)$ and hence form orthogonal basis (Peter-Weyl Theorem)



$$f(\alpha) = \alpha \mod \pi/2$$

$$f(\alpha \,|\, \hat{\mathbf{w}}) = \sum_{l=-L}^{L} \overline{\hat{w}_l}\; Y_l(\alpha)$$

Example: Steerable basis on S^1 (circular harmonics)

$Y(\alpha - \theta)$

$$\rho(-\theta) = \bigoplus_{l=-L}^{L} \rho_l(-\theta)$$

$Y(\alpha)$

$$\begin{pmatrix} \\ \\ \\ \\ \\ \\ \\ \end{pmatrix} = \begin{pmatrix} e^{i3\theta} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{i2\theta} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{i1\theta} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{-i1\theta} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-i2\theta} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-i3\theta} \end{pmatrix} \begin{pmatrix} \\ \\ \\ \\ \\ \\ \\ \end{pmatrix}$$

Example: Steerable basis on S^1 (circular harmonics)

$Y(\alpha - \theta)$

$$\rho(-\theta) = \bigoplus_{l=-L}^{L} \rho_l(-\theta)$$

$Y(\alpha)$

$$\begin{pmatrix} \\ \\ \\ \\ \\ \\ \\ \end{pmatrix} = \begin{pmatrix} e^{i3\theta} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{i2\theta} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{i1\theta} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{-i1\theta} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-i2\theta} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-i3\theta} \end{pmatrix} \begin{pmatrix} \\ \\ \\ \\ \\ \\ \\ \end{pmatrix}$$

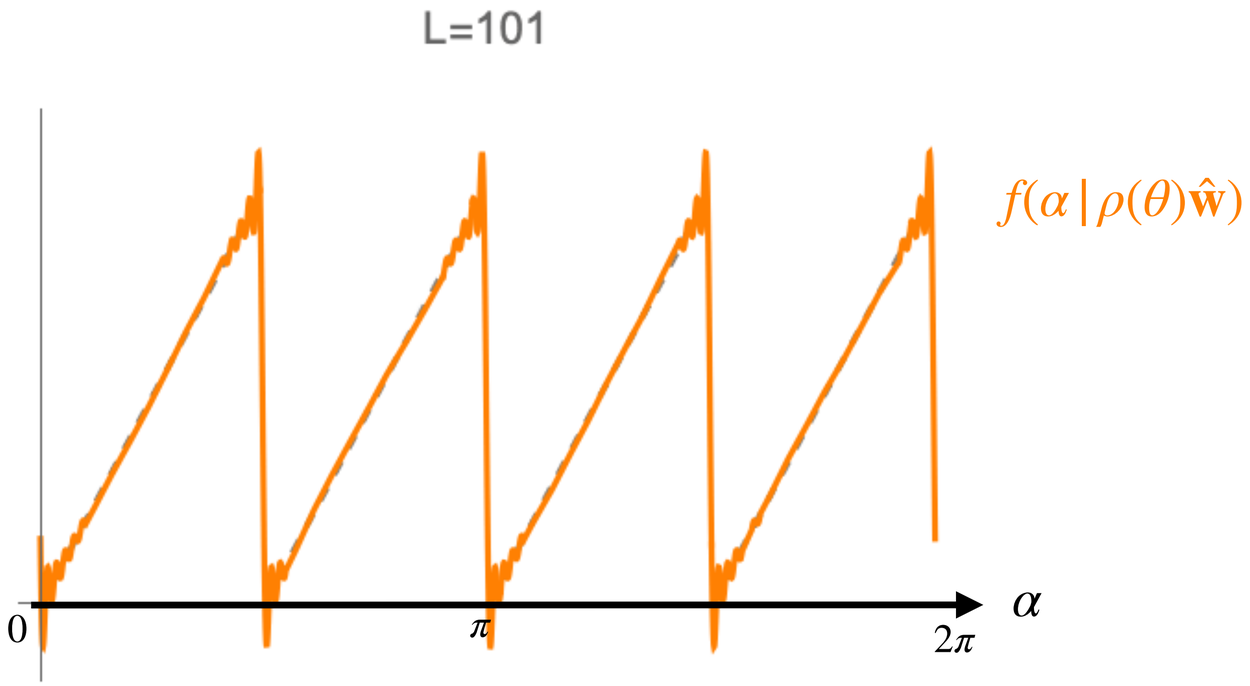
Example: Steerable basis on S^1 (circular harmonics)

Let $$f(\alpha \,|\, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^\dagger Y(\alpha)$$

Then we can steer/shift this function by transforming the weights $\hat{\mathbf{w}}$

$$f(\alpha - \theta \,|\, \hat{\mathbf{w}}) = f(\alpha \,|\, \rho(\theta)\hat{\mathbf{w}})$$

Proof:
$$\begin{aligned} f(\alpha - \theta \,|\, \hat{\mathbf{w}}) &= \hat{\mathbf{w}}^\dagger Y(\alpha - \theta) \\ &= \hat{\mathbf{w}}^\dagger \rho(-\theta) Y(\alpha) \\ &= \hat{\mathbf{w}}^\dagger \rho(\theta)^\dagger Y(\alpha) \\ &= (\rho(\theta)\hat{\mathbf{w}})^\dagger Y(\alpha) \\ &= f(\alpha \,|\, \rho(\theta)\hat{\mathbf{w}}) \end{aligned}$$



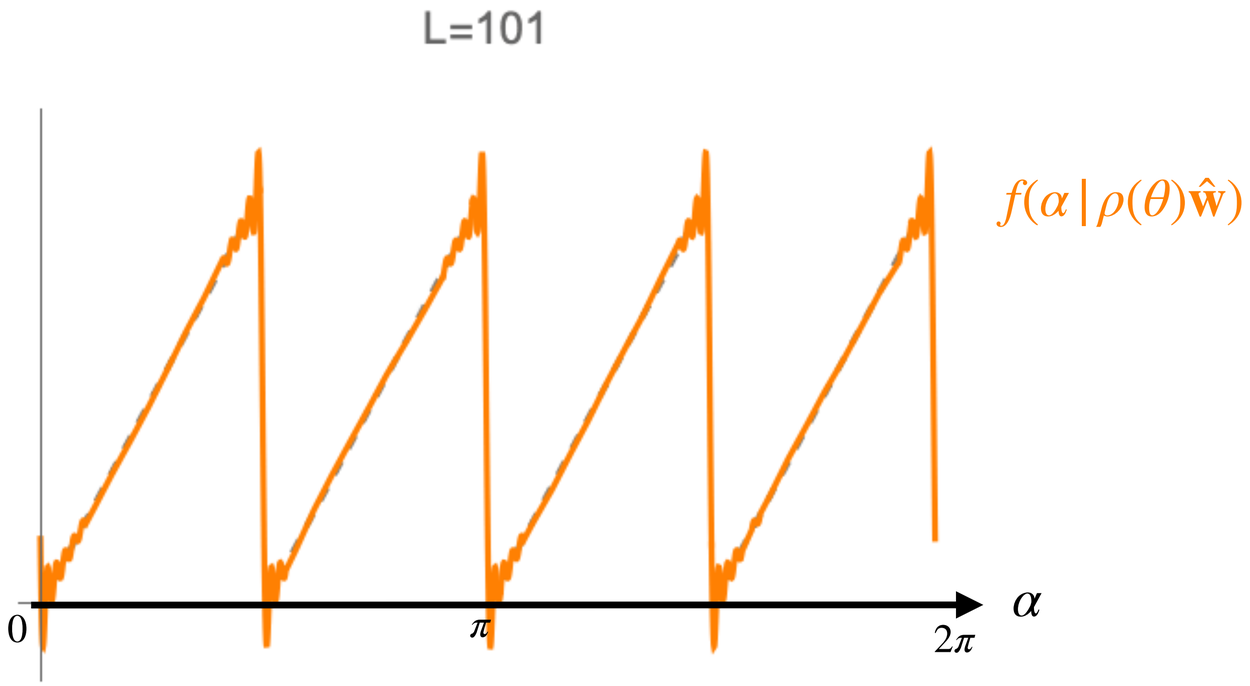
Example: Steerable basis on S^1 (circular harmonics)

Let $$f(\alpha \,|\, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^\dagger Y(\alpha)$$

Then we can steer/shift this function by transforming the weights $\hat{\mathbf{w}}$

$$f(\alpha - \theta \,|\, \hat{\mathbf{w}}) = f(\alpha \,|\, \rho(\theta)\hat{\mathbf{w}})$$

Proof:
$$\begin{aligned} f(\alpha - \theta \,|\, \hat{\mathbf{w}}) &= \hat{\mathbf{w}}^\dagger Y(\alpha - \theta) \\ &= \hat{\mathbf{w}}^\dagger \rho(-\theta) Y(\alpha) \\ &= \hat{\mathbf{w}}^\dagger \rho(\theta)^\dagger Y(\alpha) \\ &= (\rho(\theta)\hat{\mathbf{w}})^\dagger Y(\alpha) \\ &= f(\alpha \,|\, \rho(\theta)\hat{\mathbf{w}}) \end{aligned}$$

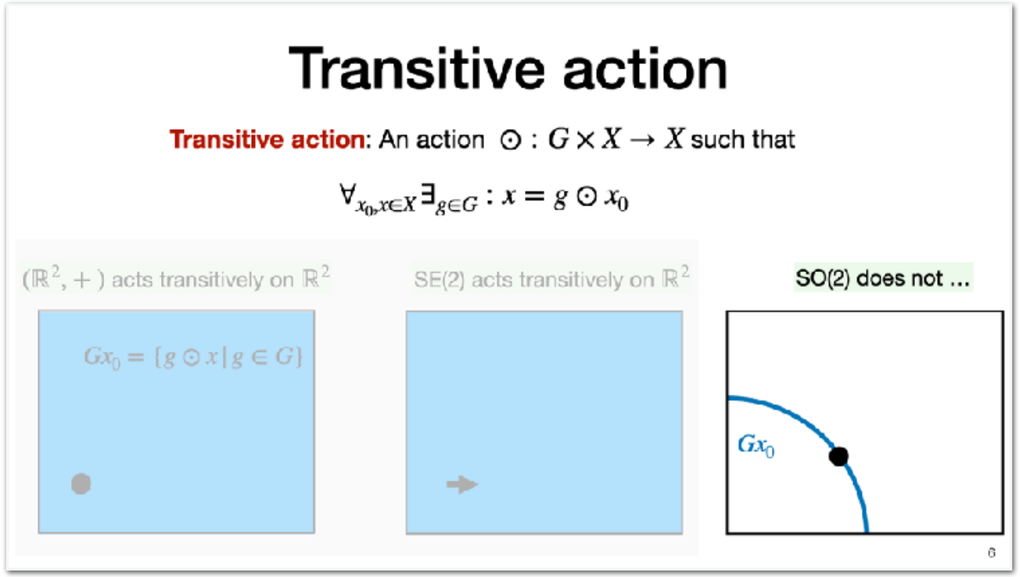


Two dimensional rotation-steerable functions

- The previous functions $\rho_l(\theta) = e^{i\,l\,\theta}$ are (irreducible) representations of $SO(2)$

- The group $SO(2)$ can also act on $\mathbb{R}^2$
  - Though not transitively…
  - It does act transitively on S^1 though

Recall lecture 1.6 (Group Theory | Homogeneous/quotient spaces)



- Use polar coordinates $\mathbb{R}^2 \ni \mathbf{x} \quad \leftrightarrow \quad (r, \alpha) \in \mathbb{R}^+ \times S^1$ to come up with a rotation-steerable basis for $\mathbb{L}_2(\mathbb{R}^2)$!

Two dimensional rotation-steerable functions

- Consider a function $f(\mathbf{x}) = \bar{f}(r, \alpha)$ in polar coordinates

$$\mathbf{x} = (r\cos\alpha, r\sin\alpha)$$

- The action of $SO(2)$ on $\mathbb{R}^2$ in polar coords translates to

$$\mathbf{x} \mapsto \mathbf{R}_\theta \mathbf{x} \quad \leftrightarrow \quad (r, \alpha) \mapsto (r, \alpha + \theta)$$

Proof: $\mathbf{R}_\theta \mathbf{x} = \mathbf{R}_\theta \begin{pmatrix} r\cos\alpha \\ r\sin\alpha \end{pmatrix}$

$$= \begin{pmatrix} r(\cos\theta\cos\alpha - \sin\alpha\sin\theta) \\ r(\cos\theta\sin\alpha + \cos\alpha\sin\theta) \end{pmatrix}$$

$$= \begin{pmatrix} r\cos(\theta + \alpha) \\ r\sin(\theta + \alpha) \end{pmatrix}$$

Two dimensional rotation-steerable functions

- Consider a function $f(\mathbf{x}) = \bar{f}(r, \alpha)$ in polar coordinates

$$\mathbf{x} = (r\cos\alpha, r\sin\alpha)$$

- The action of $SO(2)$ on $\mathbb{R}^2$ in polar coords translates to

$$\mathbf{x} \mapsto \mathbf{R}_\theta \mathbf{x} \quad \leftrightarrow \quad (r, \alpha) \mapsto (r, \alpha + \theta)$$

- Then, functions are rotated simply by a shift in the angular axis

$$\mathscr{L}_\theta^{SO(2)} f(\mathbf{x}) = f(\mathbf{R}_\theta^{-1}\mathbf{x}) \quad \leftrightarrow \quad \mathscr{L}_\theta^{SO(2)} \bar{f}(r, \alpha) = \bar{f}(r, \alpha - \theta)$$

Proof: $\mathbf{R}_\theta \mathbf{x} = \mathbf{R}_\theta \begin{pmatrix} r\cos\alpha \\ r\sin\alpha \end{pmatrix}$

$$= \begin{pmatrix} r(\cos\theta\cos\alpha - \sin\alpha\sin\theta) \\ r(\cos\theta\sin\alpha + \cos\alpha\sin\theta) \end{pmatrix}$$

$$= \begin{pmatrix} r\cos(\theta + \alpha) \\ r\sin(\theta + \alpha) \end{pmatrix}$$

Two dimensional rotation-steerable functions

- Consider a function $f(\mathbf{x}) = \bar{f}(r, \alpha)$ in polar coordinates

$$\mathbf{x} = (r\cos\alpha, r\sin\alpha)$$

- The action of $SO(2)$ on $\mathbb{R}^2$ in polar coords translates to

$$\mathbf{x} \mapsto \mathbf{R}_\theta \mathbf{x} \quad \leftrightarrow \quad (r, \alpha) \mapsto (r, \alpha + \theta)$$

- Then, functions are rotated simply by a shift in the angular axis

$$\mathscr{L}_\theta^{SO(2)} f(\mathbf{x}) = f(\mathbf{R}_\theta^{-1}\mathbf{x}) \quad \leftrightarrow \quad \mathscr{L}_\theta^{SO(2)} \bar{f}(r, \alpha) = \bar{f}(r, \alpha - \theta)$$

- Now, let's use this to paramatrize polar-separable conv kernels and focus on the angular part

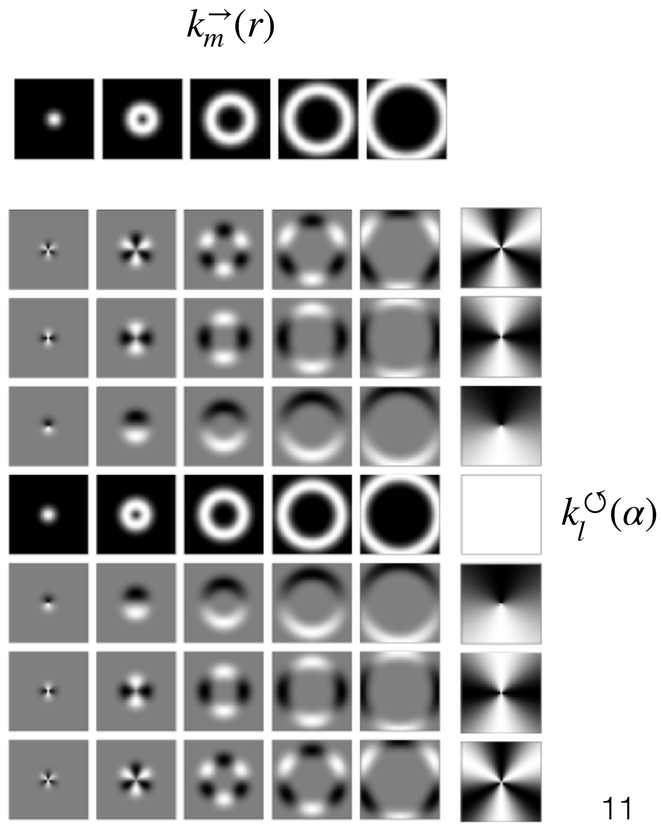
$$k(\mathbf{x}\,|\,\mathbf{w}) = k^{\rightarrow}(r\,|\,\mathbf{w})\, k^{\circlearrowleft}(\alpha\,|\,\mathbf{w})$$

A function on S^1 !!!

Proof: $\mathbf{R}_\theta \mathbf{x} = \mathbf{R}_\theta \begin{pmatrix} r\cos\alpha \\ r\sin\alpha \end{pmatrix}$

$$= \begin{pmatrix} r(\cos\theta\cos\alpha - \sin\alpha\sin\theta) \\ r(\cos\theta\sin\alpha + \cos\alpha\sin\theta) \end{pmatrix}$$

$$= \begin{pmatrix} r\cos(\theta + \alpha) \\ r\sin(\theta + \alpha) \end{pmatrix}$$



Two dimensional rotation-steerable functions

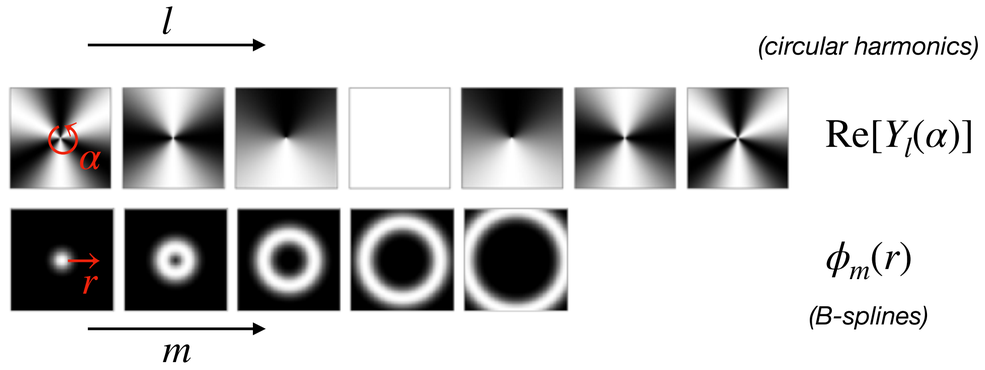
- Consider polar-separable convolution kernel:

$$k(\mathbf{x}\,|\,\mathbf{w}) = k^{\rightarrow}(r\,|\,\mathbf{w})\,k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}),$$

- with $k^{\circlearrowleft}$ in an $SO(2)$ steerable basis, and $k^{\rightarrow}$ in some radial basis:

$$k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}) = \sum_l \overline{w}_l Y_l(\alpha), \qquad \text{e.g., with} \qquad Y_l(\alpha) = e^{i\,l\,\alpha},$$

$$k^{\rightarrow}(r\,|\,\mathbf{w}) = \sum_m w_m \phi_m(r)$$



Two dimensional rotation-steerable functions

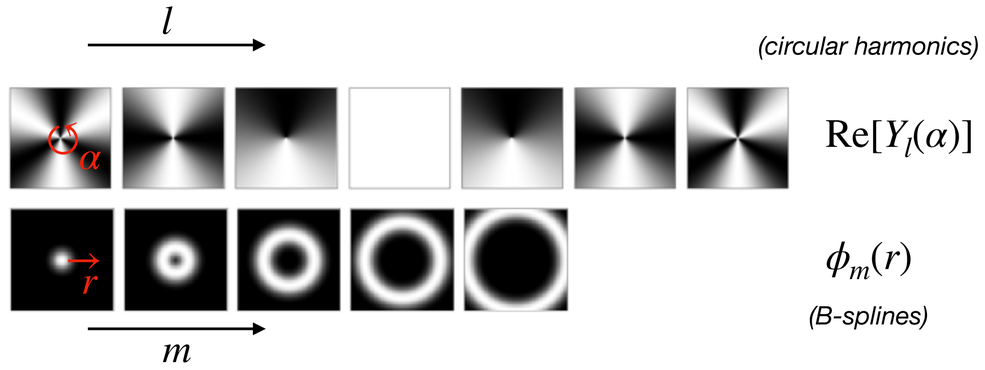
- Consider polar-separable convolution kernel:

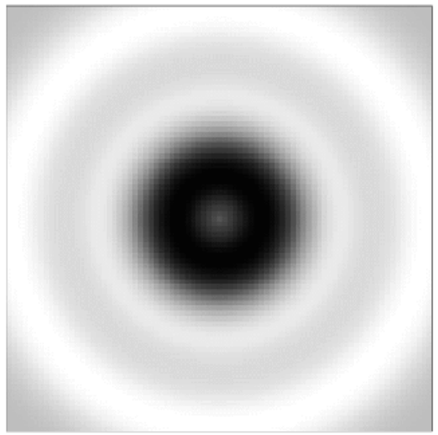
$$k(\mathbf{x}\,|\,\mathbf{w}) = k^{\rightarrow}(r\,|\,\mathbf{w})\,k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}),$$

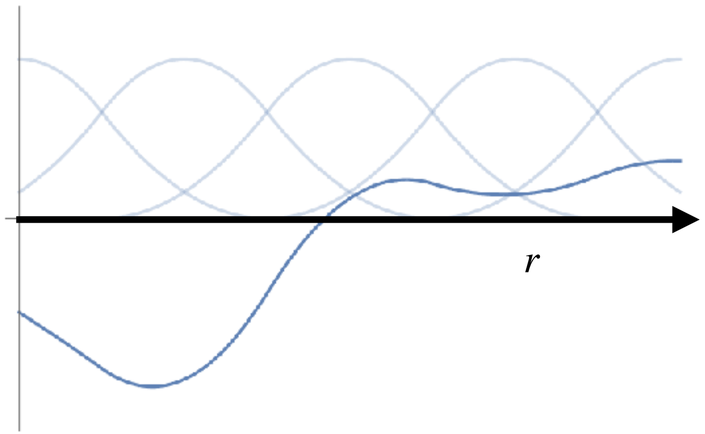
- with $k^{\circlearrowleft}$ in an $SO(2)$ steerable basis, and $k^{\rightarrow}$ in some radial basis:

$$k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}) = \sum_l \overline{w}_l Y_l(\alpha), \qquad \text{e.g., with} \qquad Y_l(\alpha) = e^{i\,l\,\alpha},$$

$$k^{\rightarrow}(r\,|\,\mathbf{w}) = \sum_m w_m \phi_m(r)$$







Two dimensional rotation-steerable functions

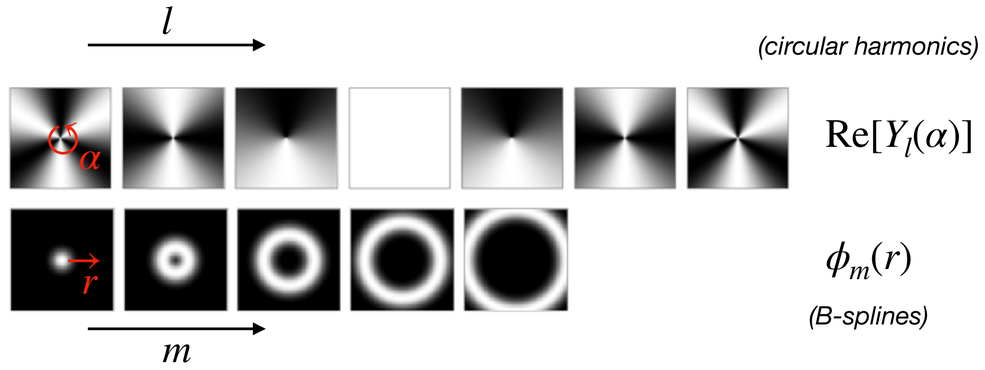
- Consider polar-separable convolution kernel:

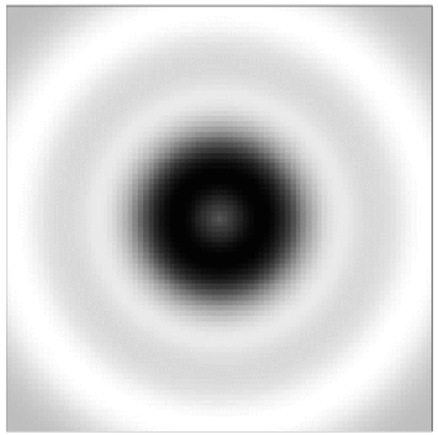
$$k(\mathbf{x}\,|\,\mathbf{w}) = k^{\rightarrow}(r\,|\,\mathbf{w})\,k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}),$$

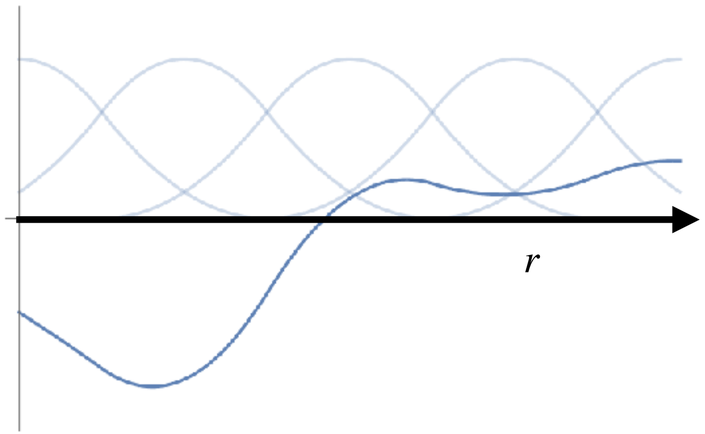
- with $k^{\circlearrowleft}$ in an $SO(2)$ steerable basis, and $k^{\rightarrow}$ in some radial basis:

$$k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}) = \sum_l \overline{w}_l Y_l(\alpha), \qquad \text{e.g., with} \qquad Y_l(\alpha) = e^{i\,l\,\alpha},$$

$$k^{\rightarrow}(r\,|\,\mathbf{w}) = \sum_m w_m \phi_m(r)$$







Two dimensional rotation-steerable functions

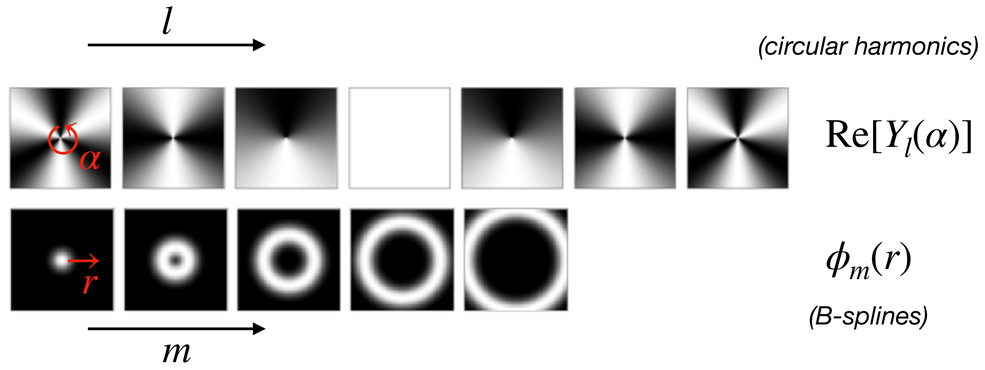
- Consider polar-separable convolution kernel:

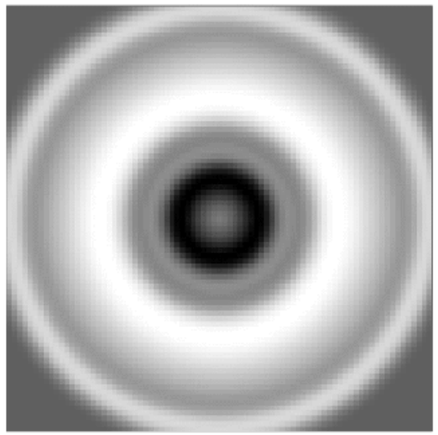
$$k(\mathbf{x}\,|\,\mathbf{w}) = k^{\rightarrow}(r\,|\,\mathbf{w})\,k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}),$$

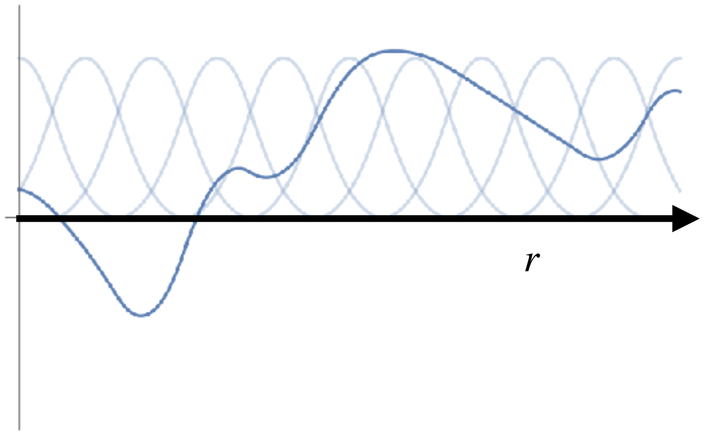
- with $k^{\circlearrowleft}$ in an $SO(2)$ steerable basis, and $k^{\rightarrow}$ in some radial basis:

$$k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}) = \sum_l \overline{w}_l Y_l(\alpha), \qquad \text{e.g., with} \qquad Y_l(\alpha) = e^{i\,l\,\alpha},$$

$$k^{\rightarrow}(r\,|\,\mathbf{w}) = \sum_m w_m \phi_m(r)$$







Two dimensional rotation-steerable functions

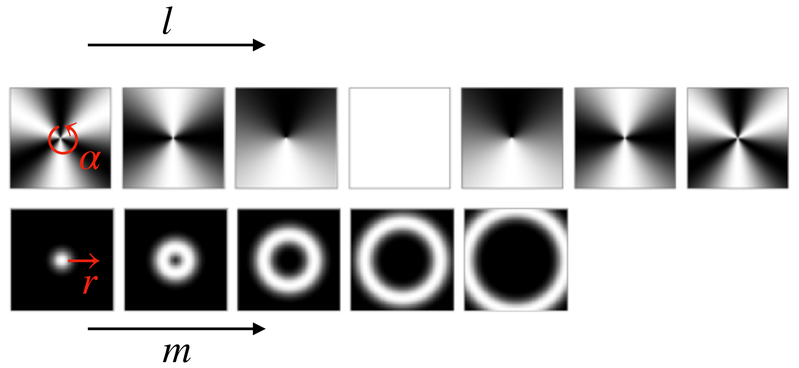
- Consider polar-separable convolution kernel:

$$k(\mathbf{x}\,|\,\mathbf{w}) = k^{\rightarrow}(r\,|\,\mathbf{w})\,k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}),$$

- with $k^{\circlearrowleft}$ in an $SO(2)$ steerable basis, and $k^{\rightarrow}$ in some radial basis:

$$k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}) = \sum_l \overline{w}_l Y_l(\alpha), \qquad \text{e.g., with} \qquad Y_l(\alpha) = e^{i\,l\,\alpha},$$

$$k^{\rightarrow}(r\,|\,\mathbf{w}) = \sum_m w_m \phi_m(r)$$



- Then we may as well write it as

$$\begin{aligned} k(\mathbf{x}\,|\,\mathbf{w}) &= \sum_l \sum_m w_m \overline{w}_l\, \phi_m(r)\, Y_l(\alpha) \\ &= \sum_l \sum_m \overline{w}_{ml}\, \phi_m(r)\, Y_l(\alpha) \qquad \text{(“absorb” weights)} \\ &= \sum_l \overline{\hat{w}}_l(r)\, Y_l(\alpha) \qquad \text{with radius dependent weights } \hat{w}_l(r) = \sum_m w_{ml}\, \phi_m(r) \end{aligned}$$

- Then such kernel is clearly rotation steerable!

$$k(\mathbf{R}_\theta^{-1}\mathbf{x}\,|\,\hat{\mathbf{w}}(r)) = k(\mathbf{x}\,|\,\rho(\theta)\hat{\mathbf{w}}(r))$$

Two dimensional rotation-steerable functions

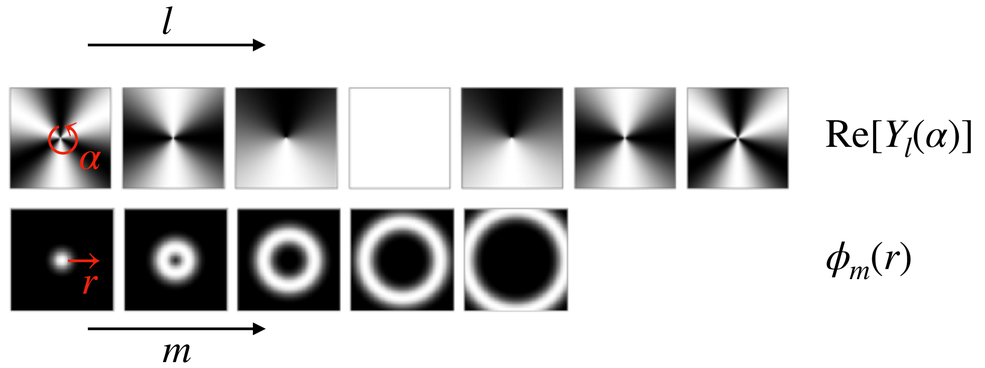
- Consider polar-separable convolution kernel:

$$k(\mathbf{x}\,|\,\mathbf{w}) = k^{\rightarrow}(r\,|\,\mathbf{w})\,k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}),$$

- with $k^{\circlearrowleft}$ in an $SO(2)$ steerable basis, and $k^{\rightarrow}$ in some radial basis:

$$k^{\circlearrowleft}(\alpha\,|\,\mathbf{w}) = \sum_l \overline{w}_l Y_l(\alpha), \qquad \text{e.g., with} \qquad Y_l(\alpha) = e^{i\,l\,\alpha},$$

$$k^{\rightarrow}(r\,|\,\mathbf{w}) = \sum_m w_m \phi_m(r)$$



- Then we may as well write it as

$$\begin{aligned} k(\mathbf{x}\,|\,\mathbf{w}) &= \sum_l \sum_m w_m \overline{w}_l\, \phi_m(r)\, Y_l(\alpha) \\ &= \sum_l \sum_m \overline{w}_{ml}\, \phi_m(r)\, Y_l(\alpha) \qquad \text{(“absorb” weights)} \\ &= \sum_l \overline{\hat{w}}_l(r)\, Y_l(\alpha) \qquad \text{with radius dependent weights } \hat{w}_l(r) = \sum_m w_{ml}\, \phi_m(r) \end{aligned}$$

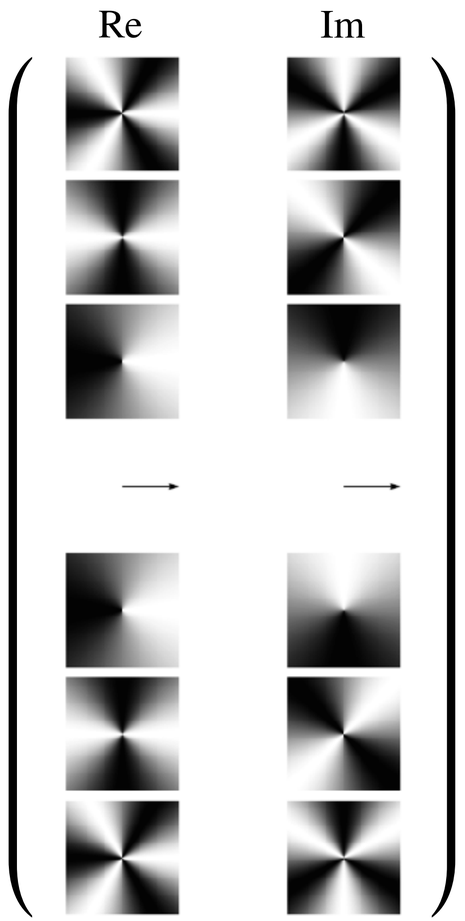
- Then such kernel is clearly rotation steerable!

$$k(\mathbf{R}_\theta^{-1}\mathbf{x}\,|\,\hat{\mathbf{w}}(r)) = k(\mathbf{x}\,|\,\rho(\theta)\hat{\mathbf{w}}(r))$$

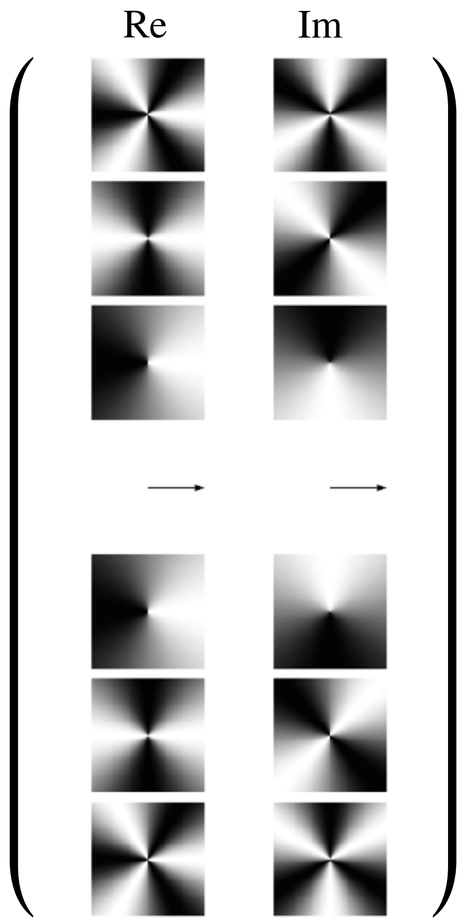
Or directly parametrize as $\hat{\mathbf{w}}(r) = \mathrm{MLP}(r\,|\,\mathbf{w})$!

