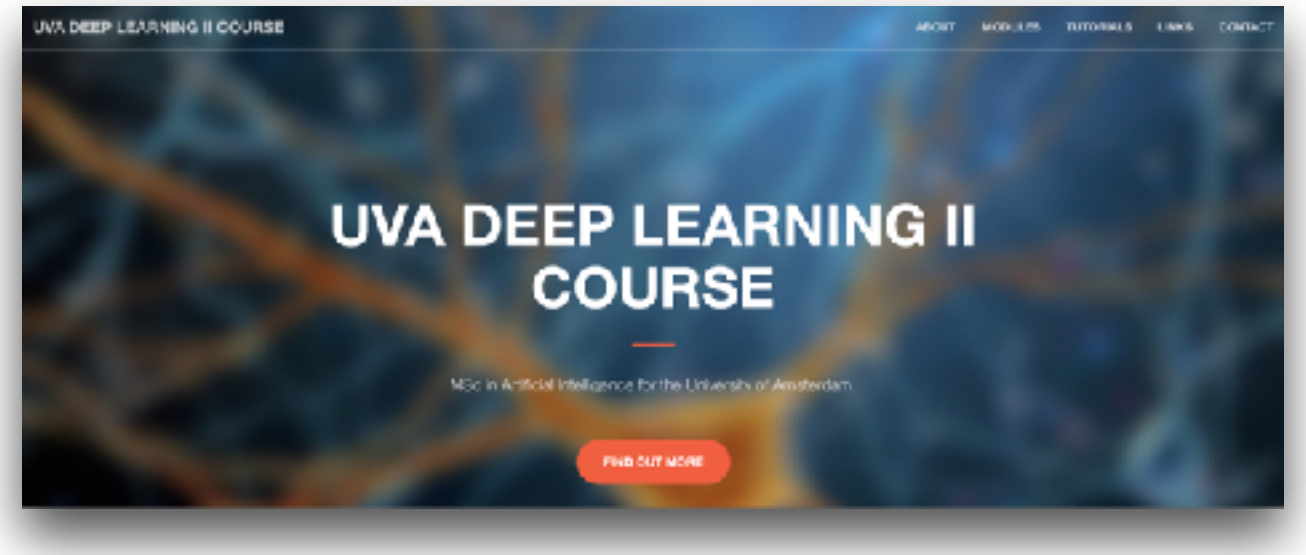
UNIVERSITY OF AMSTERDAM

AMLAB Amsterdam Machine Learning Lab

Introduction to group equivariant deep learning

Erik Bekkers
e.j.bekkers@uva.nl
Amsterdam Machine Learning Lab, University of Amsterdam

UvA course on group equivariant deep learning (https://uvagedl.github.io)

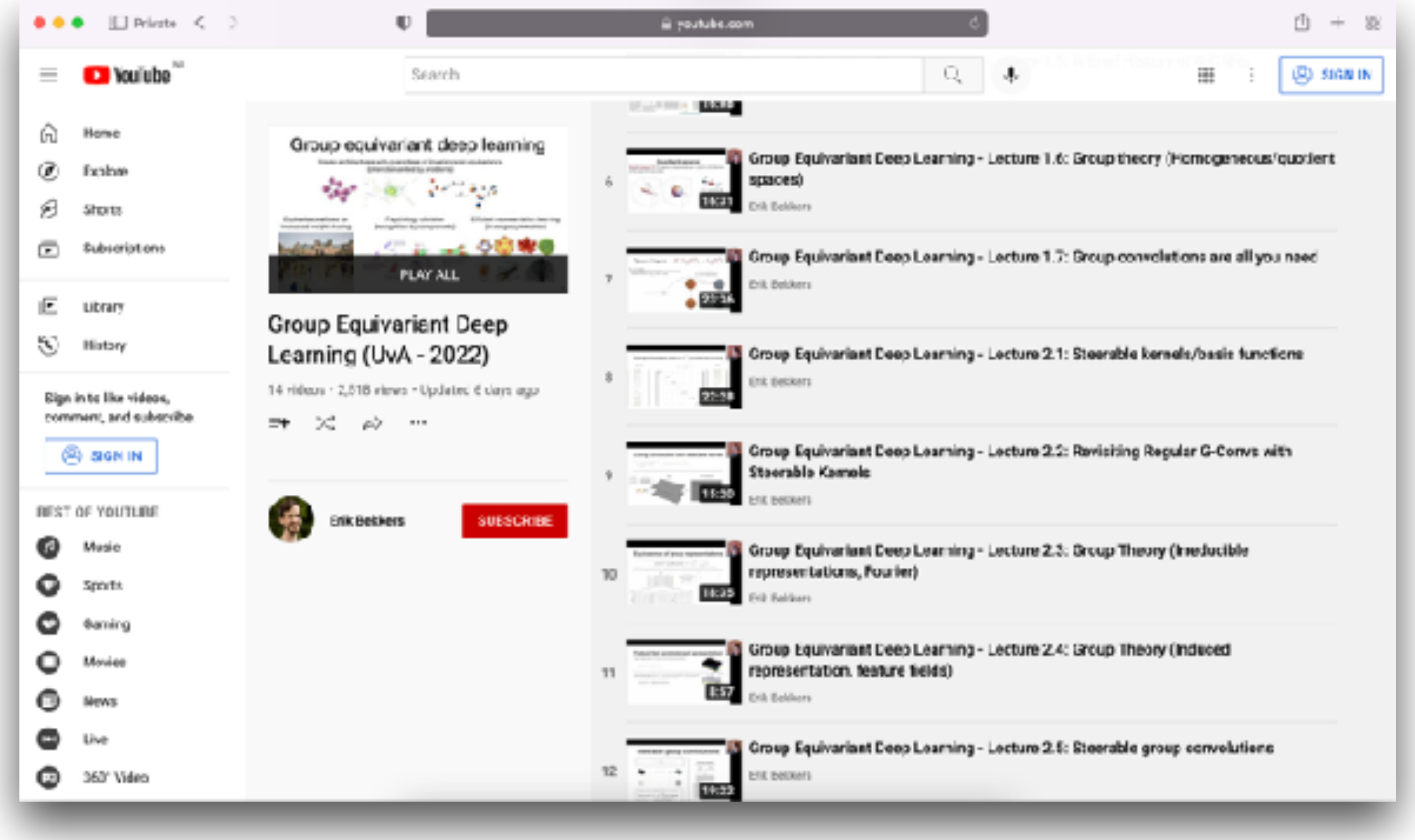


UVA DEEP LEARNING II COURSE

MSc in Artificial Intelligence for the University of Amsterdam

FIND OUT MORE

Youtube playlist



Group equivariant deep learning

PLAY ALL

Group Equivarient Deep Learning (UvA - 2022)

14 videos · 2,518 views · Updated 4 days ago

Erik Bekkers SUBSCRIBE

6 Group Equivariant Deep Learning - Lecture 1.6: Group theory (Homogeneous/quotient spaces) Erik Bekkers

7 Group Equivariant Deep Learning - Lecture 1.7: Group convolutions are all you need Erik Bekkers

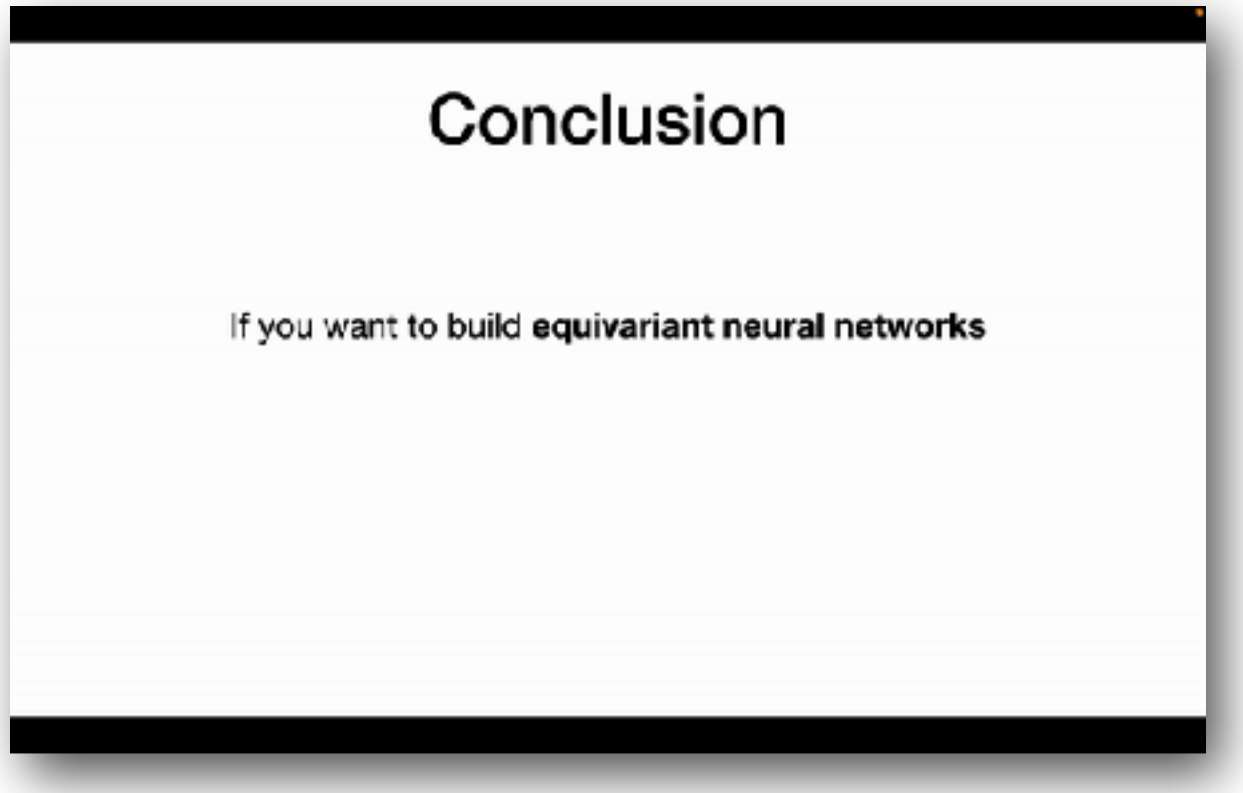
8 Group Equivariant Deep Learning - Lecture 2.1: Steerable kernels/basis functions Erik Bekkers

9 Group Equivariant Deep Learning - Lecture 2.2: Revisiting Regular G-Convs with Steerable Kernels Erik Bekkers

10 Group Equivariant Deep Learning - Lecture 2.3: Group Theory (Irreducible representations, Fourier) Erik Bekkers

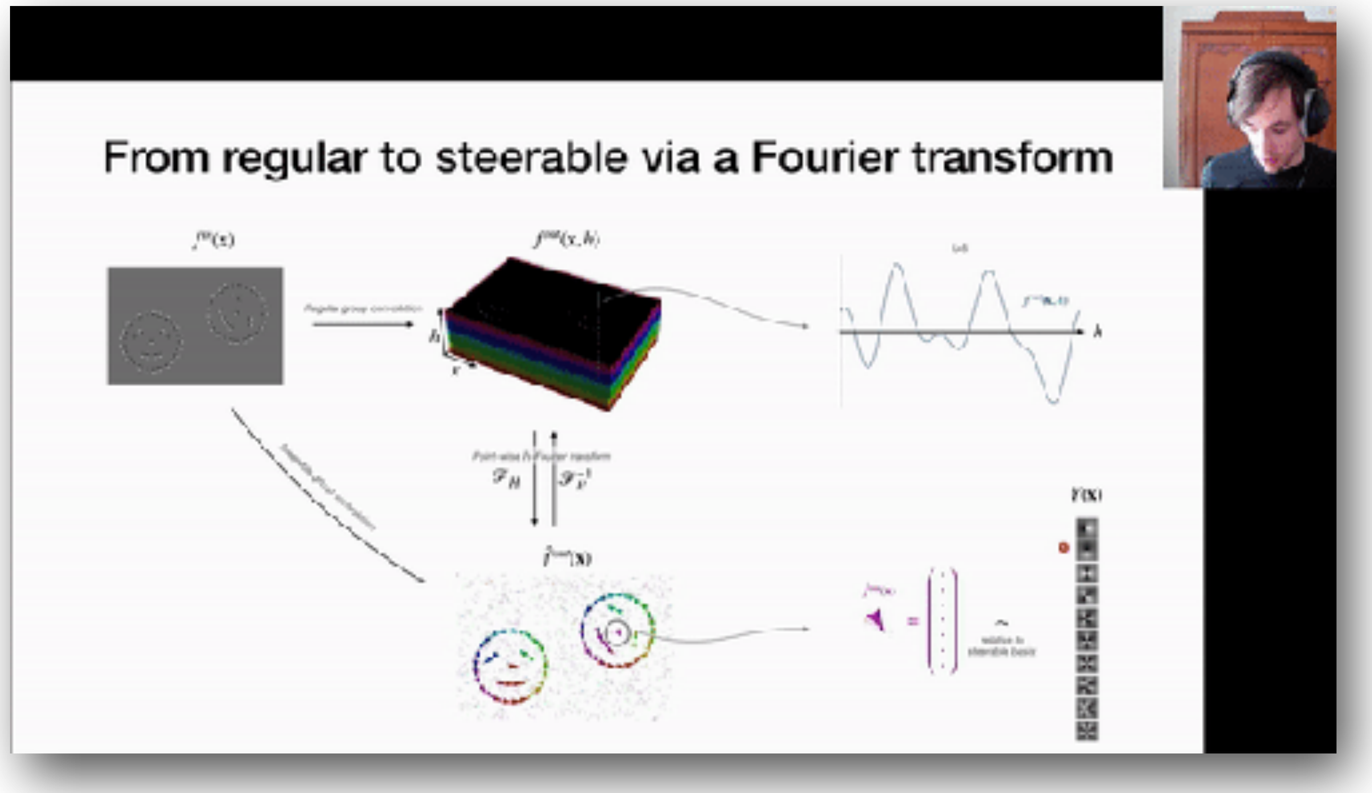
11 Group Equivariant Deep Learning - Lecture 2.4: Group Theory (Induced representation, feature fields) Erik Bekkers