Complex (irreducible) representations

$$Y(\mathbf{R}_\theta^{-1}\,\mathbf{x}) = \rho(\mathbf{R}_\theta^{-1})\, Y(\mathbf{x})$$

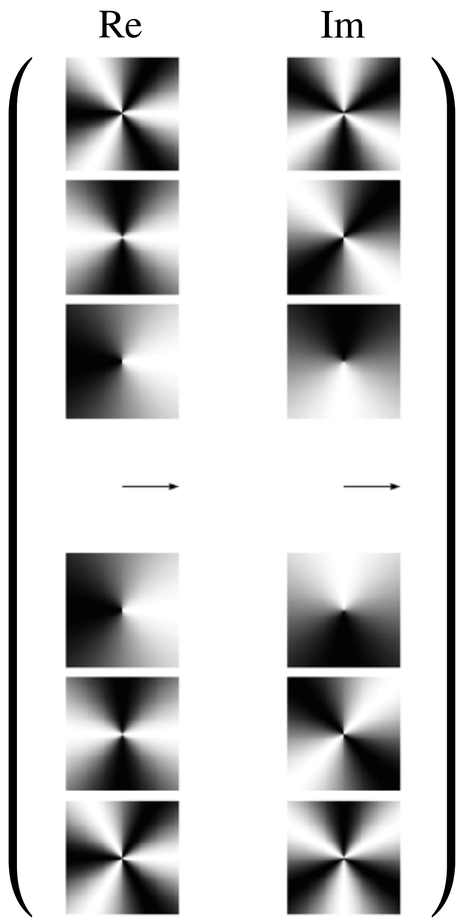


$$= \begin{pmatrix} e^{3\,i\theta} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{2\,i\theta} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{1\,i\theta} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{-1\,i\theta} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-2\,i\theta} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-3\,i\theta} \end{pmatrix}$$

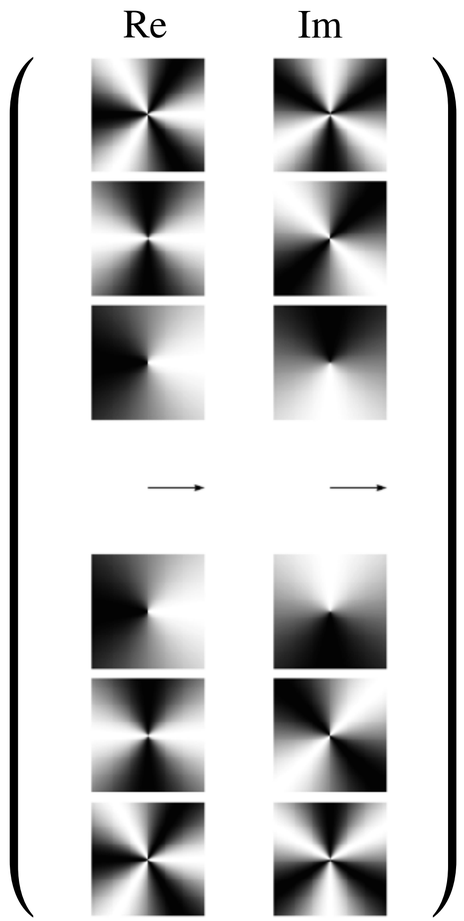


Complex (irreducible) representations

$$Y(\mathbf{R}_\theta^{-1}\,\mathbf{x}) = \rho(\mathbf{R}_\theta^{-1})\, Y(\mathbf{x})$$

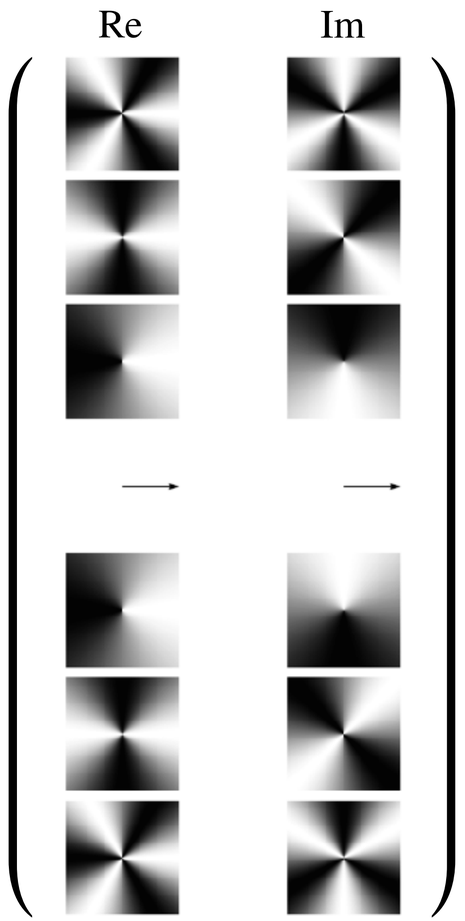


$$= \begin{pmatrix} e^{3\,i\theta} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{2\,i\theta} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{1\,i\theta} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{-1\,i\theta} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-2\,i\theta} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-3\,i\theta} \end{pmatrix}$$

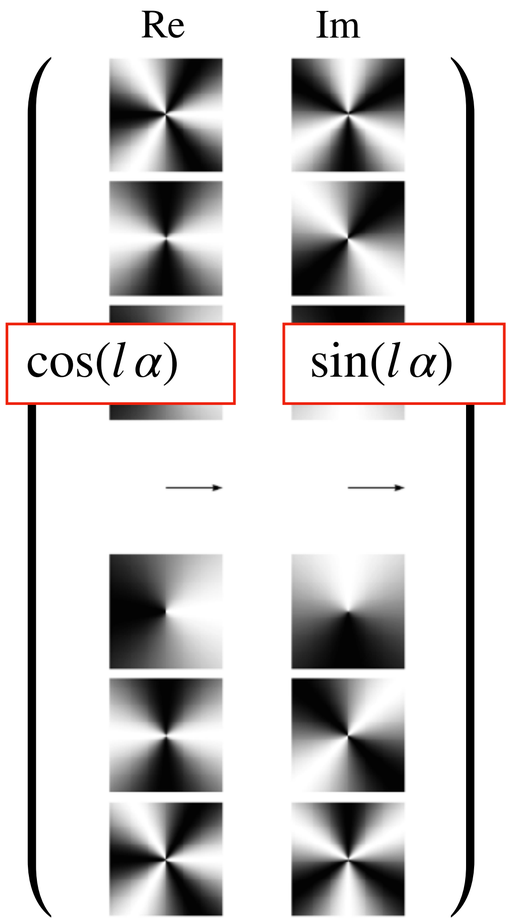


Complex (irreducible) representations

$$Y(\mathbf{R}_\theta^{-1}\,\mathbf{x}) = \rho(\mathbf{R}_\theta^{-1})\, Y(\mathbf{x})$$

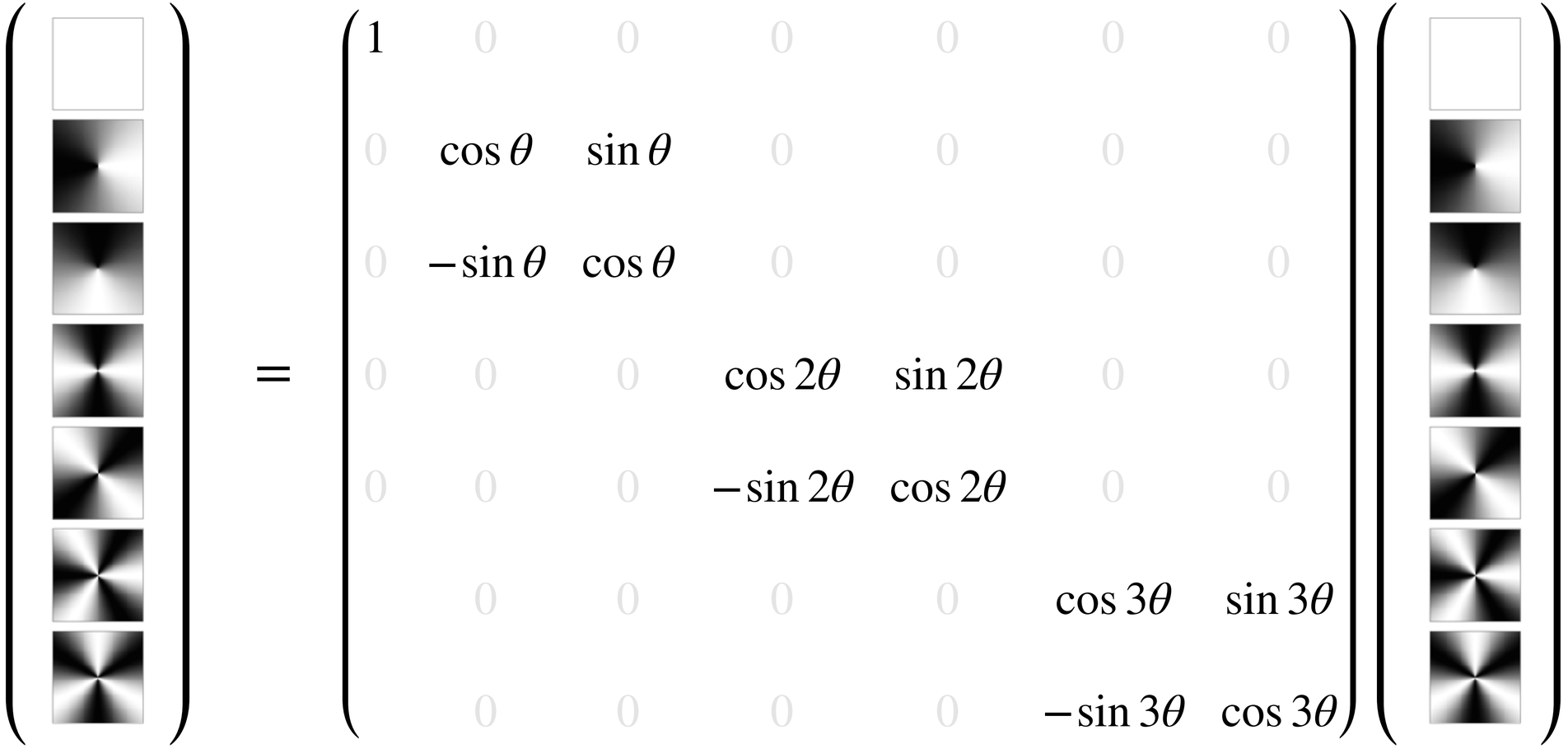


$$= \begin{pmatrix} e^{3\,i\theta} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{2\,i\theta} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{1\,i\theta} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{-1\,i\theta} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-2\,i\theta} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-3\,i\theta} \end{pmatrix}$$



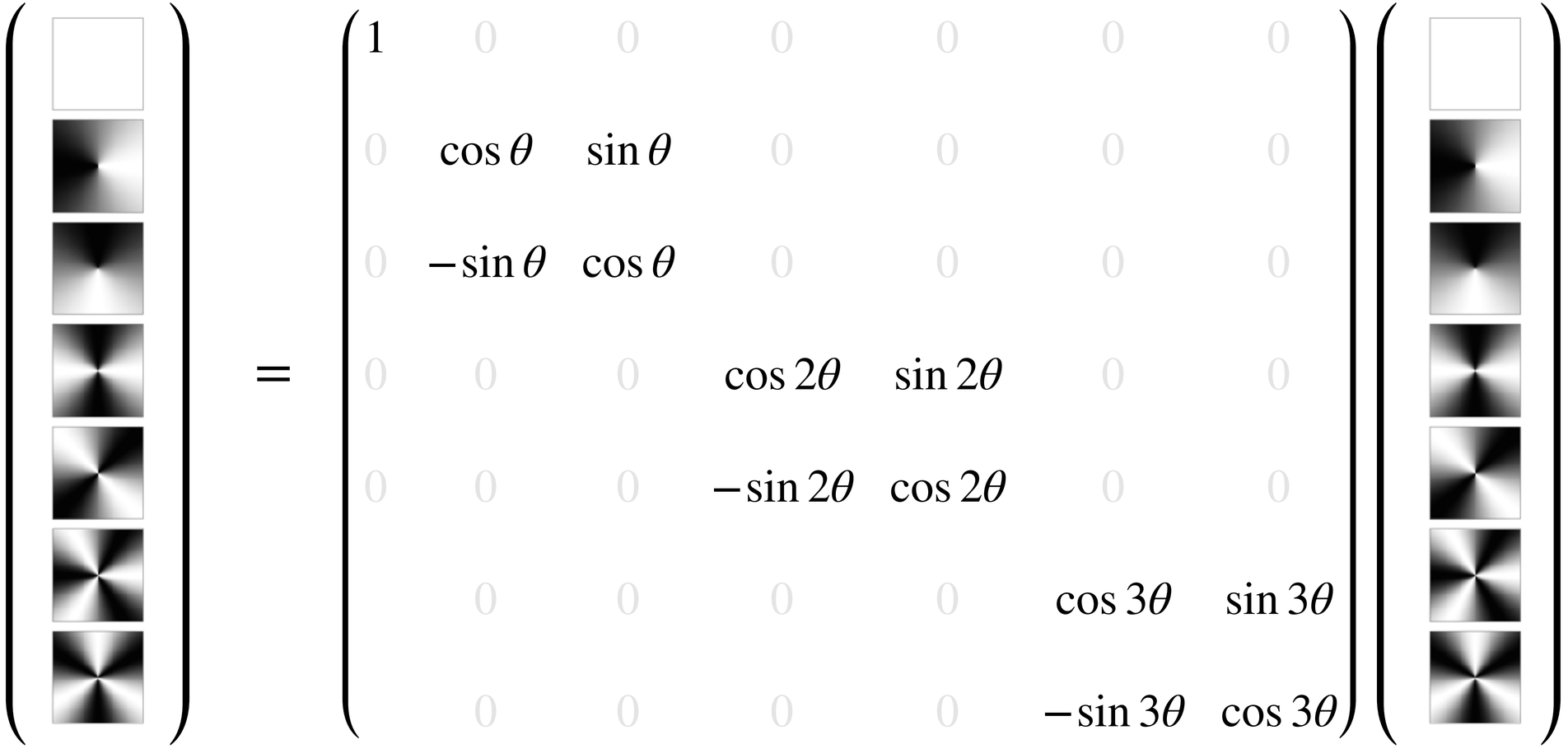
Real (irreducible) representations

$$Y(\mathbf{R}_\theta^{-1}\,\mathbf{x}) \;=\; \rho(\mathbf{R}_\theta^{-1}) \;\; Y(\mathbf{x})$$



Real (irreducible) representations

$$Y(\mathbf{R}_\theta^{-1}\,\mathbf{x}) \;=\; \rho(\mathbf{R}_\theta^{-1}) \;\; Y(\mathbf{x})$$

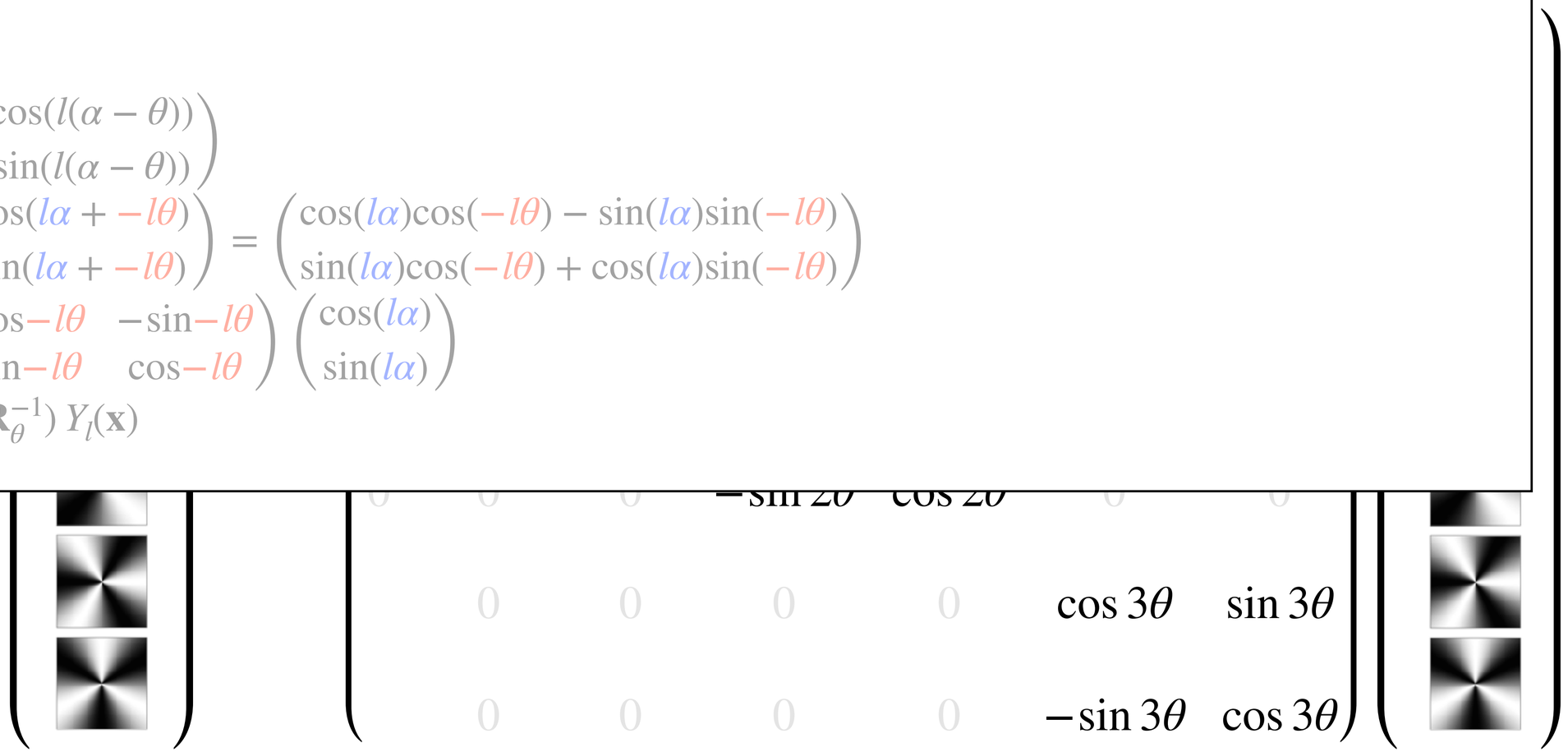


Real (irreducible) representations

The real basis functions $Y_l(\mathbf{x}) = \begin{pmatrix} \cos(l\alpha) \\ \sin(l\alpha) \end{pmatrix}$ are steerable using $\rho_l(\mathbf{R}_\theta) = \begin{pmatrix} \cos l\theta & -\sin l\theta \\ \sin l\theta & \cos l\theta \end{pmatrix}$

Proof:

$$
\begin{aligned}
Y_l(\mathbf{R}_\theta^{-1}\mathbf{x}) &= \begin{pmatrix} \cos(l(\alpha-\theta)) \\ \sin(l(\alpha-\theta)) \end{pmatrix} \\
&= \begin{pmatrix} \cos(l\alpha + -l\theta) \\ \sin(l\alpha + -l\theta) \end{pmatrix} = \begin{pmatrix} \cos(l\alpha)\cos(-l\theta) - \sin(l\alpha)\sin(-l\theta) \\ \sin(l\alpha)\cos(-l\theta) + \cos(l\alpha)\sin(-l\theta) \end{pmatrix} \\
&= \begin{pmatrix} \cos -l\theta & -\sin -l\theta \\ \sin -l\theta & \cos -l\theta \end{pmatrix} \begin{pmatrix} \cos(l\alpha) \\ \sin(l\alpha) \end{pmatrix} \\
&= \rho_l(\mathbf{R}_\theta^{-1})\, Y_l(\mathbf{x})
\end{aligned}
$$

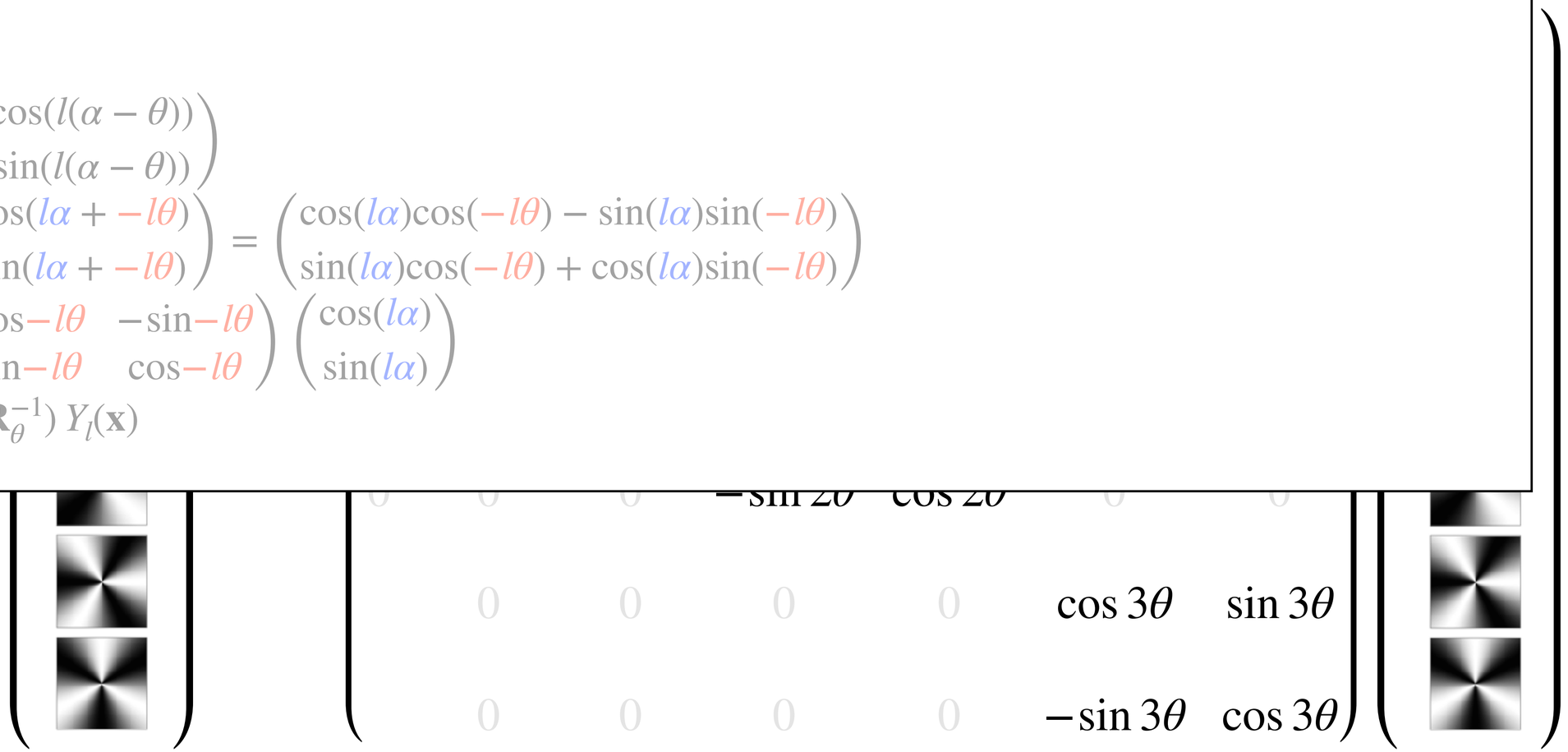


Real (irreducible) representations

The real basis functions $Y_l(\mathbf{x}) = \begin{pmatrix} \cos(l\alpha) \\ \sin(l\alpha) \end{pmatrix}$ are steerable using $\rho_l(\mathbf{R}_\theta) = \begin{pmatrix} \cos l\theta & -\sin l\theta \\ \sin l\theta & \cos l\theta \end{pmatrix}$

Proof:

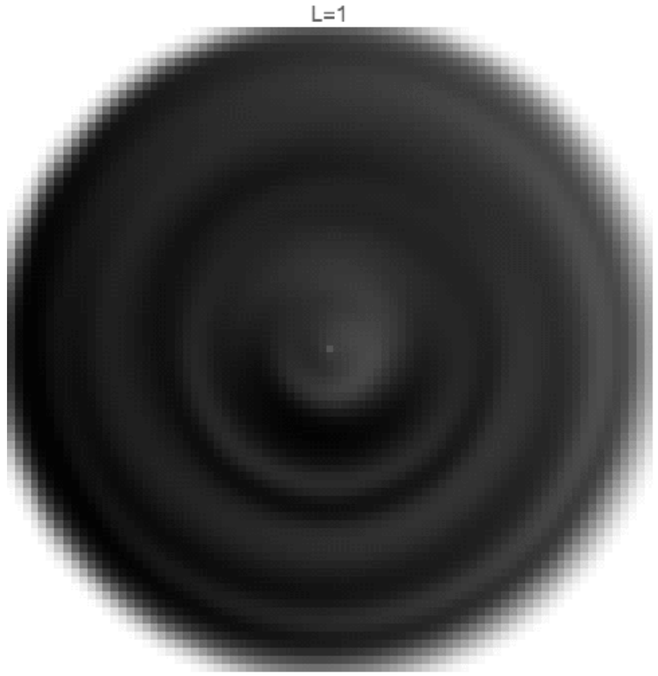
$$
\begin{aligned}
Y_l(\mathbf{R}_\theta^{-1}\mathbf{x}) &= \begin{pmatrix} \cos(l(\alpha-\theta)) \\ \sin(l(\alpha-\theta)) \end{pmatrix} \\
&= \begin{pmatrix} \cos(l\alpha + -l\theta) \\ \sin(l\alpha + -l\theta) \end{pmatrix} = \begin{pmatrix} \cos(l\alpha)\cos(-l\theta) - \sin(l\alpha)\sin(-l\theta) \\ \sin(l\alpha)\cos(-l\theta) + \cos(l\alpha)\sin(-l\theta) \end{pmatrix} \\
&= \begin{pmatrix} \cos -l\theta & -\sin -l\theta \\ \sin -l\theta & \cos -l\theta \end{pmatrix} \begin{pmatrix} \cos(l\alpha) \\ \sin(l\alpha) \end{pmatrix} \\
&= \rho_l(\mathbf{R}_\theta^{-1})\, Y_l(\mathbf{x})
\end{aligned}
$$



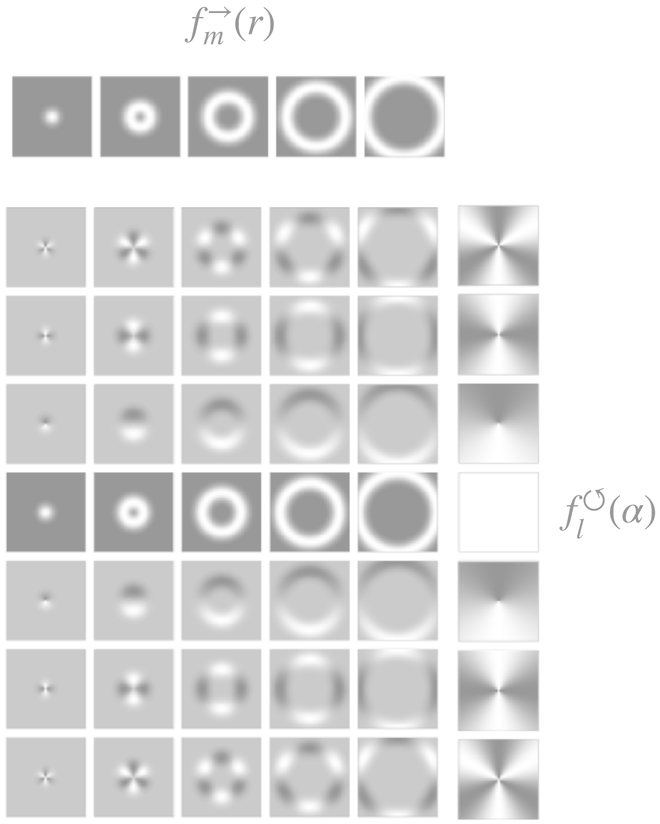
Representing interesting convolution kernels in a steerable basis!

Exercise:

1. Tune the weights $\hat{\mathbf{w}}$ until you get something interesting.
2. Add more detail by increasing maximum frequency!



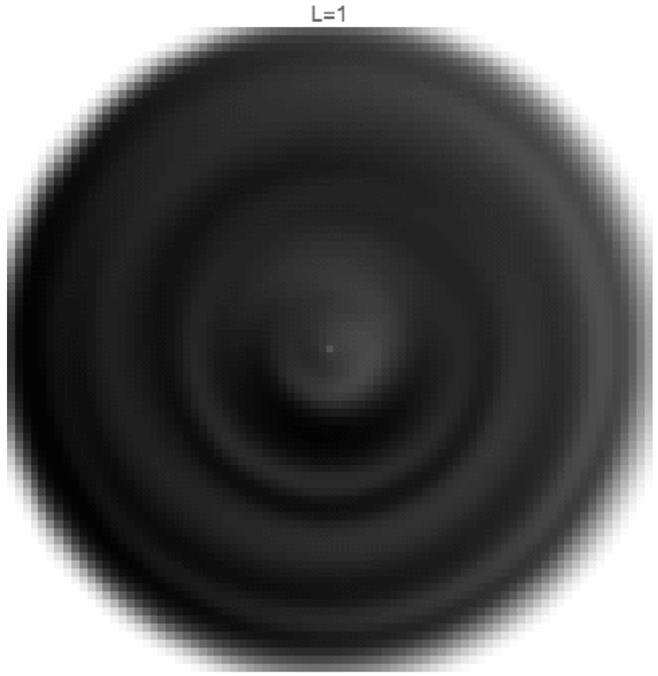
$$k(\mathbf{x} \mid \hat{\mathbf{w}}(r))$$



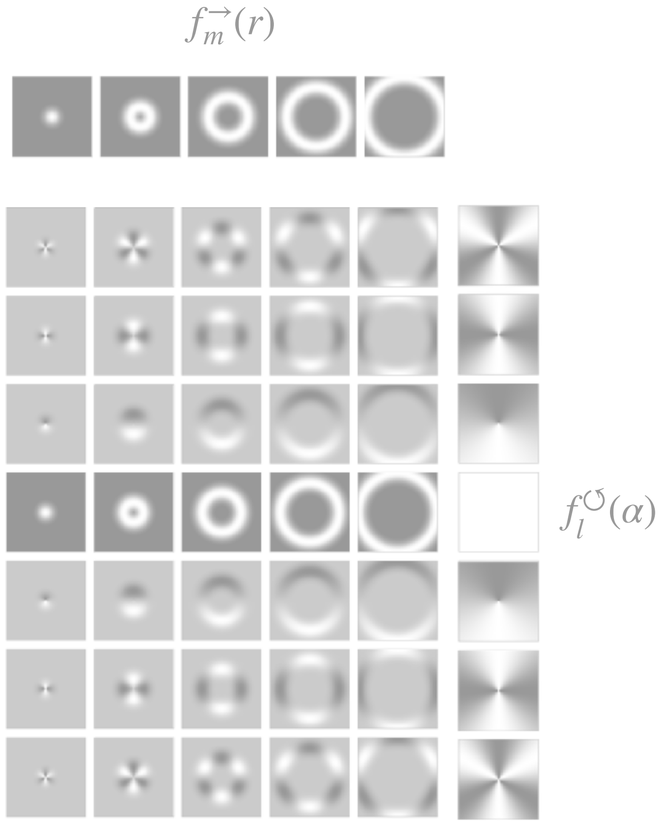
Representing interesting convolution kernels in a steerable basis!

Exercise:

1. Tune the weights $\hat{\mathbf{w}}$ until you get something interesting.
2. Add more detail by increasing maximum frequency!



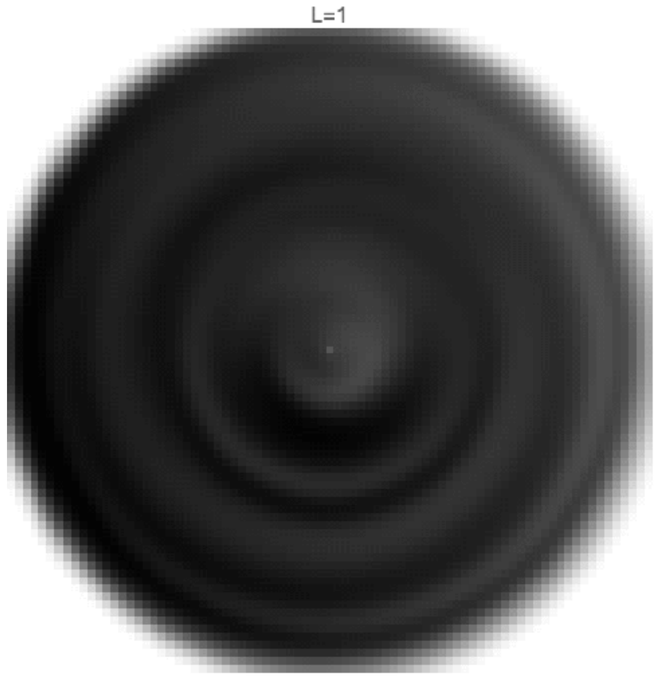
$$k(\mathbf{x} \mid \hat{\mathbf{w}}(r))$$



Representing interesting convolution kernels in a steerable basis!

Exercise:

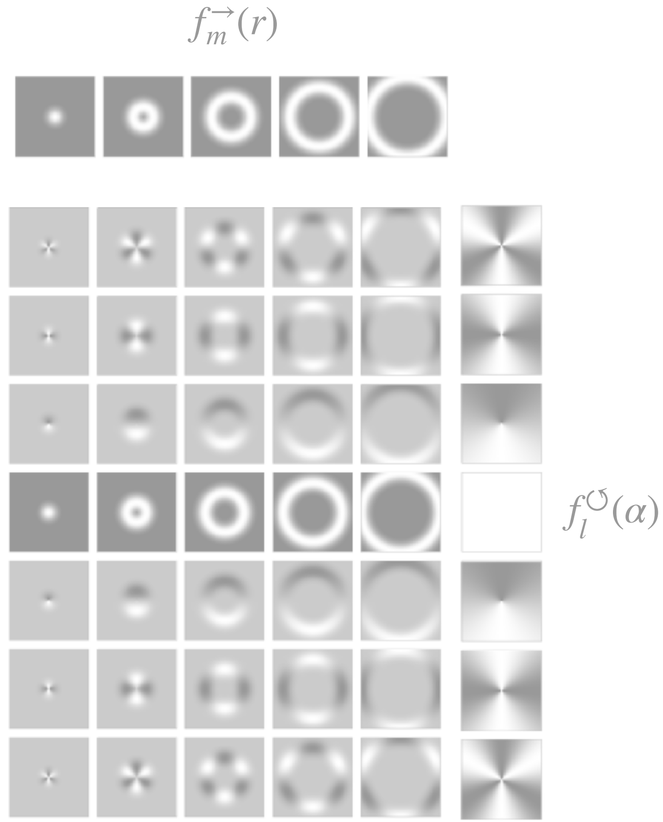
1. Tune the weights $\hat{\mathbf{w}}$ until you get something interesting.
2. Add more detail by increasing maximum frequency!



$$k(\mathbf{x} \mid \hat{\mathbf{w}}(r))$$

3. Go crazy and steer it by transforming the weights!

$$k(\mathbf{x} \mid \rho(\theta)\hat{\mathbf{w}}(r))$$



Group Equivariant Deep Learning

Lecture 2 - Steerable group convolutions

Lecture 2.2 - Revisiting regular G-convs with steerable kernels

Motivating the Fourier transform on H and showing we now no longer need a grid on the sub-group H!

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/

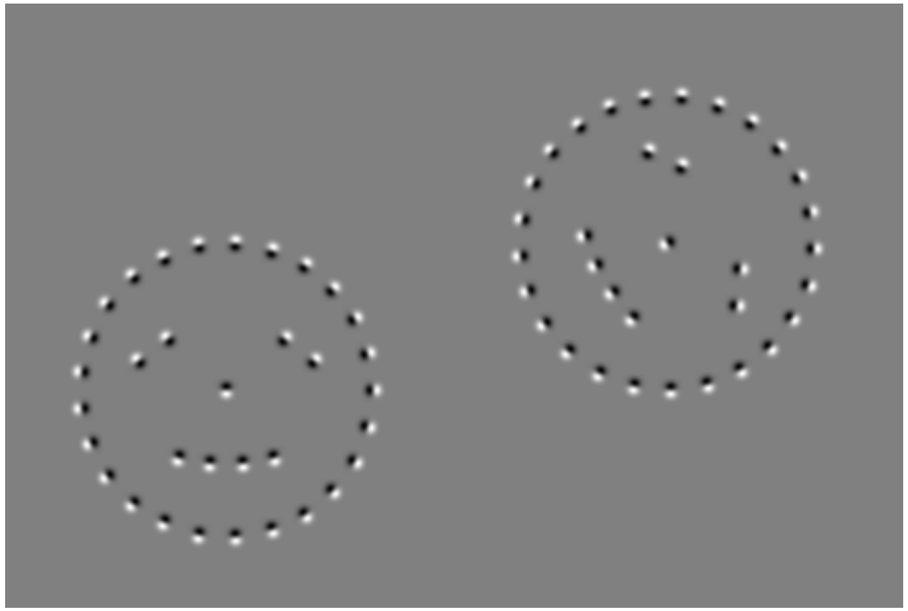
Regular group convolutions revisited

Group convolution ($G = \mathbb{R}^d \rtimes H$):

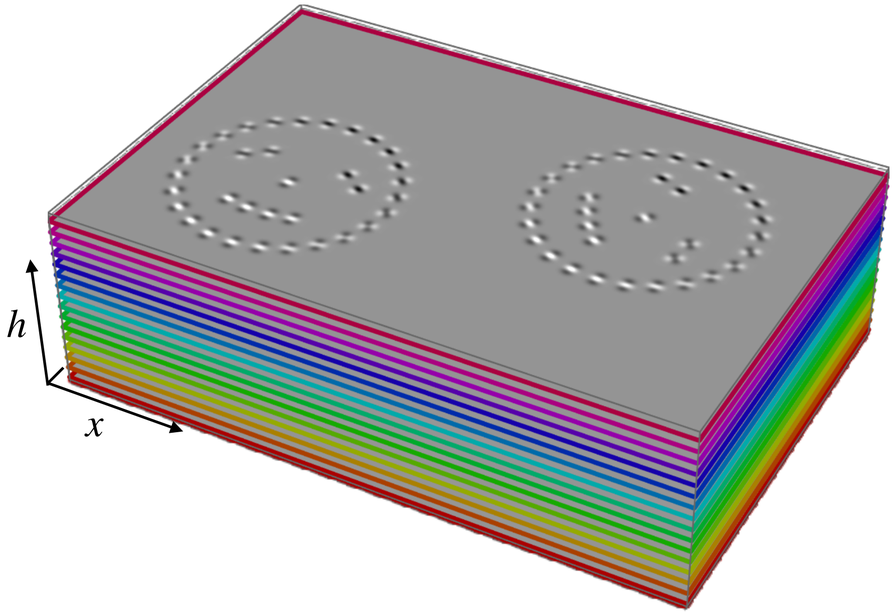
(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)}$$

$\tilde{\star}$



$=$



2D convolution kernel

2D input feature map

SE(2) output feature map

Regular group convolutions revisited

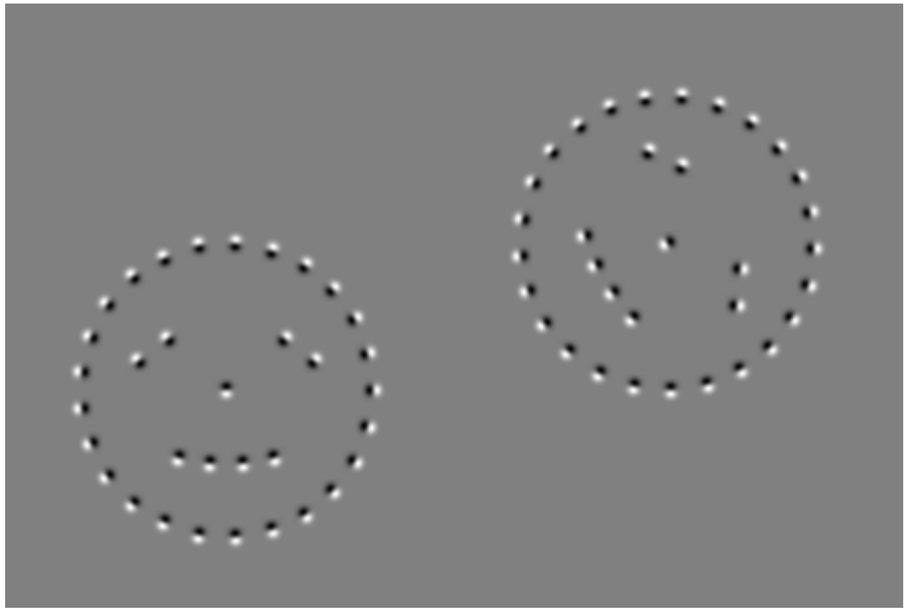
Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

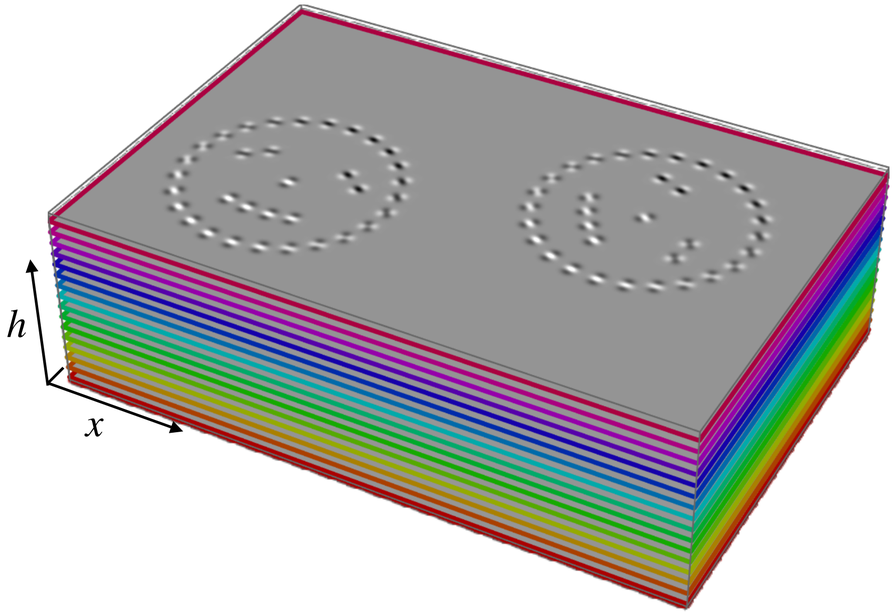
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}')f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

$\tilde{\star}$



$=$



2D convolution kernel

2D input feature map

SE(2) output feature map

Regular group convolutions revisited

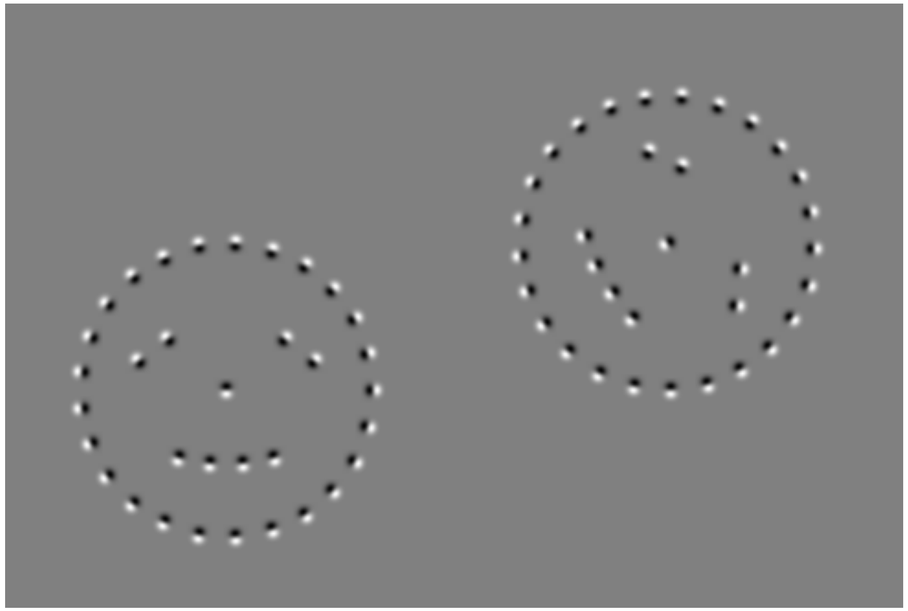
Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

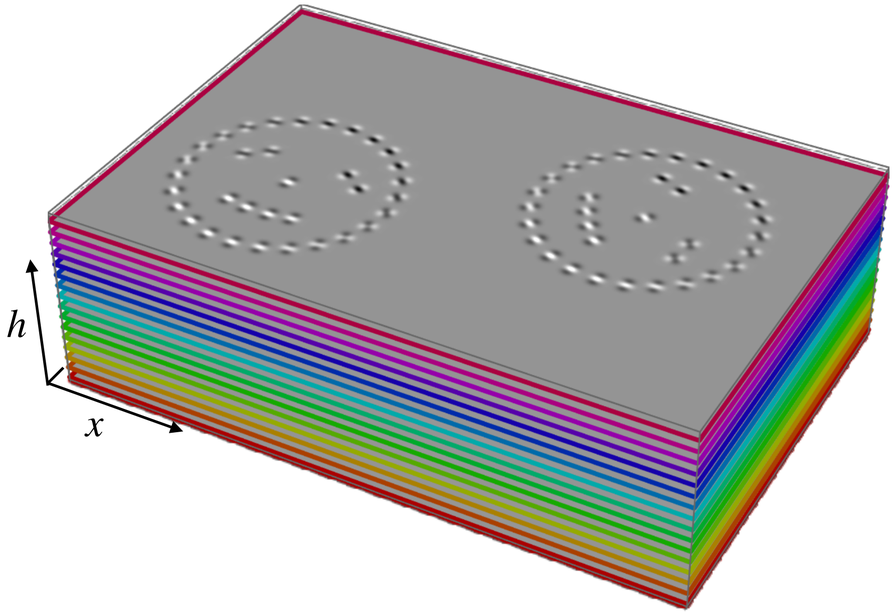
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$\tilde{\star}$



=



2D convolution kernel

2D input feature map

SE(2) output feature map

Regular group convolutions revisited

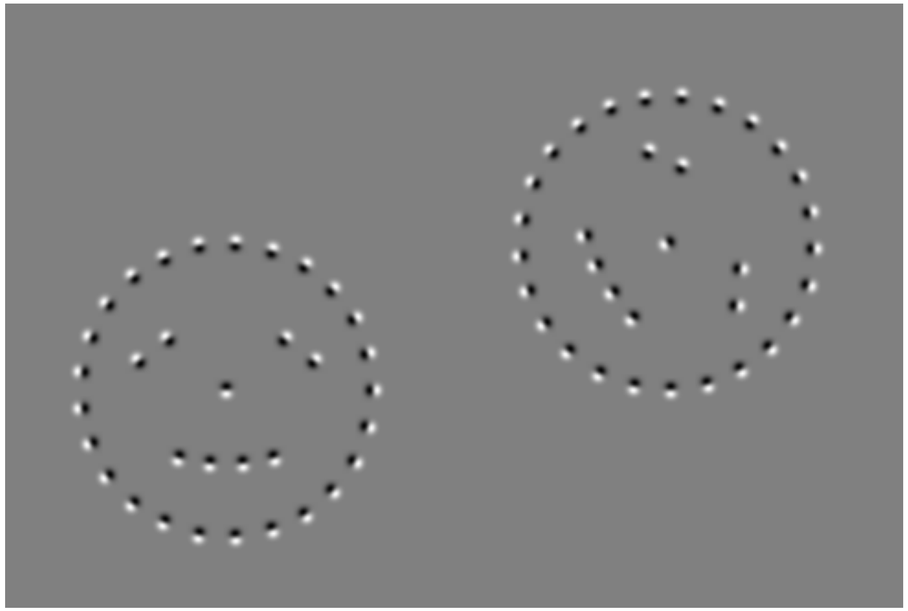
Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

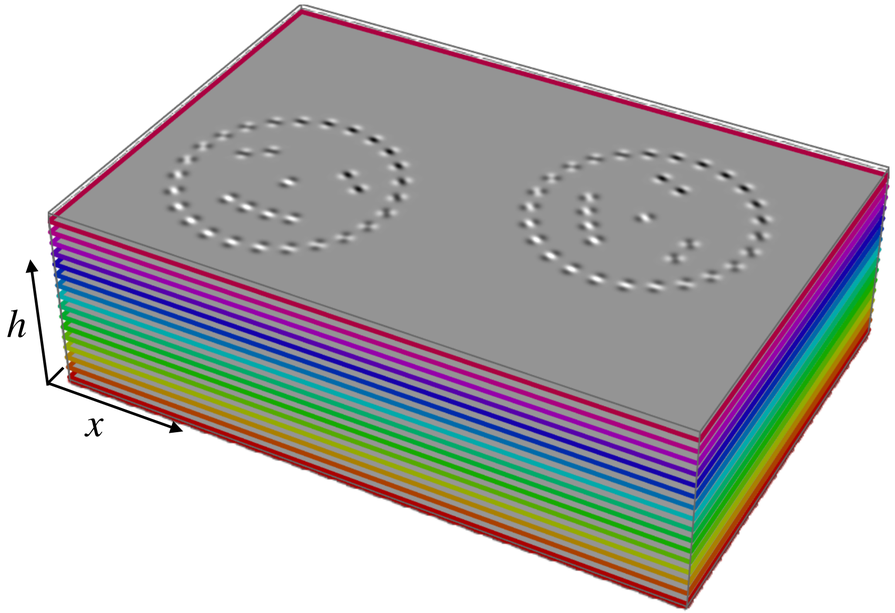
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}')f(\mathbf{x}')\mathrm{d}\mathbf{x}' = \int_{\mathbb{R}^2} k(\mathbf{R}_\theta^{-1}(\mathbf{x}'-\mathbf{x}))f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

$\tilde{\star}$



$=$



2D convolution kernel

2D input feature map

SE(2) output feature map

Regular group convolutions revisited

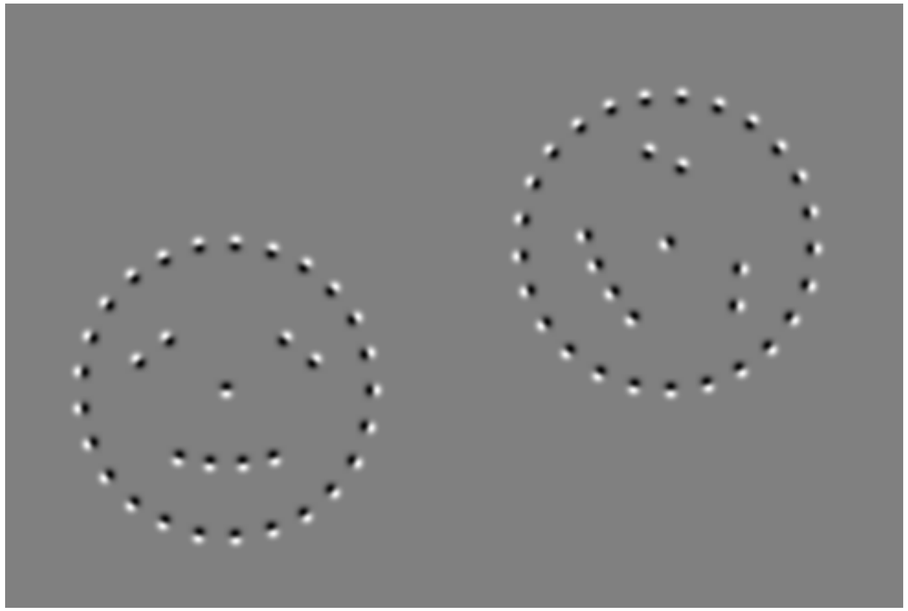
Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

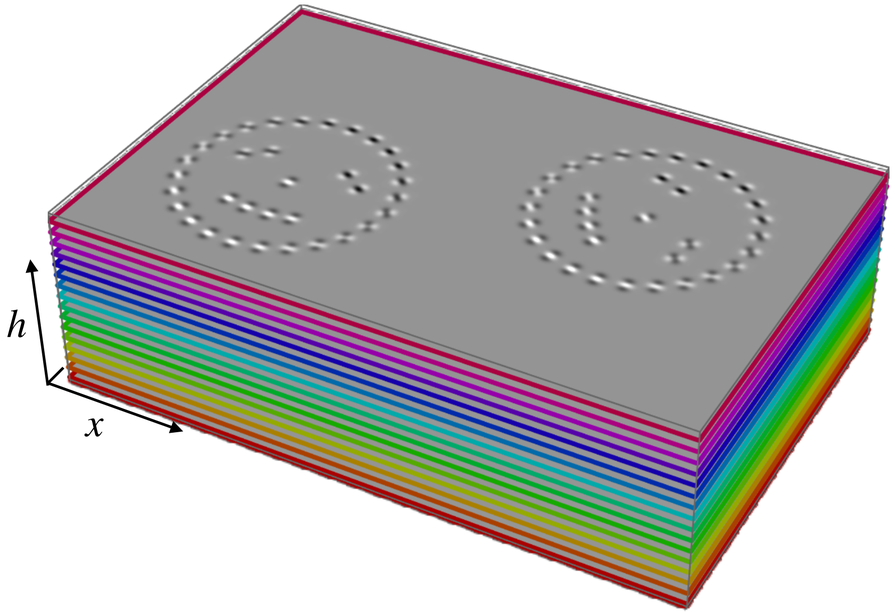
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}' \qquad = \int_{\mathbb{R}^2} k_\theta(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$\tilde{\star}$



=



2D convolution kernel

2D input feature map

SE(2) output feature map

Regular group convolutions revisited

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

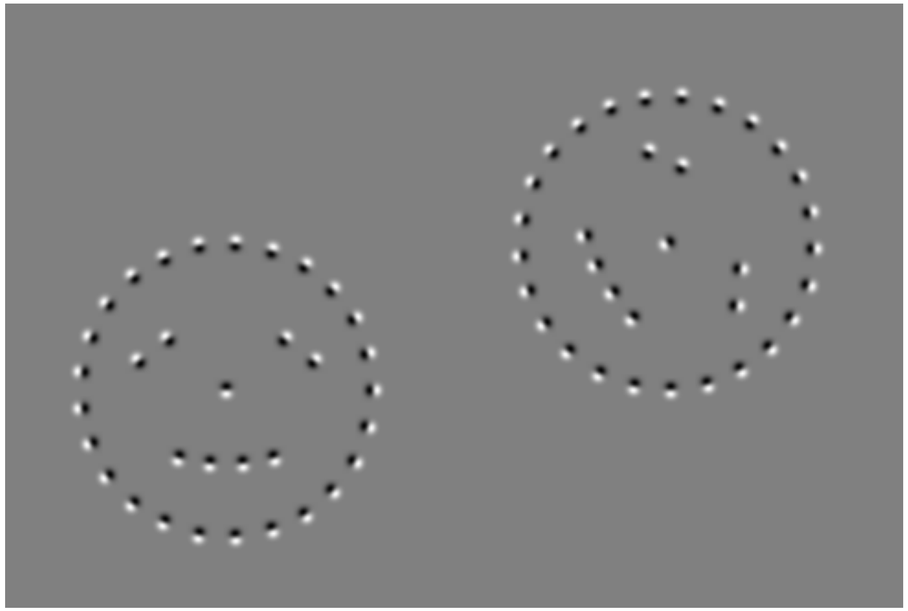
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}' \qquad = \int_{\mathbb{R}^2} k_\theta(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

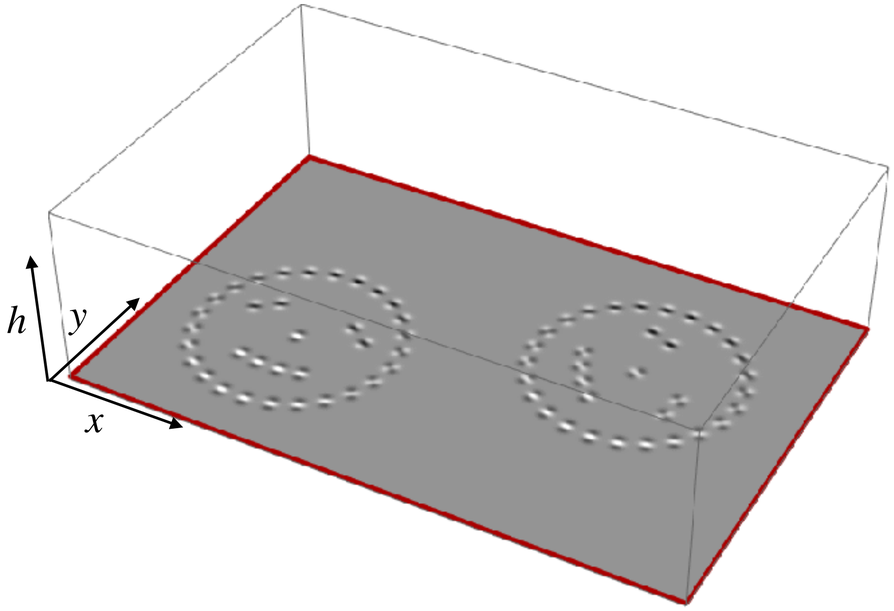
---

translation
"template matching"

$\star_{\mathbb{R}^2}$



$=$



Rotated
2D convolution kernel

2D input feature map

SE(2) output feature map

Regular group convolutions revisited

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

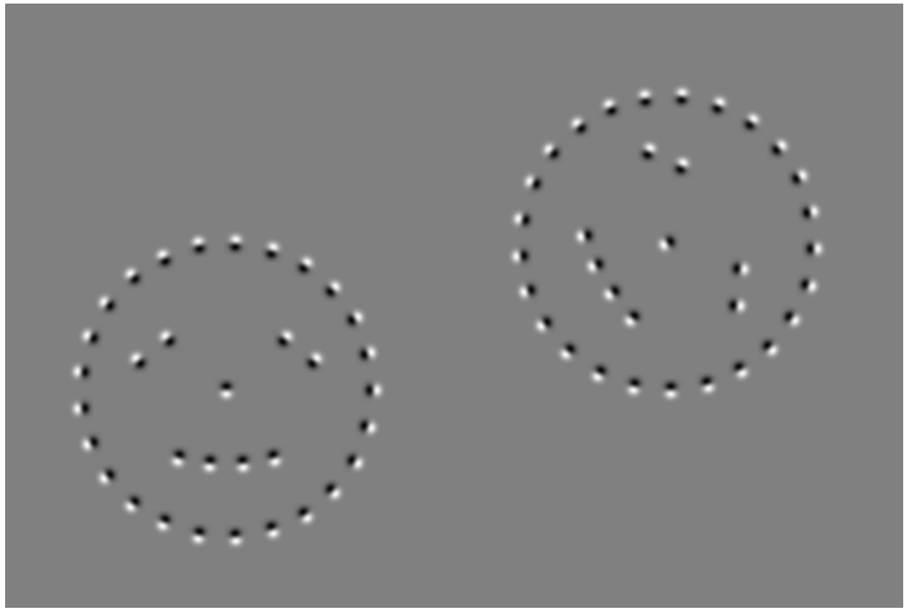
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}' \qquad = \int_{\mathbb{R}^2} k_\theta(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

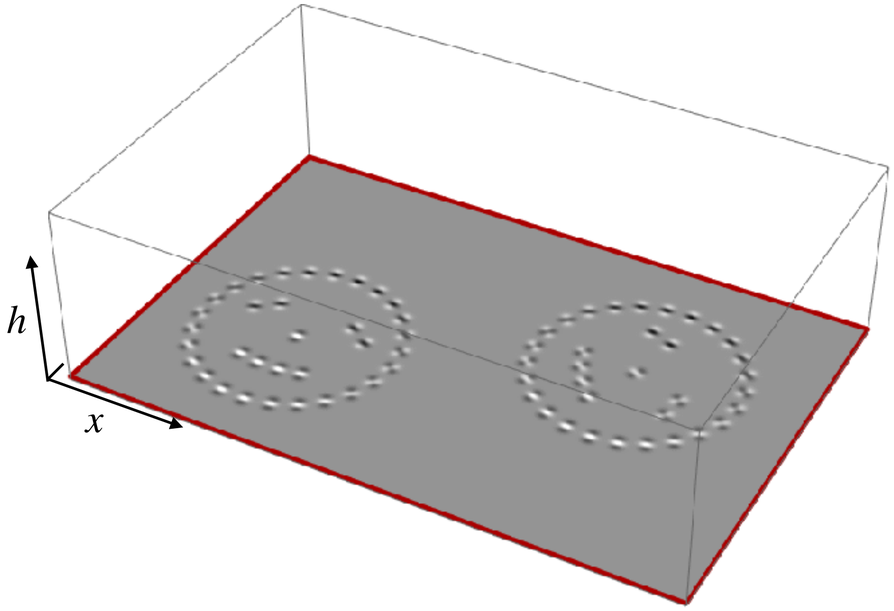
---

translation
"template matching"

$\star_{\mathbb{R}^2}$



$=$



Rotated
2D convolution kernel

2D input feature map

SE(2) output feature map

Regular group convolutions revisited

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

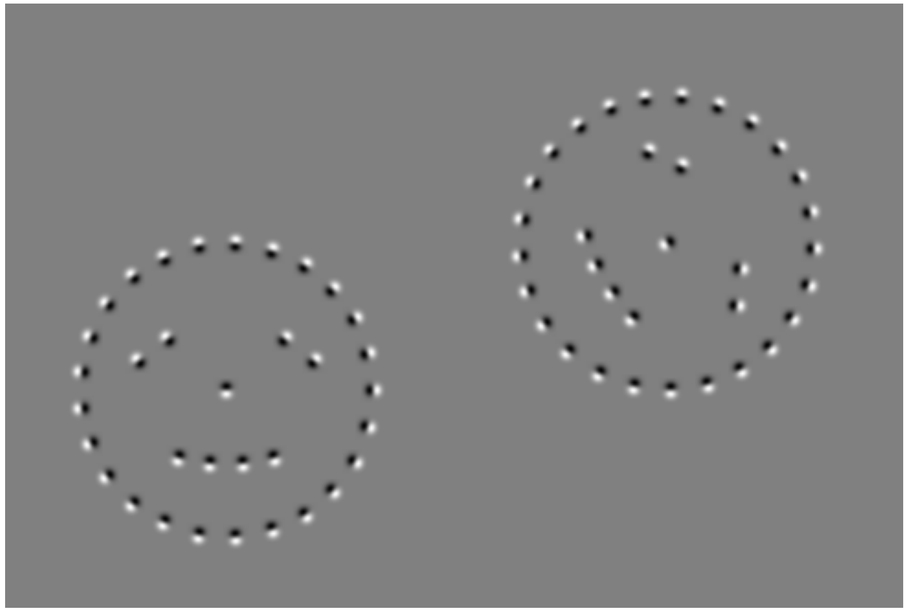
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}')f(\mathbf{x}')\mathrm{d}\mathbf{x}' \qquad = \int_{\mathbb{R}^2} k_\theta(\mathbf{x}' - \mathbf{x})f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

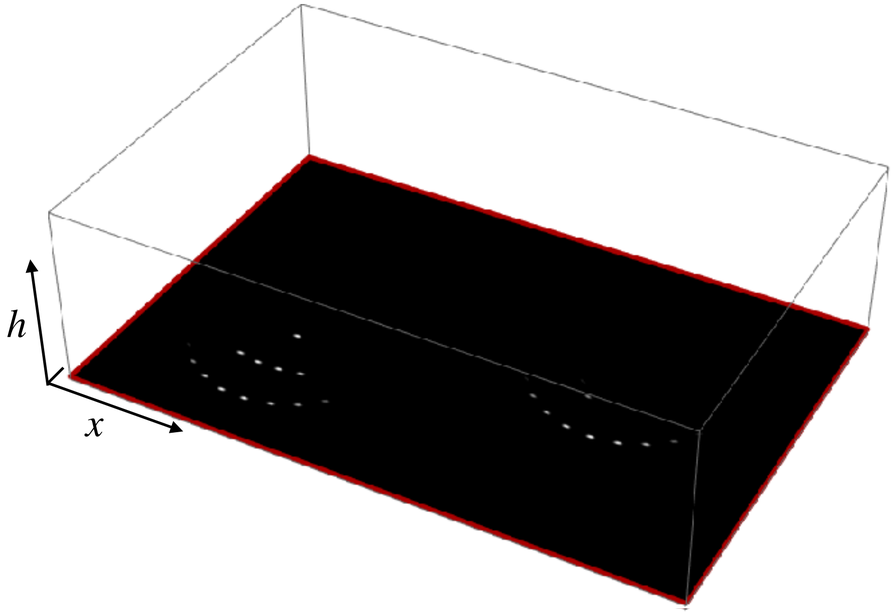
---

translation
"template matching"

$\star_{\mathbb{R}^2}$



$=$



Rotated
2D convolution kernel

2D input feature map

SE(2) output feature map (after ReLU)

Regular group convolutions revisited

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

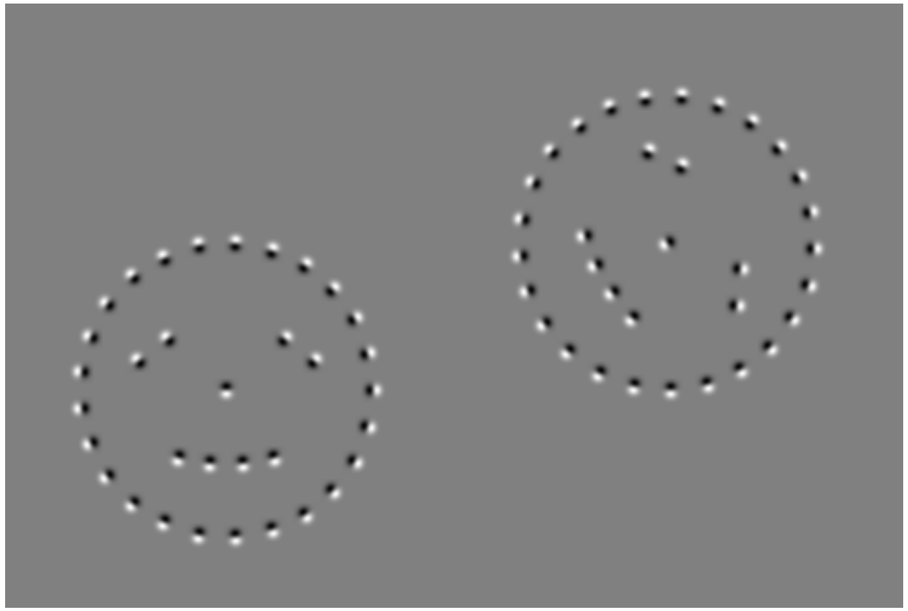
$$(k \tilde{\star} f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)} k \, , f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)} k \, , f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}' = \int_{\mathbb{R}^2} k_\theta(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

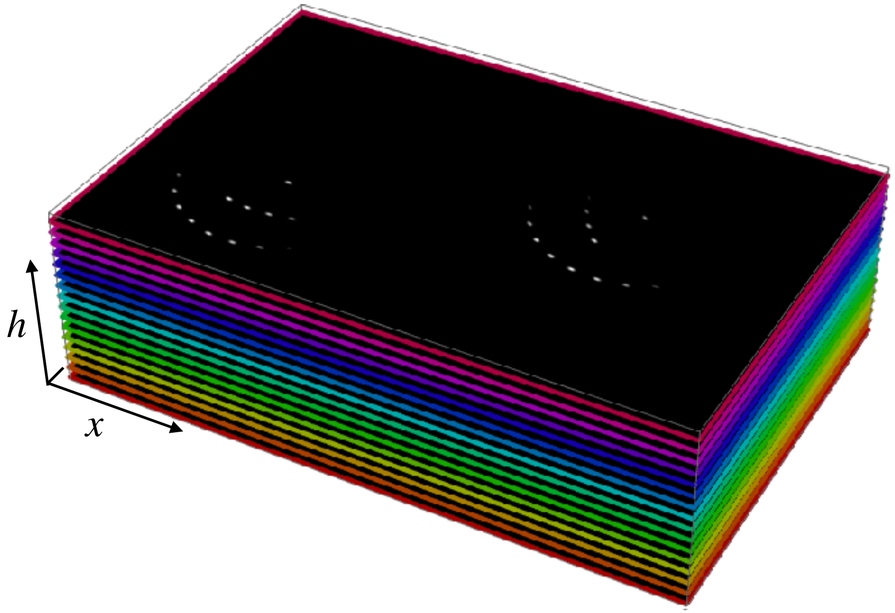
---

translation
"template matching"

$\star_{\mathbb{R}^2}$



$=$



Rotated
2D convolution kernel

2D input feature map

SE(2) output feature map (after ReLU)

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

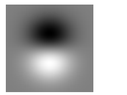
$$(k \,\tilde{\star}\, f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

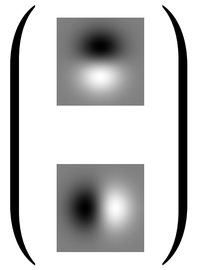
Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \,\tilde{\star}\, f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$k(\mathbf{x} \,|\, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^{\dagger} Y(\mathbf{x})$$



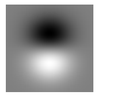


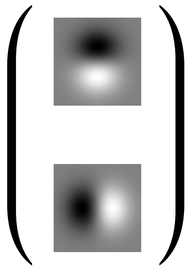
Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \,\tilde{\star}\, f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$k(h^{-1}\,\mathbf{x} \,|\, \hat{\mathbf{w}}) = (\rho(h)\hat{\mathbf{w}})^{\dagger} \quad Y(\mathbf{x})$$





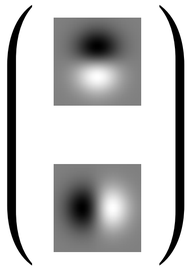
Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \tilde{\star} f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$k(h^{-1}\,\mathbf{x} \,|\, \hat{\mathbf{w}}) = (\rho(h)\hat{\mathbf{w}})^\dagger \quad Y(\mathbf{x})$$



Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \tilde{\star} f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger \int_{\mathbb{R}^d} Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$k(h^{-1}\,\mathbf{x} \,|\, \hat{\mathbf{w}}) = (\rho(h)\hat{\mathbf{w}})^\dagger \quad Y(\mathbf{x})$$

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

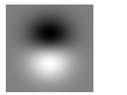
$$(k \tilde{\star} f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

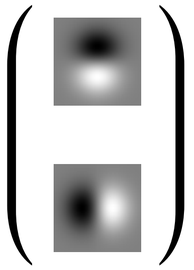
$$= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger \int_{\mathbb{R}^d} Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger\ \hat{f}^Y(\mathbf{x})$$

$$k(h^{-1}\,\mathbf{x} \,|\, \hat{\mathbf{w}}) = (\rho(h)\hat{\mathbf{w}})^\dagger \quad Y(\mathbf{x})$$





Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \tilde{\star} f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger \int_{\mathbb{R}^d} Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger\ \hat{f}^Y(\mathbf{x})$$

$$k(h^{-1}\,\mathbf{x}\,|\,\hat{\mathbf{w}}) = (\rho(h)\hat{\mathbf{w}})^\dagger \quad Y(\mathbf{x})$$

! Freeman, W. T., & Adelson, E. H. (1991). The design and use of steerable filters. IEEE Transactions on Pattern analysis and machine intelligence, 13(9), 891-906.

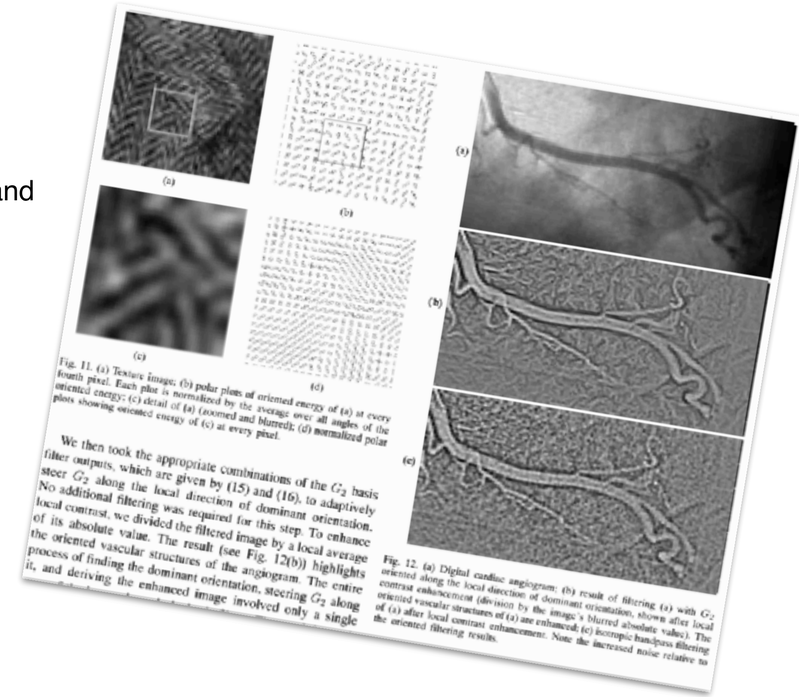


Fig. 11. (a) Texture image. (b) polar plots of oriented energy of (a) at every fourth pixel. Each plot is normalized by the average over all angles of the oriented energy; (c) detail of (a) (zoomed and blurred); (d) normalized polar plots showing oriented energy of (c) at every pixel.

We then took the appropriate combinations of the G_2 basis filter outputs, which are given by (15) and (16), to adaptively steer G_2 along the local direction of dominant orientation. No additional filtering was required for this step. To enhance local contrast, we divided the filtered image by a local average of its absolute value. The result (see Fig. 12(b)) highlights the oriented vascular structures of the angiogram. The entire process of finding the dominant orientation, steering G_2 along it, and deriving the enhanced image involved only a single

Fig. 12. (a) Digital cardiac angiogram. (b) result of filtering (a) with G_2 oriented along the local direction of dominant orientation, shown after local contrast enhancement (division by the image's blurred absolute value). The oriented vascular structures of (a) are enhanced. (c) Isotropic bandpass filtering of (a) after local contrast enhancement. Note the increased noise relative to the oriented filtering results.

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

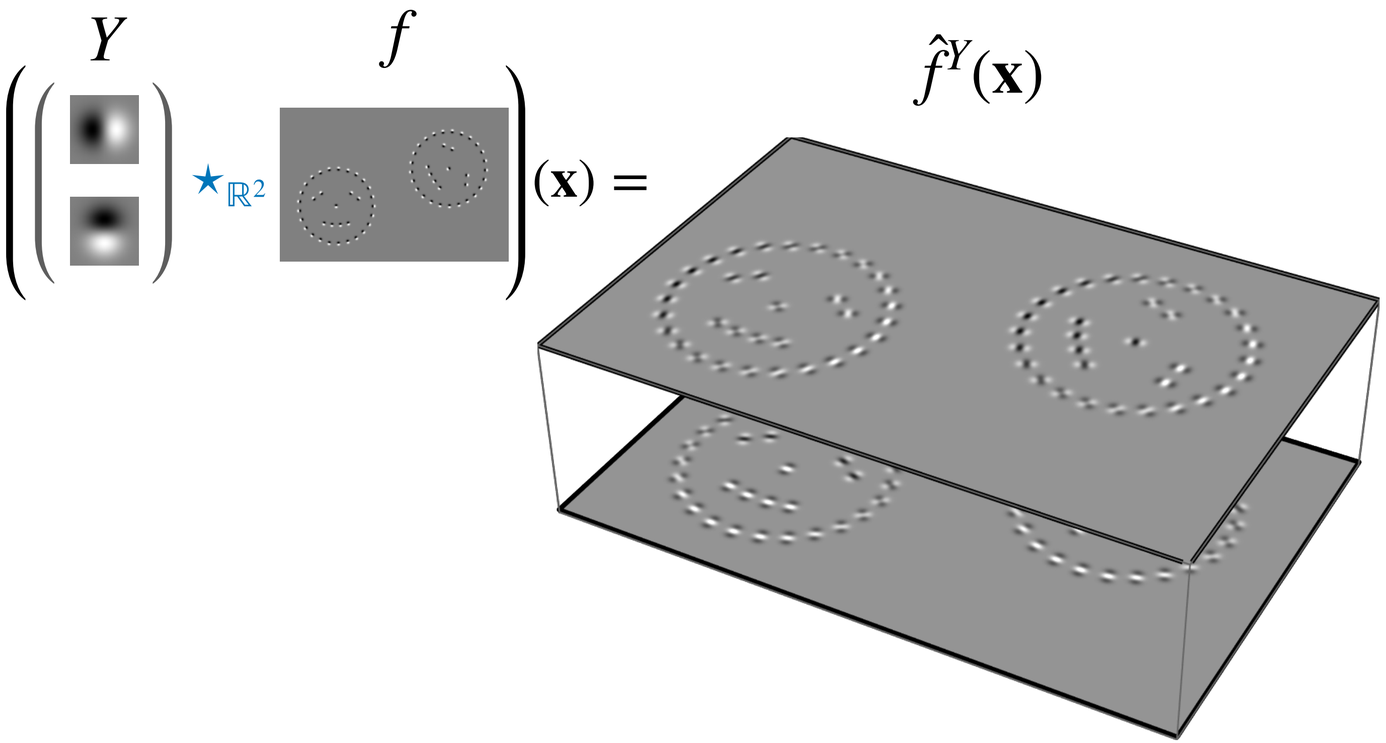
$$(k \,\tilde{\star}\, f)(\mathbf{x}, \theta) = (\rho(\mathbf{R}_\theta)\hat{\mathbf{w}})^\dagger \, \hat{f}^Y(\mathbf{x})$$

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

$$(k \,\tilde{\star}\, f)(\mathbf{x}, \theta) = (\rho(\mathbf{R}_\theta)\hat{\mathbf{w}})^\dagger \, \hat{f}^Y(\mathbf{x})$$

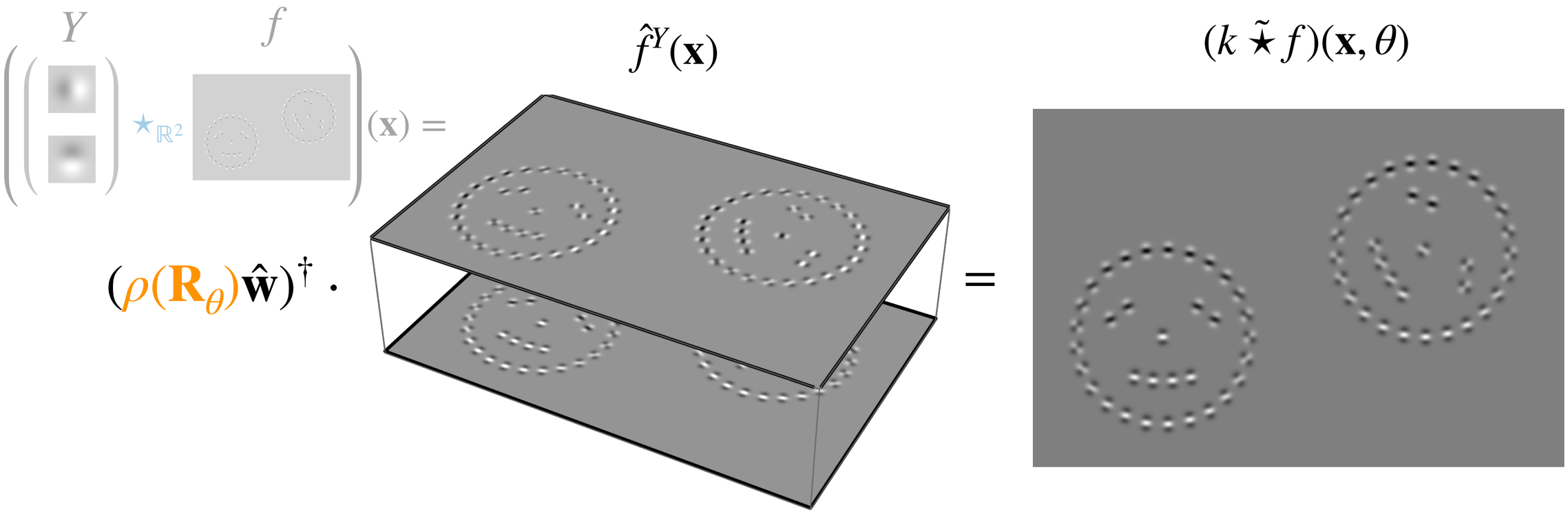


Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

$$(k \,\tilde{\star}\, f)(\mathbf{x}, \theta) = (\rho(\mathbf{R}_\theta)\hat{\mathbf{w}})^\dagger \, \hat{f}^Y(\mathbf{x})$$



Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \tilde{\star} f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \mid \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger \int_{\mathbb{R}^d} Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger\ \hat{f}^Y(\mathbf{x})$$

$$k(h^{-1}\,\mathbf{x} \mid \hat{\mathbf{w}}) = (\rho(h^{-1})\hat{\mathbf{w}})^T\ Y(\mathbf{x})$$

! Freeman, W. T., & Adelson, E. H. (1991). The design and use of steerable filters. IEEE Transactions on Pattern analysis and machine intelligence, 13(9), 891-906.

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \tilde{\star} f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \mid \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger \int_{\mathbb{R}^d} Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger\ \hat{f}^Y(\mathbf{x})$$

$$k(h^{-1}\,\mathbf{x} \mid \hat{\mathbf{w}}) = (\rho(h^{-1})\hat{\mathbf{w}})^T\ Y(\mathbf{x})$$

! Freeman, W. T., & Adelson, E. H. (1991). The design and use of steerable filters. IEEE Transactions on Pattern analysis and machine intelligence, 13(9), 891-906.

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \,\tilde{\star}\, f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\,\hat{\mathbf{w}})^\dagger \int_{\mathbb{R}^d} Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\,\hat{\mathbf{w}})^\dagger \; \hat{f}^Y(\mathbf{x})$$

$$= \mathrm{tr}(\; \hat{f}^Y(\mathbf{x}) \; \hat{\mathbf{w}}^\dagger \; \rho(h^{-1}) \;)$$

$$k(h^{-1}\,\mathbf{x} \,|\, \hat{\mathbf{w}}) \;=\; (\rho(h^{-1})\hat{\mathbf{w}})^T \; Y(\mathbf{x})$$

! Freeman, W. T., & Adelson, E. H. (1991). The design and use of steerable filters. IEEE Transactions on Pattern analysis and machine intelligence, 13(9), 891-906.

$\mathbf{a}^T \mathbf{b} = \mathrm{tr}(\mathbf{b}\,\mathbf{a}^T)$ and $\rho(h)^\dagger = \rho(h^{-1})$

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \,\tilde{\star}\, f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger \int_{\mathbb{R}^d} Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger\ \hat{f}^Y(\mathbf{x})$$

$$= \mathrm{tr}(\ \hat{f}^Y(\mathbf{x})\ \hat{\mathbf{w}}^\dagger\ \rho(h^{-1})\ )$$

$$= \mathrm{tr}(\ \hat{f}(\mathbf{x})\ \rho(h^{-1})\ )$$

$$k(h^{-1}\,\mathbf{x} \,|\, \hat{\mathbf{w}}) = (\rho(h^{-1})\hat{\mathbf{w}})^T\ Y(\mathbf{x})$$

! Freeman, W. T., & Adelson, E. H. (1991). The design and use of steerable filters. IEEE Transactions on Pattern analysis and machine intelligence, 13(9), 891-906.

$\mathbf{a}^T\,\mathbf{b} = \mathrm{tr}(\mathbf{b}\,\mathbf{a}^T)$ and $\rho(h)^\dagger = \rho(h^{-1})$

$\hat{f}(\mathbf{x}) = \hat{f}^Y(\mathbf{x})\ \hat{w}^\dagger$

Lifting convolution with steerable kernel

Group convolution ($G = \mathbb{R}^d \rtimes H$):

$$(k \tilde{\star} f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger \int_{\mathbb{R}^d} Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

$$= (\rho(h)\ \hat{\mathbf{w}})^\dagger\ \hat{f}^Y(\mathbf{x})$$

$$= \mathrm{tr}(\ \hat{f}^Y(\mathbf{x})\ \hat{\mathbf{w}}^\dagger\ \rho(h^{-1})\ )$$

$$= \mathrm{tr}(\ \hat{f}(\mathbf{x})\ \rho(h^{-1})\ )$$

$$= \mathcal{F}_H^{-1}[\ \hat{f}(\mathbf{x})\ ](h)$$

$$k(h^{-1}\,\mathbf{x} \,|\, \hat{\mathbf{w}}) = (\rho(h^{-1})\hat{\mathbf{w}})^T\ Y(\mathbf{x})$$

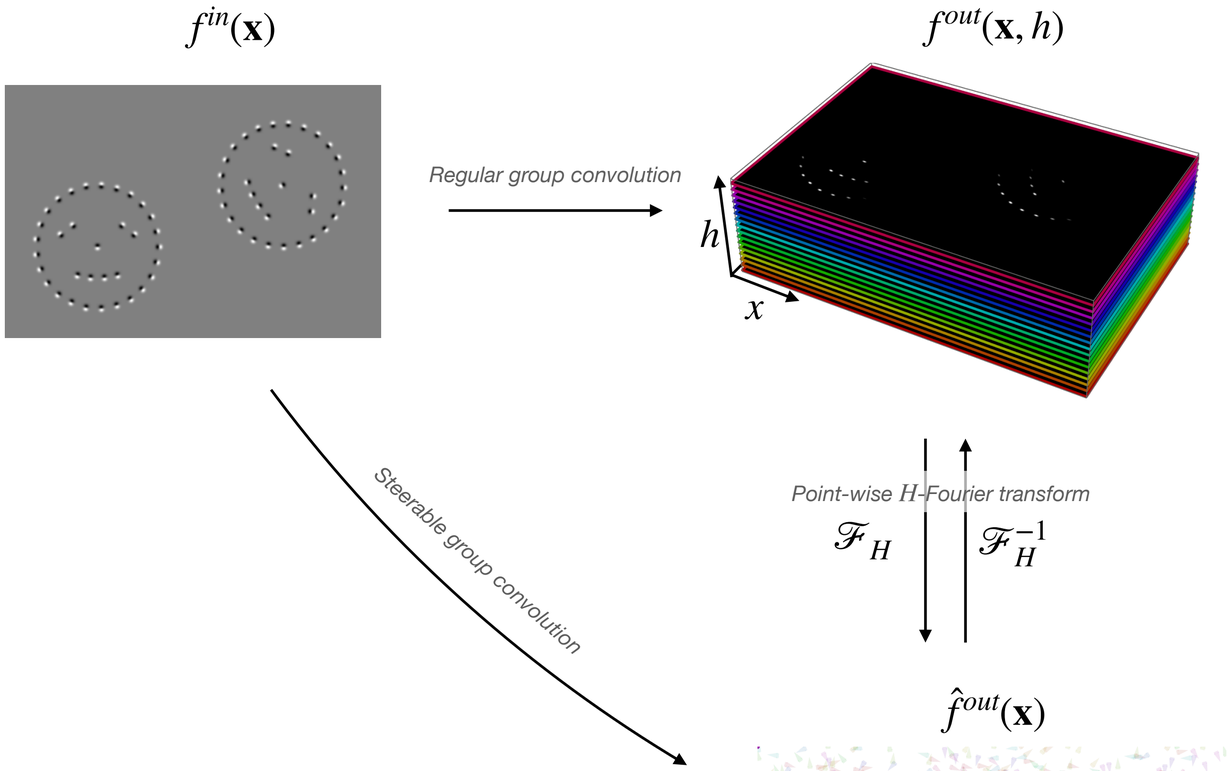
! Freeman, W. T., & Adelson, E. H. (1991). The design and use of steerable filters. IEEE Transactions on Pattern analysis and machine intelligence, 13(9), 891-906.

$\mathbf{a}^T\,\mathbf{b} = \mathrm{tr}(\mathbf{b}\,\mathbf{a}^T)$ and $\rho(h)^\dagger = \rho(h^{-1})$

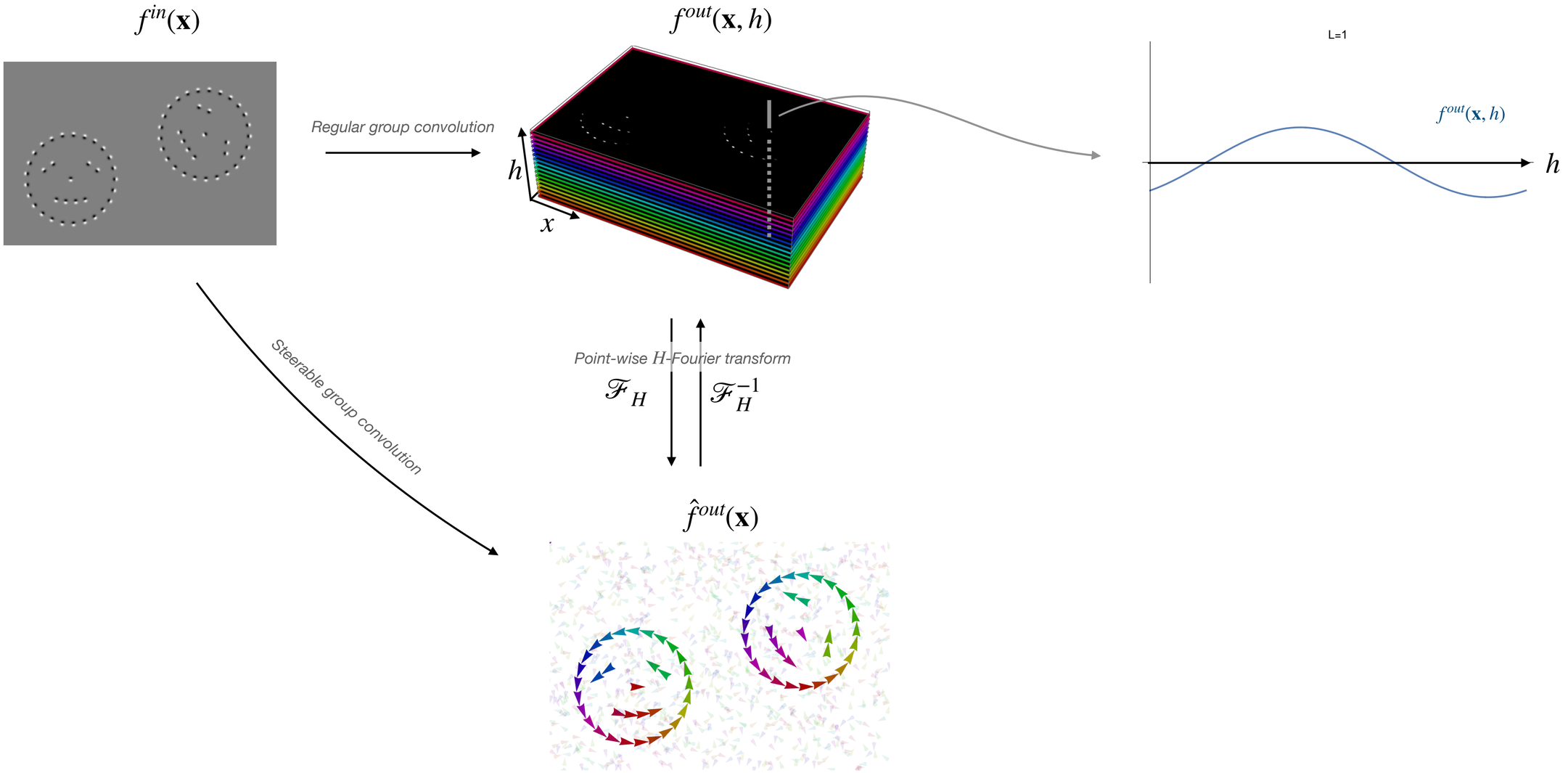
$\hat{f}(\mathbf{x}) = \hat{f}^Y(\mathbf{x})\ \hat{w}^\dagger$

Inverse H-Fourier transform!