12 Group Equivariant Deep Learning - Lecture 2.5: Steerable group convolutions Erik Bekkers



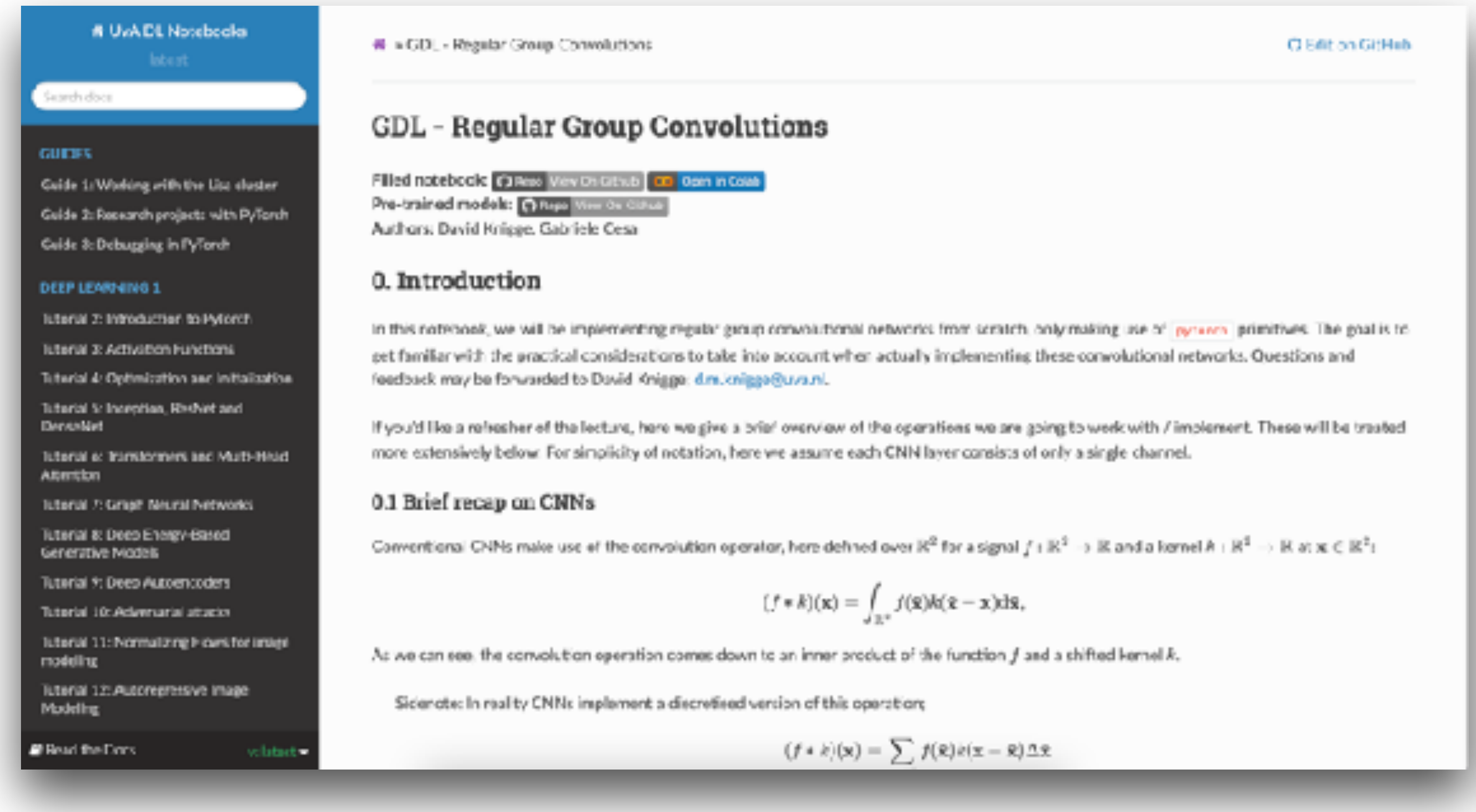
Conclusion

If you want to build **equivariant neural networks**



From regular to steerable via a Fourier transform

Tutorial notebooks



UvA DL Notebooks

GDL - Regular Group Convolutions

Filled notebook: Repo View On Github | Open in Colab
Pre-trained models: Repo View On Github
Authors: David Knigge, Gabriele Cesa

0. Introduction

In this notebook, we will be implementing regular group convolutional networks from scratch, only making use of pytorch primitives. The goal is to get familiar with the practical considerations to take into account when actually implementing these convolutional networks. Questions and feedback may be forwarded to David Knigge: d.m.knigge@uva.nl.

If you'd like a refresher of the lecture, here we give a brief overview of the operations we are going to work with / implement. These will be treated more extensively below. For simplicity of notation, here we assume each CNN layer consists of only a single channel.

0.1 Brief recap on CNNs

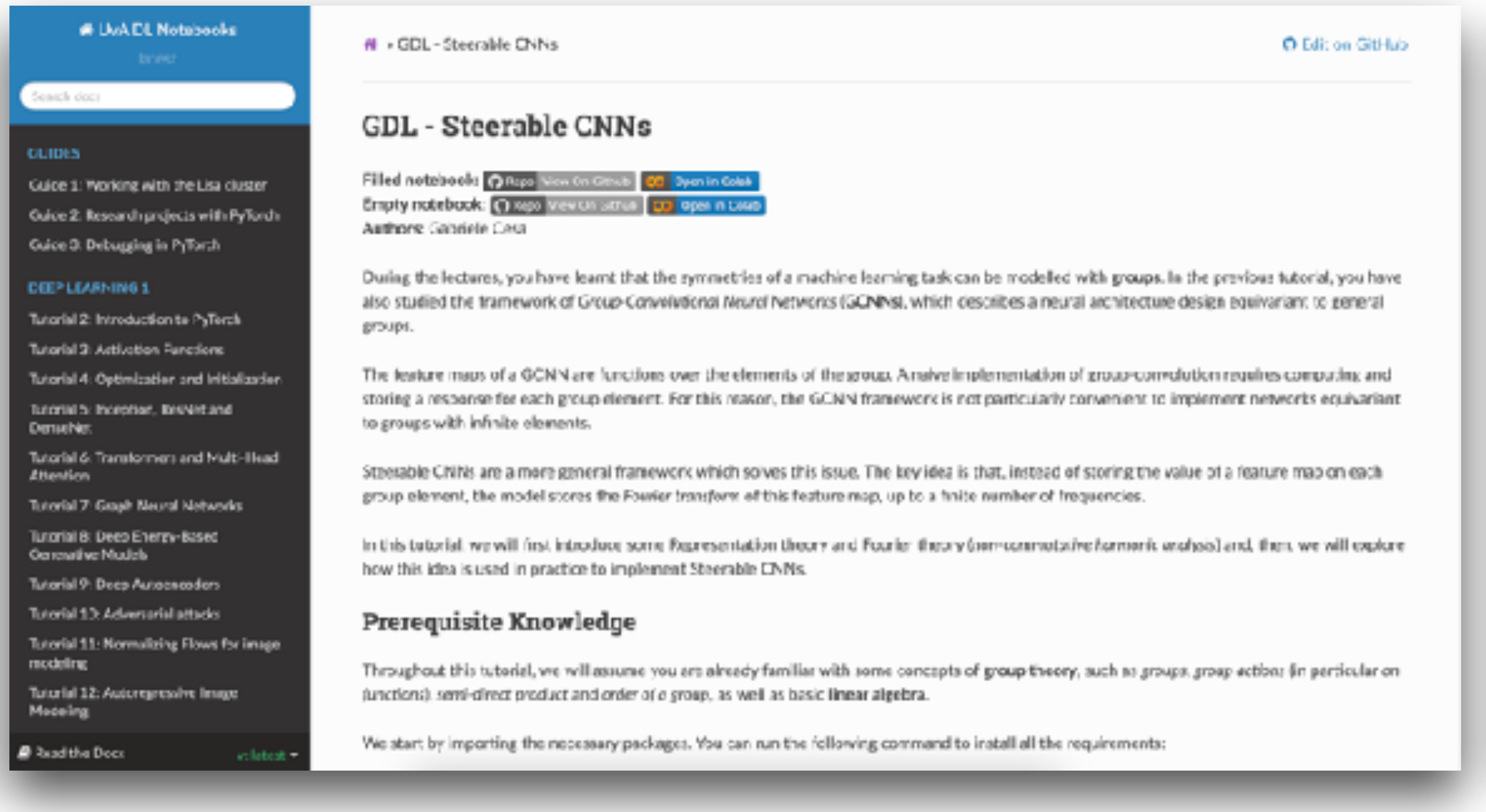
Conventional CNNs make use of the convolution operator, here defined over $\mathbb{R}^2$ for a signal $f : \mathbb{R}^2 \to \mathbb{R}$ and a kernel $k : \mathbb{R}^2 \to \mathbb{R}$ at $\mathbf{x} \in \mathbb{R}^2$:

$$(f * k)(\mathbf{x}) = \int_{\mathbb{R}^2} f(\tilde{\mathbf{x}})k(\tilde{\mathbf{x}} - \mathbf{x})d\tilde{\mathbf{x}},$$

As we can see, the convolution operation comes down to an inner product of the function f and a shifted kernel k.

Sidenote: In reality CNNs implement a discretized version of this operation:

$$(f * k)(\mathbf{x}) = \sum f(\tilde{\mathbf{x}})k(\tilde{\mathbf{x}} - \mathbf{x})\Delta\tilde{\mathbf{x}}$$



UvA DL Notebooks

GDL - Steerable CNNs

Filled notebook: Repo View On Github | Open in Colab
Empty notebook: Repo View On Github | Open in Colab
Authors: Gabriele Cesa

During the lectures, you have learnt that the symmetries of a machine learning task can be modelled with **groups**. In the previous tutorial, you have also studied the framework of *Group-Convolutional Neural Networks* (**GCNNs**), which describes a neural architecture design equivariant to general groups.

The feature maps of a GCNN are functions over the elements of the group. A naive implementation of group-convolution requires computing and storing a response for each group element. For this reason, the GCNN framework is not particularly convenient to implement networks equivariant to groups with infinite elements.

Steerable CNNs are a more general framework which solves this issue. The key idea is that, instead of storing the value of a feature map on each group element, the model stores the Fourier transform of this feature map, up to a finite number of frequencies.

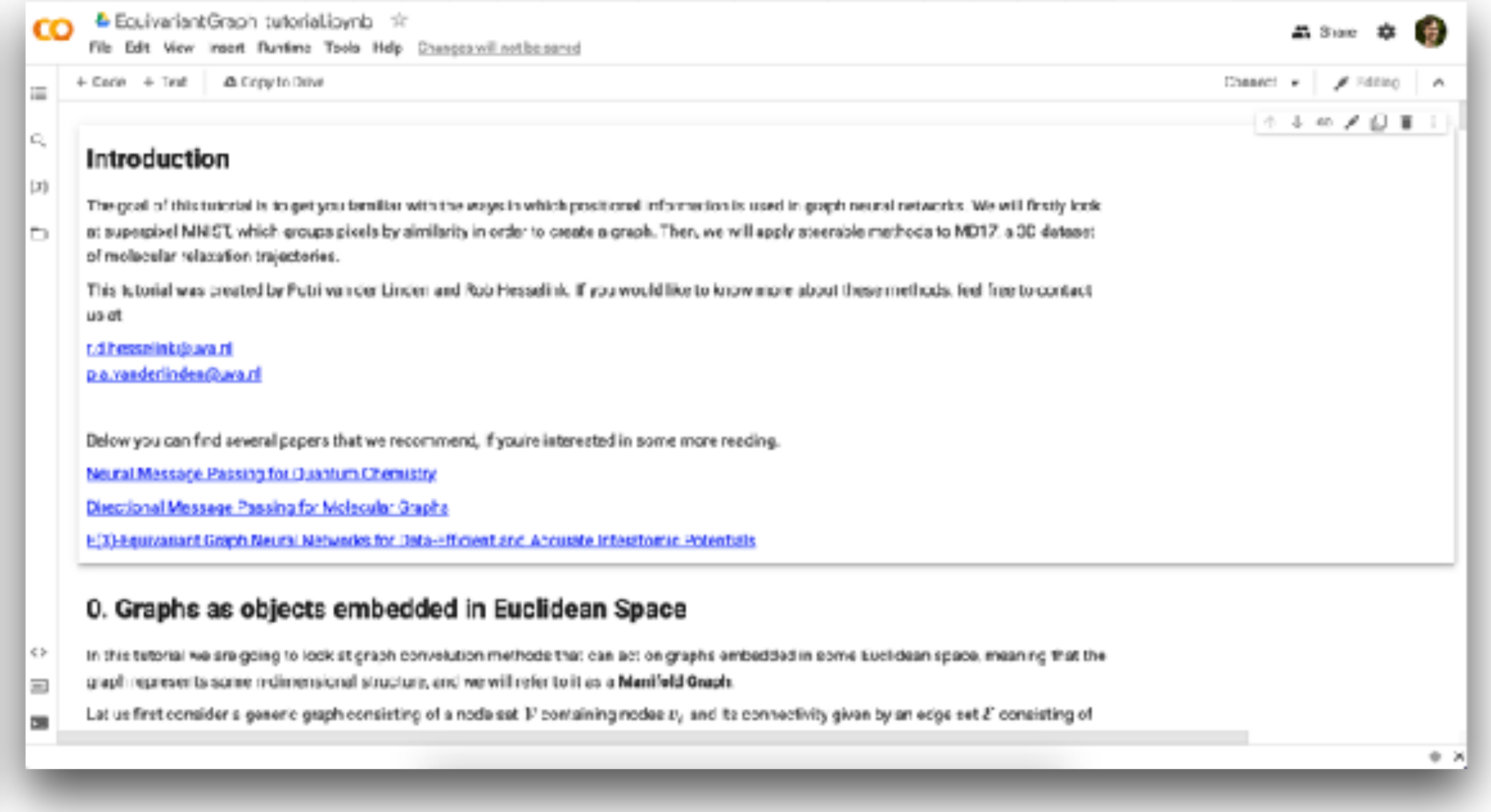
In this tutorial, we will first introduce some Representation theory and Fourier theory (*non-commutative harmonic analysis*) and, then, we will explore how this idea is used in practice to implement Steerable CNNs.

Prerequisite Knowledge

Throughout this tutorial, we will assume you are already familiar with some concepts of **group theory**, such as *groups*, *group actions* (in particular on functions), *semi-direct product* and *order of a group*, as well as basic **linear algebra**.

We start by importing the necessary packages. You can run the following command to install all the requirements:

GUIDES
Guide 1: Working with the Lisa cluster
Guide 2: Research projects with PyTorch
Guide 3: Debugging in PyTorch
DEEP LEARNING 1
Tutorial 2: Introduction to PyTorch
Tutorial 3: Activation Functions
Tutorial 4: Optimization and Initialization
Tutorial 5: Inception, ResNet and DenseNet
Tutorial 6: Transformers and Multi-Head Attention
Tutorial 7: Graph Neural Networks
Tutorial 8: Deep Energy-Based Generative Models
Tutorial 9: Deep Autoencoders
Tutorial 10: Adversarial attacks
Tutorial 11: Normalizing Flows for image modeling
Tutorial 12: Autoregressive Image Modeling
Read the Docs
Edit on GitHub



EquivariantGraph tutorial.ipynb

Introduction

The goal of this tutorial is to get you familiar with the ways in which positional information is used in graph neural networks. We will firstly look at superpixel MNIST, which groups pixels by similarity in order to create a graph. Then, we will apply steerable methods to MD17, a 3D dataset of molecular relaxation trajectories.

This tutorial was created by Putri van der Linden and Rob Hesselink. If you would like to know more about these methods, feel free to contact us at

r.d.hesselink@uva.nl
p.a.vanderlinden@uva.nl

Below you can find several papers that we recommend, if youre interested in some more reading.

Neural Message Passing for Quantum Chemistry

Directional Message Passing for Molecular Graphs

E(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials

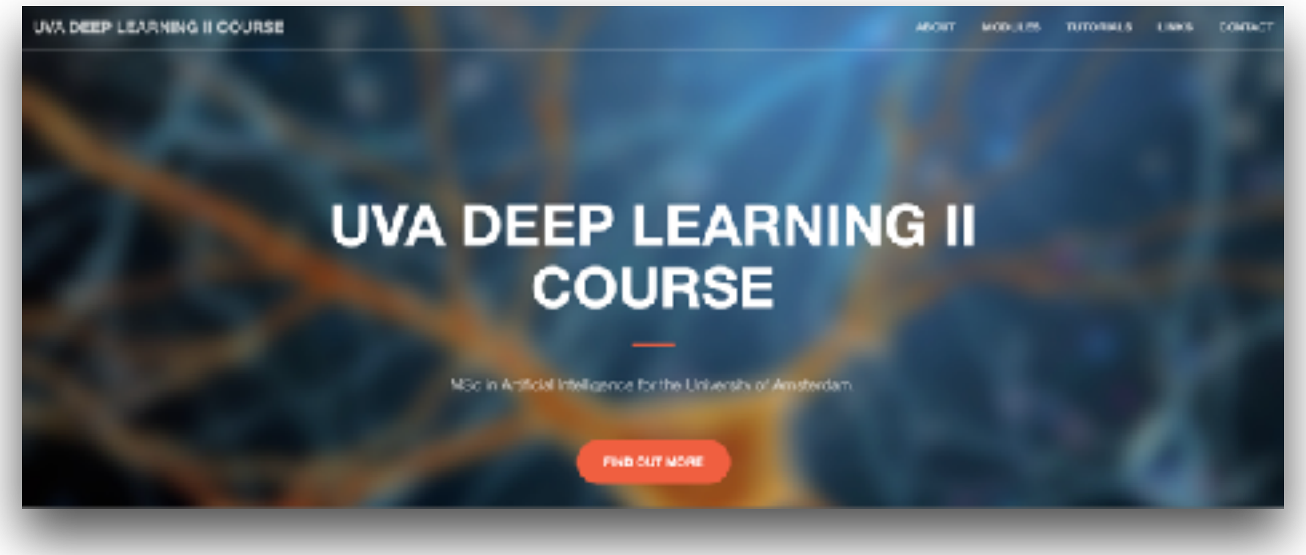
0. Graphs as objects embedded in Euclidean Space

In this tutorial we are going to look at graph convolution methods that can act on graphs embedded in some Euclidean space, meaning that the graph represents some n-dimensional structure, and we will refer to it as a **Manifold Graph**.

Let us first consider a generic graph consisting of a node set V containing nodes v_i and its connectivity given by an edge set E consisting of

File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text
Copy to Drive
Connect
Editing

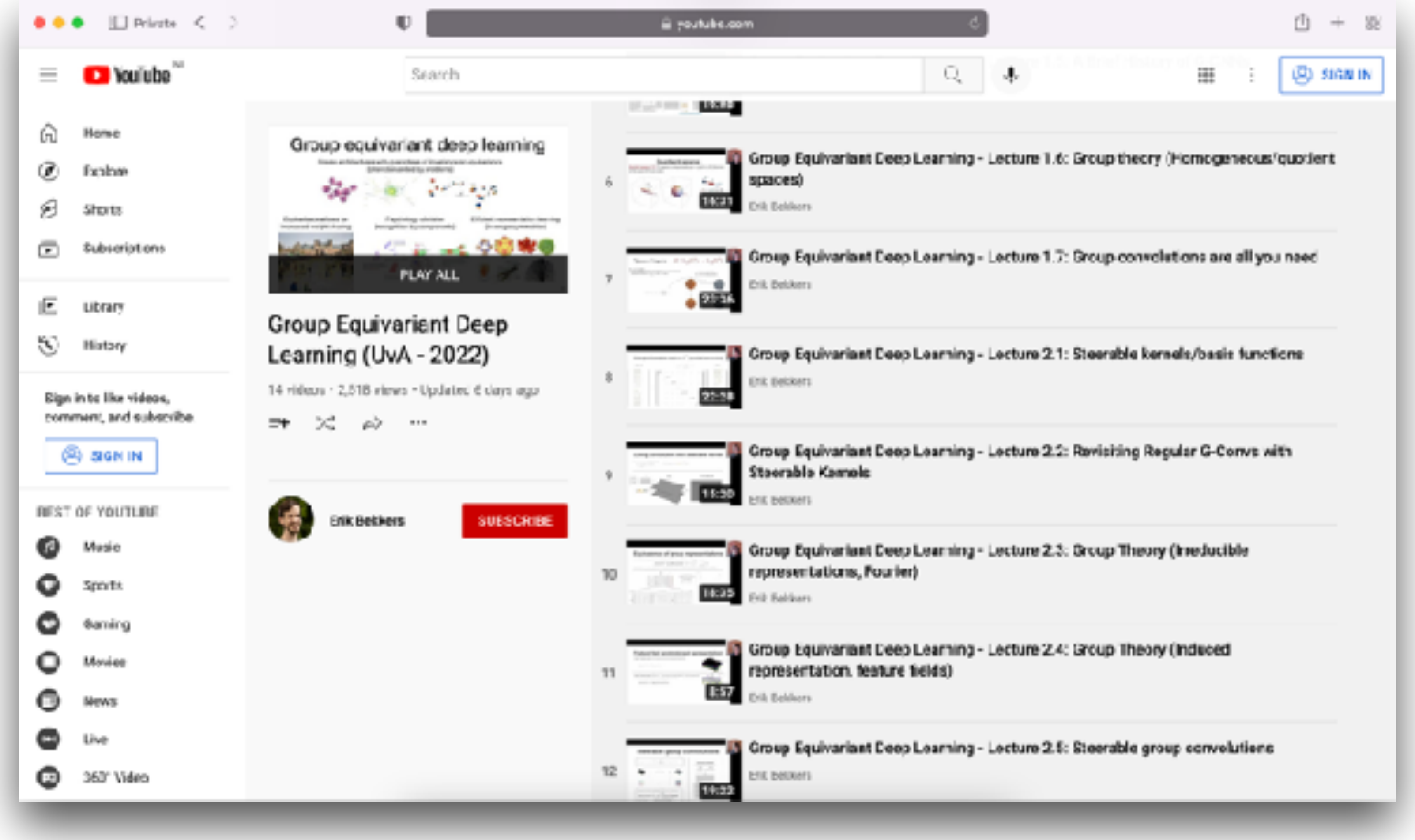
UvA course on group equivariant deep learning (https://uvagedl.github.io)



UVA DEEP LEARNING II COURSE

MSc in Artificial Intelligence for the University of Amsterdam

Youtube playlist



Group equivariant deep learning

Group Equivarient Deep Learning (UvA - 2022)