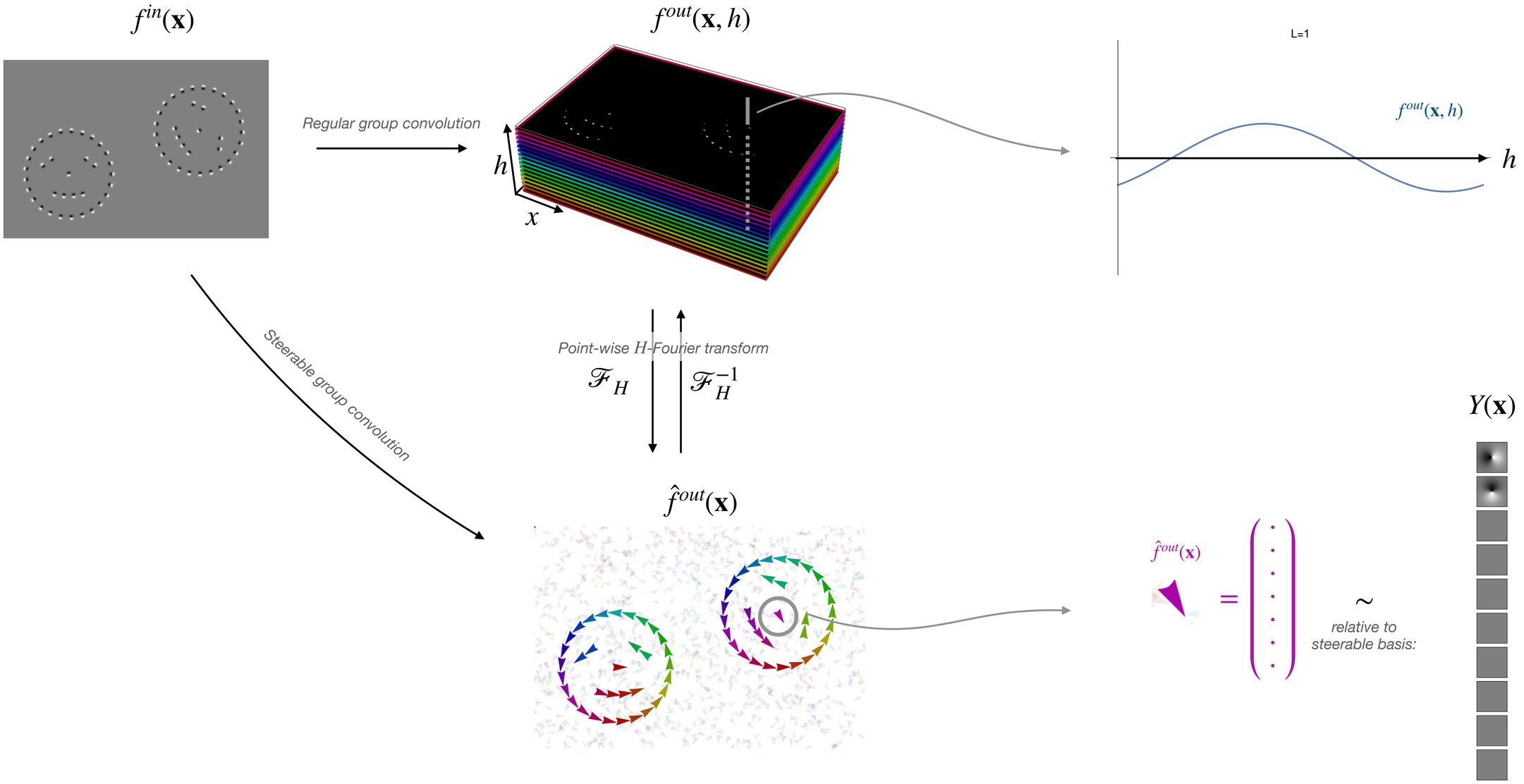
From regular to steerable via a Fourier transform



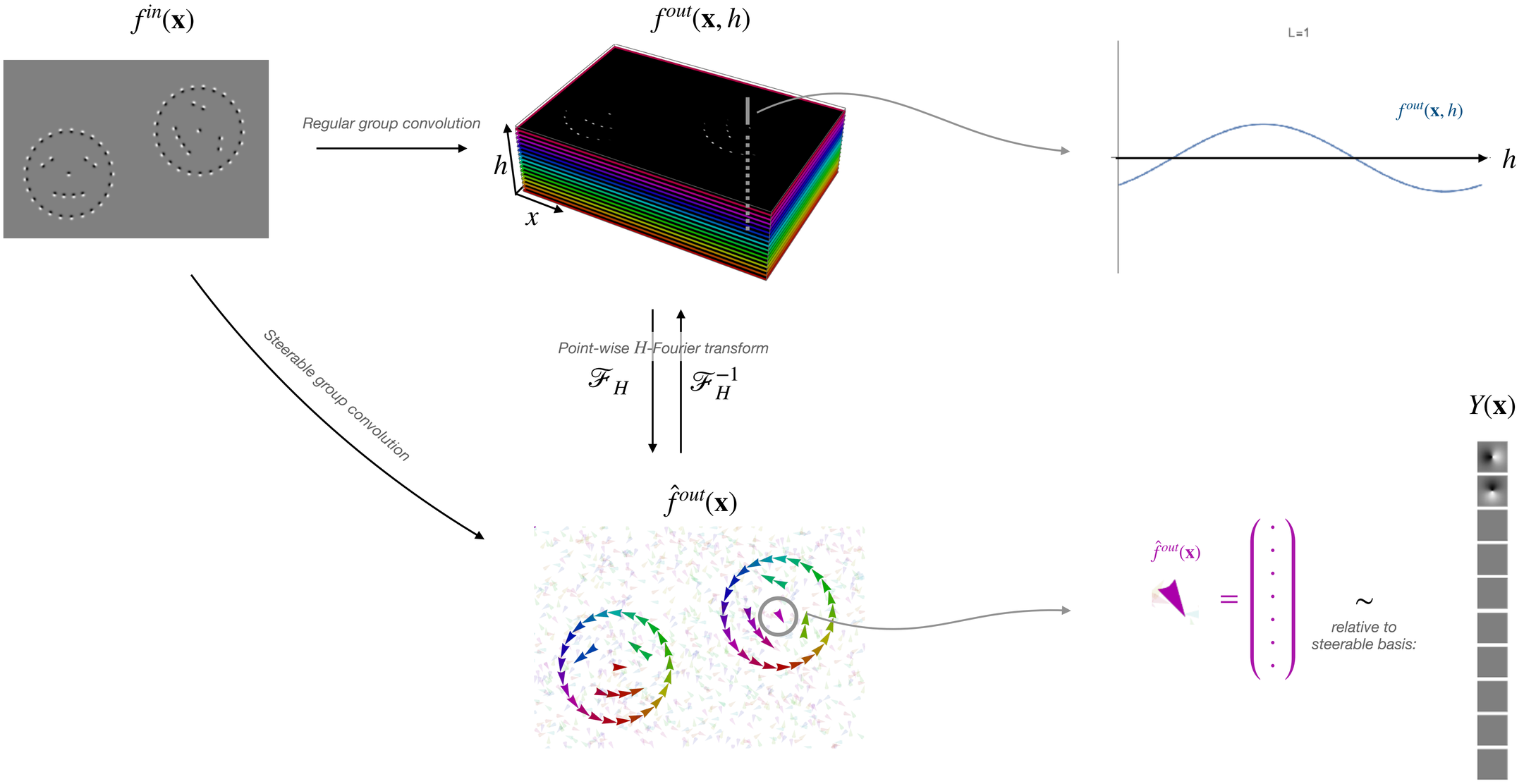
From regular to steerable via a Fourier transform



From regular to steerable via a Fourier transform



From regular to steerable via a Fourier transform



From regular to steerable via a Fourier transform

Regular group convolutions:
Domain expanded feature maps

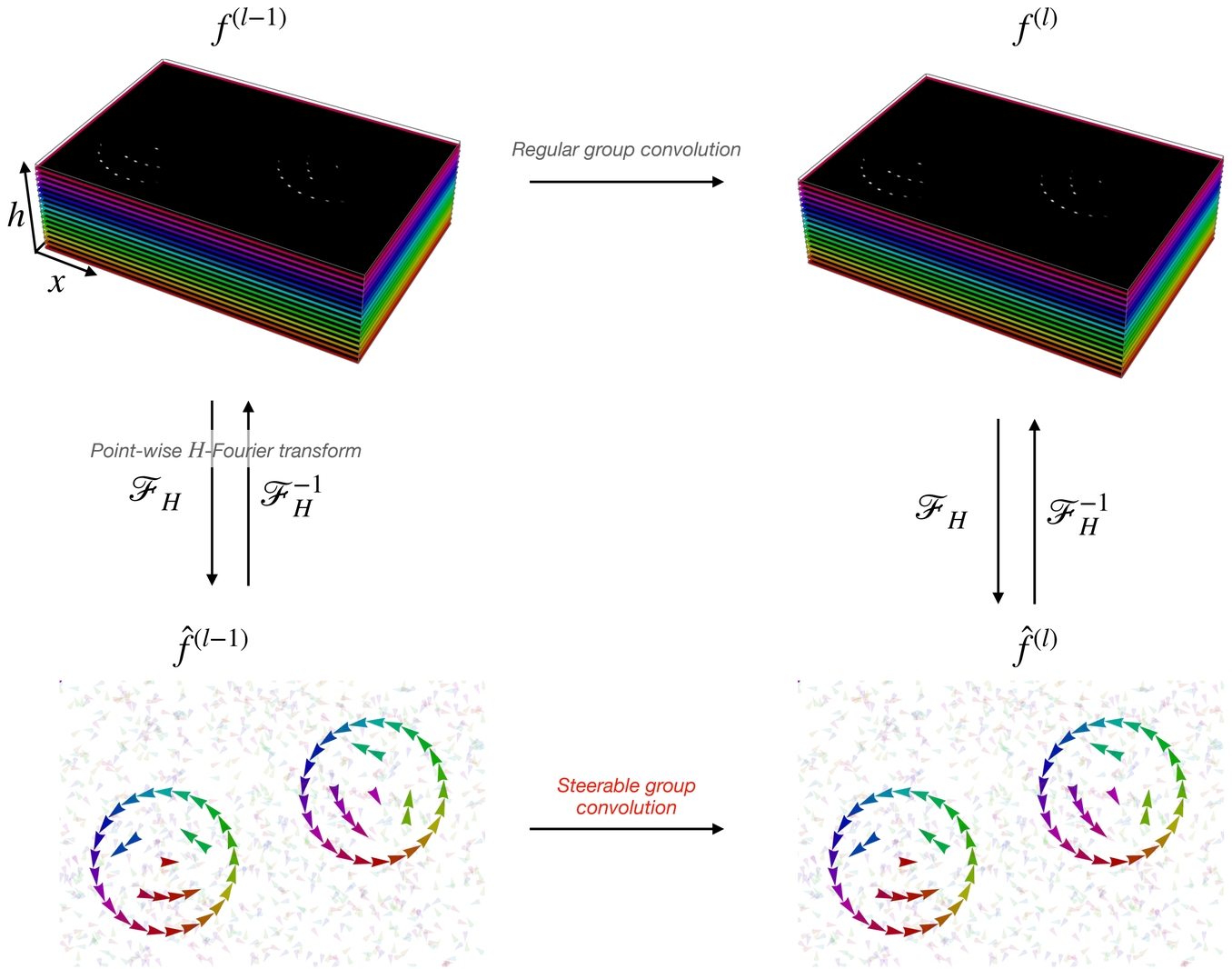
$$f^{(l)} : \mathbb{R}^d \times H \to \mathbb{R}$$

added axis

Steerable group convolutions:
Co-domain expanded feature maps (feature fields)

$$\hat{f}^{(l)} : \mathbb{R}^d \to V_H$$

vector field instead of scalar field
(vectors in V_H transform via group H representations)



UNIVERSITY OF AMSTERDAM

AMLAB Amsterdam Machine Learning Lab

Group Equivariant Deep Learning

Lecture 2 - Steerable group convolutions

Lecture 2.3 - Group Theory | Irreducible representations and Fourier trafo

Preliminaries for steerable feature fields and steerable g-conv intuition

With a focus on SO(2)

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/

Equivalence of group representations

Two representations $\rho^A(g)$ and $\rho^B(g)$ are said to be **equivalent** if they relate via a similarity transform

$$\rho^B(g) = Q^{-1}\,\rho^A(g)\,Q$$

in which Q carries out the change of basis.

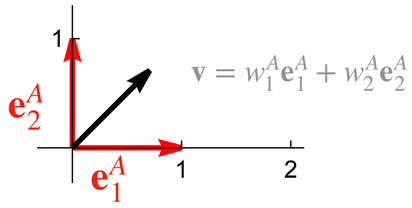
Equivalence of group representations

Two representations $\rho^A(g)$ and $\rho^B(g)$ are said to be **equivalent** if they relate via a similarity transform

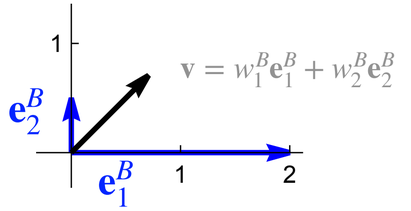
$$\rho^B(g) = Q^{-1}\,\rho^A(g)\,Q$$

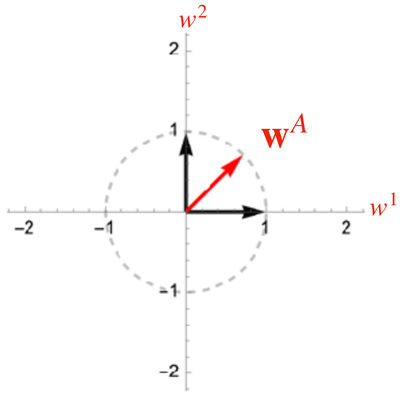
in which Q carries out the change of basis.

Standard basis for $\mathbb{R}^2$



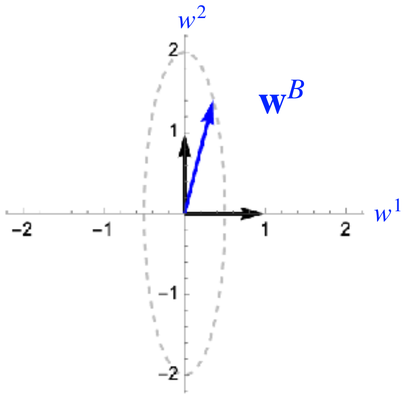
Scaled basis $Q = \begin{pmatrix} 2 & 0 \\ 0 & \frac{1}{2} \end{pmatrix} = (\mathbf{e}_1^B, \mathbf{e}_2^B)$





$$\mathbf{w}^A = Q\,\mathbf{w}^B$$

$$\mathbf{w}^B = Q^{-1}\,\mathbf{w}^A$$



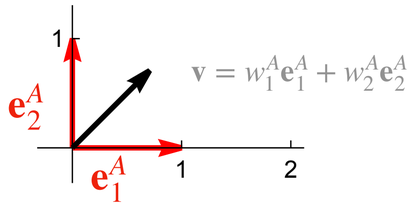
Equivalence of group representations

Two representations $\rho^A(g)$ and $\rho^B(g)$ are said to be **equivalent** if they relate via a similarity transform

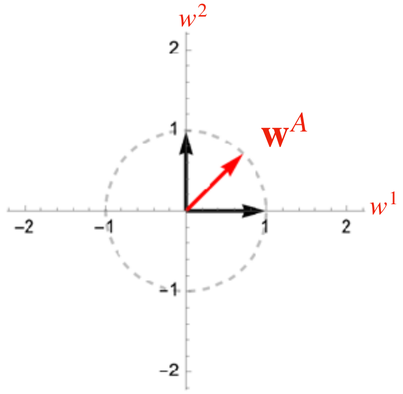
$$\rho^B(g) = Q^{-1}\,\rho^A(g)\,Q$$

in which Q carries out the change of basis.

Standard basis for $\mathbb{R}^2$

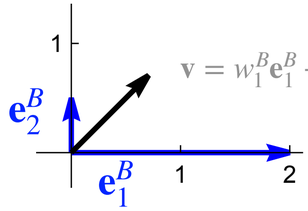


Transforms via $\rho^A(\mathbf{R}_\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

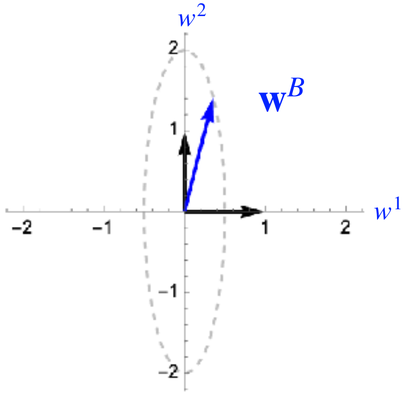


$$\mathbf{w}^A = Q\,\mathbf{w}^B \quad \rightleftarrows \quad \mathbf{w}^B = Q^{-1}\,\mathbf{w}^A$$

Scaled basis $Q = \begin{pmatrix} 2 & 0 \\ 0 & \frac{1}{2} \end{pmatrix} = (\mathbf{e}_1^B, \mathbf{e}_2^B)$



Transforms via $\rho^B(\mathbf{R}_\theta) = \begin{pmatrix} \cos\theta & -\frac{1}{4}\sin\theta \\ 4\sin\theta & \cos\theta \end{pmatrix} = Q^{-1}\,\rho^A(\mathbf{R}_\theta)\,Q$



Equivalence of group representations

A (matrix) representation is called **reducible** if it can written as

$$\rho(g) = Q^{-1} \left(\rho_1(g) \oplus \rho_2(g)\right) Q = Q^{-1} \begin{pmatrix} \rho_1(g) & 0 \\ 0 & \rho_2(g) \end{pmatrix} Q$$

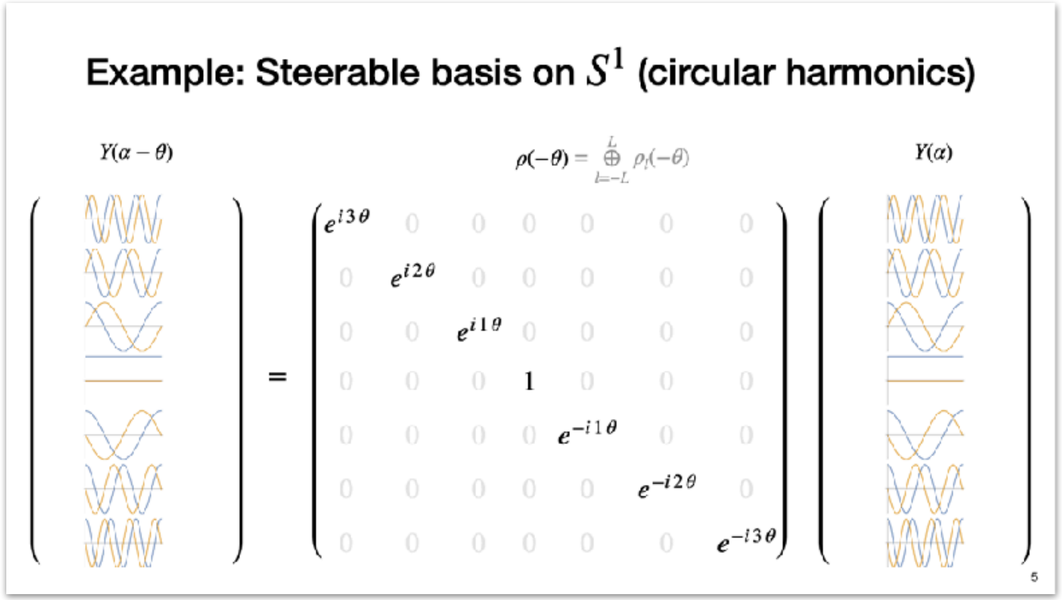
If the blocks $\rho_1(g)$, $\rho_2(g)$ are not reducible they are called **irreducible representations (irreps)**

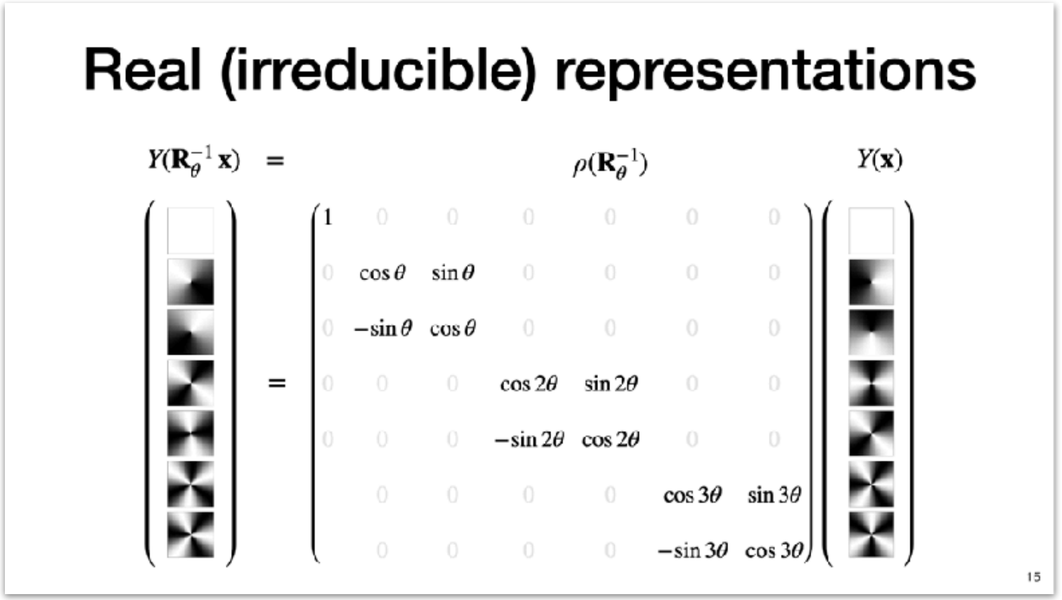
Equivalence of group representations

A (matrix) representation is called **reducible** if it can written as

$$\rho(g) = Q^{-1}\left(\rho_1(g) \oplus \rho_2(g)\right) Q = Q^{-1} \begin{pmatrix} \rho_1(g) & 0 \\ 0 & \rho_2(g) \end{pmatrix} Q$$

If the blocks $\rho_1(g)$, $\rho_2(g)$ are not reducible they are called **irreducible representations (irreps)**





Equivalence of group representations

A (matrix) representation is called reducible if it can written as

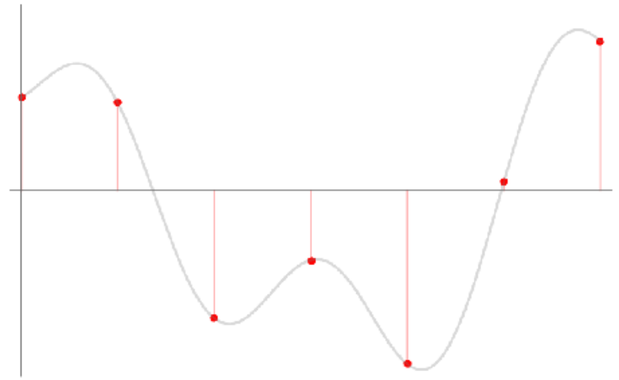
$$\rho(g) = Q^{-1} \left(\rho_1(g) \oplus \rho_2(g)\right) Q = Q^{-1} \begin{pmatrix} \rho_1(g) & 0 \\ 0 & \rho_2(g) \end{pmatrix} Q$$

If the blocks $\rho_1(g)$, $\rho_2(g)$ are not reducible they are called **irreducible representations (irreps)**

$$\mathbf{v} = \begin{pmatrix} 0.21 \\ 0.20 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}$$

Represents discrete signal on S^1
(on C_7 in this case)

$v_n \leftrightarrow f(\alpha_n)$



Equivalence of group representations

A (matrix) representation is called reducible if it can written as

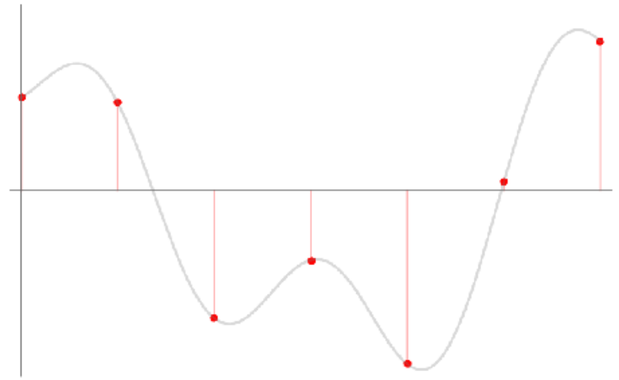
$$\rho(g) = Q^{-1} \left(\rho_1(g) \oplus \rho_2(g)\right) Q = Q^{-1} \begin{pmatrix} \rho_1(g) & 0 \\ 0 & \rho_2(g) \end{pmatrix} Q$$

If the blocks $\rho_1(g)$, $\rho_2(g)$ are not reducible they are called **irreducible representations (irreps)**

$$\mathbf{v} = \begin{pmatrix} 0.21 \\ 0.20 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}$$

Represents discrete signal on S^1
(on C_7 in this case)

$v_n \leftrightarrow f(\alpha_n)$



Equivalence of group representations

A (matrix) representation is called reducible if it can written as

$$\rho(g) = Q^{-1}\left(\rho_1(g) \oplus \rho_2(g)\right) Q = Q^{-1} \begin{pmatrix} \rho_1(g) & 0 \\ 0 & \rho_2(g) \end{pmatrix} Q$$

If the blocks $\rho_1(g)$, $\rho_2(g)$ are not reducible they are called **irreducible representations (irreps)**

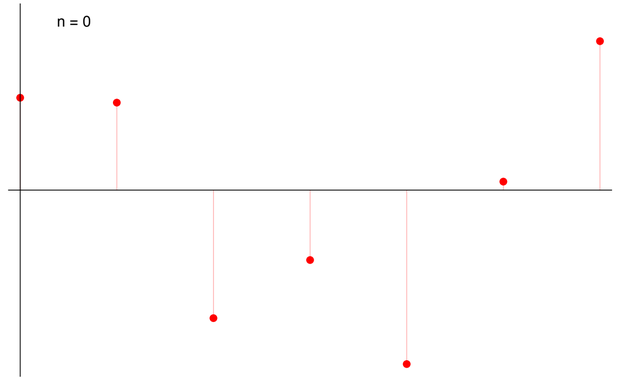
$$\rho(\theta_n) \qquad \mathbf{v}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}^n \begin{pmatrix} 0.21 \\ 0.20 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}$$

Represents discrete signal on S^1
(on C_7 in this case)

$v_n \leftrightarrow f(\alpha_n)$

$\rho(\theta_n)\,\mathbf{v} \leftrightarrow \mathscr{L}_{\theta_n} f(\alpha)$



Equivalence of group representations

A (matrix) representation is called reducible if it can written as

$$\rho(g) = Q^{-1}\left(\rho_1(g) \oplus \rho_2(g)\right) Q = Q^{-1} \begin{pmatrix} \rho_1(g) & 0 \\ 0 & \rho_2(g) \end{pmatrix} Q$$

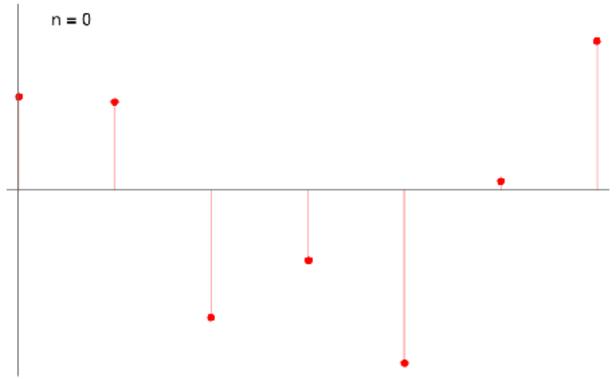
If the blocks $\rho_1(g)$, $\rho_2(g)$ are not reducible they are called **irreducible representations (irreps)**

$$\begin{pmatrix} 0.21 \\ 0.2 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix} = \overset{\rho(\theta_n)}{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}^n} \overset{\mathbf{v}}{\begin{pmatrix} 0.21 \\ 0.20 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}}$$

Represents discrete signal on S^1
(on C_7 in this case)

$v_n \leftrightarrow f(\alpha_n)$

$\rho(\theta_n)\,\mathbf{v} \leftrightarrow \mathscr{L}_{\theta_n} f(\alpha)$



Equivalence of group representations

A (matrix) representation is called reducible if it can written as

$$\rho(g) = Q^{-1}\left(\rho_1(g) \oplus \rho_2(g)\right) Q = Q^{-1} \begin{pmatrix} \rho_1(g) & 0 \\ 0 & \rho_2(g) \end{pmatrix} Q$$

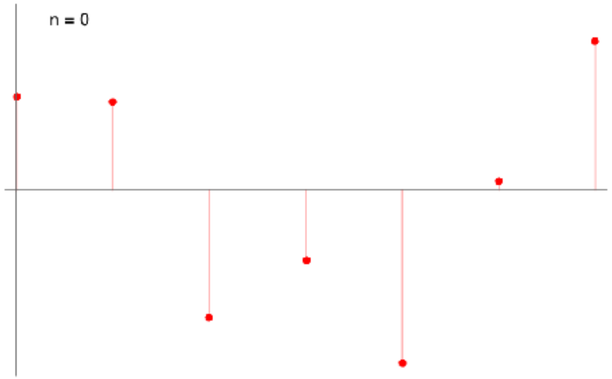
If the blocks $\rho_1(g)$, $\rho_2(g)$ are not reducible they are called **irreducible representations (irreps)**

$$\begin{pmatrix} 0.21 \\ 0.2 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix} = \overset{\rho(\theta_n)}{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}}^{n} \overset{\mathbf{v}}{\begin{pmatrix} 0.21 \\ 0.20 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}}$$

Represents discrete signal on S^1 (on C_7 in this case)

$$v_n \leftrightarrow f(\alpha_n)$$

$$\rho(\theta_n)\,\mathbf{v} \leftrightarrow \mathscr{L}_{\theta_n} f(\alpha)$$



$$Q^{-1} \qquad \oplus_{l=-3}^{3} \rho_l(\theta_n) \qquad Q$$

$$\frac{1}{\sqrt{7}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ e^{\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} & 1 & e^{-\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} \\ e^{-\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} & 1 & e^{-\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} \\ e^{\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} & 1 & e^{-\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} \\ e^{-\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} & 1 & e^{\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} \\ e^{\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} & 1 & e^{\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} \\ e^{-\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} & 1 & e^{\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} \end{pmatrix} \begin{pmatrix} e^{-3i\theta_n} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{-2i\theta_n} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{-i\theta_n} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{i\theta_n} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{2i\theta_n} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{3i\theta_n} \end{pmatrix} \frac{1}{\sqrt{7}} \begin{pmatrix} 1 & e^{-\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} \\ 1 & e^{-\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} \\ 1 & e^{-\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & e^{\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} \\ 1 & e^{\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} \\ 1 & e^{\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} \end{pmatrix}$$

Equivalence of group representations

A (matrix) representation is called reducible if it can written as

$$\rho(g) = Q^{-1}\left(\rho_1(g) \oplus \rho_2(g)\right) Q = Q^{-1}\begin{pmatrix}\rho_1(g) & 0\\ 0 & \rho_2(g)\end{pmatrix} Q$$

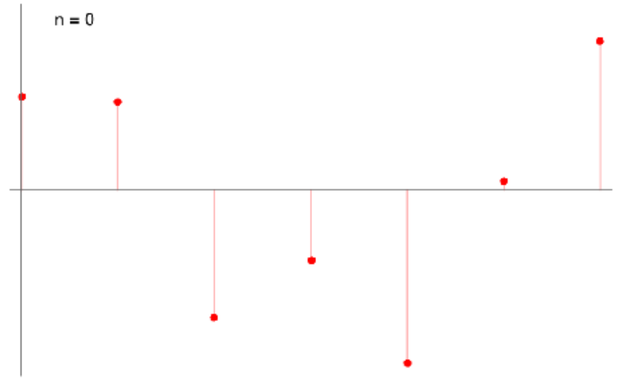
If the blocks $\rho_1(g), \rho_2(g)$ are not reducible they are called **irreducible representations (irreps)**

$$\begin{pmatrix}0.21\\0.2\\-0.29\\-0.16\\-0.39\\0.02\\0.34\end{pmatrix} = \overset{\rho(\theta_n)}{\begin{pmatrix}0&0&0&0&0&0&1\\1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\end{pmatrix}^n} \overset{\mathbf{v}}{\begin{pmatrix}0.21\\0.20\\-0.29\\-0.16\\-0.39\\0.02\\0.34\end{pmatrix}}$$

Represents discrete signal on S^1
(on C_7 in this case)

$v_n \leftrightarrow f(\alpha_n)$

$\rho(\theta_n)\,\mathbf{v} \leftrightarrow \mathscr{L}_{\theta_n} f(\alpha)$

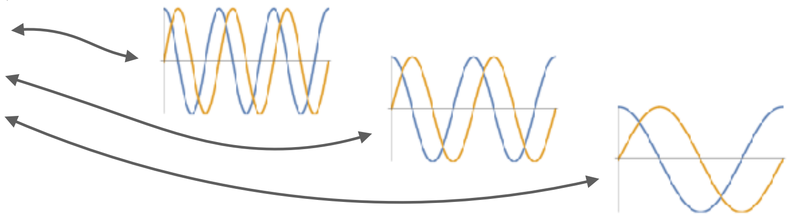


Q^{-1} $\quad \oplus_{l=-3}^{3} \rho_l(\theta_n)$ $\quad Q$

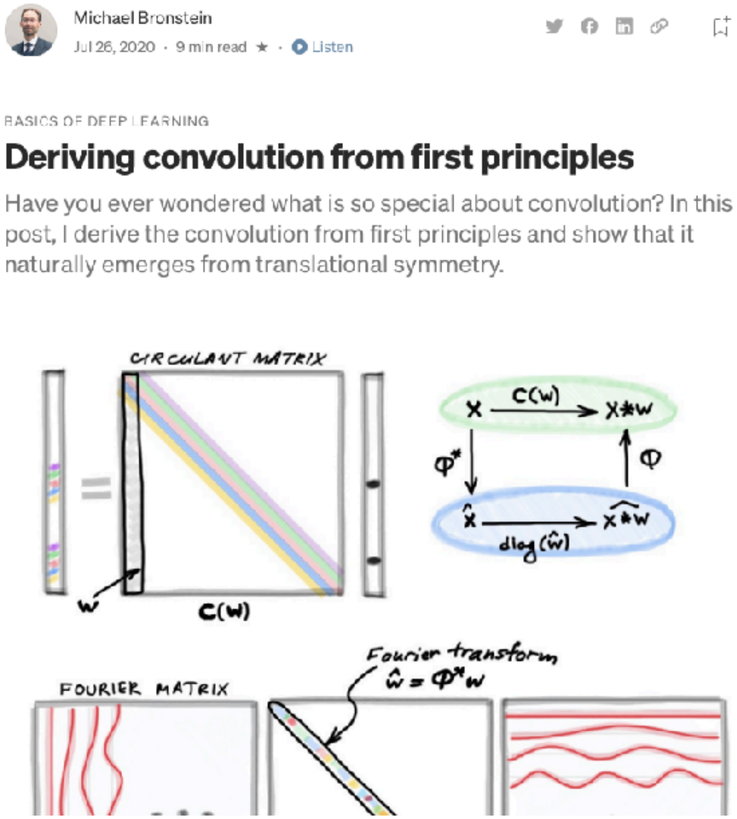
$$\frac{1}{\sqrt{7}}\begin{pmatrix}1&1&1&1&1&1&1\\e^{\frac{6i\pi}{7}}&e^{\frac{4i\pi}{7}}&e^{\frac{2i\pi}{7}}&1&e^{-\frac{2i\pi}{7}}&e^{-\frac{4i\pi}{7}}&e^{-\frac{6i\pi}{7}}\\e^{\frac{2i\pi}{7}}&e^{-\frac{6i\pi}{7}}&e^{\frac{4i\pi}{7}}&1&e^{-\frac{4i\pi}{7}}&e^{\frac{6i\pi}{7}}&e^{\frac{2i\pi}{7}}\\e^{\frac{4i\pi}{7}}&e^{-\frac{2i\pi}{7}}&e^{\frac{6i\pi}{7}}&1&e^{-\frac{6i\pi}{7}}&e^{\frac{2i\pi}{7}}&e^{-\frac{4i\pi}{7}}\\e^{-\frac{4i\pi}{7}}&e^{\frac{2i\pi}{7}}&e^{-\frac{6i\pi}{7}}&1&e^{\frac{6i\pi}{7}}&e^{-\frac{2i\pi}{7}}&e^{\frac{4i\pi}{7}}\\e^{-\frac{2i\pi}{7}}&e^{\frac{6i\pi}{7}}&e^{-\frac{4i\pi}{7}}&1&e^{\frac{4i\pi}{7}}&e^{-\frac{6i\pi}{7}}&e^{-\frac{2i\pi}{7}}\\e^{-\frac{6i\pi}{7}}&e^{-\frac{4i\pi}{7}}&e^{-\frac{2i\pi}{7}}&1&e^{\frac{2i\pi}{7}}&e^{\frac{4i\pi}{7}}&e^{\frac{6i\pi}{7}}\end{pmatrix}$$

$$\begin{pmatrix}e^{-3i\theta_n}&0&0&0&0&0&0\\0&e^{-2i\theta_n}&0&0&0&0&0\\0&0&e^{-i\theta_n}&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&e^{i\theta_n}&0&0\\0&0&0&0&0&e^{2i\theta_n}&0\\0&0&0&0&0&0&e^{3i\theta_n}\end{pmatrix}$$

$$\frac{1}{\sqrt{7}}\begin{pmatrix}1&e^{-\frac{6i\pi}{7}}&e^{\frac{2i\pi}{7}}&e^{-\frac{4i\pi}{7}}&e^{\frac{4i\pi}{7}}&e^{-\frac{2i\pi}{7}}&e^{\frac{6i\pi}{7}}\\1&e^{-\frac{4i\pi}{7}}&e^{\frac{6i\pi}{7}}&e^{\frac{2i\pi}{7}}&e^{-\frac{2i\pi}{7}}&e^{-\frac{6i\pi}{7}}&e^{\frac{4i\pi}{7}}\\1&e^{-\frac{2i\pi}{7}}&e^{-\frac{4i\pi}{7}}&e^{-\frac{6i\pi}{7}}&e^{\frac{6i\pi}{7}}&e^{\frac{4i\pi}{7}}&e^{\frac{2i\pi}{7}}\\1&1&1&1&1&1&1\\1&e^{\frac{2i\pi}{7}}&e^{\frac{4i\pi}{7}}&e^{\frac{6i\pi}{7}}&e^{-\frac{6i\pi}{7}}&e^{-\frac{4i\pi}{7}}&e^{-\frac{2i\pi}{7}}\\1&e^{\frac{4i\pi}{7}}&e^{-\frac{6i\pi}{7}}&e^{-\frac{2i\pi}{7}}&e^{\frac{2i\pi}{7}}&e^{\frac{6i\pi}{7}}&e^{-\frac{4i\pi}{7}}\\1&e^{\frac{6i\pi}{7}}&e^{-\frac{2i\pi}{7}}&e^{\frac{4i\pi}{7}}&e^{-\frac{4i\pi}{7}}&e^{\frac{2i\pi}{7}}&e^{-\frac{6i\pi}{7}}\end{pmatrix}$$



Derive Q (the irreps) yourself through an eigendecomposition of the circular shift matrix



Michael Bronstein

Jul 26, 2020 · 9 min read ★ · Listen

BASICS OF DEEP LEARNING

Deriving convolution from first principles

Have you ever wondered what is so special about convolution? In this post, I derive the convolution from first principles and show that it naturally emerges from translational symmetry.

https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028

DISCOVERING TRANSFORMS: A TUTORIAL ON CIRCULANT MATRICES, CIRCULAR CONVOLUTION, AND THE DISCRETE FOURIER TRANSFORM

BASSAM BAMIEH*

4.1. Construction of Eigenvectors/Eigenvalues of S^*. Let w be an eigenvector (with eigenvalue λ) of the shift operator S^*. Note that it is also an eigenvector (with eigenvalue λ^l) of any power $(S^*)^l$ of S^*. Applying the definition (3.3) to the relation $S^*w = \lambda w$ will reveal that an eigenvector w has a very special structure

$$(4.1)\qquad \begin{array}{rcl} S^*w &=& \lambda w \\ (S^*)^l w &=& \lambda^l w \end{array} \iff \begin{array}{rcll} w_{k+1} &=& \lambda\, w_k, & k \in \mathbb{Z}_n, \\ w_{k+l} &=& \lambda^l\, w_k, & k \in \mathbb{Z}_n,\ l \in \mathbb{Z}, \end{array}$$

i.e. each entry w_{k+1} of w is equal to the previous entry w_k multiplied by the eigenvalue λ. These relations can be used to compute all eigenvectors/eigenvalues of S^*. First, observe that although (4.1) is valid for all $l \in \mathbb{Z}$, this relation "repeats" for $l \geq n$. In particular, for $l = n$ we have for each index k

$$(4.2)\qquad w_{k+n} = \lambda^n w_k \iff w_k = \lambda^n w_k$$

since $k + n \equiv_n k$. Now since the vector $w \neq 0$, then for at least one index k, $w_k \neq 0$, and the last equality implies that $\lambda^n = 1$, i.e. any eigenvalue of S must be an *nth root of unity*

$$\boxed{\lambda^n = 1 \iff \lambda = \rho_m := e^{i\frac{2\pi}{n}m},\ m \in \mathbb{Z}_n.}$$

https://arxiv.org/pdf/1805.05533.pdf

Overview: Block-diagonalization via Fourier transform

	Finite-dimensional vectors $\mathbf{v} \in \mathbb{R}^d$	**Infinite-dimensional vectors** $f \in \mathbb{L}_2(G)$
Regular representation	$\rho(g_n)\ \mathbf{v}$	$\mathscr{L}_g f$
Decomposed into irreps	$Q^{-1}\left[\oplus_{l=-L}^{L} \rho_l(g_n)\right] Q$	$\mathscr{F}_G^{-1} \circ \left[\oplus_{l=-L}^{L} \rho_l(g)\right] \circ \mathscr{F}_G$
Fourier transform	$Q\,\mathbf{v} = \frac{1}{\sqrt{7}} \begin{pmatrix} 1 & e^{-\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} \\ 1 & e^{-\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} \\ 1 & e^{-\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & e^{\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} \\ 1 & e^{\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} \\ 1 & e^{\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} \end{pmatrix} \begin{pmatrix} 0.21 \\ 0.20 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}$	$[\mathscr{F}_G f]_l = \int_G \rho_l(g) f(g) \mathrm{d}g$
Inverse Fourier transform	$Q^{-1}\,\hat{\mathbf{v}}$	$\mathscr{F}^{-1}[\hat{f}](g) = \sum_l \hat{f}(\rho_l)\rho_l(g^{-1})$
		$= \sum_l d_{\rho_l} \mathrm{tr}\left[\hat{f}(\rho_l)\rho_l(h^{-1})\right]$

General case $d_l \times d_l$ ***matrix irreps of compact groups***

See e.g. Kondor, R., & Trivedi, S. (2018, July). On the generalization of equivariance and convolution in neural networks to the action of compact groups. In International Conference on Machine Learning (pp. 2747-2755). PMLR.

Overview: Block-diagonalization via Fourier transform

	Finite-dimensional vectors $\mathbf{v} \in \mathbb{R}^d$	**Infinite-dimensional vectors** $f \in \mathbb{L}_2(G)$
Regular representation	$\rho(g_n)\ \mathbf{v}$	$\mathscr{L}_g f$
Decomposed into irreps	$Q^{-1}\left[\oplus_{l=-L}^{L} \rho_l(g_n)\right] Q$	$\mathscr{F}_G^{-1} \circ \left[\oplus_{l=-L}^{L} \rho_l(g)\right] \circ \mathscr{F}_G$
Fourier transform	$Q\,\mathbf{v} = \frac{1}{\sqrt{7}} \begin{pmatrix} 1 & e^{-\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} \\ 1 & e^{-\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} \\ 1 & e^{-\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & e^{\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} \\ 1 & e^{\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} \\ 1 & e^{\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} \end{pmatrix} \begin{pmatrix} 0.21 \\ 0.20 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}$	$[\mathscr{F}_G f]_l = \int_G \rho_l(g) f(g) \mathrm{d}g$
Inverse Fourier transform	$Q^{-1}\,\hat{\mathbf{v}}$	$\mathscr{F}^{-1}[\hat{f}](g) = \sum_l \hat{f}(\rho_l)\rho_l(g^{-1})$
		$= \sum_l d_{\rho_l} \mathrm{tr}\left[\hat{f}(\rho_l)\rho_l(h^{-1})\right]$

General case $d_l \times d_l$ ***matrix irreps of compact groups***

See e.g. Kondor, R., & Trivedi, S. (2018, July). On the generalization of equivariance and convolution in neural networks to the action of compact groups. In International Conference on Machine Learning (pp. 2747-2755). PMLR.