14 videos · 2,518 views · Updated 4 days ago

Sign in to like videos, comment, and subscribe

Erik Bekkers

6 Group Equivariant Deep Learning - Lecture 1.6: Group theory (Homogeneous/quotient spaces) Erik Bekkers 14:31

7 Group Equivariant Deep Learning - Lecture 1.7: Group convolutions are all you need Erik Bekkers

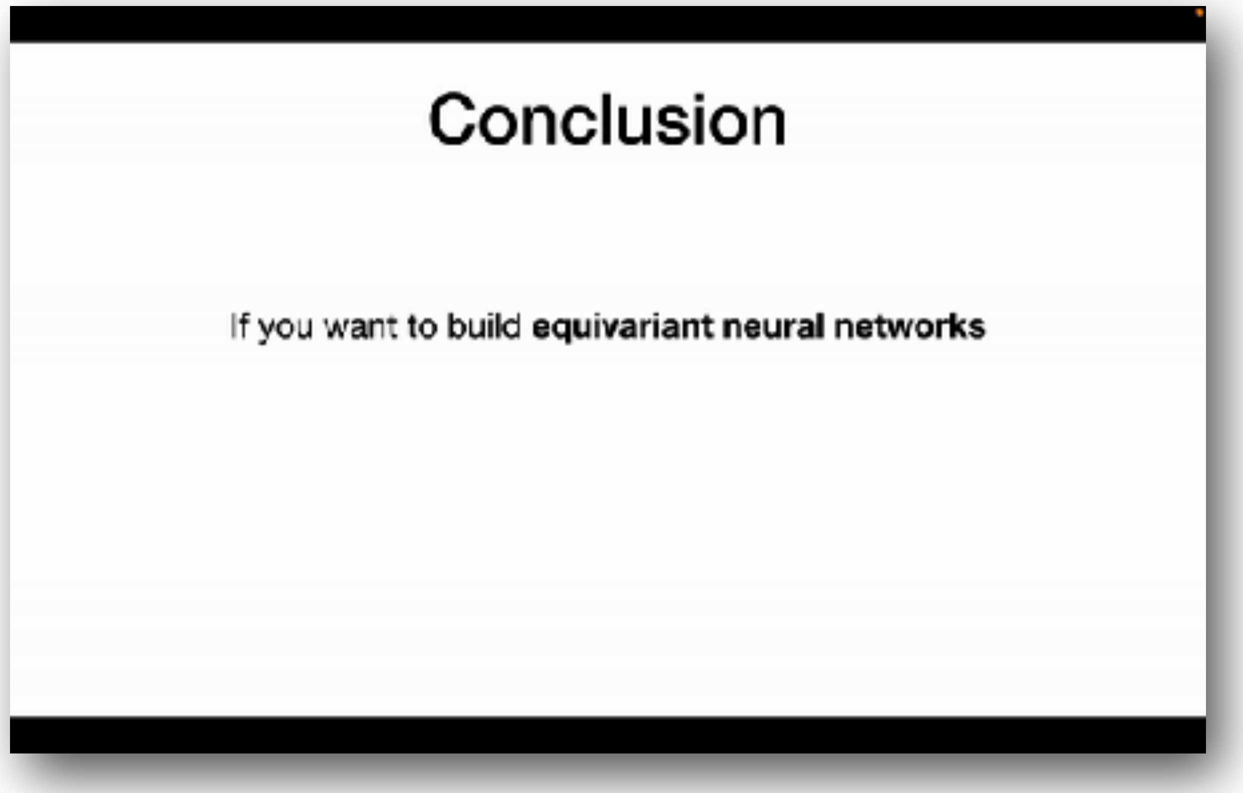
8 Group Equivariant Deep Learning - Lecture 2.1: Steerable kernels/basis functions Erik Bekkers 22:19

9 Group Equivariant Deep Learning - Lecture 2.2: Revisiting Regular G-Convs with Steerable Kernels Erik Bekkers 11:30

10 Group Equivariant Deep Learning - Lecture 2.3: Group Theory (Irreducible representations, Fourier) Erik Bekkers 18:25

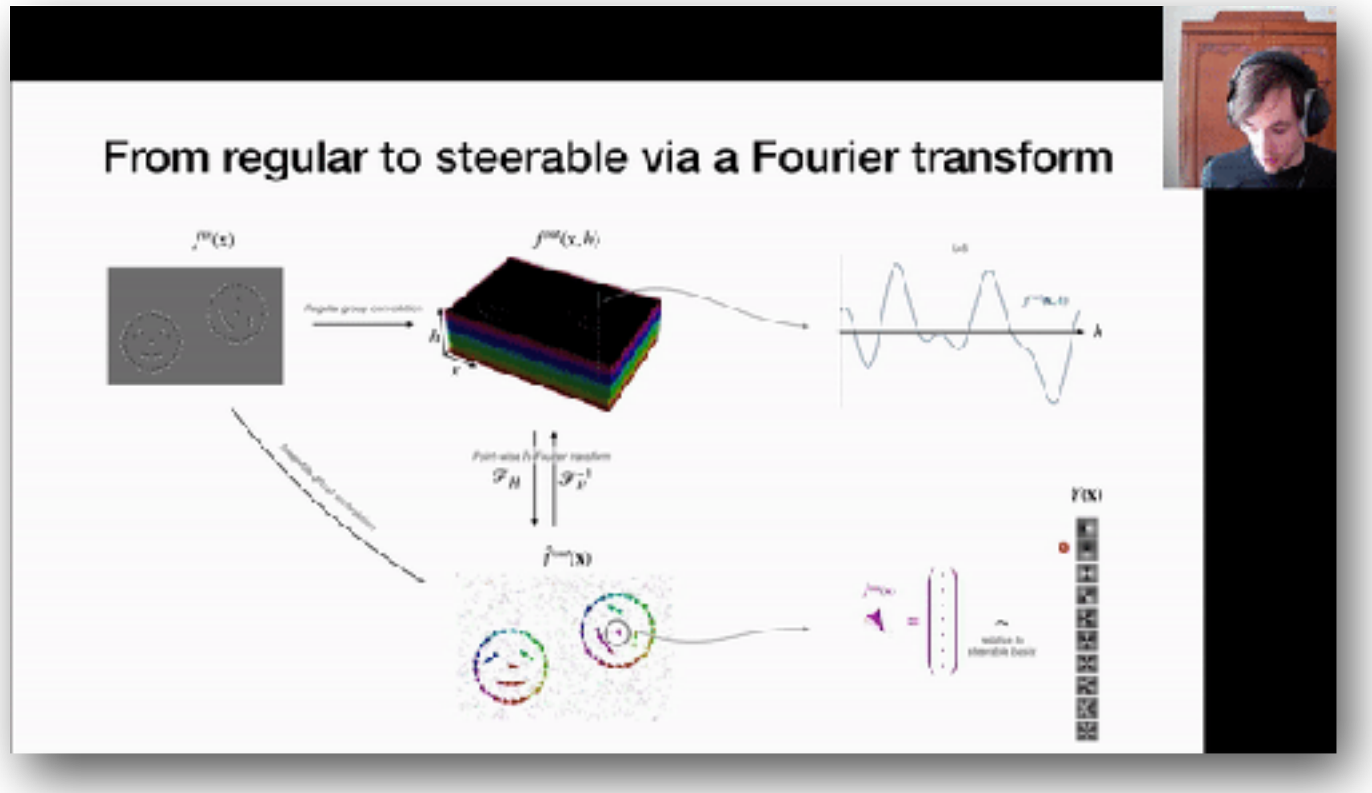
11 Group Equivariant Deep Learning - Lecture 2.4: Group Theory (Induced representation, feature fields) Erik Bekkers 8:57

12 Group Equivariant Deep Learning - Lecture 2.5: Steerable group convolutions Erik Bekkers 14:33



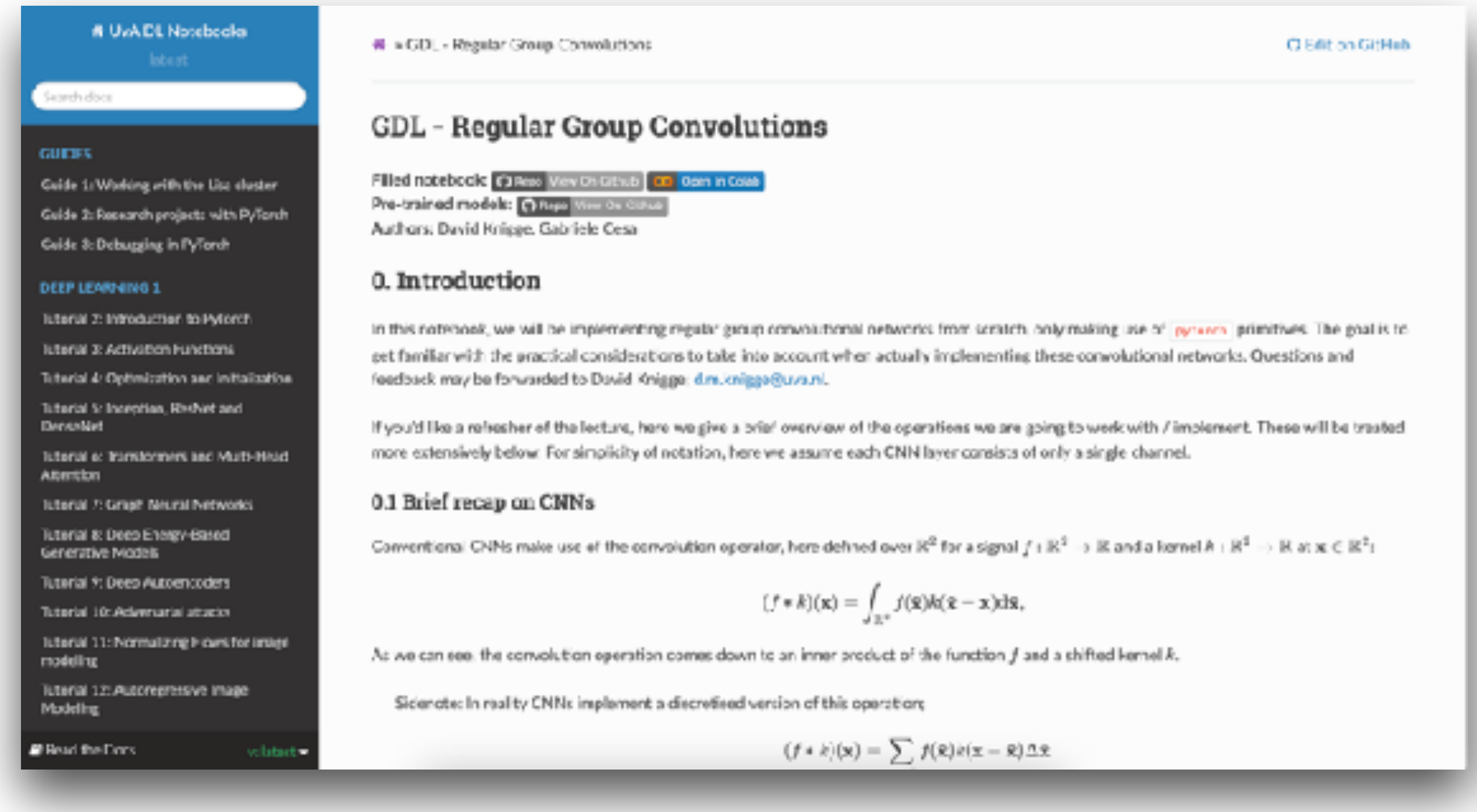
Conclusion

If you want to build **equivariant neural networks**



From regular to steerable via a Fourier transform

Tutorial notebooks


UvA DL Notebooks
latest
Search docs
GUIDES
Guide 1: Working with the Lisa cluster
Guide 2: Research projects with PyTorch
Guide 3: Debugging in PyTorch
DEEP LEARNING 1
Tutorial 2: Introduction to PyTorch
Tutorial 3: Activation Functions
Tutorial 4: Optimization and Initialization
Tutorial 5: Inception, ResNet and DenseNet
Tutorial 6: Transformers and Multi-Head Attention
Tutorial 7: Graph Neural Networks
Tutorial 8: Deep Energy-Based Generative Models
Tutorial 9: Deep Autoencoders
Tutorial 10: Adversarial attacks
Tutorial 11: Normalizing Flows for image modeling
Tutorial 12: Autoregressive Image Modeling
Read the Docs
v: latest
» GDL - Regular Group Convolutions
Edit on GitHub
Repo
View On Github
Open in Colab

GDL - Regular Group Convolutions

Filled notebook: Repo View On Github Open in Colab
Pre-trained models: Repo View On Github
Authors: David Knigge, Gabriele Cesa

0. Introduction

In this notebook, we will be implementing regular group convolutional networks from scratch, only making use of pytorch primitives. The goal is to get familiar with the practical considerations to take into account when actually implementing these convolutional networks. Questions and feedback may be forwarded to David Knigge: d.m.knigge@uva.nl.

If you'd like a refresher of the lecture, here we give a brief overview of the operations we are going to work with / implement. These will be treated more extensively below. For simplicity of notation, here we assume each CNN layer consists of only a single channel.

0.1 Brief recap on CNNs

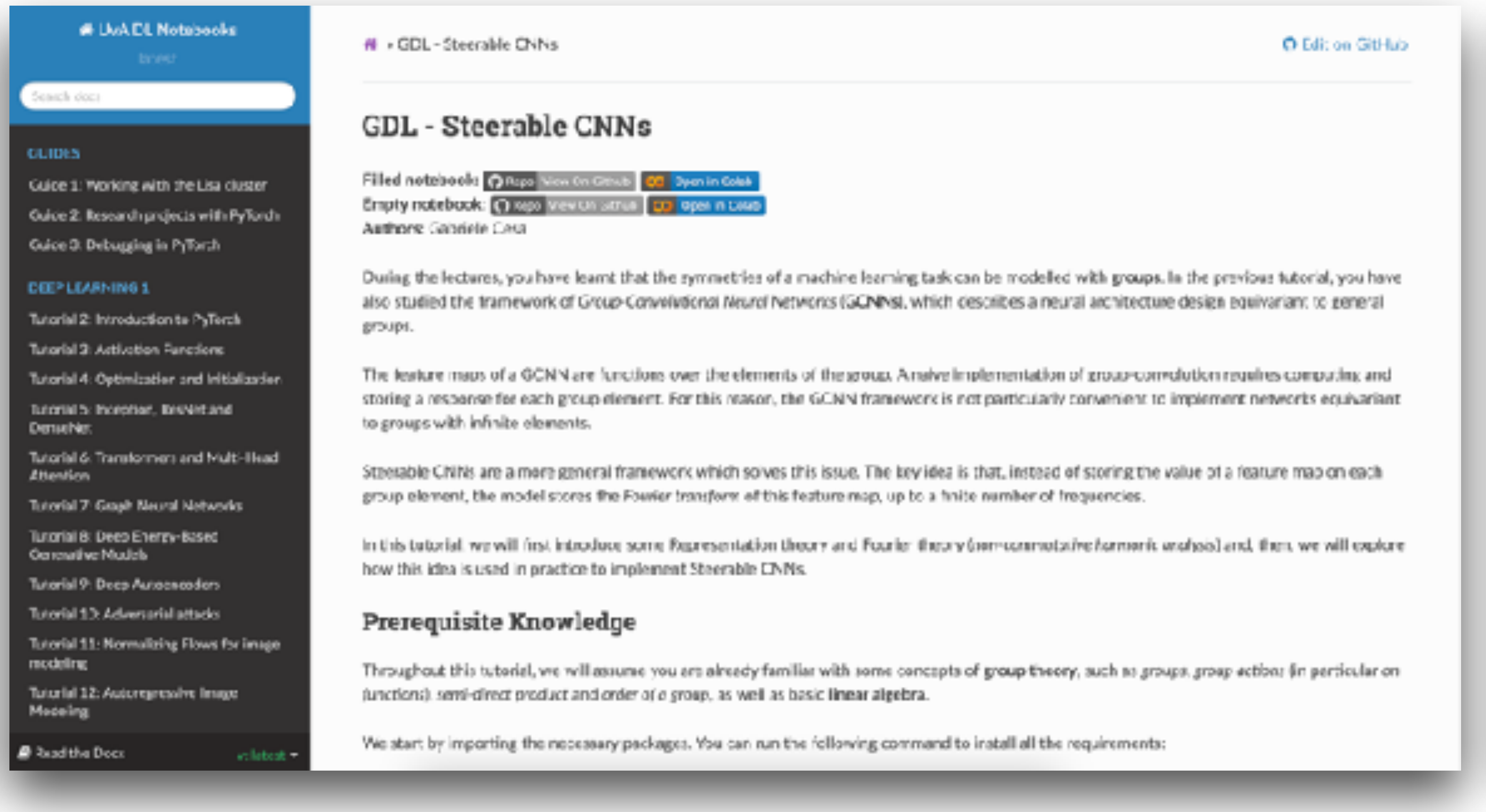
Conventional CNNs make use of the convolution operator, here defined over $\mathbb{R}^2$ for a signal $f:\mathbb{R}^2\rightarrow\mathbb{R}$ and a kernel $k:\mathbb{R}^2\rightarrow\mathbb{R}$ at $\mathbf{x}\in\mathbb{R}^2$:

$$(f*k)(\mathbf{x})=\int_{\mathbb{R}^2} f(\tilde{\mathbf{x}})k(\tilde{\mathbf{x}}-\mathbf{x})d\tilde{\mathbf{x}},$$

As we can see, the convolution operation comes down to an inner product of the function f and a shifted kernel k.

Sidenote: In reality CNNs implement a discretized version of this operation;

$$(f*k)(\mathbf{x})=\sum f(\tilde{\mathbf{x}})k(\tilde{\mathbf{x}}-\mathbf{x})\Delta\tilde{\mathbf{x}}$$


UvA DL Notebooks
latest
Search docs
GUIDES
Guide 1: Working with the Lisa cluster
Guide 2: Research projects with PyTorch
Guide 3: Debugging in PyTorch
DEEP LEARNING 1
Tutorial 2: Introduction to PyTorch
Tutorial 3: Activation Functions
Tutorial 4: Optimization and Initialization
Tutorial 5: Inception, ResNet and DenseNet
Tutorial 6: Transformers and Multi-Head Attention
Tutorial 7: Graph Neural Networks
Tutorial 8: Deep Energy-Based Generative Models
Tutorial 9: Deep Autoencoders
Tutorial 10: Adversarial attacks
Tutorial 11: Normalizing Flows for image modeling
Tutorial 12: Autoregressive Image Modeling
Read the Docs
v: latest
» GDL - Steerable CNNs
Edit on GitHub
Repo
View On Github
Open in Colab

GDL - Steerable CNNs

Filled notebook: Repo View On Github Open in Colab
Empty notebook: Repo View On Github Open in Colab
Authors: Gabriele Cesa

During the lectures, you have learnt that the symmetries of a machine learning task can be modelled with **groups**. In the previous tutorial, you have also studied the framework of *Group-Convolutional Neural Networks* (**GCNNs**), which describes a neural architecture design equivariant to general groups.

The feature maps of a GCNN are functions over the elements of the group. A naive implementation of group-convolution requires computing and storing a response for each group element. For this reason, the GCNN framework is not particularly convenient to implement networks equivariant to groups with infinite elements.

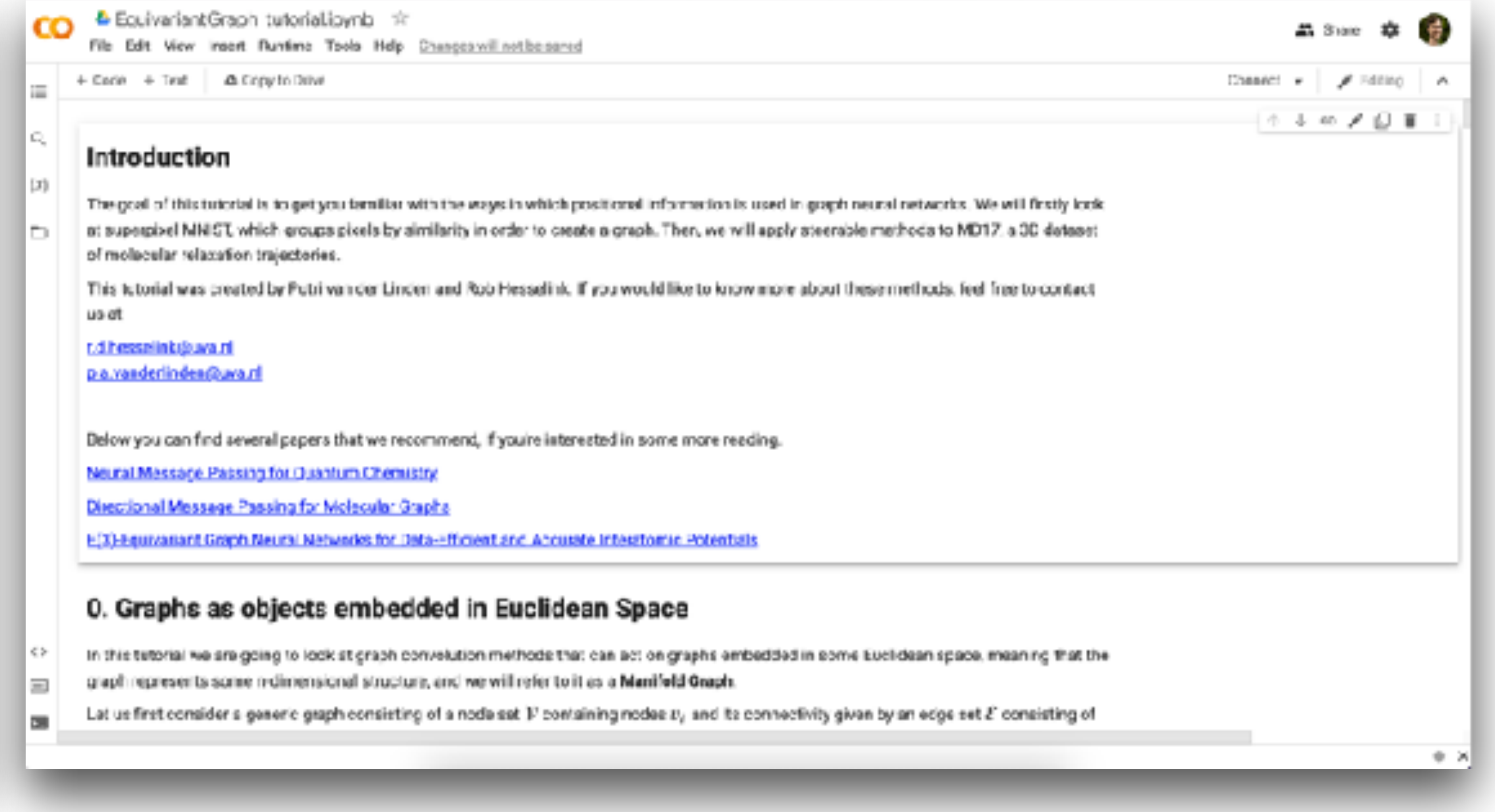
Steerable CNNs are a more general framework which solves this issue. The key idea is that, instead of storing the value of a feature map on each group element, the model stores the Fourier transform of this feature map, up to a finite number of frequencies.

In this tutorial, we will first introduce some Representation theory and Fourier theory (*non-commutative harmonic analysis*) and, then, we will explore how this idea is used in practice to implement Steerable CNNs.

Prerequisite Knowledge

Throughout this tutorial, we will assume you are already familiar with some concepts of **group theory**, such as *groups*, *group actions* (in particular on functions), *semi-direct product* and *order of a group*, as well as basic **linear algebra**.

We start by importing the necessary packages. You can run the following command to install all the requirements:



Introduction

The goal of this tutorial is to get you familiar with the ways in which positional information is used in graph neural networks. We will firstly look at superpixel MNIST, which groups pixels by similarity in order to create a graph. Then, we will apply steerable methods to MD17, a 3D dataset of molecular relaxation trajectories.

This tutorial was created by Putri van der Linden and Rob Hesselink. If you would like to know more about these methods, feel free to contact us at

r.d.hesselink@uva.nl
p.a.vanderlinden@uva.nl

Below you can find several papers that we recommend, if you're interested in some more reading.

Neural Message Passing for Quantum Chemistry

Directional Message Passing for Molecular Graphs

E(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials

0. Graphs as objects embedded in Euclidean Space

In this tutorial we are going to look at graph convolution methods that can act on graphs embedded in some Euclidean space, meaning that the graph represents some n-dimensional structure, and we will refer to it as a **Manifold Graph**.

Let us first consider a generic graph consisting of a node set V containing nodes v_i and its connectivity given by an edge set E consisting of

1. Motivation

2. Pattern matching using group theory

3. Group convolutions

- **4.** Example

- **5.** G-convs are all you need!

6. Steerable group convolutions

- **7.** Feature fields and escnn library

8. Equivariant tensor product layers

9. Equivariant graph NNs

1. Motivation

2. Pattern matching using group theory

3. Group convolutions

  - **4.** Example

  - **5.** G-convs are all you need!