Overview: Block-diagonalization via Fourier transform

	Finite-dimensional vectors $\mathbf{v} \in \mathbb{R}^d$	**Infinite-dimensional vectors** $f \in \mathbb{L}_2(G)$
Regular representation	$\rho(g_n)\ \mathbf{v}$	$\mathscr{L}_g f$
Decomposed into irreps	$Q^{-1}\left[\oplus_{l=-L}^{L} \rho_l(g_n)\right] Q$	$\mathscr{F}_G^{-1} \circ \left[\oplus_{l=-L}^{L} \rho_l(g)\right] \circ \mathscr{F}_G$
Fourier transform	$Q\,\mathbf{v} = \frac{1}{\sqrt{7}} \begin{pmatrix} 1 & e^{-\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} \\ 1 & e^{-\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} \\ 1 & e^{-\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & e^{\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} \\ 1 & e^{\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} \\ 1 & e^{\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} \end{pmatrix} \begin{pmatrix} 0.21 \\ 0.20 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}$	$[\mathscr{F}_G f]_l = \int_G \rho_l(g) f(g) \mathrm{d}g$
Inverse Fourier transform	$Q^{-1}\,\hat{\mathbf{v}}$	$\mathscr{F}^{-1}[\hat{f}](g) = \sum_l \hat{f}(\rho_l)\rho_l(g^{-1})$
		$= \sum_l d_{\rho_l} \mathrm{tr}\left[\hat{f}(\rho_l)\rho_l(h^{-1})\right]$

General case $d_l \times d_l$ ***matrix irreps of compact groups***

See e.g. Kondor, R., & Trivedi, S. (2018, July). On the generalization of equivariance and convolution in neural networks to the action of compact groups. In International Conference on Machine Learning (pp. 2747-2755). PMLR.

Overview: Block-diagonalization via Fourier transform

	Finite-dimensional vectors $\mathbf{v} \in \mathbb{R}^d$	**Infinite-dimensional vectors** $f \in \mathbb{L}_2(G)$
Regular representation	$\rho(g_n)\ \mathbf{v}$	$\mathscr{L}_g f$
Decomposed into irreps	$Q^{-1}\left[\oplus_{l=-L}^{L} \rho_l(g_n)\right] Q$	$\mathscr{F}_G^{-1} \circ \left[\oplus_{l=-L}^{L} \rho_l(g)\right] \circ \mathscr{F}_G$
Fourier transform	$Q\,\mathbf{v} = \frac{1}{\sqrt{7}} \begin{pmatrix} 1 & e^{-\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} \\ 1 & e^{-\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} \\ 1 & e^{-\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & e^{\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} \\ 1 & e^{\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} \\ 1 & e^{\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} \end{pmatrix} \begin{pmatrix} 0.21 \\ 0.20 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}$	$[\mathscr{F}_G f]_l = \int_G \rho_l(g) f(g) \mathrm{d}g$
Inverse Fourier transform	$Q^{-1}\,\hat{\mathbf{v}}$	$\mathscr{F}^{-1}[\hat{f}](g) = \sum_l \hat{f}(\rho_l)\rho_l(g^{-1})$
		$= \sum_l d_{\rho_l} \mathrm{tr}\left[\hat{f}(\rho_l)\rho_l(h^{-1})\right]$

Fourier shift theorem!!!

General case $d_l \times d_l$ ***matrix irreps of compact groups***

See e.g. Kondor, R., & Trivedi, S. (2018, July). On the generalization of equivariance and convolution in neural networks to the action of compact groups. In International Conference on Machine Learning (pp. 2747-2755). PMLR.

Overview: Block-diagonalization via Fourier transform

	Finite-dimensional vectors $\mathbf{v} \in \mathbb{R}^d$	**Infinite-dimensional vectors** $f \in \mathbb{L}_2(G)$
Regular representation	$\rho(g_n)\ \mathbf{v}$	$\mathscr{L}_g f$
Decomposed into irreps	$Q^{-1}\left[\oplus_{l=-L}^{L} \rho_l(g_n)\right] Q$	$\mathscr{F}_G^{-1} \circ \left[\oplus_{l=-L}^{L} \rho_l(g)\right] \circ \mathscr{F}_G$
Fourier transform	$Q\,\mathbf{v} = \frac{1}{\sqrt{7}} \begin{pmatrix} 1 & e^{-\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} \\ 1 & e^{-\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} \\ 1 & e^{-\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & e^{\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} \\ 1 & e^{\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} \\ 1 & e^{\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} \end{pmatrix} \begin{pmatrix} 0.21 \\ 0.20 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}$	$[\mathscr{F}_G f]_l = \int_G \rho_l(g) f(g) \mathrm{d}g$
Inverse Fourier transform	$Q^{-1}\,\hat{\mathbf{v}}$	$\mathscr{F}^{-1}[\hat{f}](g) = \sum_l \hat{f}(\rho_l)\rho_l(g^{-1})$
		$= \sum_l d_{\rho_l} \operatorname{tr}\left[\hat{f}(\rho_l)\rho_l(h^{-1})\right]$

Fourier shift theorem!!!

General case $d_l \times d_l$ ***matrix irreps of compact groups***

See e.g. Kondor, R., & Trivedi, S. (2018, July). On the generalization of equivariance and convolution in neural networks to the action of compact groups. In International Conference on Machine Learning (pp. 2747-2755). PMLR.

Overview: Block-diagonalization via Fourier transform

	Finite-dimensional vectors $\mathbf{v} \in \mathbb{R}^d$	**Infinite-dimensional vectors** $f \in \mathbb{L}_2(G)$
Regular representation	$\rho(g_n)\ \mathbf{v}$	$\mathscr{L}_g f$
Decomposed into irreps	$Q^{-1}\left[\oplus_{l=-L}^{L} \rho_l(g_n)\right] Q$	$\mathscr{F}_G^{-1} \circ \left[\oplus_{l=-L}^{L} \rho_l(g)\right] \circ \mathscr{F}_G$
Fourier transform	$Q\,\mathbf{v} = \frac{1}{\sqrt{7}} \begin{pmatrix} 1 & e^{-\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} \\ 1 & e^{-\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} \\ 1 & e^{-\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & e^{\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} \\ 1 & e^{\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} \\ 1 & e^{\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} \end{pmatrix} \begin{pmatrix} 0.21 \\ 0.20 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}$	$[\mathscr{F}_G f]_l = \int_G \rho_l(g) f(g) \mathrm{d}g$
Inverse Fourier transform	$Q^{-1}\,\hat{\mathbf{v}}$	$\mathscr{F}^{-1}[\hat{f}](g) = \sum_l \hat{f}(\rho_l)\rho_l(g^{-1})$
		$= \sum_l d_{\rho_l} \operatorname{tr}\left[\hat{f}(\rho_l)\rho_l(h^{-1})\right]$

Fourier shift theorem!!!

General case $d_l \times d_l$ ***matrix irreps of compact groups***

See e.g. Kondor, R., & Trivedi, S. (2018, July). On the generalization of equivariance and convolution in neural networks to the action of compact groups. In International Conference on Machine Learning (pp. 2747-2755). PMLR.

Overview: Block-diagonalization via Fourier transform

	Finite-dimensional vectors $\mathbf{v} \in \mathbb{R}^d$	**Infinite-dimensional vectors** $f \in \mathbb{L}_2(G)$
Regular representation	$\rho(g_n)\,\mathbf{v}$	$\mathscr{L}_g f$
Decomposed into irreps	$Q^{-1}\left[\oplus_{l=-L}^{L}\rho_l(g_n)\right]Q$	$\mathscr{F}_G^{-1}\circ\left[\oplus_{l=-L}^{L}\rho_l(g)\right]\circ\mathscr{F}_G$
Fourier transform	$Q\,\mathbf{v} = \frac{1}{\sqrt{7}}\begin{pmatrix} 1 & e^{-\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} \\ 1 & e^{-\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} \\ 1 & e^{-\frac{2i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & e^{\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{-\frac{2i\pi}{7}} \\ 1 & e^{\frac{4i\pi}{7}} & e^{-\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{\frac{6i\pi}{7}} & e^{-\frac{4i\pi}{7}} \\ 1 & e^{\frac{6i\pi}{7}} & e^{-\frac{2i\pi}{7}} & e^{\frac{4i\pi}{7}} & e^{-\frac{4i\pi}{7}} & e^{\frac{2i\pi}{7}} & e^{-\frac{6i\pi}{7}} \end{pmatrix}\begin{pmatrix} 0.21 \\ 0.20 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}$	$[\mathscr{F}_G f]_l = \int_G f(g)\,\rho_l(g)\,\mathrm{d}g$
Inverse Fourier transform	$Q^{-1}\,\hat{\mathbf{v}}$	$\mathscr{F}^{-1}[\hat{f}](g) = \sum_l \hat{f}(\rho_l)\rho_l(g^{-1})$

Fourier shift theorem!!!

General case $d_l \times d_l$ ***matrix irreps of compact groups***

See e.g. Kondor, R., & Trivedi, S. (2018, July). On the generalization of equivariance and convolution in neural networks to the action of compact groups. In International Conference on Machine Learning (pp. 2747-2755). PMLR.

$$= \sum_l d_{\rho_l}\,\mathrm{tr}\left[\hat{f}(\rho_l)\rho_l(h^{-1})\right]$$

Group Equivariant Deep Learning

Lecture 2 - Steerable group convolutions

Lecture 2.4 - Group Theory | Induced representations and feature fields

Preliminaries (and intuition) for steerable group convolutions

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/

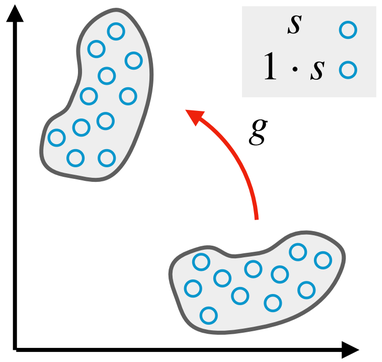
Feature field and induced representation

We call $\hat{f} : \mathbb{R}^d \to \mathbb{R}^{d_\rho}$ a feature vector field, or simply a **feature field**, if its

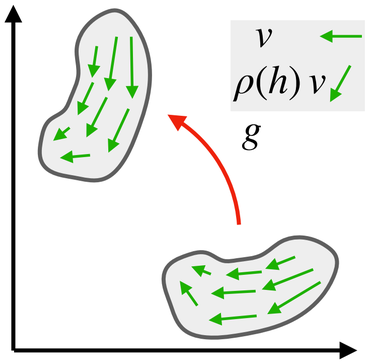
codomain	transforms via a *representation*	$\rho(h)$	of H
domain	transforms via the action	g^{-1}	of $G = (\mathbb{R}^d, +) \rtimes H$

Representation ρ defines the **type** of the field, and together with the group action of $G = (\mathbb{R}^d, +) \rtimes H$ defines the **induced representation**

$$\left(\mathrm{Ind}_H^G[\rho](\mathbf{x}, h)\,\hat{f}\right)(\mathbf{x}') := \rho(h)\,\hat{f}(h^{-1}(\mathbf{x}' - \mathbf{x}))$$



Scalar field ($\rho(h) = 1$)



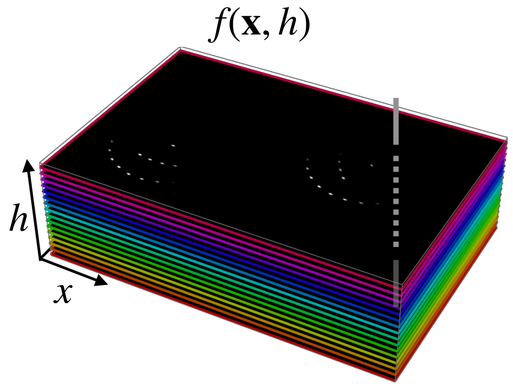
vector field

Figure adapted from: Weiler, M., & Cesa, G. (2019). General e (2)-equivariant steerable cnns. NeurIPS
See also https://github.com/QUVA-Lab/e2cnn

Feature field and induced representation

Regular G feature maps: $f(\mathbf{x}, h)$ considered so far can be considered feature fields.

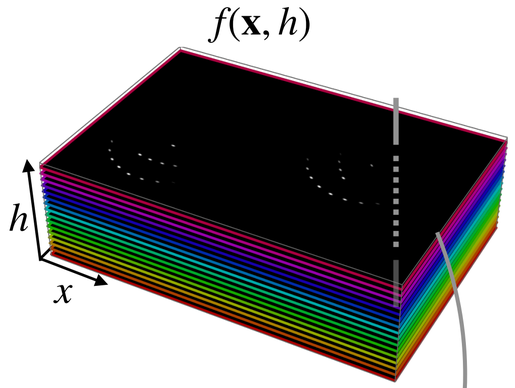
$$(\mathscr{L}_g f)(\mathbf{x}', h') = f(h^{-1}(\mathbf{x}' - \mathbf{x}), h^{-1}h)$$



Feature field and induced representation

Regular G feature maps: $f(\mathbf{x}, h)$ considered so far can be considered feature fields.

$$(\mathscr{L}_g f)(\mathbf{x}', h') = f(h^{-1}(\mathbf{x}' - \mathbf{x}), h^{-1}h)$$

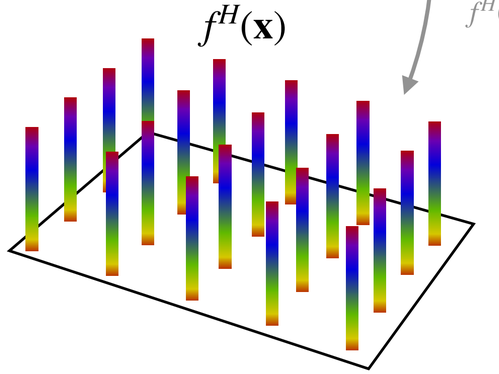


Regular H feature fields: Let $f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$ be the field of functions $f^H(\mathbf{x}) : H \to \mathbb{R}$ on the subgroup H, then the functions (**fibers**) transform via the regular representation $\mathscr{L}^H_h$

(recall. $\mathscr{L}^H_h f(h') = f(h^{-1}h')$)

$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}^G_H[\ \mathscr{L}^H_h\ ](\mathbf{x}, h) f^H\right)(\mathbf{x}')$$

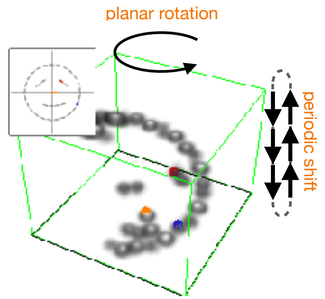
$f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$

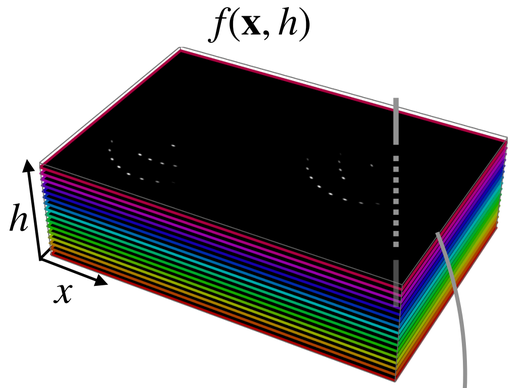


Feature field and induced representation

Regular G feature maps: $f(\mathbf{x}, h)$ considered so far can be considered feature fields.

$$(\mathscr{L}_g f)(\mathbf{x}', h') = f(h^{-1}(\mathbf{x}' - \mathbf{x}), h^{-1}h)$$



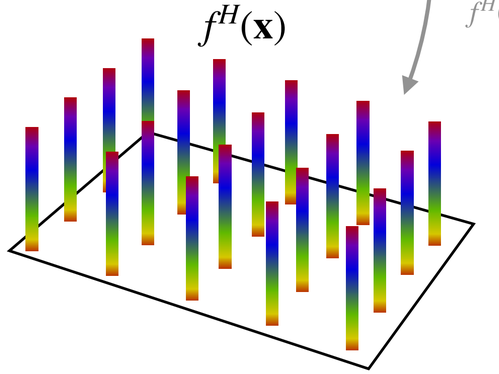


Regular H feature fields: Let $f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$ be the field of functions $f^H(\mathbf{x}) : H \to \mathbb{R}$ on the subgroup H, then the functions (**fibers**) transform via the regular representation $\mathscr{L}^H_h$

(recall. $\mathscr{L}^H_h f(h') = f(h^{-1}h')$)

$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}^G_H[\ \mathscr{L}^H_h\ ](\mathbf{x}, h) f^H\right)(\mathbf{x}')$$

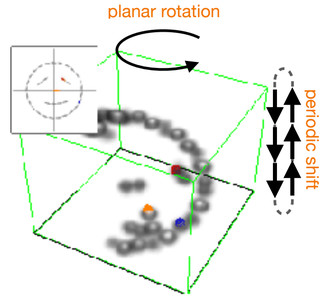
$f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$

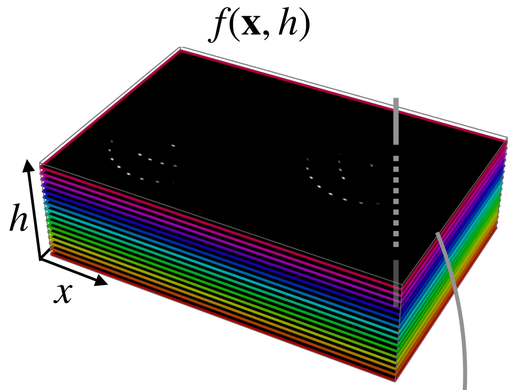


Feature field and induced representation

Regular G feature maps: $f(\mathbf{x}, h)$ considered so far can be considered feature fields.

$$(\mathscr{L}_g f)(\mathbf{x}', h') = f(h^{-1}(\mathbf{x}' - \mathbf{x}), h^{-1}h)$$



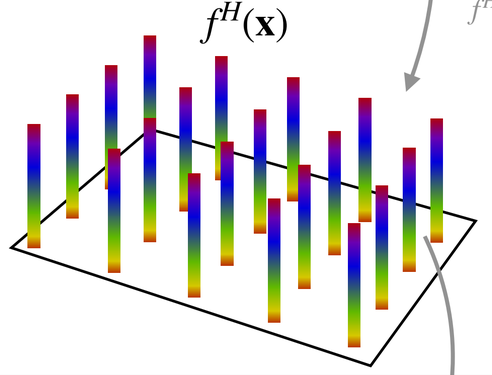


Regular H feature fields: Let $f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$ be the field of functions $f^H(\mathbf{x}) : H \to \mathbb{R}$ on the subgroup H, then the functions (**fibers**) transform via the regular representation $\mathscr{L}^H_h$

(recall. $\mathscr{L}^H_h f(h') = f(h^{-1}h')$)

$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}^G_H[\ \mathscr{L}^H_h\ ](\mathbf{x}, h) f^H\right)(\mathbf{x}')$$

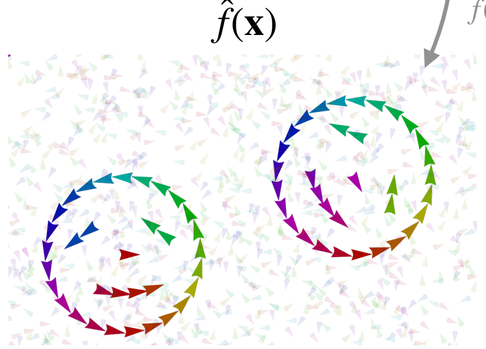
$f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$



Steerable H feature fields: Since the fibers $f^H(\mathbf{x})$ are functions on H we can represent them via their Fourier coefficients $\hat{f}(\mathbf{x}) = \mathscr{F}_H[f^H(\mathbf{x})]$. These vectors of coefficients transform via irreps $\rho(h) = \oplus_l \rho_l(h)$

$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}^G_H[\ \mathscr{L}^H_h\ ](\mathbf{x}, h) \hat{f}\right)(\mathbf{x}') \iff \left(\mathrm{Ind}^G_H[\ \rho(h)\ ](\mathbf{x}, h) \hat{f}\right)(\mathbf{x}')$$

$\hat{f}(\mathbf{x}) = \mathscr{F}_H[f^H(\mathbf{x})]$



Group Equivariant Deep Learning

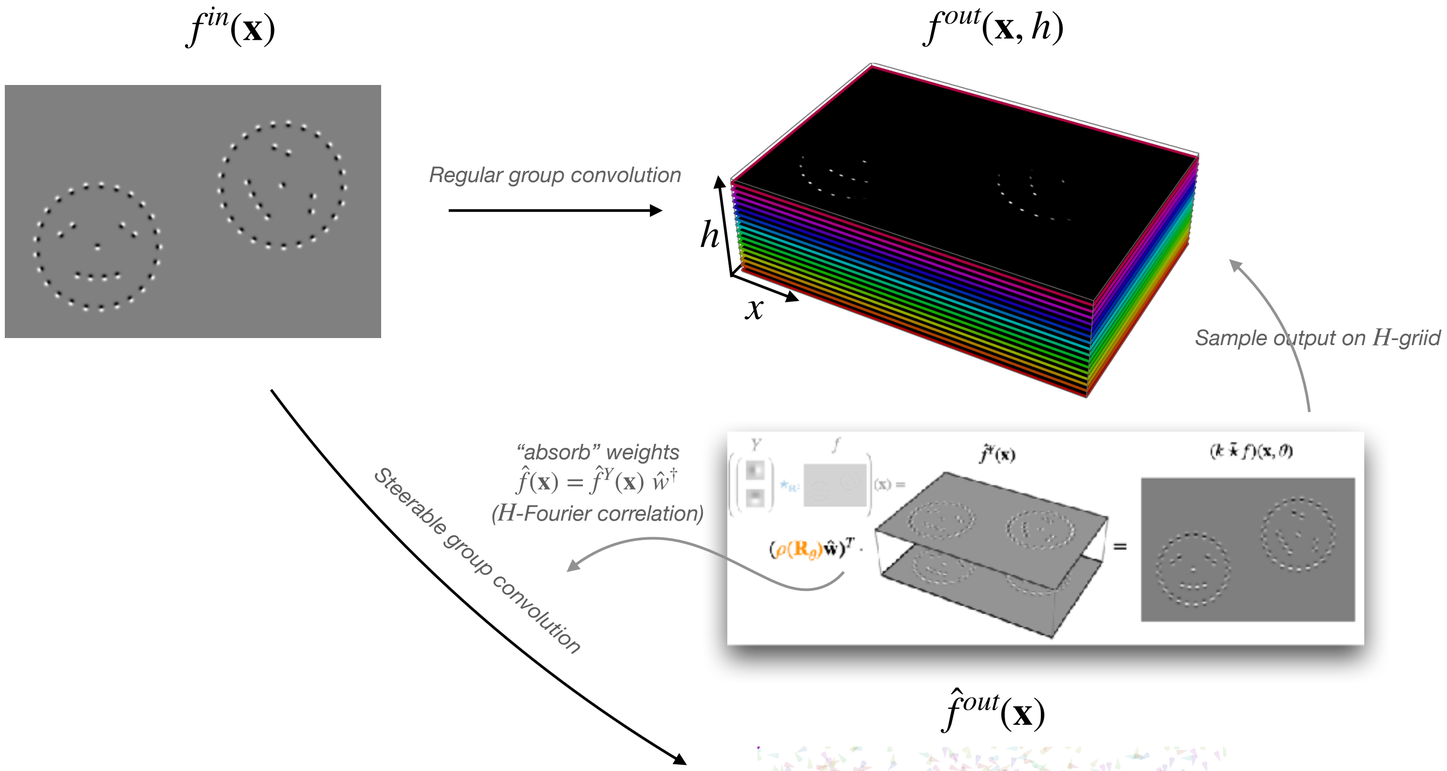
Lecture 2 - Steerable group convolutions

Lecture 2.5 - Steerable group convolutions

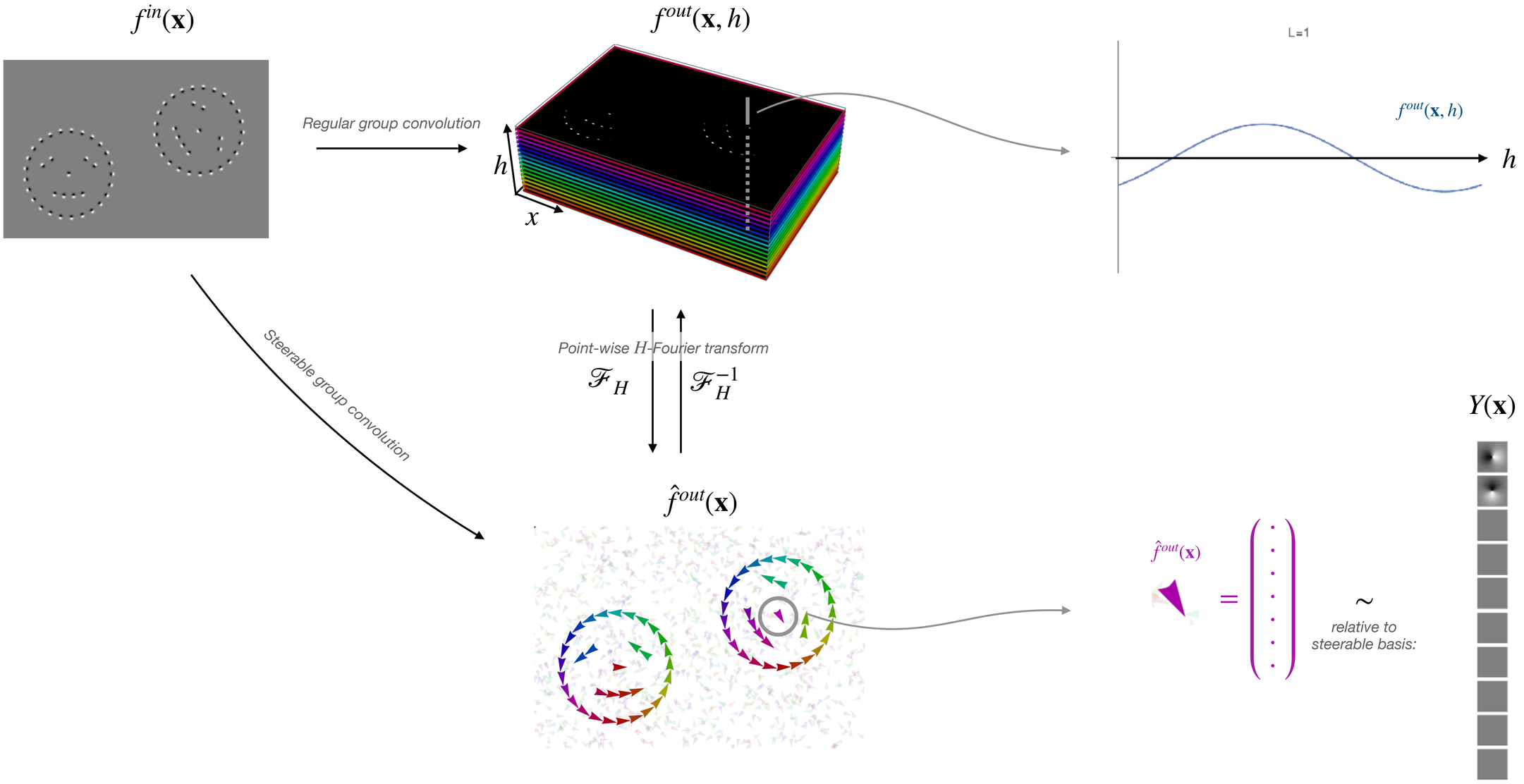
Erik Bekkers, Amsterdam Machine Learning Lab, University of Amsterdam

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

From regular to steerable via a Fourier transform



From regular to steerable via a Fourier transform



From regular to steerable via a Fourier transform

Regular group convolutions:
Domain expanded feature maps

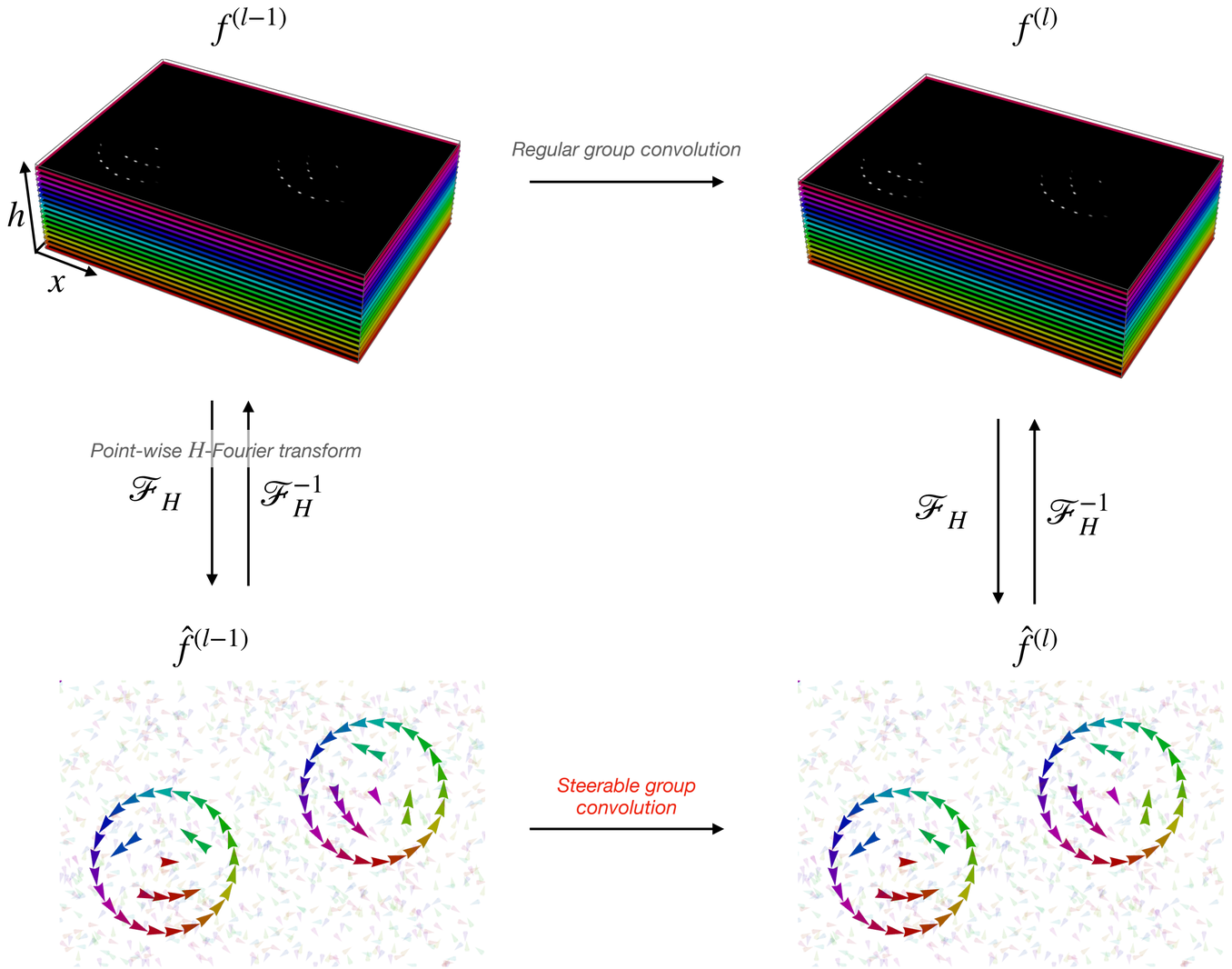
$$f^{(l)} : \mathbb{R}^d \times H \to \mathbb{R}$$

added axis

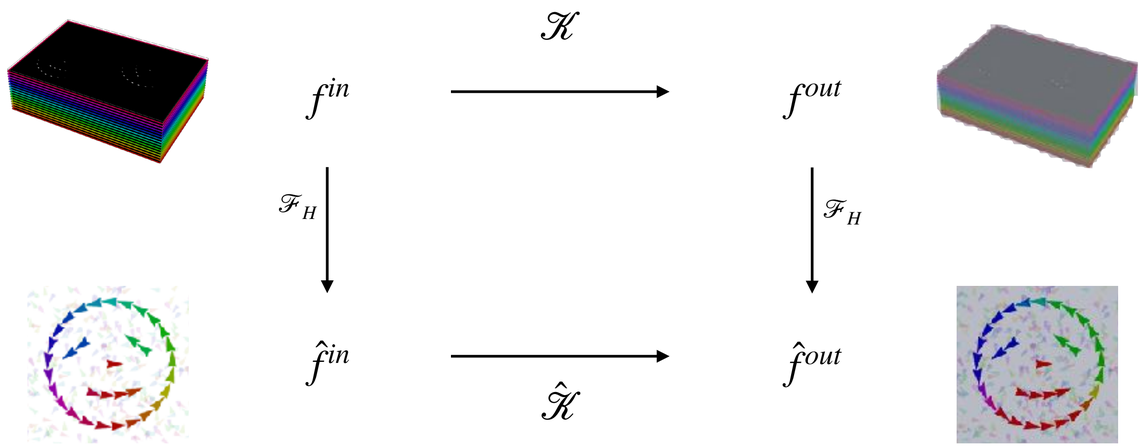
Steerable group convolutions:
Co-domain expanded feature maps (feature fields)

$$\hat{f}^{(l)} : \mathbb{R}^d \to V_H$$

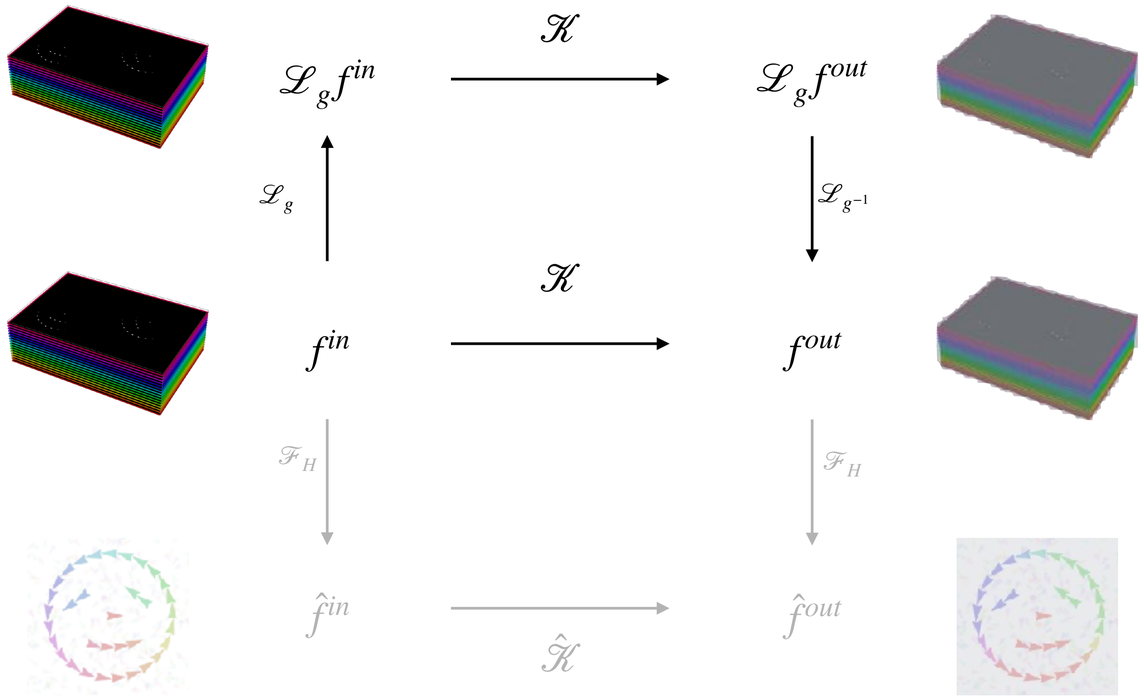
vector field instead of scalar field
(vectors in V_H transform via group H representations)



Steerable group convolutions



Steerable group convolutions



If $\mathcal{K}$ is linear

$$\mathcal{K}[f](g) = \int_G k(g, g') f(g) \mathrm{d}g$$

and equivariant

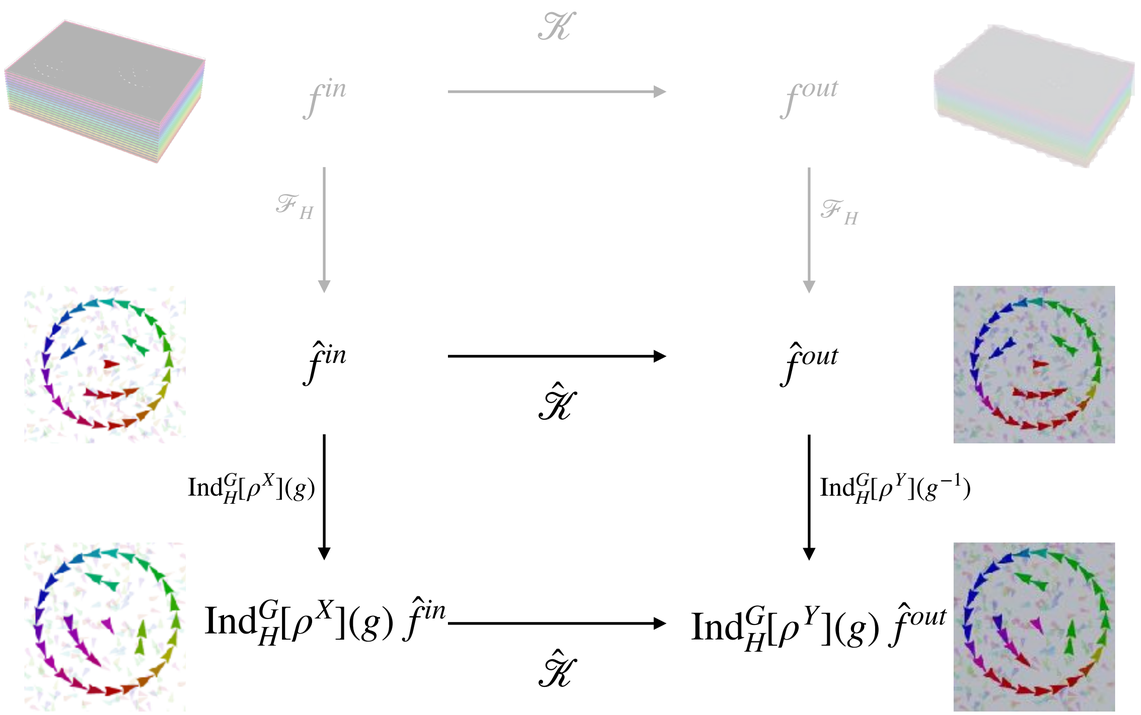
$$\mathcal{K}[\mathcal{L}_g f] = \mathcal{L}_g \mathcal{K}[f]$$

Then it is a group convolution

$$\mathcal{K}[f](g) = \int_G k(g^{-1} g') f(g) \mathrm{d}\mathbf{x}' \mathrm{d}g$$

Recall lecture 1.7 (Group convolutions are all you need)

Steerable group convolutions



If $\mathscr{K}$ is linear

$$\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}, \mathbf{x}')\hat{f}(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

and equivariant

$$\mathscr{K}[\mathrm{Ind}_H^G[\rho^X](g)\ \hat{f}] = \mathrm{Ind}_H^G[\rho^Y](g)\ \mathscr{K}[\hat{f}]$$

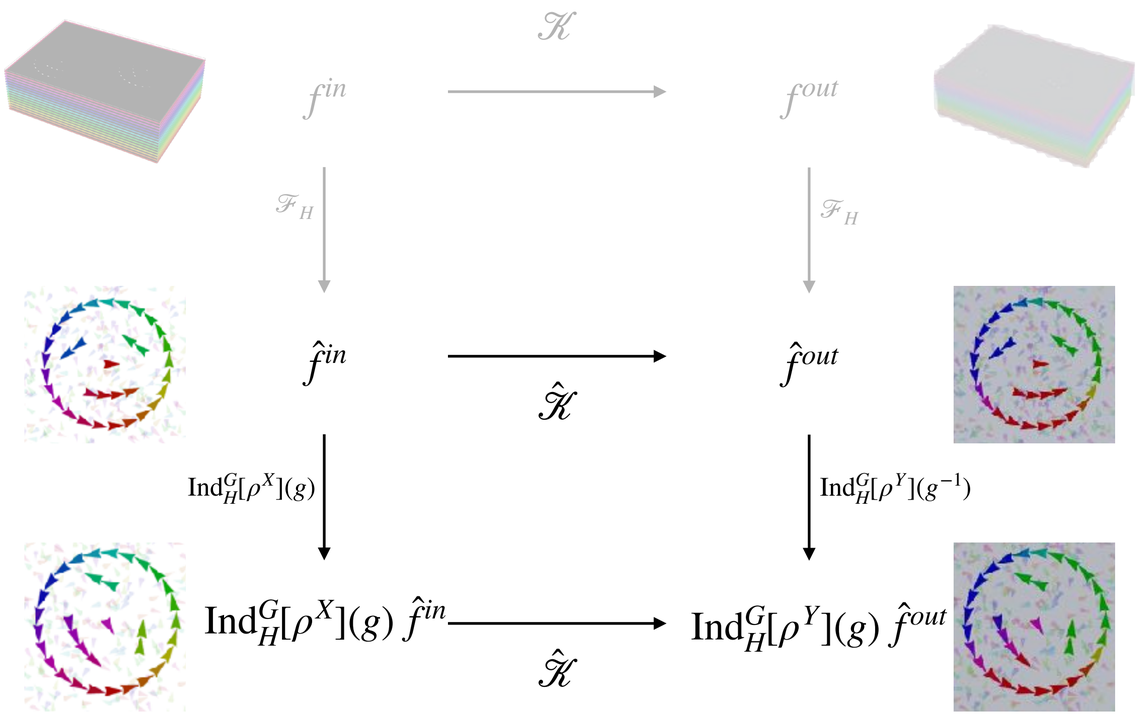
Then it is a normal convolution

$$\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}' - \mathbf{x})f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

but with kernel $k : \mathbb{R}^d \to \mathbb{R}^{d_Y \times d_X}$ satisfying **constraint**

$$\forall_{h \in H}\ \forall_{\mathbf{x} \in \mathbb{R}^d} : \qquad k(h\,\mathbf{x}) = \rho_Y(h)k(\mathbf{x})\rho_X(h^{-1})$$

Steerable group convolutions



If $\mathscr{K}$ is linear

$$\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}, \mathbf{x}')\hat{f}(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

and equivariant

$$\mathscr{K}[\mathrm{Ind}_H^G[\rho^X](g)\ \hat{f}] = \mathrm{Ind}_H^G[\rho^Y](g)\ \mathscr{K}[\hat{f}]$$

Then it is a normal convolution

$$\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}' - \mathbf{x})f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

but with kernel $k : \mathbb{R}^d \to \mathbb{R}^{d_Y \times d_X}$ satisfying **constraint**

$$\forall_{h\in H}\ \forall_{\mathbf{x}\in\mathbb{R}^d} : \qquad k(h\,\mathbf{x}) = \rho_Y(h)k(\mathbf{x})\rho_X(h^{-1})$$

Problem: The G-steerability constraint! [1,2]

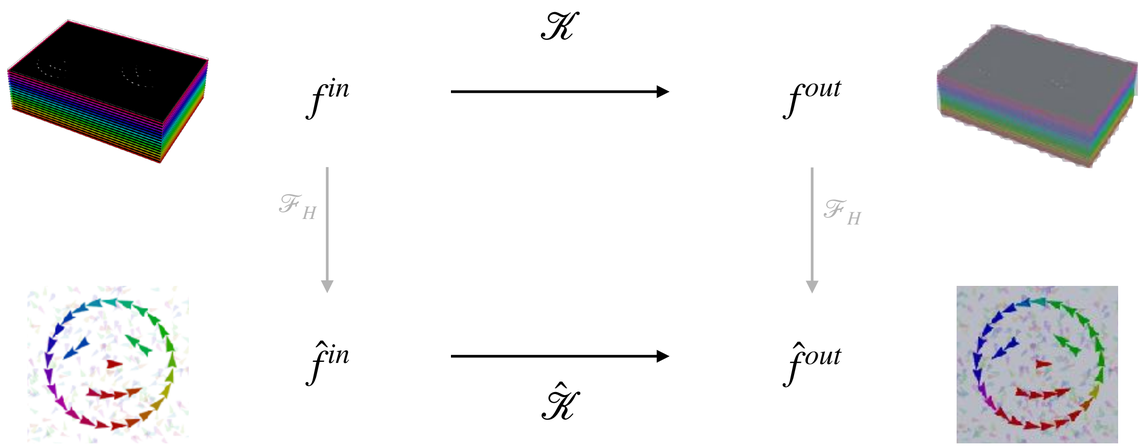
Solution: Expand kernel in steerable basis

[1] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S. Cohen. 3D steerable CNNs: Learning rotationally equivariant features in volumetric data. In Conference on Neural Information Processing Systems (NeurIPS), 2018a.