6. Steerable group convolutions

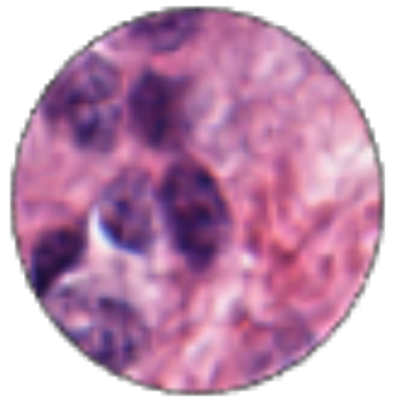
  - **7.** Feature fields and escnn library

8. Equivariant tensor product layers

9. Equivariant graph NNs

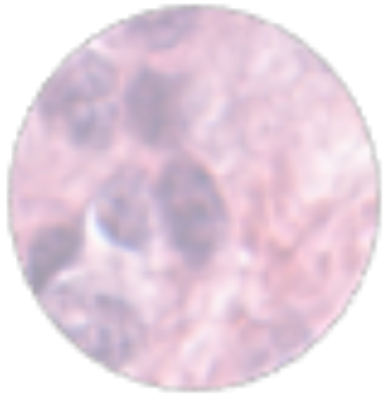
Geometric guarantees (invariance)

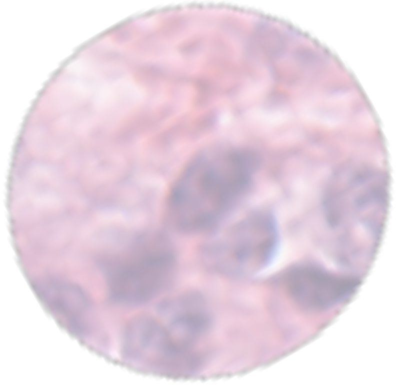
Example: Detection of pathological cells



Geometric guarantees (invariance)

Example: Detection of pathological cells





Healthy

?

Geometric guarantees (invariance)

Example: Detection of pathological cells

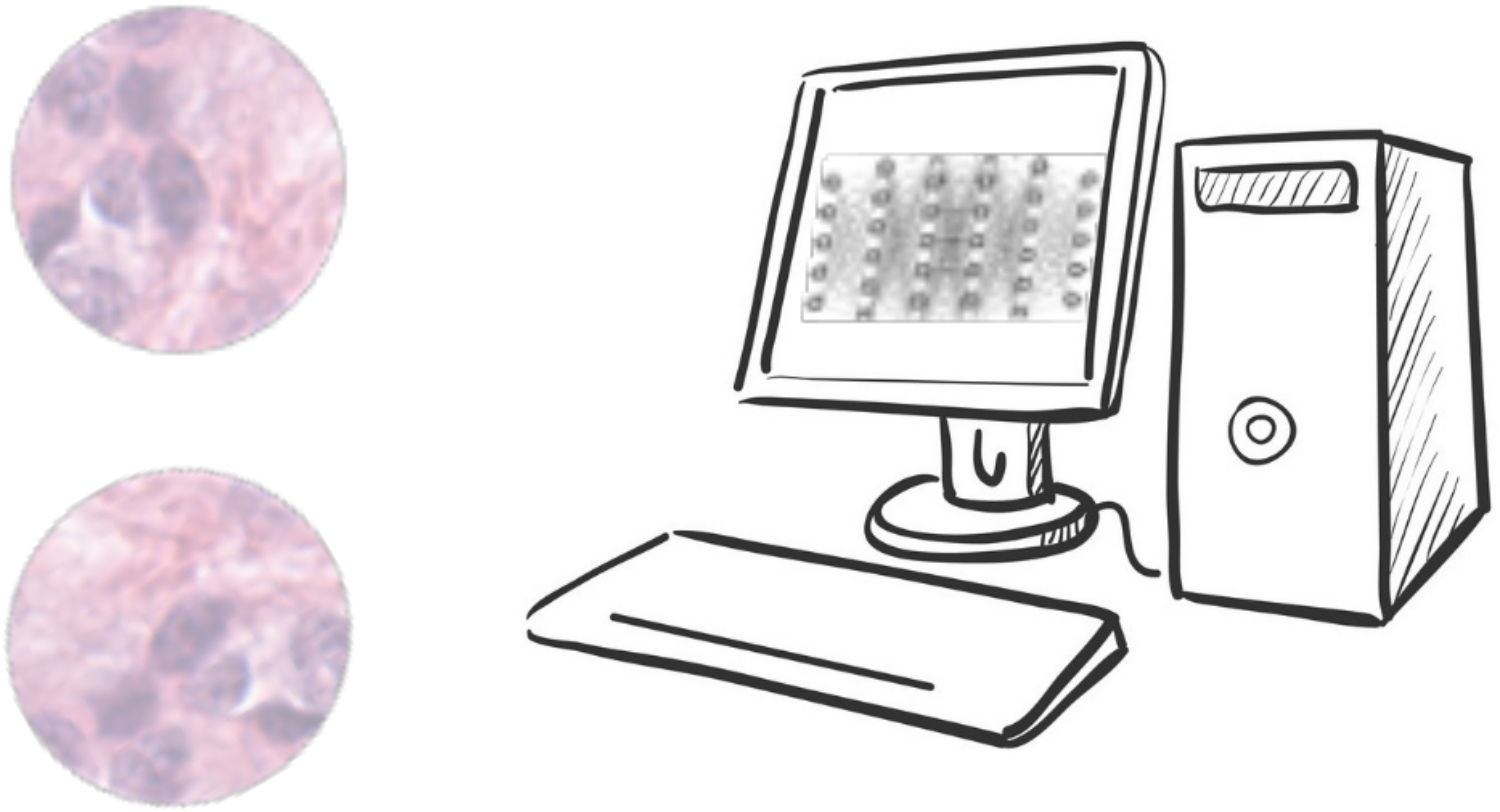
Healthy

?

Pathological

Geometric guarantees (invariance)

Example: Detection of pathological cells

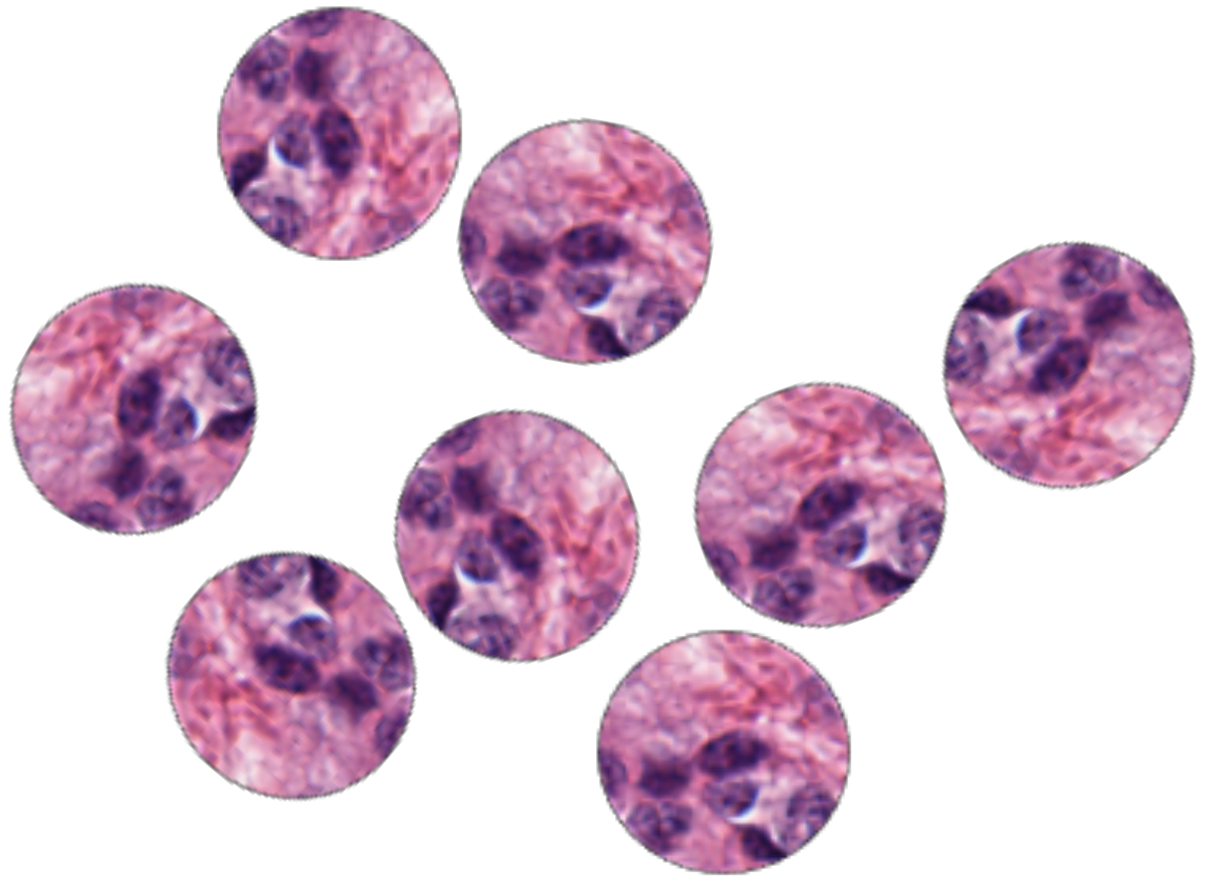


Healthy

?

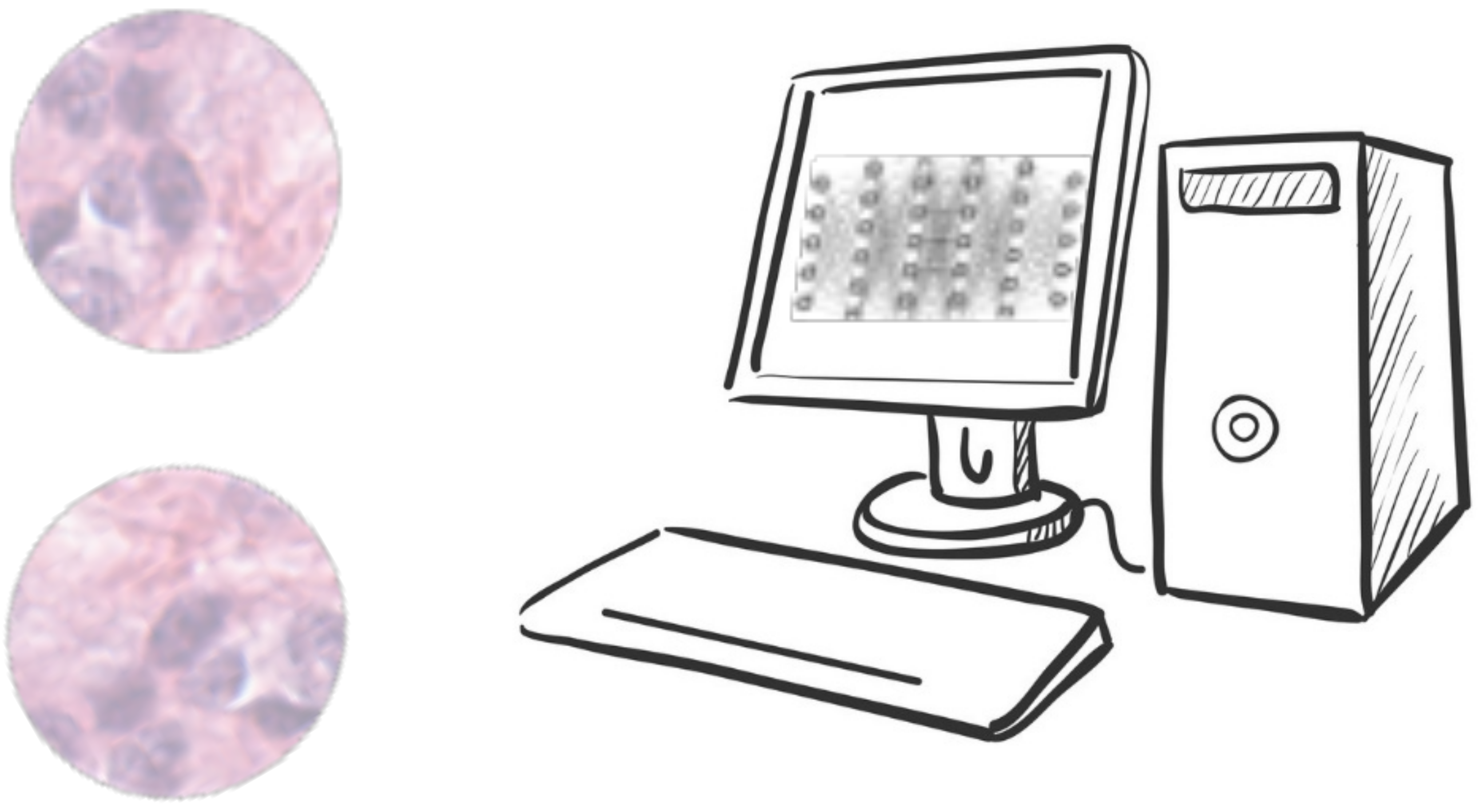
Pathological

Common approach: data-augmentation



Geometric guarantees (invariance)

Example: Detection of pathological cells

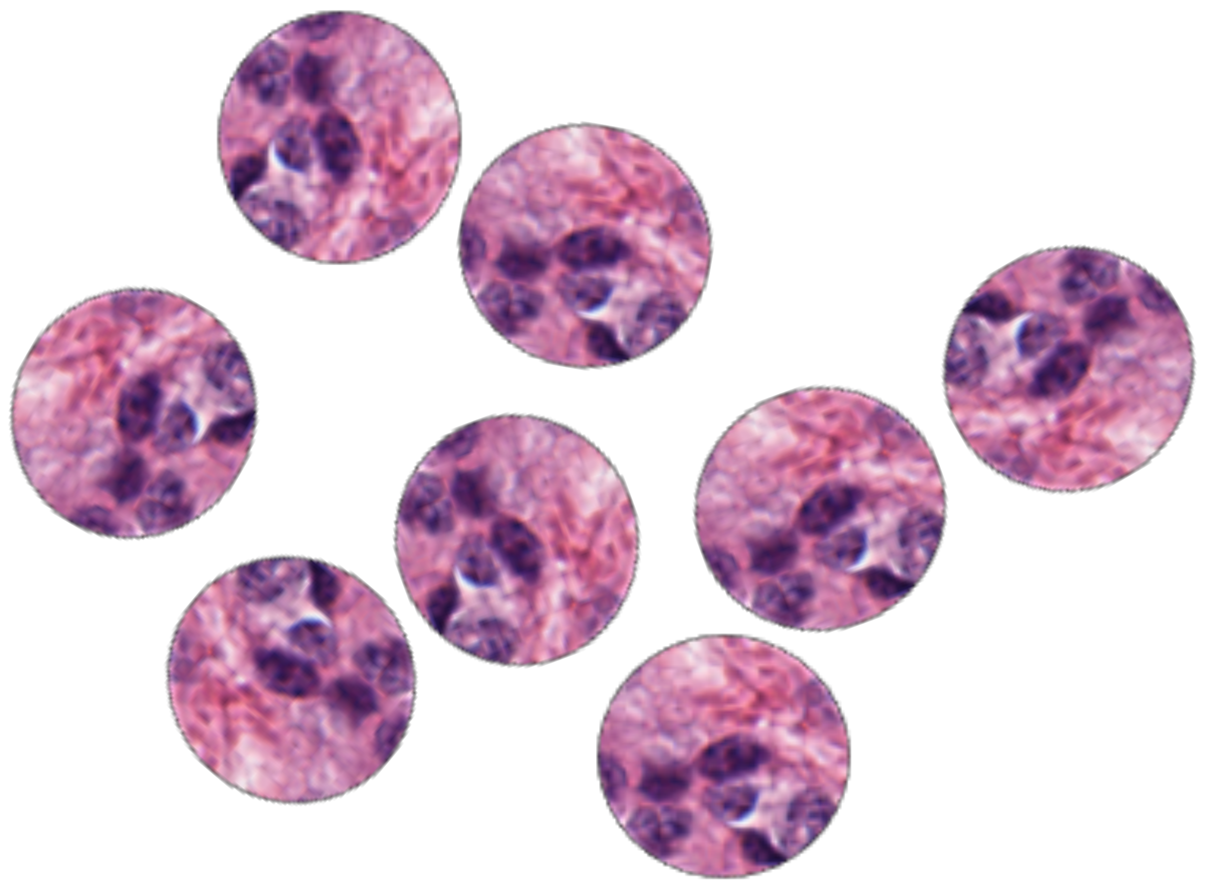


Healthy

?

Pathological

Common approach: data-augmentation

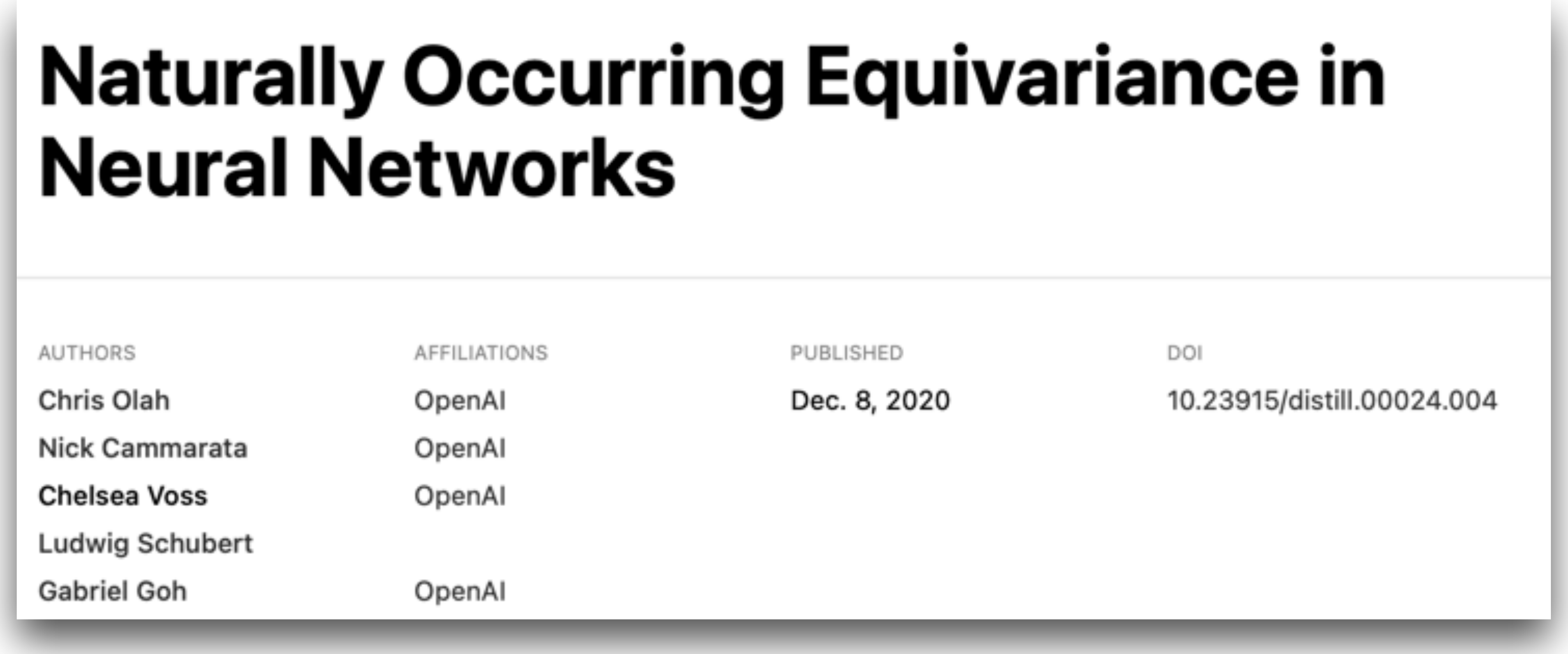


Issues:

- Still no guarantee of invariance
- Valuable net capacity is spend on learning invariance
- Redundancy in feature repr.