[2] Cesa, G., Lang, L., & Weiler, M. (2021, September). A Program to Build E (N)-Equivariant Steerable CNNs. In International Conference on Learning Representations.

Steerable group convolutions

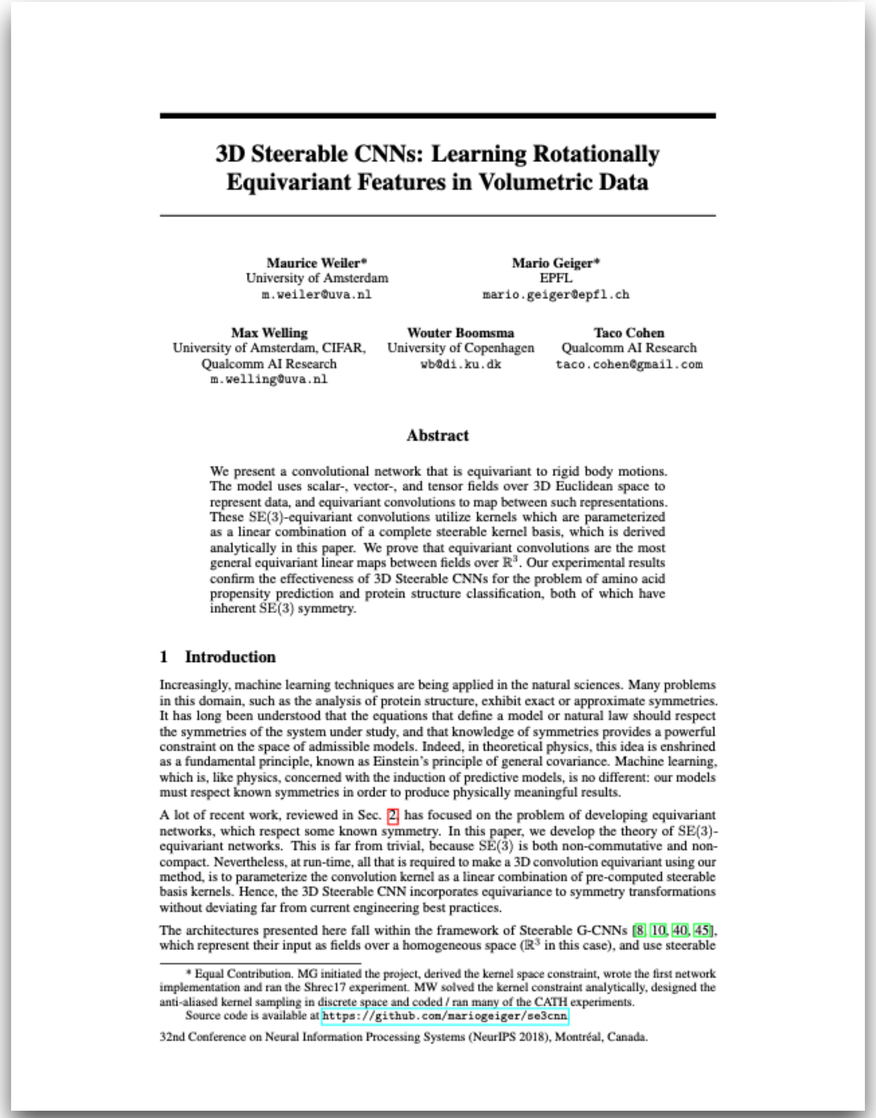
Group convolution $\mathscr{K}[f](g) = \int_G k(g^{-1}g')f(g)\mathrm{d}\mathbf{x}'\mathrm{d}g$



Normal convolution $\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}' - \mathbf{x})f(\mathbf{x}')\mathrm{d}\mathbf{x}'$

but with kernel $k : \mathbb{R}^d \to \mathbb{R}^{d_Y \times d_X}$ satisfying **constraint**

$\forall_{h\in H}\ \forall_{\mathbf{x}\in\mathbb{R}^d}$: $\quad k(g\,\mathbf{x}) = \rho_Y(h)k(\mathbf{x})\rho_X(h^{-1})$



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.

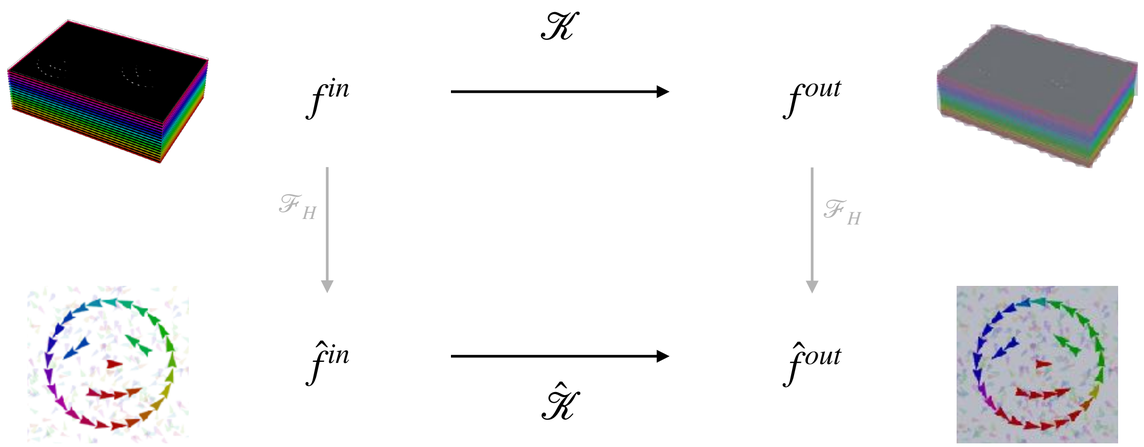
The architectures presented here fall within the framework of Steerable G-CNNs [8, 10, 40, 45], which represent their input as fields over a homogeneous space ($\mathbb{R}^3$ in this case), and use steerable

* 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.

Steerable group convolutions

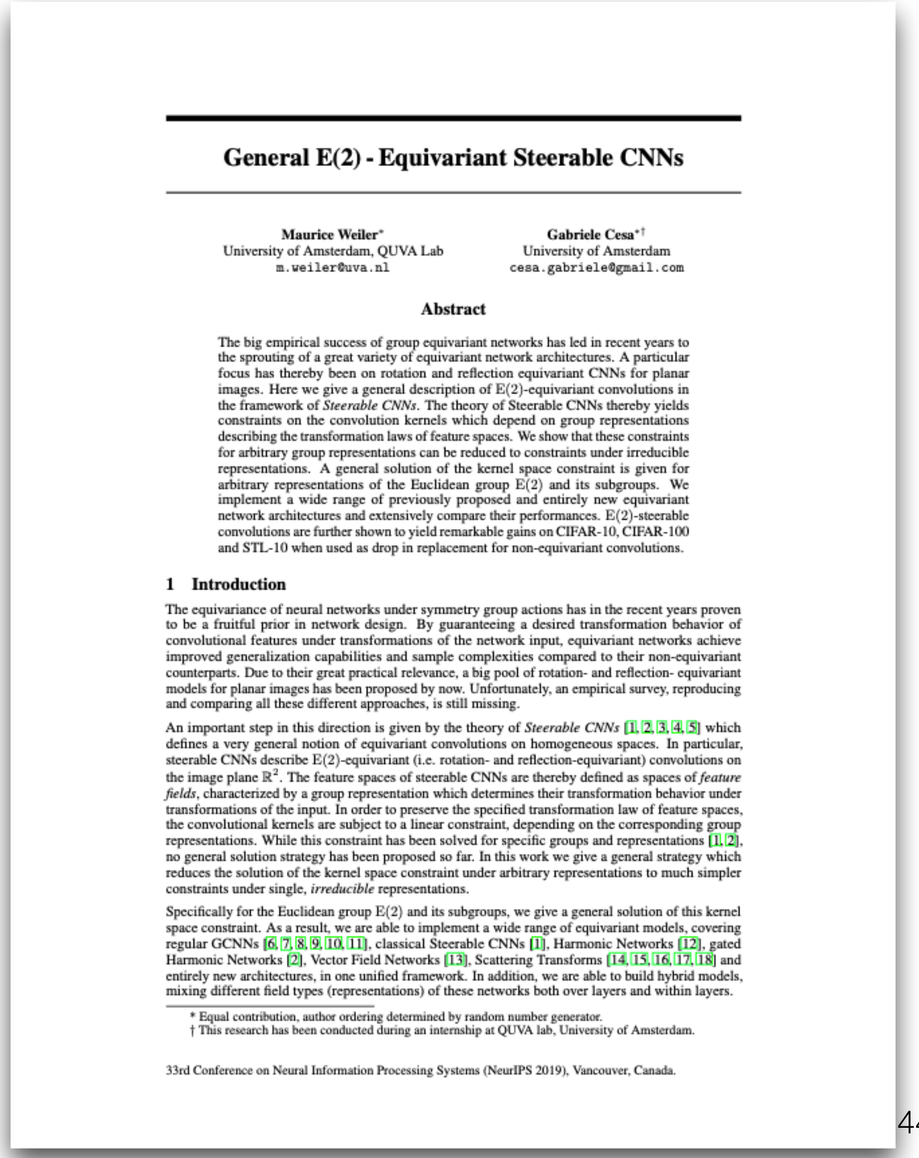
Group convolution $\mathscr{K}[f](g) = \int_G k(g^{-1}g')f(g)\mathrm{d}\mathbf{x}'\mathrm{d}g$



Normal convolution $\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}' - \mathbf{x})f(\mathbf{x}')\mathrm{d}\mathbf{x}'$

but with kernel $k : \mathbb{R}^d \to \mathbb{R}^{d_Y \times d_X}$ satisfying **constraint**

$\forall_{h\in H}\ \forall_{\mathbf{x}\in\mathbb{R}^d}$: $\quad k(g\,\mathbf{x}) = \rho_Y(h)k(\mathbf{x})\rho_X(h^{-1})$



General E(2) - Equivariant Steerable CNNs

Maurice Weiler*
University of Amsterdam, QUVA Lab
m.weiler@uva.nl

Gabriele Cesa*†
University of Amsterdam
cesa.gabriele@gmail.com

Abstract

The big empirical success of group equivariant networks has led in recent years to the sprouting of a great variety of equivariant network architectures. A particular focus has thereby been on rotation and reflection equivariant CNNs for planar images. Here we give a general description of E(2)-equivariant convolutions in the framework of *Steerable CNNs*. The theory of Steerable CNNs thereby yields constraints on the convolution kernels which depend on group representations describing the transformation laws of feature spaces. We show that these constraints for arbitrary group representations can be reduced to constraints under irreducible representations. A general solution of the kernel space constraint is given for arbitrary representations of the Euclidean group E(2) and its subgroups. We implement a wide range of previously proposed and entirely new equivariant network architectures and extensively compare their performances. E(2)-steerable convolutions are further shown to yield remarkable gains on CIFAR-10, CIFAR-100 and STL-10 when used as drop in replacement for non-equivariant convolutions.

1 Introduction

The equivariance of neural networks under symmetry group actions has in the recent years proven to be a fruitful prior in network design. By guaranteeing a desired transformation behavior of convolutional features under transformations of the network input, equivariant networks achieve improved generalization capabilities and sample complexities compared to their non-equivariant counterparts. Due to their great practical relevance, a big pool of rotation- and reflection- equivariant models for planar images has been proposed by now. Unfortunately, an empirical survey, reproducing and comparing all these different approaches, is still missing.

An important step in this direction is given by the theory of *Steerable CNNs* [1, 2, 3, 4, 5] which defines a very general notion of equivariant convolutions on homogeneous spaces. In particular, steerable CNNs describe E(2)-equivariant (i.e. rotation- and reflection-equivariant) convolutions on the image plane $\mathbb{R}^2$. The feature spaces of steerable CNNs are thereby defined as spaces of *feature fields*, characterized by a group representation which determines their transformation behavior under transformations of the input. In order to preserve the specified transformation law of feature spaces, the convolutional kernels are subject to a linear constraint, depending on the corresponding group representations. While this constraint has been solved for specific groups and representations [1, 2], no general solution strategy has been proposed so far. In this work we give a general strategy which reduces the solution of the kernel space constraint under arbitrary representations to much simpler constraints under single, *irreducible* representations.

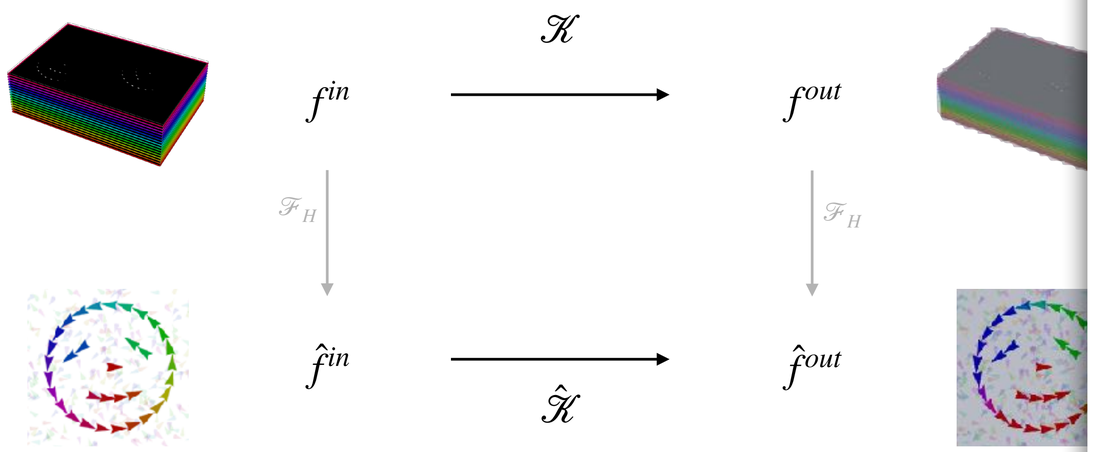
Specifically for the Euclidean group E(2) and its subgroups, we give a general solution of this kernel space constraint. As a result, we are able to implement a wide range of equivariant models, covering regular GCNNs [6, 7, 8, 9, 10, 11], classical Steerable CNNs [1], Harmonic Networks [12], gated Harmonic Networks [2], Vector Field Networks [13], Scattering Transforms [14, 15, 16, 17, 18] and entirely new architectures, in one unified framework. In addition, we are able to build hybrid models, mixing different field types (representations) of these networks both over layers and within layers.

* Equal contribution, author ordering determined by random number generator.
† This research has been conducted during an internship at QUVA lab, University of Amsterdam.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Steerable group convolutions

Group convolution $\mathscr{K}[f](g) = \int_G k(g^{-1}g')f(g)\mathrm{d}\mathbf{x}'\mathrm{d}g$



Normal convolution $\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}' - \mathbf{x})f(\mathbf{x}')\mathrm{d}\mathbf{x}'$

but with kernel $k : \mathbb{R}^d \to \mathbb{R}^{d_Y \times d_X}$ satisfying **constraint**

$\forall_{h\in H}\ \forall_{\mathbf{x}\in\mathbb{R}^d}$: $\quad k(g\,\mathbf{x}) = \rho_Y(h)k(\mathbf{x})\rho_X(h^{-1})$

Published as a conference paper at ICLR 2022

A PROGRAM TO BUILD E(n)-EQUIVARIANT STEERABLE CNNS

Gabriele Cesa
Qualcomm AI Research*
University of Amsterdam
gcesa@qti.qualcomm.com

Leon Lang
University of Amsterdam
l.lang@uva.nl

Maurice Weiler
University of Amsterdam
m.weiler.ml@gmail.com

ABSTRACT

Equivariance is becoming an increasingly popular design choice to build data efficient neural networks by exploiting prior knowledge about the symmetries of the problem at hand. Euclidean steerable CNNs are one of the most common classes of equivariant networks. While the constraints these architectures need to satisfy are understood, existing approaches are tailored to specific (classes of) groups. No generally applicable method that is *practical* for implementation has been described so far. In this work, we generalize the Wigner-Eckart theorem proposed in Lang & Weiler (2020), which characterizes general G-steerable kernel spaces for compact groups G over their homogeneous spaces, to arbitrary G-spaces. This enables us to directly parameterize filters in terms of a band-limited basis on the whole space rather than on G's orbits, but also to easily implement steerable CNNs equivariant to a large number of groups. To demonstrate its generality, we instantiate our method on a variety of isometry groups acting on the Euclidean space $\mathbb{R}^3$. Our framework allows us to build E(3) and SE(3)-steerable CNNs like previous works, but also CNNs with arbitrary $G \leq \mathrm{O}(3)$-steerable kernels. For example, we build 3D CNNs equivariant to the symmetries of platonic solids or choose $G = \mathrm{SO}(2)$ when working with 3D data having only azimuthal symmetries. We compare these models on 3D shapes and molecular datasets, observing improved performance by matching the model's symmetries to the ones of the data.

1 INTRODUCTION

In machine learning, it is common for learning tasks to present a number of *symmetries*. A symmetry in the data occurs, for example, when some property (e.g., the label) does not change if a set of transformations is applied to the data itself, e.g. translations or rotations of images. Symmetries are algebraically described by *groups*. If prior knowledge about the symmetries of a task is available, it is usually beneficial to encode them in the models used (Shawe-Taylor, 1989; Cohen & Welling, 2016a). The property of such models is referred to as *equivariance* and is obtained by introducing some *equivariance constraints* in the architecture (see Eq. 2). A classical example are convolutional neural networks (CNNs), which achieve translation equivariance by constraining linear layers to be convolution operators. A wider class of equivariant models are Euclidean steerable CNNs (Cohen & Welling, 2016b; Weiler et al., 2018a; Weiler & Cesa, 2019; Jenner & Weiler, 2022), which guarantee equivariance to isometries $\mathbb{R}^n \rtimes G$ of a Euclidean space $\mathbb{R}^n$, i.e., to translations and a group G of origin-preserving transformations, such as rotations and reflections. As proven in Weiler et al. (2018a; 2021); Jenner & Weiler (2022), this requires convolutions with G-*steerable* (equivariant) kernels.

Our goal is developing a program to parameterize with minimal requirements arbitrary G-steerable kernel spaces, with compact G, which are required to implement $\mathbb{R}^n \rtimes G$ equivariant CNNs. Lang & Weiler (2020) provides a first step in this direction by generalizing the *Wigner-Eckart theorem* from quantum mechanics to obtain a general technique to parametrize G-steerable kernel spaces over *orbits* of a compact G. The theorem reduces the task of building steerable kernel bases to that of finding some pure representation theoretic ingredients. Since the equivariance constraint only relates points $g.x \in \mathbb{R}^n$ in the same *orbit* $G.x \subset \mathbb{R}^n$, a kernel can take independent values on different orbits. Fig. 1 shows

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

https://github.com/QUVA-Lab/escnn

up convolutions

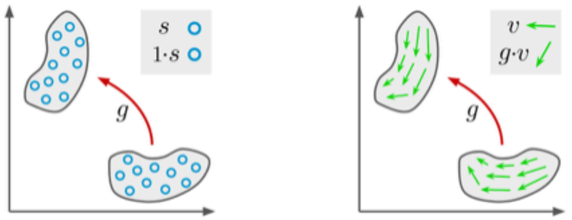
E(n)-Equivariant Steerable CNNs

Documentation | **Paper ICLR 22** | **Paper NeurIPS 19** | **e2cnn library** | **e2cnn experiments** | **Thesis**

escnn is a PyTorch extension for equivariant deep learning. *escnn* is the successor of the e2cnn library, which only supported planar isometries.

Equivariant neural networks guarantee a specified transformation behavior of their feature spaces under transformations of their input. For instance, classical convolutional neural networks (*CNNs*) are by design equivariant to translations of their input. This means that a translation of an image leads to a corresponding translation of the network's feature maps. This package provides implementations of neural network modules which are equivariant under all *isometries* E(2) of the image plane $\mathbb{R}^2$ and all *isometries* E(3) of the 3D space $\mathbb{R}^3$, that is, under *translations*, *rotations* and *reflections* (and can, potentially, be extended to all isometries E(n) of $\mathbb{R}^n$). In contrast to conventional CNNs, E(n)-equivariant models are guaranteed to generalize over such transformations, and are therefore more data efficient.

The feature spaces of E(n)-Equivariant Steerable CNNs are defined as spaces of *feature fields*, being characterized by their transformation law under rotations and reflections. Typical examples are scalar fields (e.g. gray-scale images or temperature fields) or vector fields (e.g. optical flow or electromagnetic fields).



scalar field

vector field

Instead of a number of channels, the user has to specify the field *types* and their *multiplicities* in order to define a feature space. Given a specified input- and output feature space, our `R2conv` and `R3conv` modules instantiate

Published as a conference paper at ICLR 2022

A PROGRAM TO BUILD E(*n*)-EQUIVARIANT STEERABLE CNNS

Gabriele Cesa
Qualcomm AI Research*
University of Amsterdam
gcesa@qti.qualcomm.com

Leon Lang
University of Amsterdam
l.lang@uva.nl

Maurice Weiler
University of Amsterdam
m.weiler.ml@gmail.com

ABSTRACT

Equivariance is becoming an increasingly popular design choice to build data efficient neural networks by exploiting prior knowledge about the symmetries of the problem at hand. Euclidean steerable CNNs are one of the most common classes of equivariant networks. While the constraints these architectures need to satisfy are understood, existing approaches are tailored to specific (classes of) groups. No generally applicable method that is *practical* for implementation has been described so far. In this work, we generalize the Wigner-Eckart theorem proposed in Lang & Weiler (2020), which characterizes general G-steerable kernel spaces for compact groups G over their homogeneous spaces, to arbitrary G-spaces. This enables us to directly parameterize filters in terms of a band-limited basis on the whole space rather than on G's orbits, but also to easily implement steerable CNNs equivariant to a large number of groups. To demonstrate its generality, we instantiate our method on a variety of isometry groups acting on the Euclidean space $\mathbb{R}^3$. Our framework allows us to build E(3) and SE(3)-steerable CNNs like previous works, but also CNNs with arbitrary $G \leq \mathrm{O}(3)$-steerable kernels. For example, we build 3D CNNs equivariant to the symmetries of platonic solids or choose $G = \mathrm{SO}(2)$ when working with 3D data having only azimuthal symmetries. We compare these models on 3D shapes and molecular datasets, observing improved performance by matching the model's symmetries to the ones of the data.

1 INTRODUCTION

In machine learning, it is common for learning tasks to present a number of *symmetries*. A symmetry in the data occurs, for example, when some property (e.g., the label) does not change if a set of transformations is applied to the data itself, e.g. translations or rotations of images. Symmetries are algebraically described by *groups*. If prior knowledge about the symmetries of a task is available, it is usually beneficial to encode them in the models used (Shawe-Taylor, 1989; Cohen & Welling, 2016a). The property of such models is referred to as *equivariance* and is obtained by introducing some *equivariance constraints* in the architecture (see Eq. 2). A classical example are convolutional neural networks (CNNs), which achieve translation equivariance by constraining linear layers to be convolution operators. A wider class of equivariant models are Euclidean steerable CNNs (Cohen & Welling, 2016b; Weiler et al., 2018a; Weiler & Cesa, 2019; Jenner & Weiler, 2022), which guarantee equivariance to isometries $\mathbb{R}^n \rtimes G$ of a Euclidean space $\mathbb{R}^n$, i.e., to translations and a group G of origin-preserving transformations, such as rotations and reflections. As proven in Weiler et al. (2018a; 2021); Jenner & Weiler (2022), this requires convolutions with G-*steerable* (equivariant) kernels.

Our goal is developing a program to parameterize with minimal requirements arbitrary G-steerable kernel spaces, with compact G, which are required to implement $\mathbb{R}^n \rtimes G$ equivariant CNNs. Lang & Weiler (2020) provides a first step in this direction by generalizing the *Wigner-Eckart theorem* from quantum mechanics to obtain a general technique to parametrize G-steerable kernel spaces over *orbits* of a compact G. The theorem reduces the task of building steerable kernel bases to that of finding some pure representation theoretic ingredients. Since the equivariance constraint only relates points $g.x \in \mathbb{R}^n$ in the same *orbit* $G.x \subset \mathbb{R}^n$, a kernel can take independent values on different orbits. Fig. 1 shows

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Normal convolution $\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$

but with kernel $k : \mathbb{R}^d \to \mathbb{R}^{d_Y \times d_X}$ satisfying **constraint**

$\forall_{h \in H} \; \forall_{\mathbf{x} \in \mathbb{R}^d} :$ $\quad k(g\,\mathbf{x}) = \rho_Y(h) k(\mathbf{x}) \rho_X(h^{-1})$

https://github.com/QUVA-Lab/escnn

up convolutions

E(n)-Equivariant Steerable CNNs

Documentation | **Paper ICLR 22** | **Paper NeurIPS 19** | **e2cnn library** | **e2cnn experiments** | **Thesis**

escnn is a PyTorch extension for equivariant deep learning. *escnn* is the successor of the e2cnn library, which only supported planar isometries.

Equivariant neural networks guarantee a specified transformation behavior of their feature spaces under transformations of their input. For instance, classical convolutional neural networks (*CNNs*) are by design equivariant to translations of their input. This means that a translation of an image leads to a corresponding translation of the network's feature maps. This package provides implementations of neural network modules which are equivariant under all *isometries* E(2) of the image plane $\mathbb{R}^2$ and all *isometries* E(3) of the 3D space $\mathbb{R}^3$, that is, under *translations*, *rotations* and *reflections* (and can, potentially, be extended to all isometries E(n) of $\mathbb{R}^n$). In contrast to conventional CNNs, E(n)-equivariant models are guaranteed to generalize over such transforma

The feature
characteriz
gray-scale

Instead of a
feature spa

Getting Started

escnn is easy to use since it provides a high level user interface which abstracts most intricacies of group and representation theory away. The following code snippet shows how to perform an equivariant convolution from an RGB-image to 10 *regular* feature fields (corresponding to a group convolution).

```
from escnn import gspaces                                          #  1
from escnn import nn                                               #  2
import torch                                                       #  3
                                                                   #  4
r2_act = gspaces.rot2dOnR2(N=8)                                    #  5
feat_type_in  = nn.FieldType(r2_act,  3*[r2_act.trivial_repr])     #  6
feat_type_out = nn.FieldType(r2_act, 10*[r2_act.regular_repr])     #  7
                                                                   #  8
conv = nn.R2Conv(feat_type_in, feat_type_out, kernel_size=5)       #  9
relu = nn.ReLU(feat_type_out)                                      # 10
                                                                   # 11
x = torch.randn(16, 3, 32, 32)                                     # 12
x = feat_type_in(x)                                                # 13
                                                                   # 14
y = relu(conv(x))                                                  # 15
```

Line 5 specifies the symmetry group action on the image plane $\mathbb{R}^2$ under which the network should be equivariant. We choose the *cyclic group* C_8, which describes discrete rotations by multiples of 2π/8. Line 6 specifies the input feature field types. The three color channels of an RGB image are thereby to be identified as three independent scalar fields, which transform under the *trivial representation* of C_8. Similarly, the output feature space is in line 7 specified to consist of 10 feature fields which transform under the *regular representation* of C_8. The C_8-equivariant convolution is then instantiated by passing the input and output type as well as the kernel size to the constructor (line 9). Line 10 instantiates an equivariant ReLU nonlinearity which will operate on the output field and is therefore passed the output field type.

Published as a conference paper at ICLR 2022

A PROGRAM TO BUILD E(*n*)-EQUIVARIANT STEERABLE CNNS

Gabriele Cesa
Qualcomm AI Research*
University of Amsterdam
gcesa@qti.qualcomm.com

Leon Lang
University of Amsterdam
l.lang@uva.nl

Maurice Weiler
University of Amsterdam
m.weiler.ml@gmail.com

ABSTRACT

Equivariance is becoming an increasingly popular design choice to build data efficient neural networks by exploiting prior knowledge about the symmetries of the problem at hand. Euclidean steerable CNNs are one of the most common classes of equivariant networks. While the constraints these architectures need to satisfy are understood, existing approaches are tailored to specific (classes of) groups. No generally applicable method that is *practical* for implementation has been described so far. In this work, we generalize the Wigner-Eckart theorem proposed in Lang & Weiler (2020), which characterizes general G-steerable kernel spaces for compact groups G over their homogeneous spaces, to arbitrary G-spaces. This enables us to directly parameterize filters in terms of a band-limited basis on the whole space rather than on G's orbits, but also to easily implement steerable CNNs equivariant to a large number of groups. To demonstrate its generality, we instantiate our method on a variety of isometry groups acting on the Euclidean space $\mathbb{R}^3$. Our framework allows us to build E(3) and SE(3)-steerable CNNs like previous works, but also CNNs with arbitrary $G \leq \mathrm{O}(3)$-steerable kernels. For example, we build 3D CNNs equivariant to the symmetries of platonic solids or choose $G = \mathrm{SO}(2)$ when working with 3D data having only azimuthal symmetries. We compare these models on 3D shapes and molecular datasets, observing improved performance by matching the model's symmetries to the ones of the data.

1 INTRODUCTION

In machine learning, it is common for learning tasks to present a number of *symmetries*. A symmetry in the data occurs, for example, when some property (e.g., the label) does not change if a set of transformations is applied to the data itself, e.g. translations or rotations of images. Symmetries are algebraically described by *groups*. If prior knowledge about the symmetries of a task is available, it is usually beneficial to encode them in the models used (Shawe-Taylor, 1989; Cohen & Welling, 2016a). The property of such models is referred to as *equivariance* and is obtained by introducing some *equivariance constraints* in the architecture (see Eq. 2). A classical example are convolutional neural networks (CNNs), which achieve translation equivariance by constraining linear layers to be convolution operators. A wider class of equivariant models are Euclidean steerable CNNs (Cohen & Welling, 2016b; Weiler et al., 2018a; Weiler & Cesa, 2019; Jenner & Weiler, 2022), which guarantee equivariance to isometries $\mathbb{R}^n \rtimes G$ of a Euclidean space $\mathbb{R}^n$, i.e., to translations and a group G of origin-preserving transformations, such as rotations and reflections. As proven in Weiler et al. (2018a; 2021); Jenner & Weiler (2022), this requires convolutions with *G-steerable* (equivariant) kernels.

Our goal is developing a program to parameterize with minimal requirements arbitrary G-steerable kernel spaces, with compact G, which are required to implement $\mathbb{R}^n \rtimes G$ equivariant CNNs. Lang & Weiler (2020) provides a first step in this direction by generalizing the *Wigner-Eckart theorem* from quantum mechanics to obtain a general technique to parametrize G-steerable kernel spaces over *orbits* of a compact G. The theorem reduces the task of building steerable kernel bases to that of finding some pure representation theoretic ingredients. Since the equivariance constraint only relates points $g.x \in \mathbb{R}^n$ in the same *orbit* $G.x \subset \mathbb{R}^n$, a kernel can take independent values on different orbits. Fig. 1 shows

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

https://github.com/QUVA-Lab/escnn

E(n)-Equivariant Steerable CNNs

Documentation | **Paper ICLR 22** | **Paper NeurIPS 19** | **e2cnn library** | **e2cnn experiments** | **Thesis**

escnn is a PyTorch extension for equivariant deep learning. *escnn* is the s
supported planar isometries.

Equivariant neural networks guarantee a specified transformation behavi
transformations of their input. For instance, classical convolutional neura
equivariant to translations of their input. This means that a translation of
translation of the network's feature maps. This package provides implem
which are equivariant under all *isometries* E(2) of the image plane $\mathbb{R}^2$ and
that is, under *translations*, *rotations* and *reflections* (and can, potentially,
In con
transf

The fe
chara
gray-s

Instea
featur

Getting Started

escnn is easy to use since it provides a high level user interfac
representation theory away. The following code snippet shows
RGB-image to 10 *regular* feature fields (corresponding to a gr

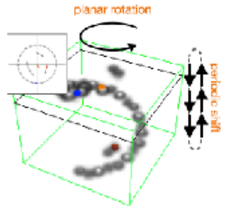
```
from escnn import gspaces
from escnn import nn
import torch

r2_act = gspaces.rot2dOnR2(N=8)
feat_type_in  = nn.FieldType(r2_act,  3*[r2_act.tri
feat_type_out = nn.FieldType(r2_act, 10*[r2_act.reg

conv = nn.R2Conv(feat_type_in, feat_type_out, kerne
relu = nn.ReLU(feat_type_out)

x = torch.randn(16, 3, 32, 32)
x = feat_type_in(x)

y = relu(conv(x))
```

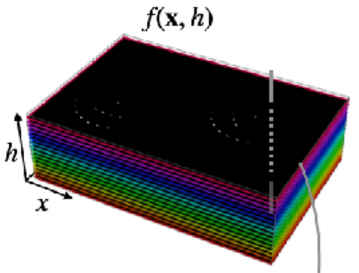
Line 5 specifies the symmetry group action on the image plane $\mathbb{R}^2$ under which the network should be equivariant. We choose the *cyclic group* C_8, which describes discrete rotations by multiples of 2π/8. Line 6 specifies the input feature field types. The three color channels of an RGB image are thereby to be identified as three independent scalar fields, which transform under the *trivial representation* of C_8. Similarly, the output feature space is in line 7 specified to consist of 10 feature fields which transform under the *regular representation* of C_8. The C_8-equivariant convolution is then instantiated by passing the input and output type as well as the kernel size to the constructor (line 9). Line 10 instantiates an equivariant ReLU nonlinearity which will operate on the output field and is therefore passed the output field type.

Feature field and induced representation

Regular G feature maps: $f(\mathbf{x}, h)$ considered so far can be considered feature fields.

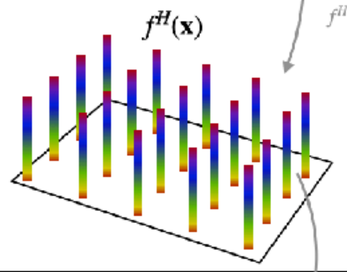
$$(\mathscr{L}_g f)(\mathbf{x}', h') = f(h^{-1}(\mathbf{x}' - \mathbf{x}), h^{-1}h)$$





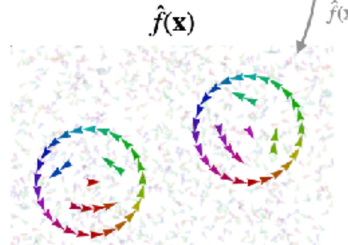
Regular H feature fields: Let $f^H(\mathbf{x}) = f(\mathbf{x}, \cdot)$ be the field of functions $f^H(\mathbf{x}) : H \to \mathbb{R}$ on the subgroup H, then the functions (**fibers**) transform via the regular representation $\mathscr{L}^H_h$ (recall. $\mathscr{L}^H_h f(h') = f(h^{-1}h')$)

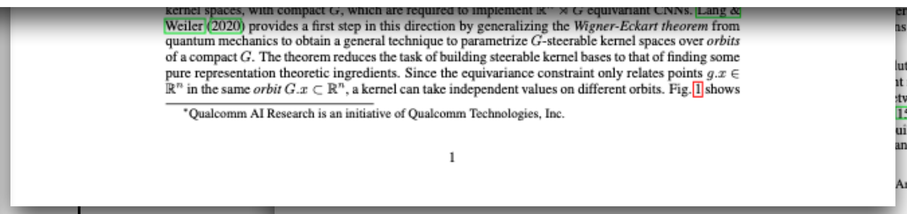
$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}_H^G[\,\mathscr{L}^H_h\,](\mathbf{x}, h) f^H\right)(\mathbf{x}')$$



Steerable H feature fields: Since the fibers $f^H(\mathbf{x})$ are functions on H we can represent them via their Fourier coefficients $\hat{f}(\mathbf{x}) = \mathscr{F}_H[f^H(\mathbf{x})]$. These vectors of coefficients transform via irreps $\rho(h) = \oplus_l \rho_l(h)$

$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}_H^G[\,\mathscr{L}^H_h\,](\mathbf{x}, h)\hat{f}\right)(\mathbf{x}') \iff \left(\mathrm{Ind}_H^G[\,\rho(h)\,](\mathbf{x}, h)\hat{f}\right)(\mathbf{x}')$$




kernel spaces, with compact G, which are required to implement $\mathbb{R}^n \rtimes G$ equivariant CNNs. Lang & Weiler (2020) provides a first step in this direction by generalizing the *Wigner-Eckart theorem* from quantum mechanics to obtain a general technique to parametrize G-steerable kernel spaces over *orbits* of a compact G. The theorem reduces the task of building steerable kernel bases to that of finding some pure representation theoretic ingredients. Since the equivariance constraint only relates points $g.x \in \mathbb{R}^n$ in the same *orbit* $G.x \subset \mathbb{R}^n$, a kernel can take independent values on different orbits. Fig. 1 shows

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

1

eral strategy which
as to much simpler
lution of this kernel
t models, covering
tworks [12], gated
15, 16, 17, 18] and
uild hybrid models,
and within layers.
Amsterdam.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Steerable group convolutions

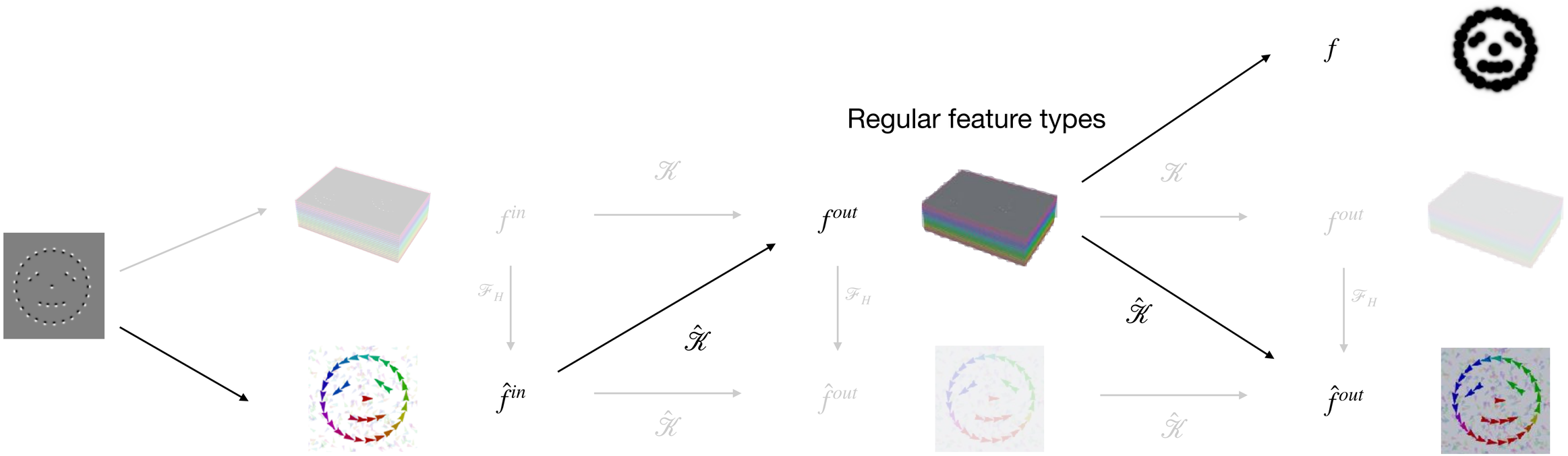
E(n)-Equivariant Steerable CNNs

Documentation | Paper ICLR 22 | Paper NeurIPS 19 | e2cnn library | e2cnn experiments | Thesis

escnn is a PyTorch extension for equivariant deep learning. *escnn* is the successor of the e2cnn library, which only supported planar isometries.

https://github.com/QUVA-Lab/escnn

Type-0 field usual 2D feature map
(e.g. for segmentation)



Steerable (irrep) feature types

Type-1 vector fields
(e.g. force/velocity vectors)

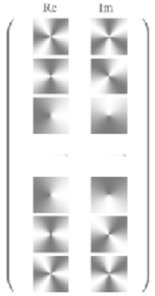
Feature field types

- A feature field is defined by its type ρ
- Feature fields, each of their own type ρ_l, can be stacked:
  - should be thought of as the channels in standard CNNs
- The sub-vectors/channels in these fields:
  - live in their own sub-vector spaces V_l
  - transform by their own representations ρ_l

Different types ρ

Complex irreps

$$\begin{pmatrix} e^{3i\theta} & & & & & & \\ & e^{2i\theta} & & & & & \\ & & e^{1i\theta} & & & & \\ & & & 1 & & & \\ & & & & e^{-1i\theta} & & \\ & & & & & e^{-2i\theta} & \\ & & & & & & e^{-3i\theta} \end{pmatrix}$$

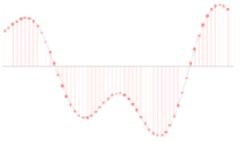


Real irreps

$$\begin{pmatrix} 1 & & & & & & \\ & \cos\theta & \sin\theta & & & & \\ & -\sin\theta & \cos\theta & & & & \\ & & & \cos 2\theta & \sin 2\theta & & \\ & & & -\sin 2\theta & \cos 2\theta & & \\ & & & & & \cos 3\theta & \sin 3\theta \\ & & & & & -\sin 3\theta & \cos 3\theta \end{pmatrix}$$

Regular reps

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}^n \begin{pmatrix} 0.21 \\ 0.20 \\ -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}$$



Feature field types

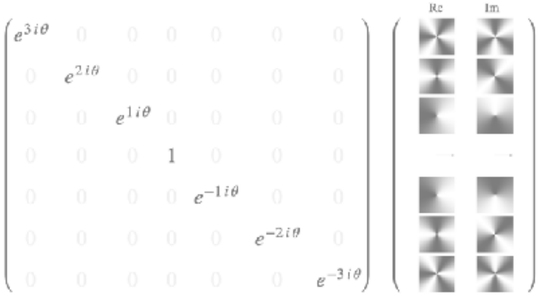
- A feature field is defined by its type ρ
- Feature fields, each of their own type ρ_l, can be stacked:
  - should be thought of as the channels in standard CNNs
- The sub-vectors/channels in these fields:
  - live in their own sub-vector spaces V_l
  - transform by their own representations ρ_l
- Example notations (ρ_l denote irreps)

$$\rho = \rho_1 \oplus \rho_2 \oplus \rho_3 \oplus \dots$$ (Steerable G-CNNs | Fourier)

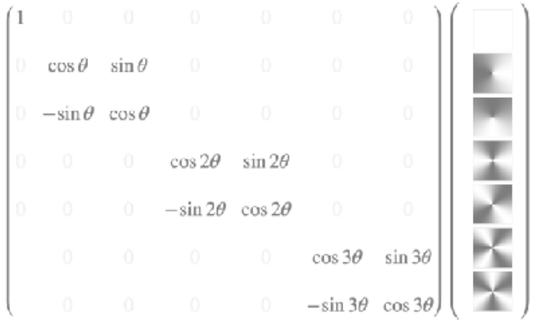
$$\rho = n\rho_0$$ (Normal CNNs with isotropic kernels)

$$\rho = n\mathscr{L}^H$$ (Regular G-CNNs)

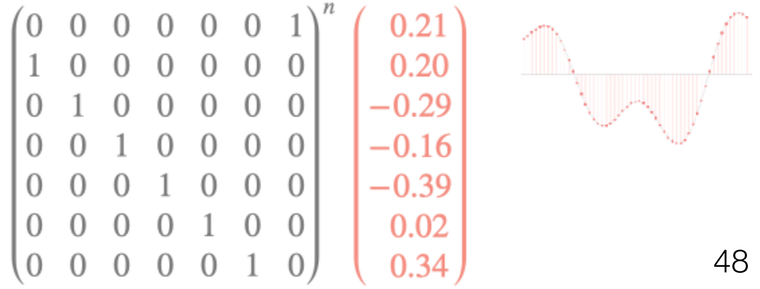
Complex irreps



Real irreps



Regular reps



Feature field t

- A feature field is defined by its
- Feature fields, each of their ov
  - should be thought of as th
- The sub-vectors/channels in t
  - live in their own sub-vecto
  - transform by their own rep
- Example notations (ρ_l denote irreps)

$\rho = \rho_1 \oplus \rho_2 \oplus \rho_3 \oplus \dots$ (Steerable G-CNNs | Fourier)

$\rho = 4\rho_0 \oplus 9\rho_1 \oplus \dots$ (Steerable G-CNNs | General)

define $n\rho_l = \underbrace{\rho_l \oplus \rho_l \oplus \dots \oplus \rho_l}_{\text{n times}}$

$\rho = n\rho_0$ (Normal CNNs with isotropic kernels)

$\rho = n\mathscr{L}^H$ (Regular G-CNNs)

Equation (12) describes the behavior of spherical harmonic *vectors* under rotations, while (15) describes the behavior of Fourier *matrices*. However, the latter is equivalent to saying that each column of the matrices separately transforms according to (12). One of the key ideas of the present paper is to take this property as the basis for the definition of covariance to rotations in neural nets. Thus we have the following definition.

Definition 1 *Let $\mathcal{N}$ be an $S+1$ layer feed-forward neural network whose input is a spherical function $f^0 \colon S^2 \to \mathbb{C}^d$. We say that $\mathcal{N}$ is a* ***generalized*** *$\mathrm{SO}(3)$***–covariant spherical CNN** *if the output of each layer s can be expressed as a collection of vectors*

$$\widehat{f}^s = (\underbrace{\widehat{f}^s_{0,1}, \widehat{f}^s_{0,2}, \dots, \widehat{f}^s_{0,\tau^s_0}}_{\ell=0}, \underbrace{\widehat{f}^s_{1,1}, \widehat{f}^s_{1,2}, \dots, \widehat{f}^s_{1,\tau^s_1}}_{\ell=1}, \dots\dots\dots, \underbrace{\dots \widehat{f}^s_{L,\tau^s_L}}_{\ell=L}), \tag{14}$$

where each $\widehat{f}^s_{\ell,j} \in \mathbb{C}^{2\ell+1}$ is a ρ_ℓ–covariant vector in the sense that if the input image is rotated by some rotation R, then $\widehat{f}^s_{\ell,j}$ transforms as

$$\widehat{f}^s_{\ell,j} \mapsto \rho(R) \cdot \widehat{f}^s_{\ell,j}. \tag{15}$$

We call the individual $\widehat{f}^s_{\ell,j}$ vectors the irreducible ***fragments*** *of $\widehat{f}^s$, and the integer vector $\tau^s = (\tau^s_0, \tau^s_1, \dots, \tau^s_L)$ counting the number of fragments for each ℓ the* ***type*** *of $\widehat{f}^s$.*

There are a few things worth noting about Definition 1. First, since the (15) maps are linear, clearly any $\mathrm{SO}(3)$–covariant spherical CNN is equivariant to rotations, as defined in the introduction. Second, since in [1] the inputs are functions on the sphere, whereas in higher layers the activations are functions on $\mathrm{SO}(3)$, their architecture is a special case of Definition 1 with $\tau^0 = (1, 1, \dots, 1)$ and $\tau^s = (1, 3, 5, \dots, 2L+1)$ for $s \geq 1$.

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 a generalization of this work that generally exhibits improved performace, but from an implementation point of view is actually simpler. An unusual feature of the proposed architecture is that it uses the Clebsch–Gordan transform as its only source of nonlinearity, thus avoiding repeated forward and backward Fourier transforms. The underlying ideas of the paper generalize to constructing neural networks that are invariant to the action of other compact groups.

1 Introduction

Despite the many recent breakthroughs in deep learning, we still do not have a satisfactory understanding of how deep neural networks are able to achieve such spectacular perfomance on a wide range of learning problems. One thing that is clear, however, is that certain architectures pick up on natural invariances in data, and this is a key component to their success. The classic example is of course Convolutional Neural Networks (CNNs) for image classification [2]. Recall that, fundamentally, each layer of a CNN realizes two simple operations: a linear one consisting of convolving the previous layer's activations with a (typically small) learnable filter, and a nonlinear but pointwise one, such as a ReLU operator[2]. This architecture is sufficient to guarantee *translation equivariance*, meaning that if the input image is translated by some vector $\boldsymbol{t}$, then the activation pattern in each higher layer of the network will translate by the same amount. Equivariance is crucial to image recognition for two closely related reasons: (a) It guarantees that exactly the same filters are applied to each part the input image regardless of position. (b) Assuming that finally, at the very top of the network, we add some layer that is translation *invariant*, the entire network will be invariant, ensuring that it can detect any given object equally well regardless of its location.

Recently, a number of papers have appeared that examine equivariance from the theoretical point of view, motivated by the understanding that the natural way to generalize convolutional networks to other types of data will likely lead through generalizing the notion of equivariance itself to other transformation groups [3, 4, 5, 6, 7]. Letting f^s denote the activations of the neurons in layer s of a hypothetical generalized convolution-like neural network, mathematically, equivariance to a group G means that if the inputs to the network are transformed by some transformation $g \in G$, then f^s transforms to $T^s_g(f^s)$ for some fixed set of linear transformations $\{T^s_g\}_{g\in G}$. s(Note that in some contexts this is called "covariance", the difference between the two words being only one of emphasis.)

*Authors are arranged alphabetically
[2]Real CNNs typically of course have multiple channels, and correspondingly multiple filters per layer, but this does not fundamentally change the network's invariance properties.

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

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} -0.29 \\ -0.16 \\ -0.39 \\ 0.02 \\ 0.34 \end{pmatrix}$$

Feature field types

- A feature field is defined by its type ρ
- Feature fields, each of their own type ρ_l, can be stacked:
  - should be thought of as the channels in standard CNNs
- The sub-vectors/channels in these fields:
  - live in their own sub-vector spaces V_l
  - transform by their own representations ρ_l
- Example notations (ρ_l denote irreps)

$$\rho = \rho_1 \oplus \rho_2 \oplus \rho_3 \oplus \dots$$ (Steerable G-CNNs | Fourier)

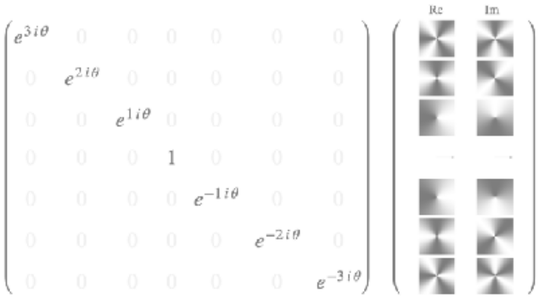
$$\rho = 4\rho_0 \oplus 9\rho_1 \oplus \dots$$ (Steerable G-CNNs | General)

define $n\rho_l = \underbrace{\rho_l \oplus \rho_l \oplus \dots \oplus \rho_l}_{\text{n times}}$

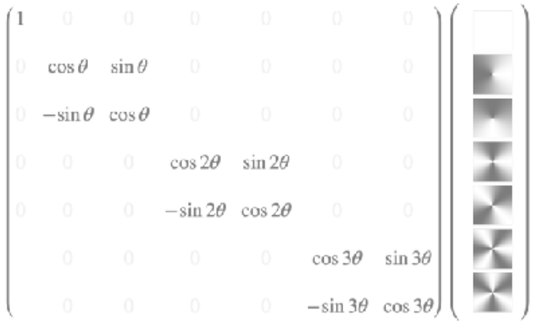
$$\rho = n\rho_0$$ (Normal CNNs with isotropic kernels)

$$\rho = n\mathscr{L}^H$$ (Regular G-CNNs)

Complex irreps

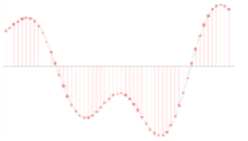


Real irreps



Regular reps

$$\begin{pmatrix} 0&0&0&0&0&0&1\\ 1&0&0&0&0&0&0\\ 0&1&0&0&0&0&0\\ 0&0&1&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&1&0&0\\ 0&0&0&0&0&1&0 \end{pmatrix}^n \begin{pmatrix} 0.21\\ 0.20\\ -0.29\\ -0.16\\ -0.39\\ 0.02\\ 0.34 \end{pmatrix}$$



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/

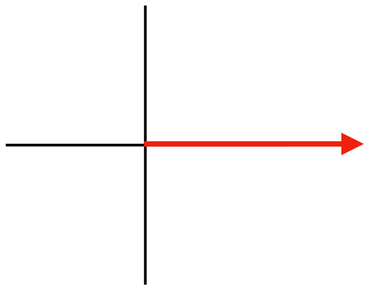
Activation functions

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**

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



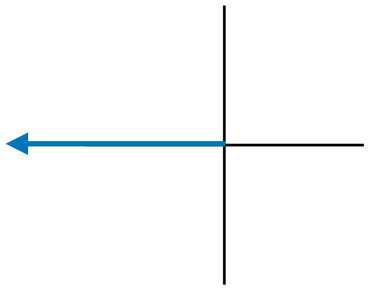
Activation functions

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**

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



Activation functions

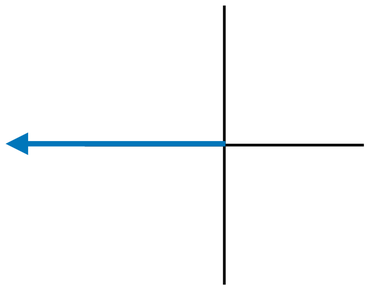
Activation functions should commute with the representation of the fibers

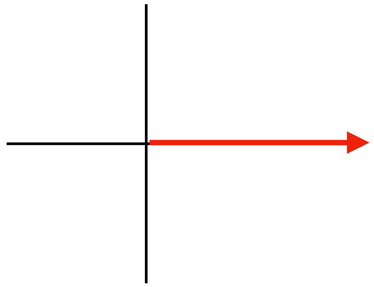
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

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

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

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





Activation functions

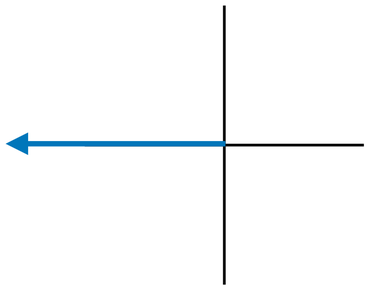
Activation functions should commute with the representation of the fibers

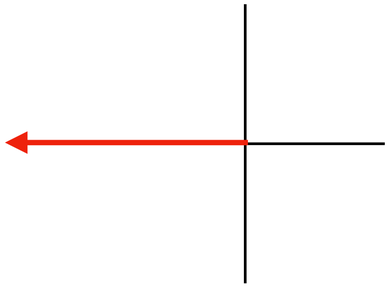
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

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

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

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





Activation functions

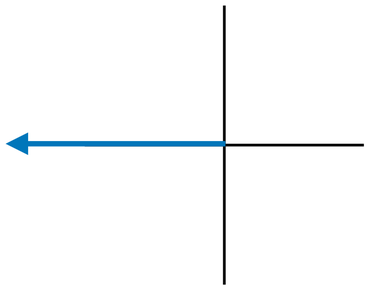
Activation functions should commute with the representation of the fibers

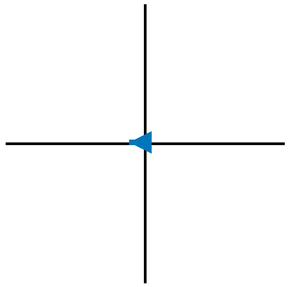
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

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

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

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





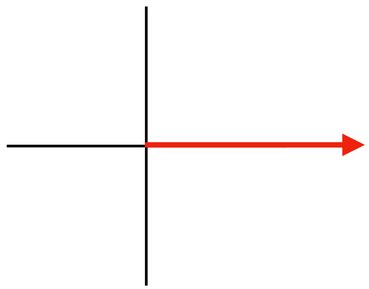
Activation functions

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**

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



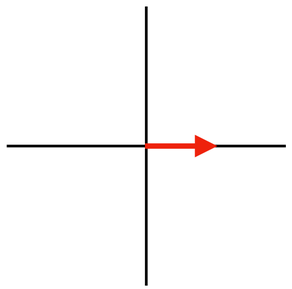
Activation functions

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**

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



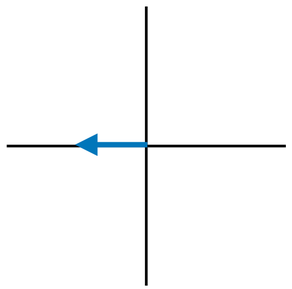
Activation functions

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

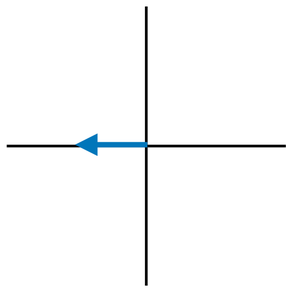
Activation functions should commute with the representation of the fibers

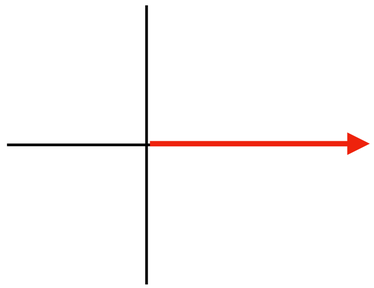
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

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

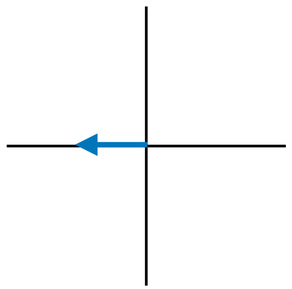
Activation functions should commute with the representation of the fibers

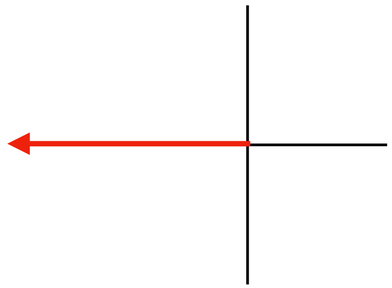
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

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

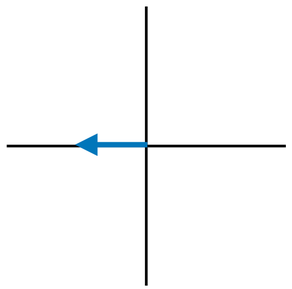
Activation functions should commute with the representation of the fibers

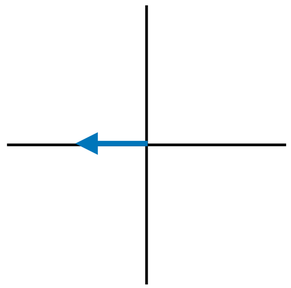
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

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

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

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





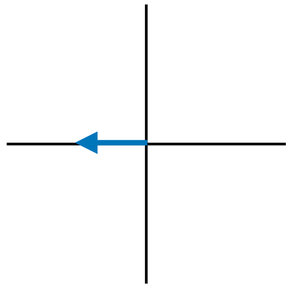
Activation functi

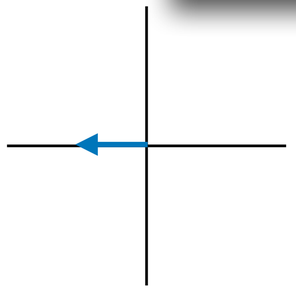
Activation functions should commute with the representation of

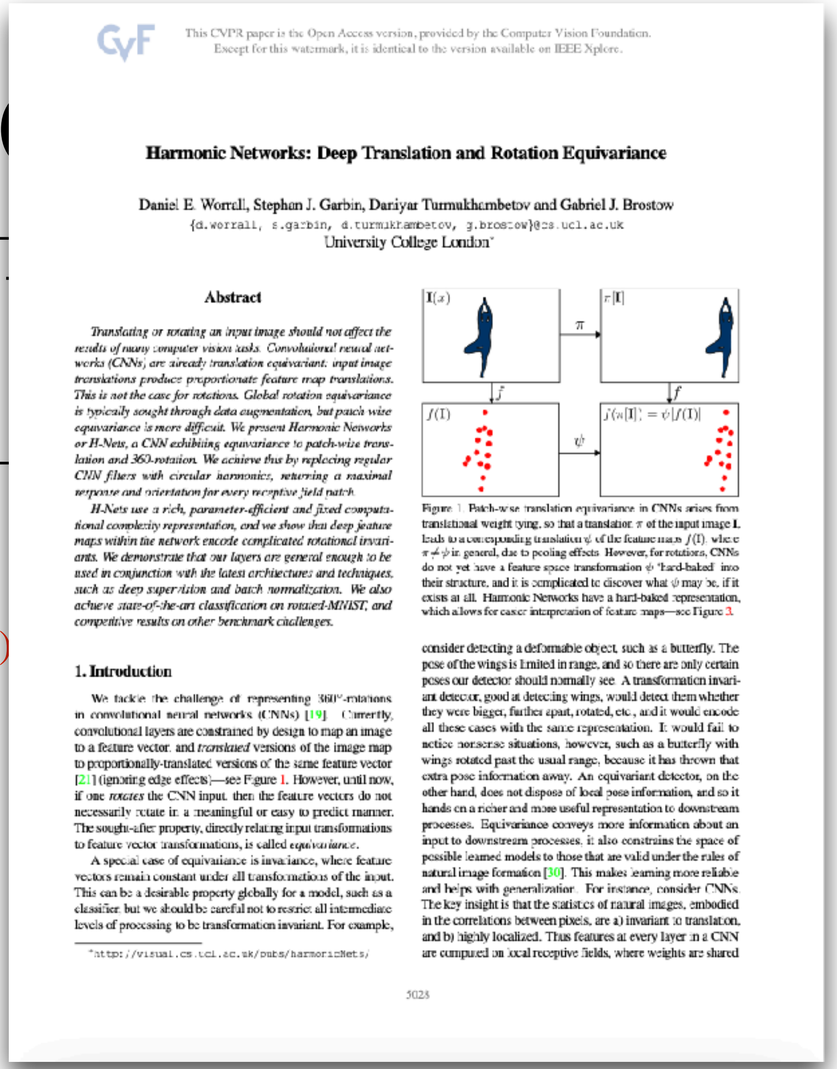
$$\sigma\left(\rho(g)\,\hat{f}(\mathbf{x})\right) = \rho'(g)\,\sigma\left(\hat{f}(\mathbf{x})\right)$$

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

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







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.

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/

Figure 1. Patch-wise translation equivariance in CNNs arises from translational weight tying, so that a translation π of the input image I, leads to a corresponding translation ψ of the feature maps $f(\mathrm{I})$, 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.

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

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**

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

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

using predicted scalar fields (type-0)

Activation functi

Activation functions should commute with the representation c

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

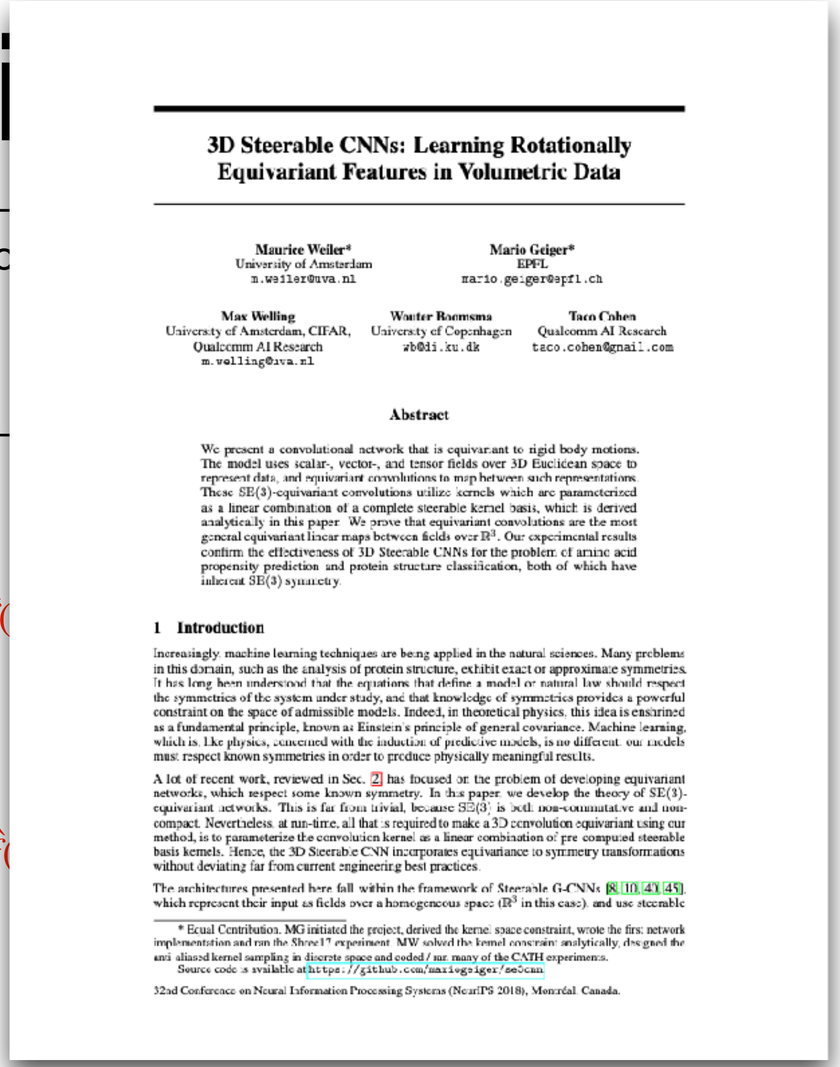
Compatible activation function: **norm-based activation functions**

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

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, 41, 45], which represent their input as fields over a homogeneous space ($\mathbb{R}^3$ in this case), and use steerable

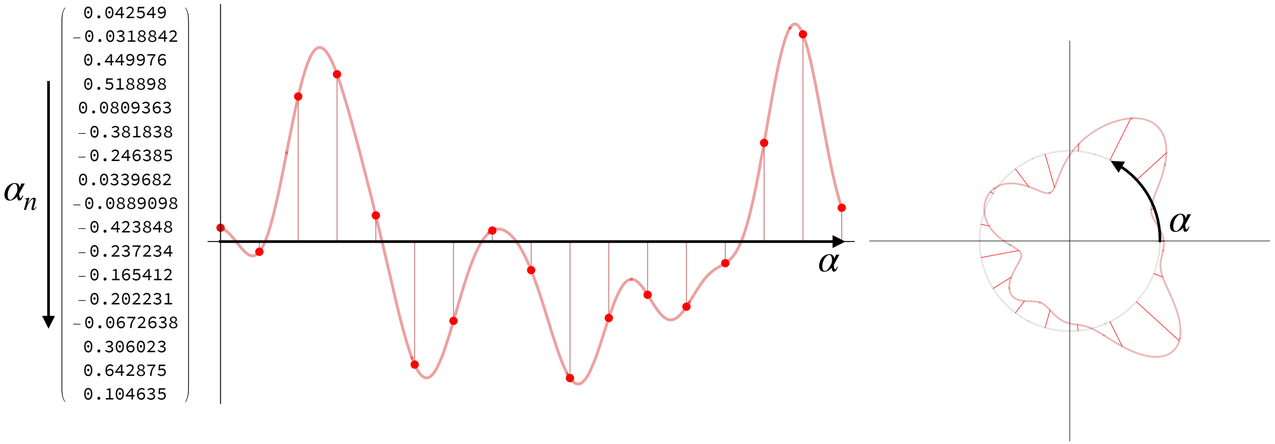
* Equal Contribution. MG initiated the project, derived the kernel space constraint, wrote the first network implementation and ran the Shrec17 experiment. MW solved the kernel constraint analytically, designed the anti-aliased kernel sampling in discrete space and coded / ran many of the CATH experiments.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

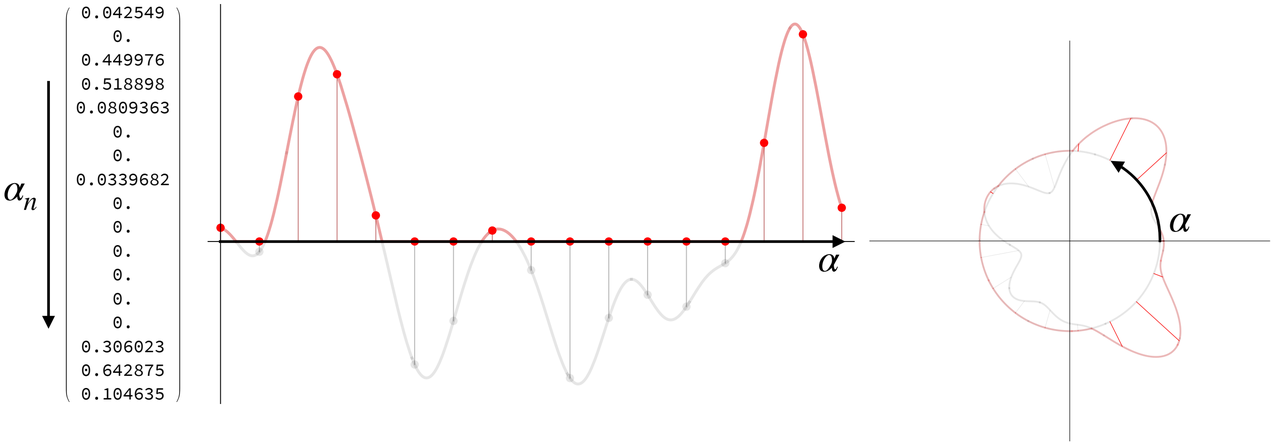
$$\mathscr{L}_\theta\,\sigma(f(\mathbf{x},\alpha)) \quad = \quad \sigma(f(\mathbf{x},\alpha-\theta)) \quad = \quad f'(\mathbf{x},\alpha-\theta)$$

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**

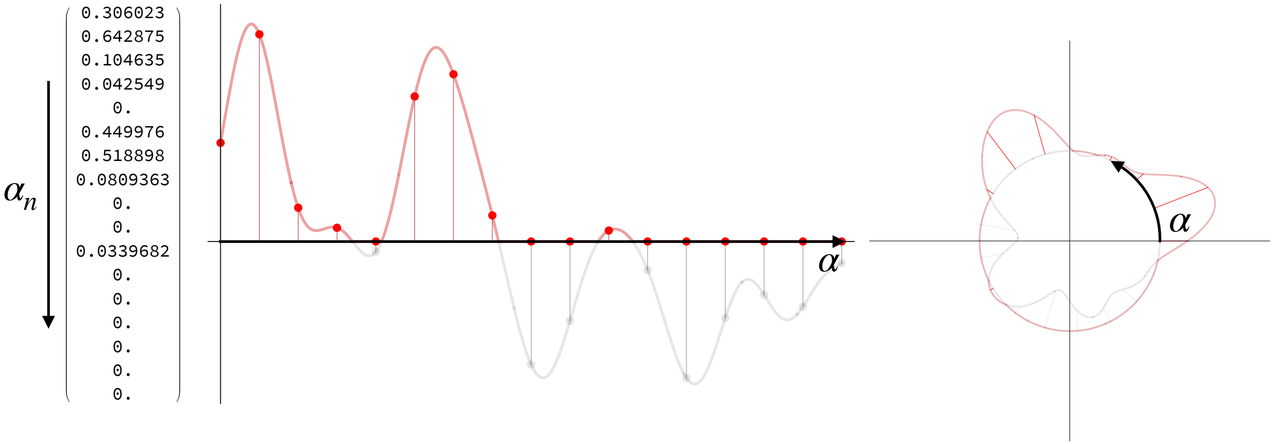
$$\mathscr{L}_\theta\,\sigma(f(\mathbf{x},\alpha)) \quad = \quad \sigma(f(\mathbf{x},\alpha-\theta)) \quad = \quad f'(\mathbf{x},\alpha-\theta)$$

element-wise activations commute with permutations

Activation functions

Activation functions should commute with the representation of the fibers

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



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

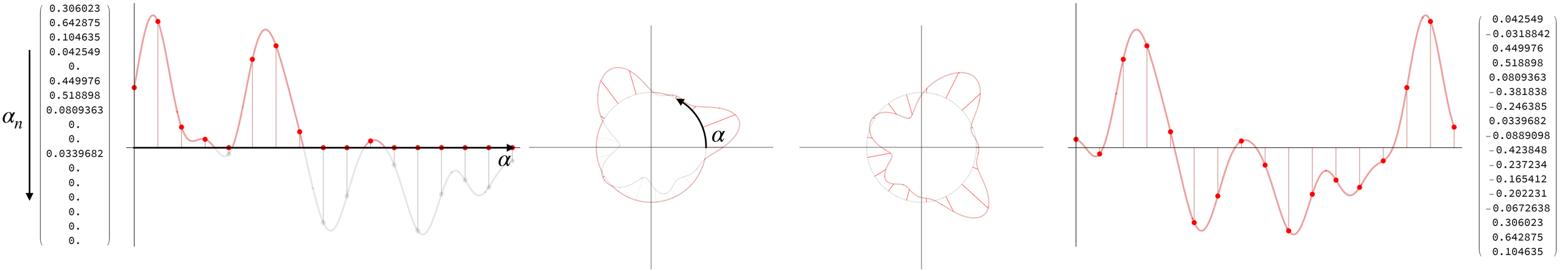
$$\mathscr{L}_\theta\,\sigma(f(\mathbf{x},\alpha)) \quad = \quad \sigma(f(\mathbf{x},\alpha-\theta)) \quad = \quad f'(\mathbf{x},\alpha-\theta)$$

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**

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

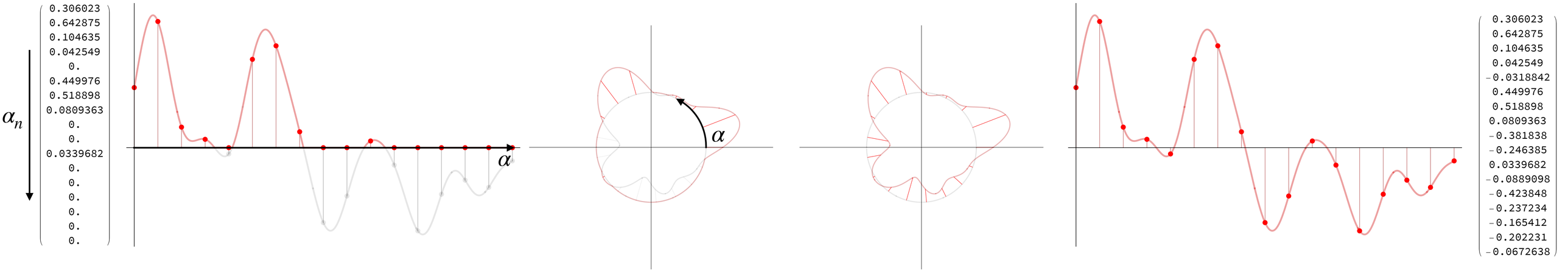
$$\sigma(\mathscr{L}_\theta f(\mathbf{x},\alpha)) \quad = \quad \sigma(f(\mathbf{x},\alpha-\theta)) \quad = \quad f'(\mathbf{x},\alpha-\theta)$$

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**

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

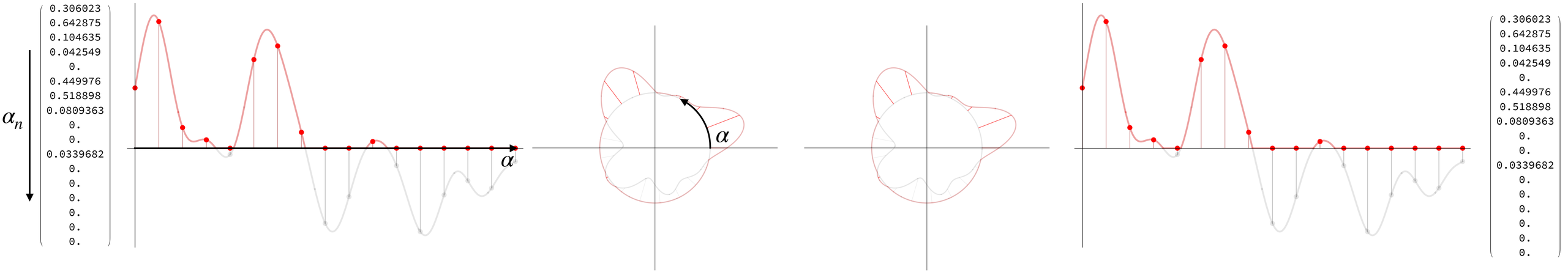
$$\sigma(\mathscr{L}_\theta f(\mathbf{x},\alpha)) \quad = \quad \sigma(f(\mathbf{x},\alpha-\theta)) \quad = \quad f'(\mathbf{x},\alpha-\theta)$$

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**

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

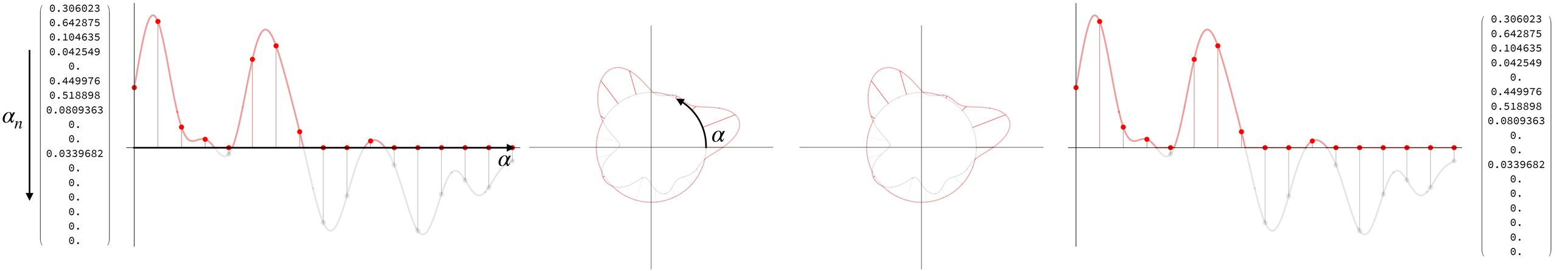
$$\sigma(\mathscr{L}_\theta f(\mathbf{x},\alpha)) \quad = \quad \sigma(f(\mathbf{x},\alpha-\theta)) \quad = \quad f'(\mathbf{x},\alpha-\theta)$$

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

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

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

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: **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

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

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

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

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

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

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

1. INTRODUC

operator T_1^g 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

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

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 $\overline{\psi}^{\ell}_m$ 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.,

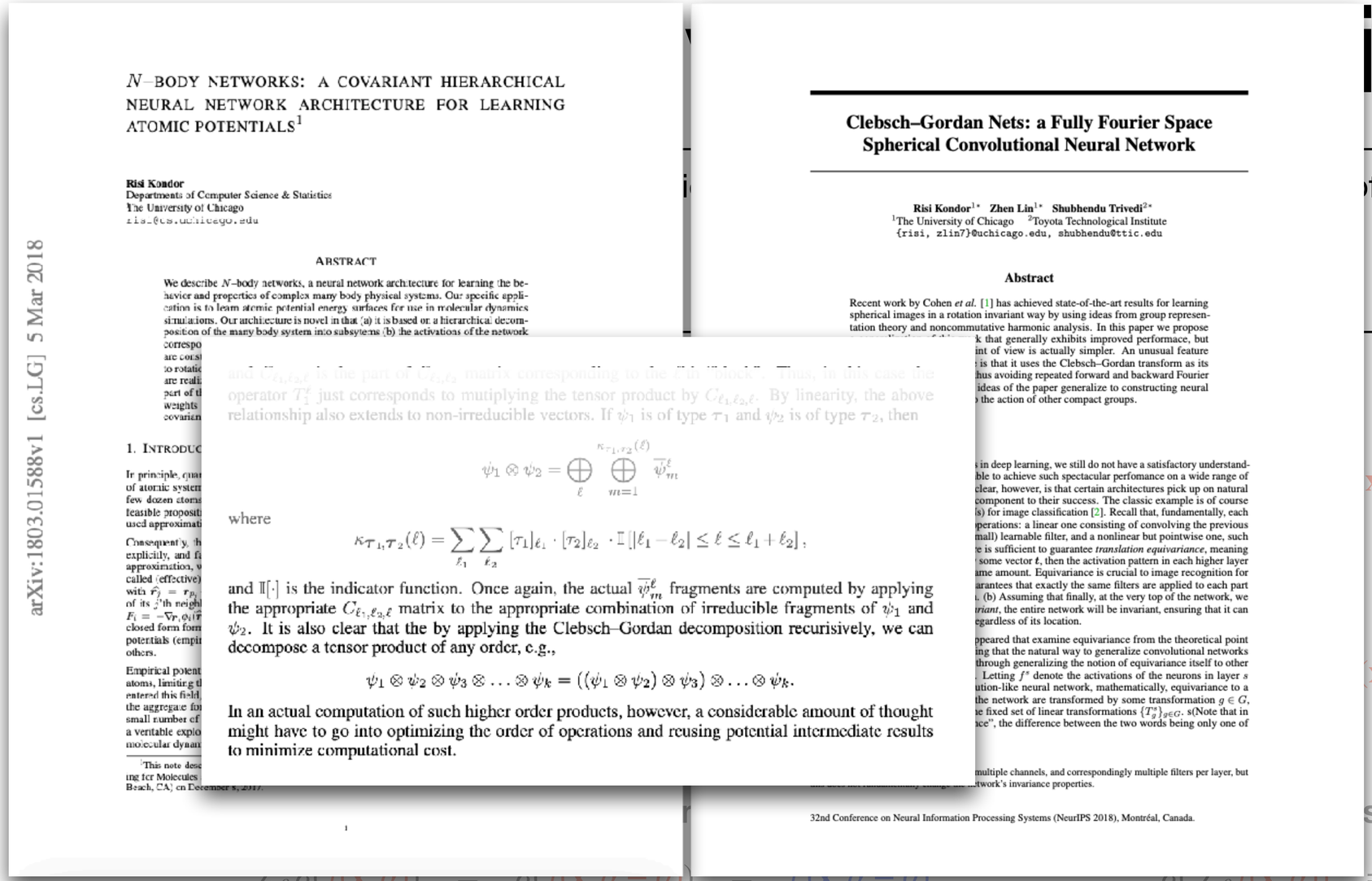
$$\psi_1 \otimes \psi_2 \otimes \psi_3 \otimes \ldots \otimes \psi_k = ((\psi_1 \otimes \psi_2) \otimes \psi_3) \otimes \ldots \otimes \psi_k.$$

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)

ions

of the fibers



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

operator T_1^g 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

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

where

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

and $\mathbb{I}[\cdot]$ is the indicator function. Once again, the actual $\overline{\psi}^{\ell}_{m}$ 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.

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.

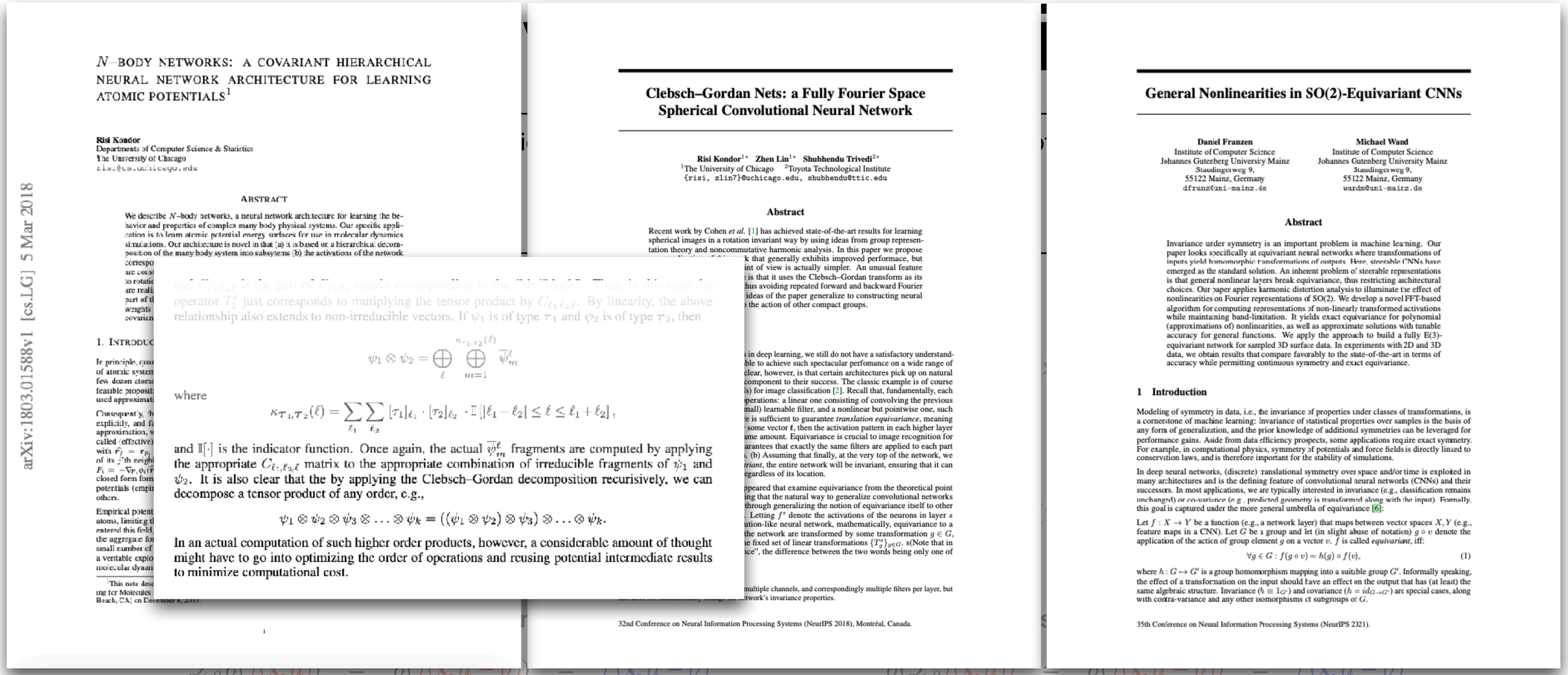
$$\mathbf{x}) \quad = \quad \rho(\mathbf{R}_\pi)\,\sigma(\|\hat{f}(\mathbf{x})\|)\hat{f}(\mathbf{x}) \quad = \quad \hat{f}'(\mathbf{x})$$

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

s or scalar fields Fourier-based

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

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



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

1. INTRODUC

operator T_1^g 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

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

where

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

and $\mathbb{I}[\cdot]$ is the indicator function. Once again, the actual $\overline{\psi}^{\ell}_{m}$ 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.

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.

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

Michael Wand
Institute of Computer Science
Johannes Gutenberg University Mainz
Staudingerweg 9,
55122 Mainz, Germany
wandm@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 $f : X \to Y$ 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:

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

where $h : G \mapsto G'$ 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 ($h = id_{G \to G'}$) are special cases, along with contra-variance and any other isomorphisms of subgroups of G.

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

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**

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

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)

Group Equivariant Deep Learning

Lecture 2 - Steerable group convolutions

Lecture 2.7 - Derivation of Harmonic[1] nets from regular g-convs

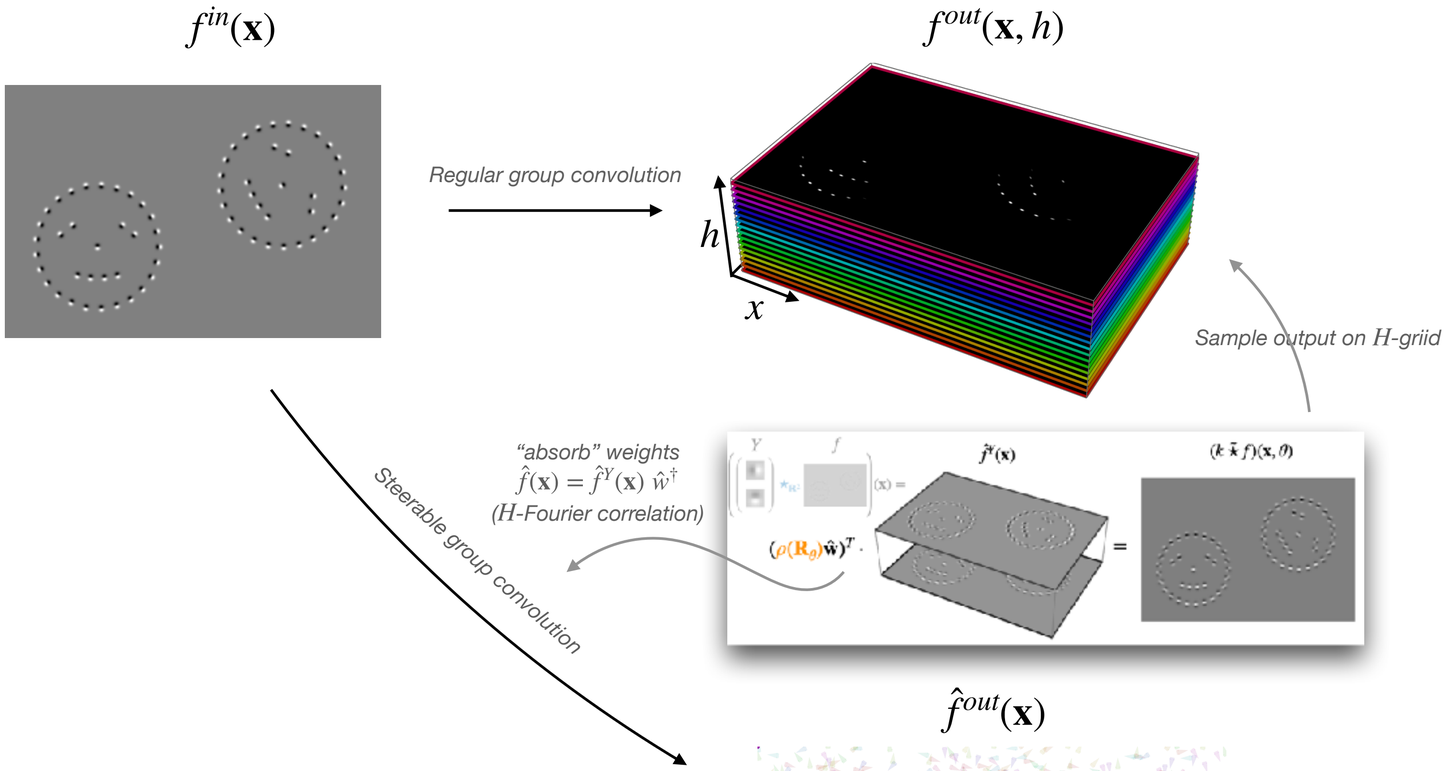
Using complex irreps of $SO(2)$

[1] Worrall, D. E., Garbin, S. J., Turmukhambetov, D., & Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. CVPR 2017

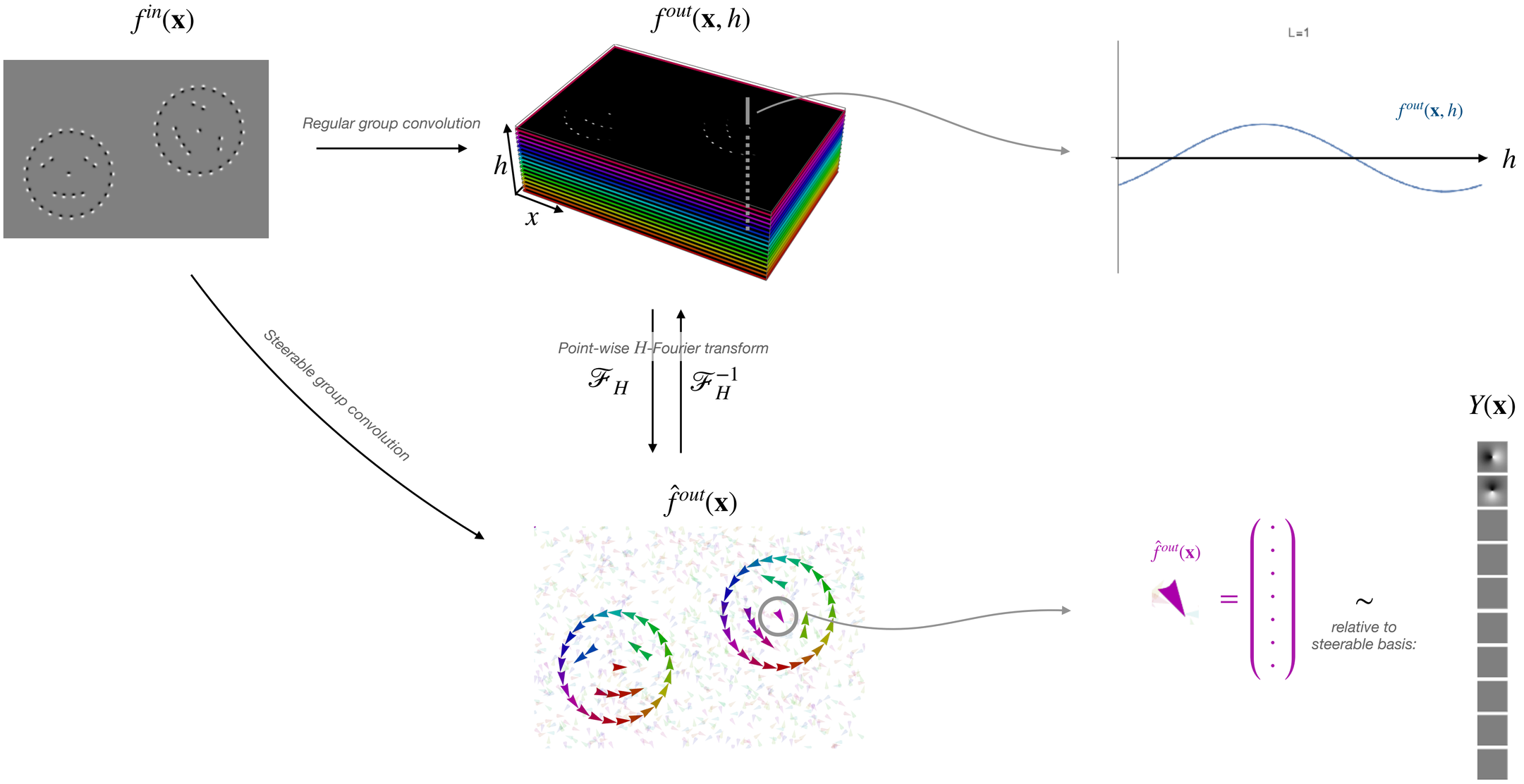
Erik Bekkers, Amsterdam Machine Learning Lab, University of Amsterdam

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

From regular to steerable via a Fourier transform



From regular to steerable via a Fourier transform



From regular to steerable via a Fourier transform

Regular group convolutions:
Domain expanded feature maps

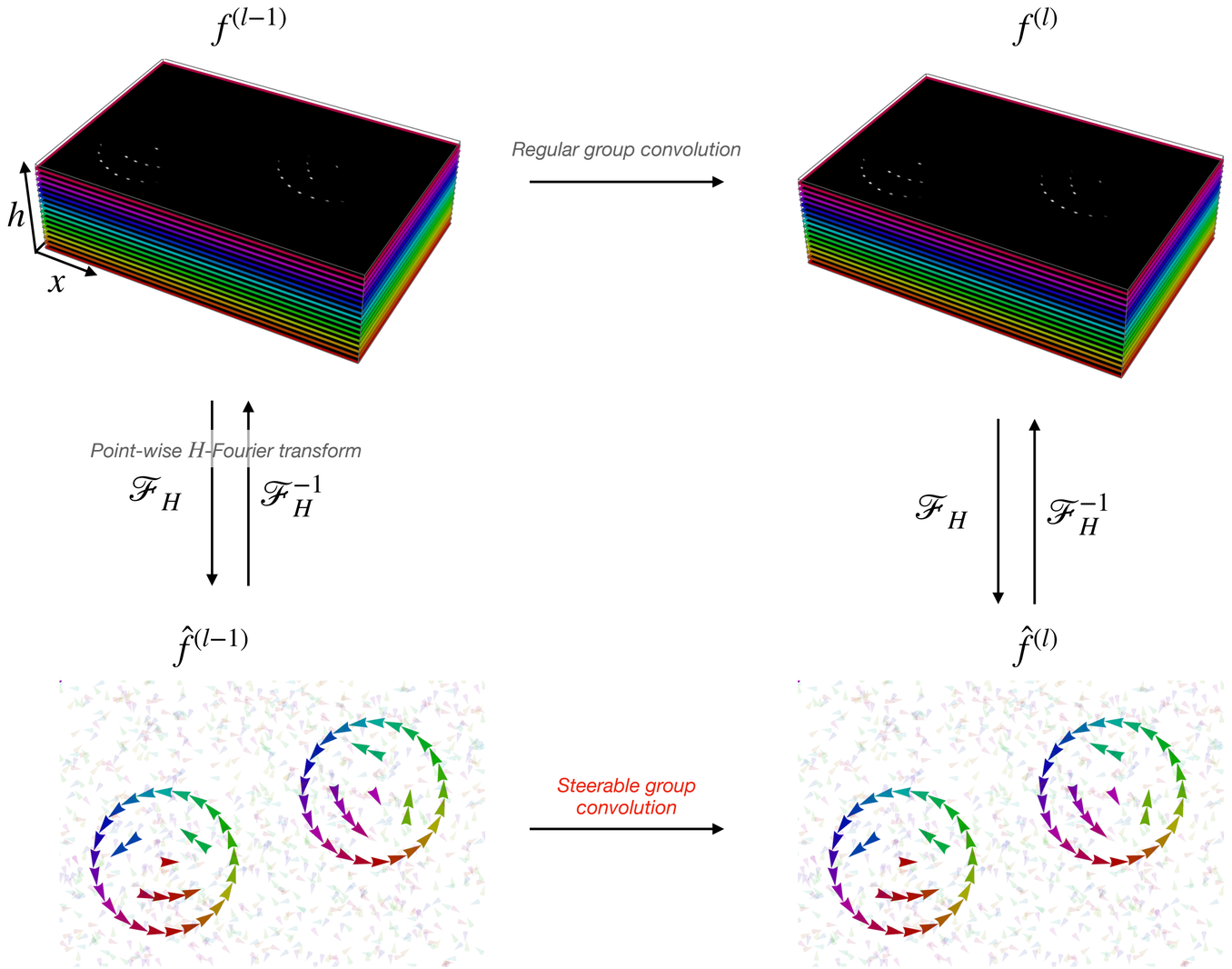
$$f^{(l)} : \mathbb{R}^d \times H \to \mathbb{R}$$

added axis

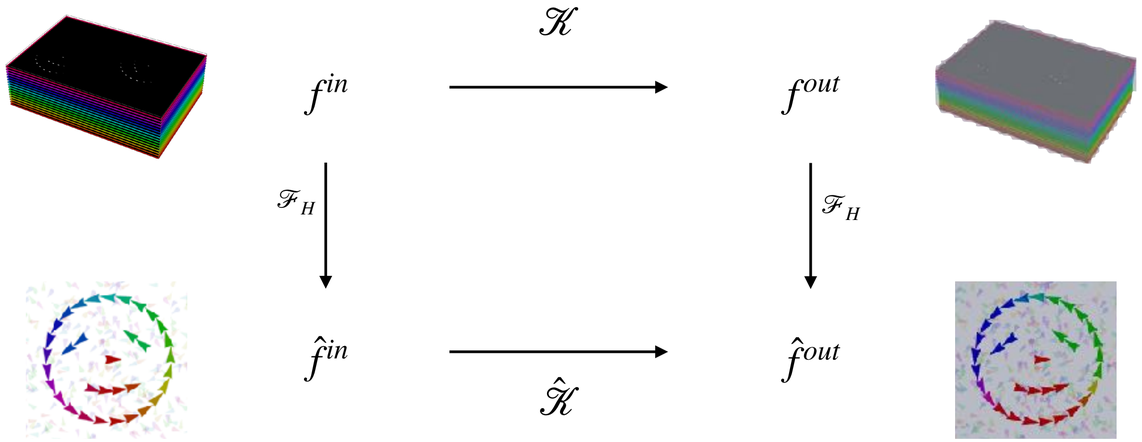
Steerable group convolutions:
Co-domain expanded feature maps (feature fields)

$$\hat{f}^{(l)} : \mathbb{R}^d \to V_H$$

vector field instead of scalar field
(vectors in V_H transform via group H representations)



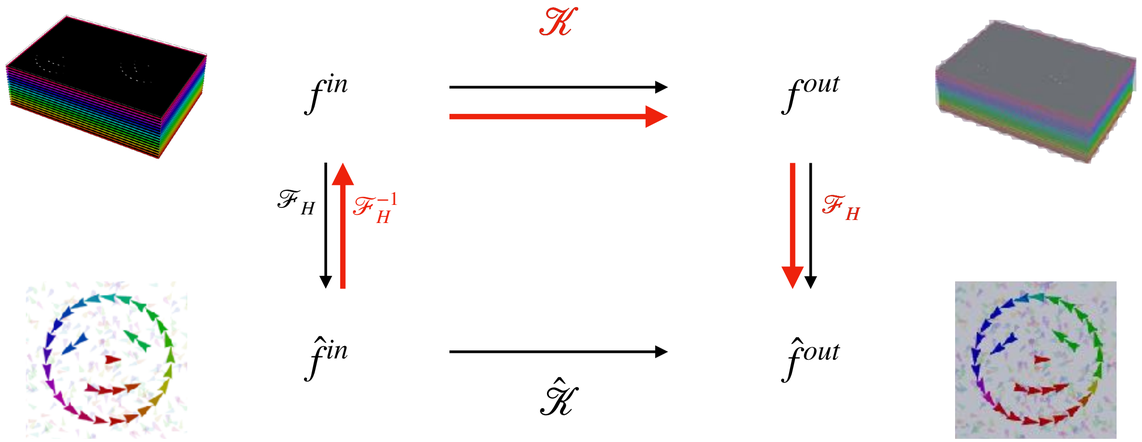
Deriving $\hat{\mathscr{K}}$ from the knowns $(\mathscr{K}, \mathscr{F}_H)$



Instead of solving the kernel constraint, let's compute

$$\hat{\mathscr{K}}(\hat{f}^{in}) = [\mathscr{F}_H \circ \mathscr{K} \circ \mathscr{F}_H^{-1}](\hat{f}^{in})$$

Deriving $\hat{\mathcal{K}}$ from the knowns $(\mathcal{K}, \mathcal{F}_H)$



Instead of solving the kernel constraint, let's compute

$$\hat{\mathcal{K}}(\hat{f}^{in}) = [\mathcal{F}_H \circ \mathcal{K} \circ \mathcal{F}_H^{-1}](\hat{f}^{in})$$

Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

$$\hat{\mathscr{K}}(\hat{f}) = \mathscr{F}_H(\mathscr{K}(\mathscr{F}_H^{-1}(\hat{f})))$$

Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

$$\hat{\mathscr{K}}(\hat{f}) = \mathscr{F}_H(\mathscr{K}(\mathscr{F}_H^{-1}(\hat{f})))$$

Deriving $\hat{\mathscr{K}}$ from the knowns $(\mathscr{K}, \mathscr{F}_H)$

$$\hat{\mathscr{K}}(\hat{f}) = \mathscr{F}_H(\mathscr{K}(\mathscr{F}_H^{-1}(\hat{f})))$$

$$\hat{\mathscr{K}}(\hat{f})_j(\mathbf{x}) = \int_{S^1} \mathscr{K}(\mathscr{F}_H^{-1}(\hat{f}))(\mathbf{x}, \theta)\, e^{ij\theta} \mathrm{d}\theta$$

Fourier transform: $\mathscr{F}_H(f(\cdot))_j = \int_{S^1} f(\theta) e^{ij\theta} \mathrm{d}\theta$

Deriving $\hat{\mathscr{K}}$ from the knowns $(\mathscr{K}, \mathscr{F}_H)$

$$\hat{\mathscr{K}}(\hat{f}) = \mathscr{F}_H(\mathscr{K}(\mathscr{F}_H^{-1}(\hat{f})))$$

$$\hat{\mathscr{K}}(\hat{f})_j(\mathbf{x}) = \int_{S^1} \mathscr{K}(\mathscr{F}_H^{-1}(\hat{f}))(\mathbf{x}, \theta)\, e^{ij\theta} \mathrm{d}\theta$$

$$= \int_{S^1} \int_{\mathbb{R}^2} \int_{S^1} k(\mathbf{R}_\theta(\mathbf{x}' - \mathbf{x}), \theta' - \theta) \sum_l \hat{f}_l(\mathbf{x}') e^{-il\theta'} \, \mathrm{d}\mathbf{x}' \mathrm{d}\theta' \, e^{ij\theta} \mathrm{d}\theta$$

Fourier transform: $\mathscr{F}_H(f(\cdot))_j = \int_{S^1} f(\theta) e^{ij\theta} \mathrm{d}\theta$

Regular group conv: $\mathscr{K}(f)(g) = \int_G k(g^{-1}g') f(g') \mathrm{d}g'$

Inverse Fourier trafo: $\mathscr{F}_H^{-1}(\hat{f})(h) = \sum_l \hat{f}_l e^{-il\theta'}$

Deriving $\hat{\mathscr{K}}$ from the knowns $(\mathscr{K}, \mathscr{F}_H)$

$$\hat{\mathscr{K}}(\hat{f}) = \mathscr{F}_H(\mathscr{K}(\mathscr{F}_H^{-1}(\hat{f})))$$

$$\hat{\mathscr{K}}(\hat{f})_j(\mathbf{x}) = \int_{S^1} \mathscr{K}(\mathscr{F}_H^{-1}(\hat{f}))(\mathbf{x}, \theta)\, e^{ij\theta} \mathrm{d}\theta$$

$$= \int_{S^1} \int_{\mathbb{R}^2} \int_{S^1} k(\mathbf{R}_\theta(\mathbf{x}' - \mathbf{x}), \theta' - \theta) \sum_l \hat{f}_l(\mathbf{x}') e^{-il\theta'} \,\mathrm{d}\mathbf{x}' \mathrm{d}\theta'\, e^{ij\theta} \mathrm{d}\theta$$

$$= \int_{\mathbb{R}^2} \sum_l \int_{S^1} \int_{S^1} k(\mathbf{R}_\theta(\mathbf{x}' - \mathbf{x}), \theta' - \theta)\, e^{-il\theta'}\, e^{ij\theta} \mathrm{d}\theta\, \mathrm{d}\theta'\, \hat{f}_l(\mathbf{x}')\, \mathrm{d}\mathbf{x}'$$

Fourier transform: $\mathscr{F}_H(f(\cdot))_j = \int_{S^1} f(\theta) e^{ij\theta} \mathrm{d}\theta$

Regular group conv: $\mathscr{K}(f)(g) = \int_G k(g^{-1}g') f(g') \mathrm{d}g'$

Inverse Fourier trafo: $\mathscr{F}_H^{-1}(\hat{f})(h) = \sum_l \hat{f}_l e^{-il\theta'}$

Deriving $\hat{\mathcal{K}}$ from the knowns $(\mathcal{K}, \mathcal{F}_H)$

$$\hat{\mathcal{K}}(\hat{f}) = \mathcal{F}_H(\mathcal{K}(\mathcal{F}_H^{-1}(\hat{f})))$$

$$\begin{aligned}
\hat{\mathcal{K}}(\hat{f})_j(\mathbf{x}) &= \int_{S^1} \mathcal{K}(\mathcal{F}_H^{-1}(\hat{f}))(\mathbf{x}, \theta)\, e^{ij\theta} \mathrm{d}\theta \\
&= \int_{S^1} \int_{\mathbb{R}^2} \int_{S^1} k(\mathbf{R}_\theta(\mathbf{x}' - \mathbf{x}), \theta' - \theta) \sum_l \hat{f}_l(\mathbf{x}') e^{-il\theta'} \,\mathrm{d}\mathbf{x}' \mathrm{d}\theta'\, e^{ij\theta} \mathrm{d}\theta \\
&= \int_{\mathbb{R}^2} \sum_l \int_{S^1} \int_{S^1} k(\mathbf{R}_\theta(\mathbf{x}' - \mathbf{x}), \theta' - \theta)\, e^{-il\theta'}\, e^{ij\theta} \mathrm{d}\theta\, \mathrm{d}\theta'\, \hat{f}_l(\mathbf{x}')\, \mathrm{d}\mathbf{x}'
\end{aligned}$$

$$\hat{\mathcal{K}}(\hat{f})(\mathbf{x}) = \int_{\mathbb{R}^2} \hat{k}(\mathbf{x}' - \mathbf{x})\, \hat{f}_l(\mathbf{x}')\, \mathrm{d}\mathbf{x}'$$

Fourier transform: $\mathcal{F}_H(f(\cdot))_j = \int_{S^1} f(\theta) e^{ij\theta} \mathrm{d}\theta$

Regular group conv: $\mathcal{K}(f)(g) = \int_G k(g^{-1}g') f(g') \mathrm{d}g'$

Inverse Fourier trafo: $\mathcal{F}_H^{-1}(\hat{f})(h) = \sum_l \hat{f}_l\, e^{-il\theta'}$

A normal conv with a kernel $\hat{k} : \mathbb{R}^d \to \mathbb{R}^{d_{out} \times d_{in}}$!

(recall lecture 2.5)

Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

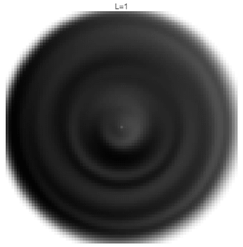
So we “just” need to compute

$$\hat{k}_{jl}(\mathbf{x}' - \mathbf{x}) = \int_{S^1} \int_{S^1} k(\mathbf{R}_\theta(\mathbf{x}' - \mathbf{x}), \theta' - \theta)\, e^{-il\theta'}\, e^{ij\theta} \mathrm{d}\theta\, \mathrm{d}\theta'$$

Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

So we “just” need to compute

$$\hat{k}_{jl}(\mathbf{x}' - \mathbf{x}) = \int_{S^1} \int_{S^1} k(\mathbf{R}_\theta(\mathbf{x}' - \mathbf{x}), \theta' - \theta)\, e^{-il\theta'}\, e^{ij\theta} \mathrm{d}\theta\, \mathrm{d}\theta'$$



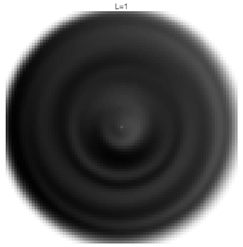
Recall that we given enough frequencies we can expand any spatial kernel in circular harmonics (lecture 2.1)

$$k(\mathbf{x}, \theta) = \sum_J w_J(r, \theta) e^{iJ\alpha}$$

Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

So we “just” need to compute

$$\hat{k}_{jl}(\mathbf{x}'-\mathbf{x}) = \int_{S^1}\int_{S^1} k(\mathbf{R}_\theta(\mathbf{x}'-\mathbf{x}), \theta'-\theta)\, e^{-il\theta'}\, e^{ij\theta} \mathrm{d}\theta\, \mathrm{d}\theta'$$



Recall that we given enough frequencies we can expand any spatial kernel in circular harmonics (lecture 2.1)

$$k(\mathbf{x}, \theta) = \sum_J w_J(r, \theta) e^{iJ\alpha}$$

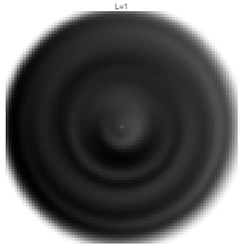
Such kernels are spatially rotation steerable via

$$\begin{aligned} k(\mathbf{R}_\theta^{-1}\mathbf{x}, \theta'-\theta) &= \sum_J w_J(r, \theta'-\theta) e^{iJ(\alpha-\theta)} \\ &= e^{-iJ\theta} \sum_J w_J(r, \theta'-\theta) e^{iJ\alpha} \\ &= e^{-iJ\theta} k(\mathbf{x}, \theta'-\theta) \end{aligned}$$

Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

So we “just” need to compute

$$\hat{k}_{jl}(\mathbf{x}'-\mathbf{x}) = \int_{S^1}\int_{S^1} k(\mathbf{R}_\theta(\mathbf{x}'-\mathbf{x}), \theta'-\theta)\, e^{-il\theta'}\, e^{ij\theta} \mathrm{d}\theta\, \mathrm{d}\theta'$$



Recall that we given enough frequencies we can expand any spatial kernel in circular harmonics (lecture 2.1)

$$k(\mathbf{x}, \theta) = \sum_J w_J(r, \theta) e^{iJ\alpha}$$

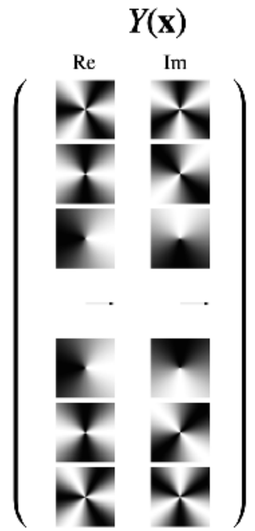
Such kernels are spatially rotation steerable via

$$\begin{aligned} k(\mathbf{R}_\theta^{-1}\mathbf{x}, \theta'-\theta) &= \sum_J w_J(r, \theta'-\theta) e^{iJ(\alpha-\theta)} \\ &= e^{-iJ\theta} \sum_J w_J(r, \theta'-\theta) e^{iJ\alpha} \\ &= e^{-iJ\theta} k(\mathbf{x}, \theta'-\theta) \end{aligned}$$

Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

So we “just” need to compute

$$\hat{k}_{jl}(\mathbf{x}' - \mathbf{x}) = \int_{S^1} \int_{S^1} e^{-iJ\theta} \sum_J w_J(\|\mathbf{x}' - \mathbf{x}\|, \theta' - \theta) e^{iJ\alpha_{\mathbf{x}'-\mathbf{x}}} e^{-il\theta'} e^{ij\theta} \mathrm{d}\theta \, \mathrm{d}\theta'$$



Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

So we “just” need to compute

$$\hat{k}_{jl}(\mathbf{x}' - \mathbf{x}) = \int_{S^1} \int_{S^1} \sum_J w_J(\|\mathbf{x}' - \mathbf{x}\|, \theta' - \theta)\, e^{i\,J\,\alpha_{\mathbf{x}'-\mathbf{x}}}\, e^{-i\,l\,\theta'}\, e^{-i\,(J-j)\,\theta} \mathrm{d}\theta\, \mathrm{d}\theta'$$

Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

So we “just” need to compute

$$\hat{k}_{jl}(\mathbf{x}' - \mathbf{x}) = \int_{S^1} \int_{S^1} \sum_J w_J(\|\mathbf{x}' - \mathbf{x}\|, \theta' - \theta)\, e^{i J \alpha_{\mathbf{x}'-\mathbf{x}}}\, e^{-i l \theta'}\, e^{-i (J-j) \theta} \mathrm{d}\theta\, \mathrm{d}\theta'$$

$$= \int_{S^1} \sum_J \overline{\hat{w}_J}(\|\mathbf{x}' - \mathbf{x}\|)\, e^{i J \alpha_{\mathbf{x}'-\mathbf{x}}}\, e^{-i l \theta}\, e^{-i (J-j) \theta} \mathrm{d}\theta$$

Fourier trafo + shift theorem
(reflection ↔ conjugate)

Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

So we “just” need to compute

$$\hat{k}_{jl}(\mathbf{x}'-\mathbf{x}) = \int_{S^1}\int_{S^1}\sum_J w_J(\|\mathbf{x}'-\mathbf{x}\|, \theta'-\theta)\, e^{iJ\alpha_{\mathbf{x}'-\mathbf{x}}}\, e^{-il\theta'}\, e^{-i(J-j)\theta}\mathrm{d}\theta\,\mathrm{d}\theta'$$

$$= \int_{S^1}\sum_J \overline{\hat{w}_J}(\|\mathbf{x}'-\mathbf{x}\|)\, e^{iJ\alpha_{\mathbf{x}'-\mathbf{x}}}\, e^{-il\theta}\, e^{-i(J-j)\theta}\mathrm{d}\theta$$

Fourier trafo + shift theorem
(reflection ↔ conjugate)

$$= \sum_J \overline{\hat{w}}_J(\|\mathbf{x}'-\mathbf{x}\|)\, e^{iJ\alpha_{\mathbf{x}'-\mathbf{x}}} \int_{S^1} e^{-i(l+J-j)\theta}\mathrm{d}\theta$$

$$\int_{S^1} e^{-i(l+J-j)\theta}\mathrm{d}\theta = \begin{cases} 2\pi & \text{if } \; l+J-j=0 \\ 0 & \text{otherwise} \end{cases}$$

Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

So we “just” need to compute

$$\hat{k}_{jl}(\mathbf{x}'-\mathbf{x}) = \int_{S^1}\int_{S^1}\sum_J w_J(\|\mathbf{x}'-\mathbf{x}\|, \theta'-\theta)\, e^{i\,J\,\alpha_{\mathbf{x}'-\mathbf{x}}}\, e^{-i\,l\,\theta'}\, e^{-i\,(J-j)\,\theta}\mathrm{d}\theta\,\mathrm{d}\theta'$$

$$= \int_{S^1}\sum_J \overline{\hat{w}_J}(\|\mathbf{x}'-\mathbf{x}\|)\, e^{i\,J\,\alpha_{\mathbf{x}'-\mathbf{x}}}\, e^{-i\,l\,\theta}\, e^{-i\,(J-j)\,\theta}\mathrm{d}\theta$$

Fourier trafo + shift theorem
(reflection ↔ conjugate)

$$= \sum_J \overline{\hat{w}}_J(\|\mathbf{x}'-\mathbf{x}\|)\, e^{i\,J\,\alpha_{\mathbf{x}'-\mathbf{x}}} \int_{S^1} e^{-i\,(l+J-j)\,\theta}\mathrm{d}\theta$$

$$\int_{S^1} e^{-i\,(l+J-j)\,\theta}\mathrm{d}\theta = \begin{cases} 2\pi & \text{if } \; l+J-j=0 \\ 0 & \text{otherwise} \end{cases}$$

$$= \sum_{J=j-l} \overline{\hat{w}}_J(\|\mathbf{x}'-\mathbf{x}\|)\, Y_J(\alpha_{\mathbf{x}'-\mathbf{x}})$$

Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

So we “just” need to compute

$$\hat{k}_{jl}(\mathbf{x}'-\mathbf{x}) = \int_{S^1}\int_{S^1}\sum_J w_J(\|\mathbf{x}'-\mathbf{x}\|,\theta'-\theta)\, e^{iJ\alpha_{\mathbf{x}'-\mathbf{x}}}\, e^{-il\theta'}\, e^{-i(J-j)\theta}\mathrm{d}\theta\,\mathrm{d}\theta'$$

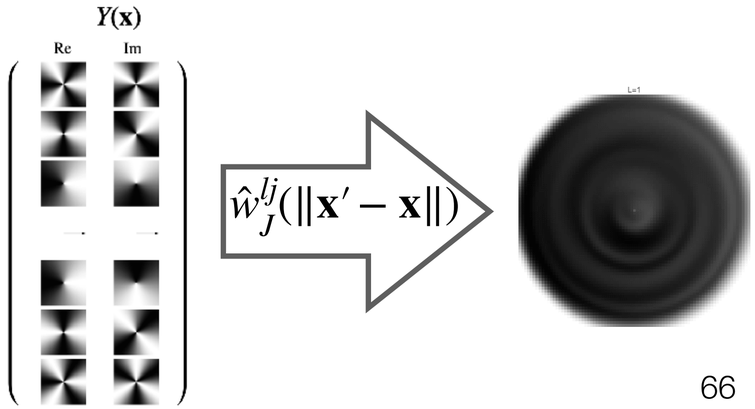
$$= \int_{S^1}\sum_J \overline{\hat{w}_J}(\|\mathbf{x}'-\mathbf{x}\|)\, e^{iJ\alpha_{\mathbf{x}'-\mathbf{x}}}\, e^{-il\theta}\, e^{-i(J-j)\theta}\mathrm{d}\theta$$

Fourier trafo + shift theorem
(reflection ↔ conjugate)

$$= \sum_J \overline{\hat{w}}_J(\|\mathbf{x}'-\mathbf{x}\|)\, e^{iJ\alpha_{\mathbf{x}'-\mathbf{x}}}\int_{S^1} e^{-i(l+J-j)\theta}\mathrm{d}\theta$$

$$\int_{S^1} e^{-i(l+J-j)\theta}\mathrm{d}\theta = \begin{cases} 2\pi & \text{if } l+J-j=0 \\ 0 & \text{otherwise} \end{cases}$$

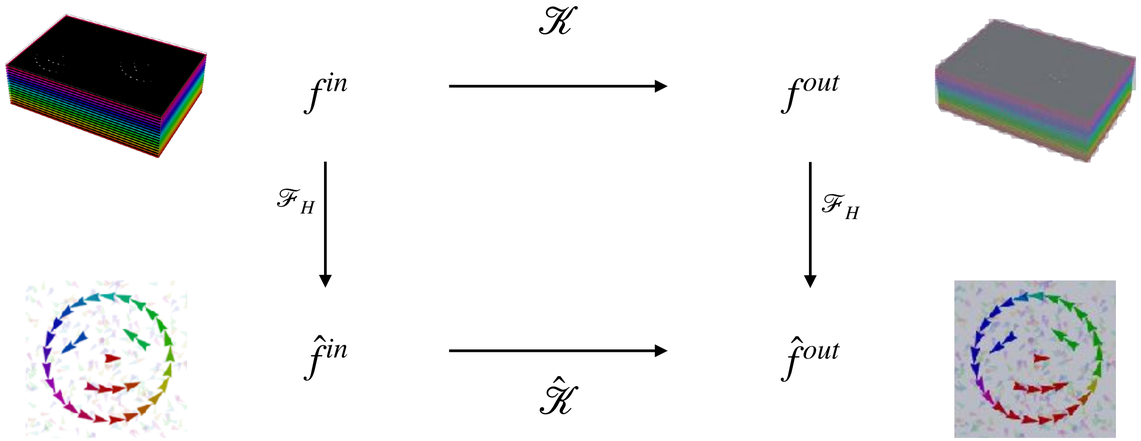
$$= \sum_{J=j-l} \overline{\hat{w}}_J(\|\mathbf{x}'-\mathbf{x}\|)\, Y_J(\alpha_{\mathbf{x}'-\mathbf{x}})$$



Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

$$\mathscr{K}(f)(g) = \int_G k(g^{-1}g')f(g')\mathrm{d}g' \qquad \text{with kernel} \quad k(\mathbf{x},\theta) = \sum_l \sum_{J=j-l} \overline{\hat{w}}_J(\|\mathbf{x}'-\mathbf{x}\|)\, Y_J(\alpha_{\mathbf{x}'-\mathbf{x}})Y_l(\theta)$$

Regular group convolutions



Steerable group convolutions

$$\hat{\mathscr{K}}(\hat{f})(\mathbf{x}) = \int_{\mathbb{R}^d} \hat{k}(\mathbf{x}'-\mathbf{x})\hat{f}(\mathbf{x}')\mathrm{d}\mathbf{x}' \qquad \text{with kernel} \quad \hat{k}_{jl}(\mathbf{x}) = \sum_{J=j-l} \hat{w}_J(\|\mathbf{x}'-\mathbf{x}\|)\, Y_J(\alpha_{\mathbf{x}'-\mathbf{x}})$$

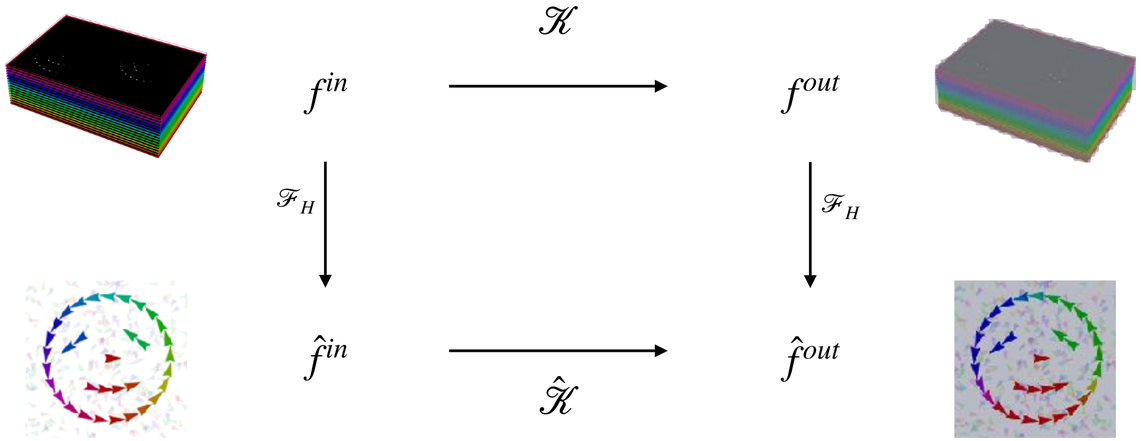
[1] Worrall, D. E., Garbin, S. J., Turmukhambetov, D., & Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. CVPR 2017

Deriving $\hat{\mathscr{K}}$ from the knowns ($\mathscr{K}$, $\mathscr{F}_H$)

$$\mathscr{K}(f)(g) = \int_G k(g^{-1}g')f(g')\mathrm{d}g'$$

with kernel $$k(\mathbf{x}, \theta) = \sum_l \sum_{J=j-l} \overline{\hat{w}}_J(\|\mathbf{x}' - \mathbf{x}\|)\, Y_J(\alpha_{\mathbf{x}'-\mathbf{x}}) Y_l(\theta)$$

Regular group convolutions



$\mathscr{F}_H^{-1}$

Given specified:
- input band-limit $l \leq l_{max}$
- output band-limit $j \leq j_{max}$
- steerable basis $Y_l(\theta) = e^{i\,l\,\theta}$

Steerable group convolutions

$$\hat{\mathscr{K}}(\hat{f})(\mathbf{x}) = \int_{\mathbb{R}^d} \hat{k}(\mathbf{x}' - \mathbf{x})\hat{f}(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

with kernel $$\hat{k}_{jl}(\mathbf{x}) = \sum_{J=j-l} \hat{w}_J(\|\mathbf{x}' - \mathbf{x}\|)\, Y_J(\alpha_{\mathbf{x}'-\mathbf{x}})$$

[1] Worrall, D. E., Garbin, S. J., Turmukhambetov, D., & Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. CVPR 2017

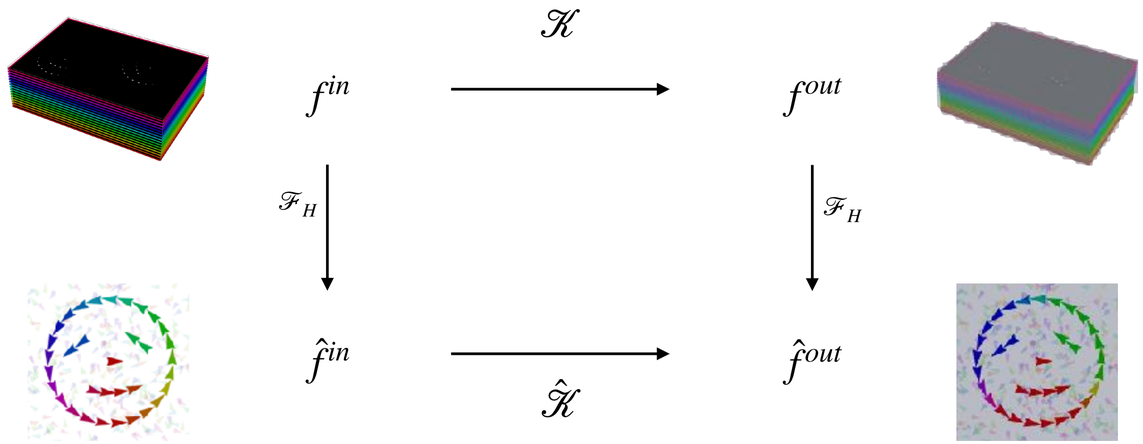
Deriving $\hat{\mathcal{K}}$ from the knowns $(\mathcal{K}, \mathcal{F}_H)$

$$\mathcal{K}(f)(g) = \int_G k(g^{-1}g')f(g')\mathrm{d}g'$$

with kernel $$k(\mathbf{x}, \theta) = \sum_l \sum_{J=j-l} \overline{\hat{w}}_J(\|\mathbf{x}' - \mathbf{x}\|)\, Y_J(\alpha_{\mathbf{x}'-\mathbf{x}}) Y_l(\theta)$$

Regular group convolutions

equivariance condition of harmonic nets!



$\mathcal{F}_H^{-1}$

Given specified:
- input band-limit $l \leq l_{max}$
- output band-limit $j \leq j_{max}$
- steerable basis $Y_l(\theta) = e^{i\,l\,\theta}$

Steerable group convolutions

$$\hat{\mathcal{K}}(\hat{f})(\mathbf{x}) = \int_{\mathbb{R}^d} \hat{k}(\mathbf{x}' - \mathbf{x})\hat{f}(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

with kernel $$\hat{k}_{jl}(\mathbf{x}) = \sum_{J=j-l} \hat{w}_J(\|\mathbf{x}' - \mathbf{x}\|)\, Y_J(\alpha_{\mathbf{x}'-\mathbf{x}})$$

[1] Worrall, D. E., Garbin, S. J., Turmukhambetov, D., & Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. CVPR 2017

Harmonic networks

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.

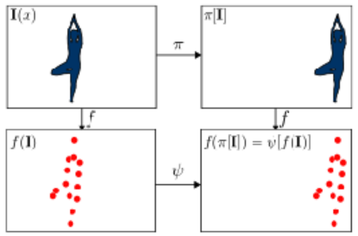


Figure 1. Patch-wise translation equivariance in CNNs arises from translational weight tying, so that a translation π of the input image $\mathbf{I}$, leads to a corresponding translation ψ of the feature maps $f(\mathbf{I})$, where $\pi \neq \psi$ in general, due to pooling effects. However, for rotations, 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/

Steerable G-CNNs as Clebsch-Gordan networks

$$\hat{\mathscr{K}}(\hat{f})(\mathbf{x}) = \int_{\mathbb{R}^d} \sum_{l} \sum_{J=j-l} \hat{w}_J(\|\mathbf{x}' - \mathbf{x}\|)\, Y_J(\alpha_{\mathbf{x}'-\mathbf{x}}) \hat{f}_l(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

Steerable G-CNNs as Clebsch-Gordan networks

$$\hat{\mathscr{K}}(\hat{f})(\mathbf{x}) = \int_{\mathbb{R}^d} \sum_l \sum_J \hat{w}^j_{Jl}(\|\mathbf{x}' - \mathbf{x}\|)\, Y_J(\alpha_{\mathbf{x}'-\mathbf{x}}) \hat{f}_l(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

Steerable G-CNNs as Clebsch-Gordan networks

$$\hat{\mathscr{K}}(\,\hat{f}\,)(\mathbf{x}) = \int_{\mathbb{R}^d} \hat{f} \;\otimes_{CG}^{\hat{w}(\|\mathbf{x}'-\mathbf{x}\|)}\; Y(\alpha_{\mathbf{x}'-\mathbf{x}})\;\mathrm{d}\mathbf{x}'$$