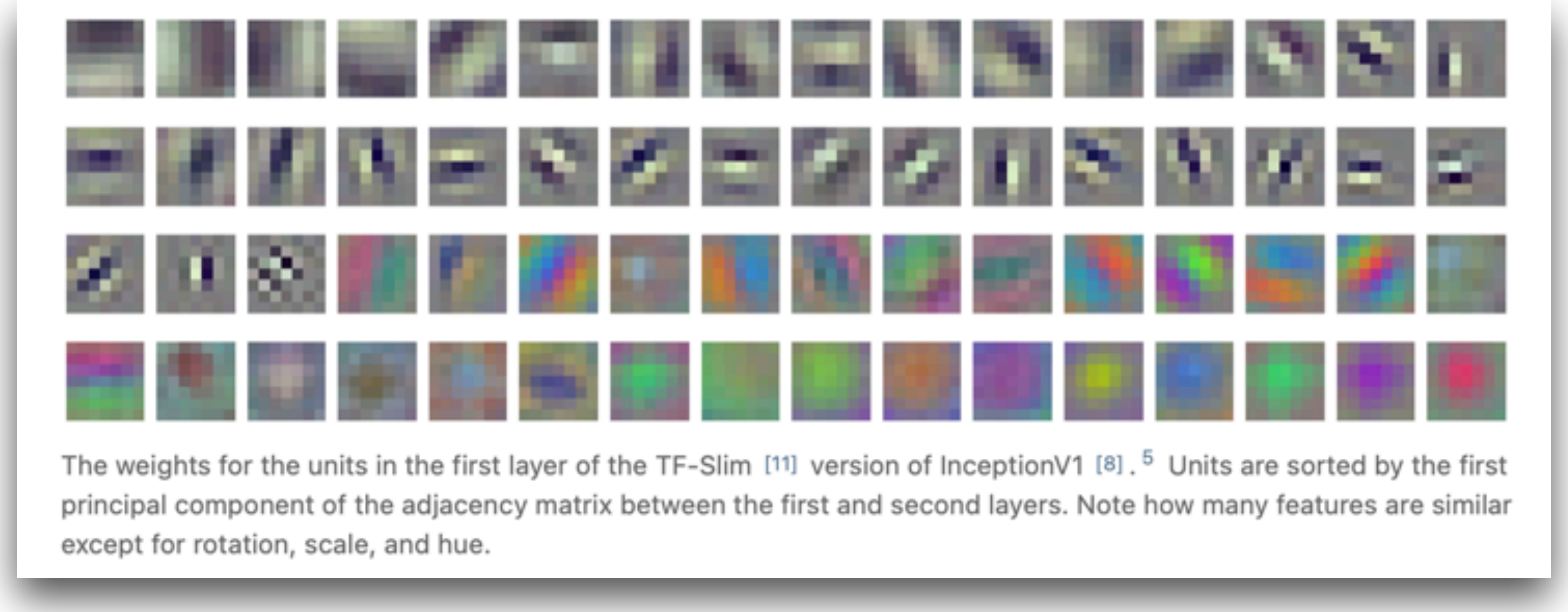
☹

https://distill.pub/2020/circuits/equivariance/

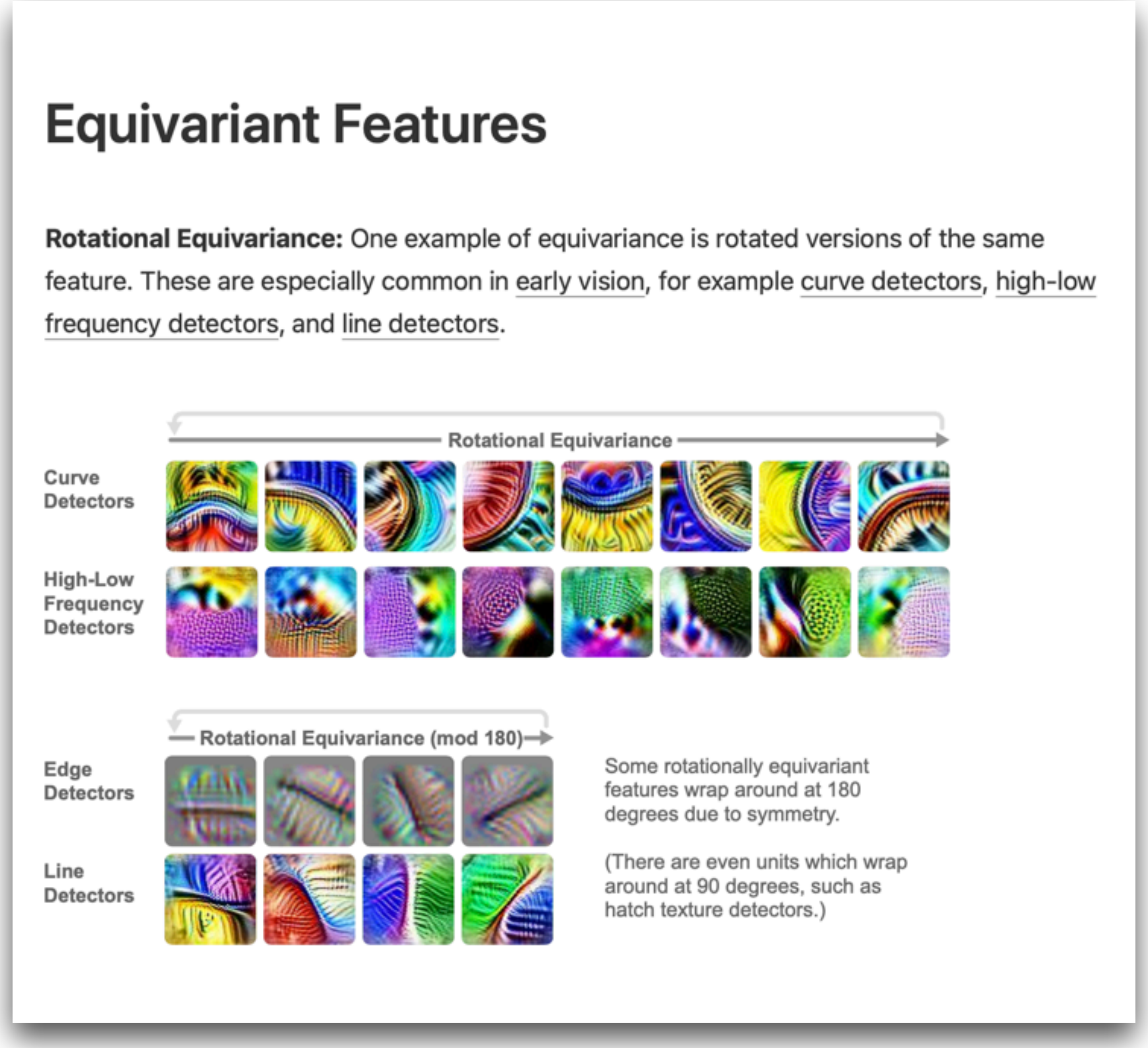


Naturally Occurring Equivariance in Neural Networks

AUTHORS	AFFILIATIONS	PUBLISHED	DOI
Chris Olah	OpenAI	Dec. 8, 2020	10.23915/distill.00024.004
Nick Cammarata	OpenAI		
Chelsea Voss	OpenAI		
Ludwig Schubert			
Gabriel Goh	OpenAI		



The weights for the units in the first layer of the TF-Slim [11] version of InceptionV1 [8] . [5] Units are sorted by the first principal component of the adjacency matrix between the first and second layers. Note how many features are similar except for rotation, scale, and hue.



Equivariant Features

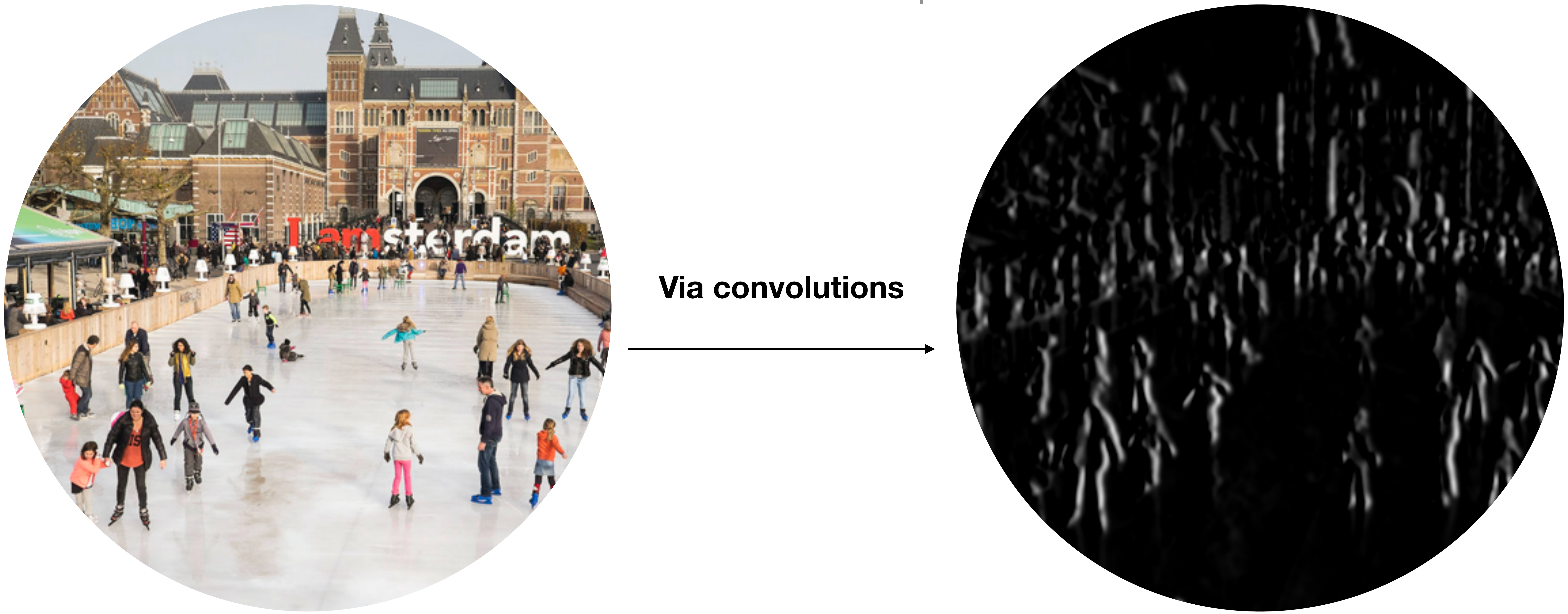
Rotational Equivariance: One example of equivariance is rotated versions of the same feature. These are especially common in early vision, for example curve detectors, high-low frequency detectors, and line detectors.

Some rotationally equivariant features wrap around at 180 degrees due to symmetry.

(There are even units which wrap around at 90 degrees, such as hatch texture detectors.)

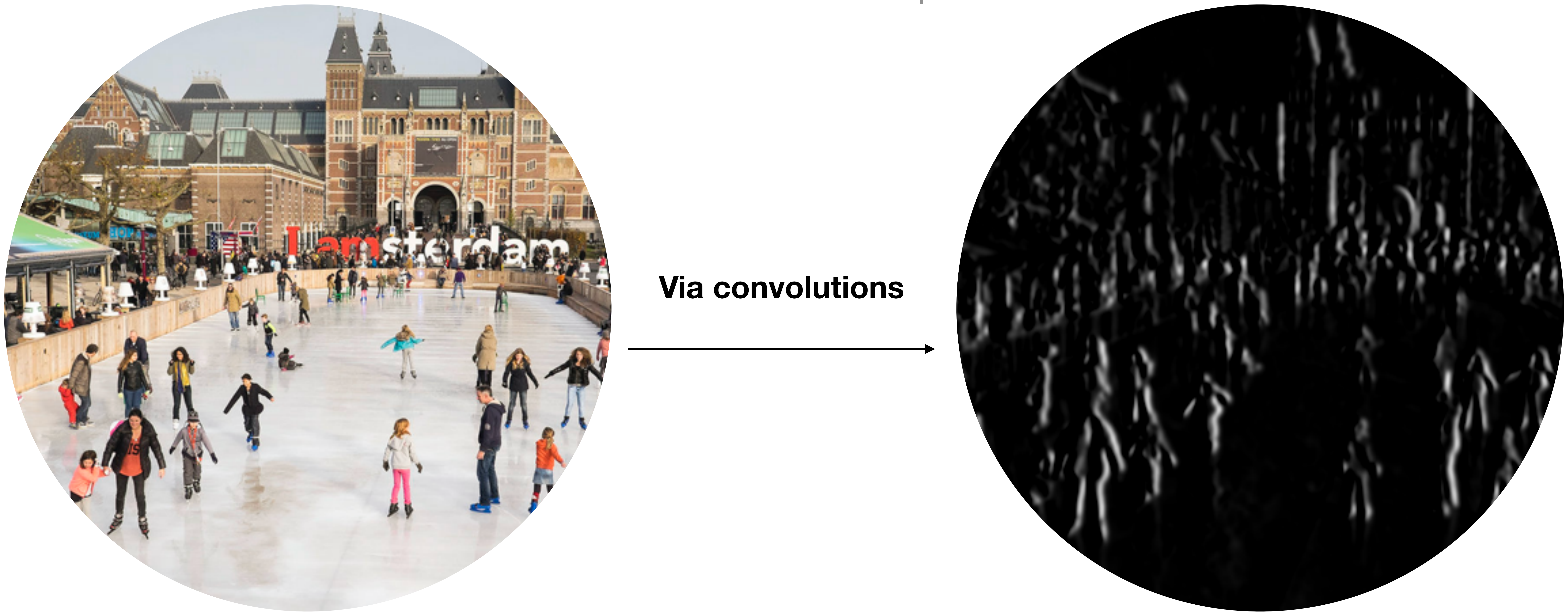
Geometric guarantees (equivariance)

CNNs are translation equivariant



Geometric guarantees (equivariance)

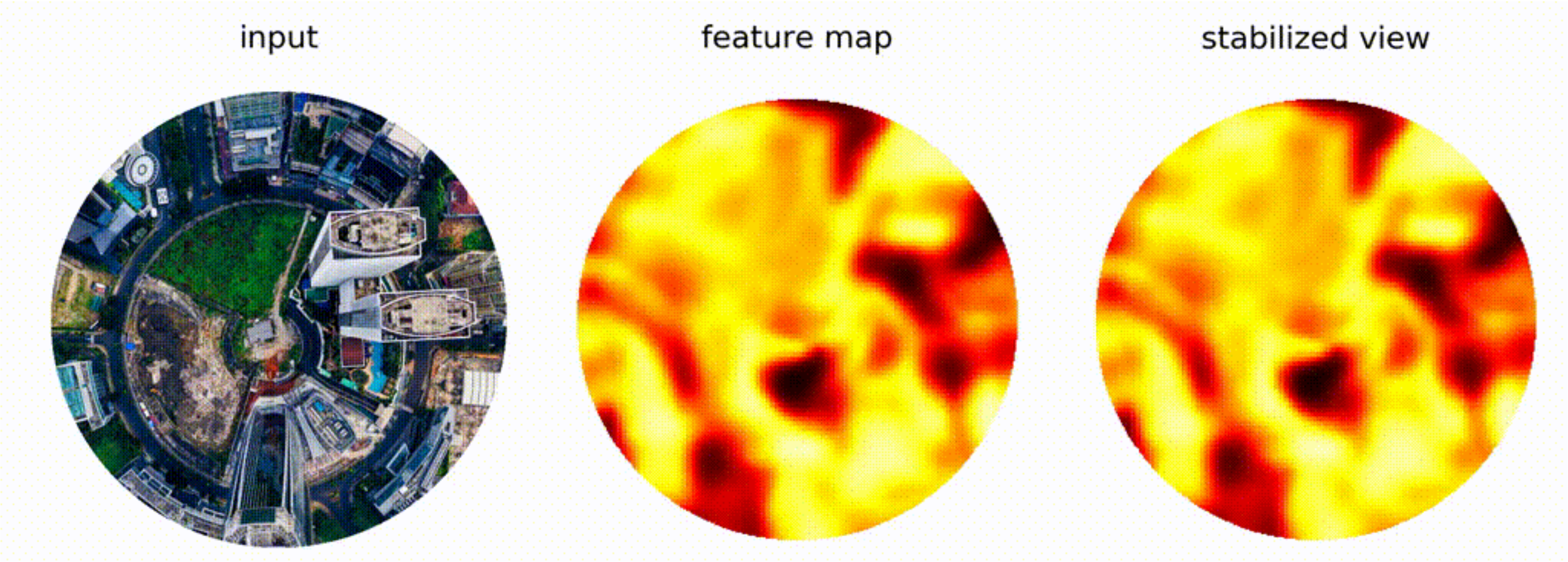
CNNs are translation equivariant



Via convolutions

Geometric guarantees (equivariance)

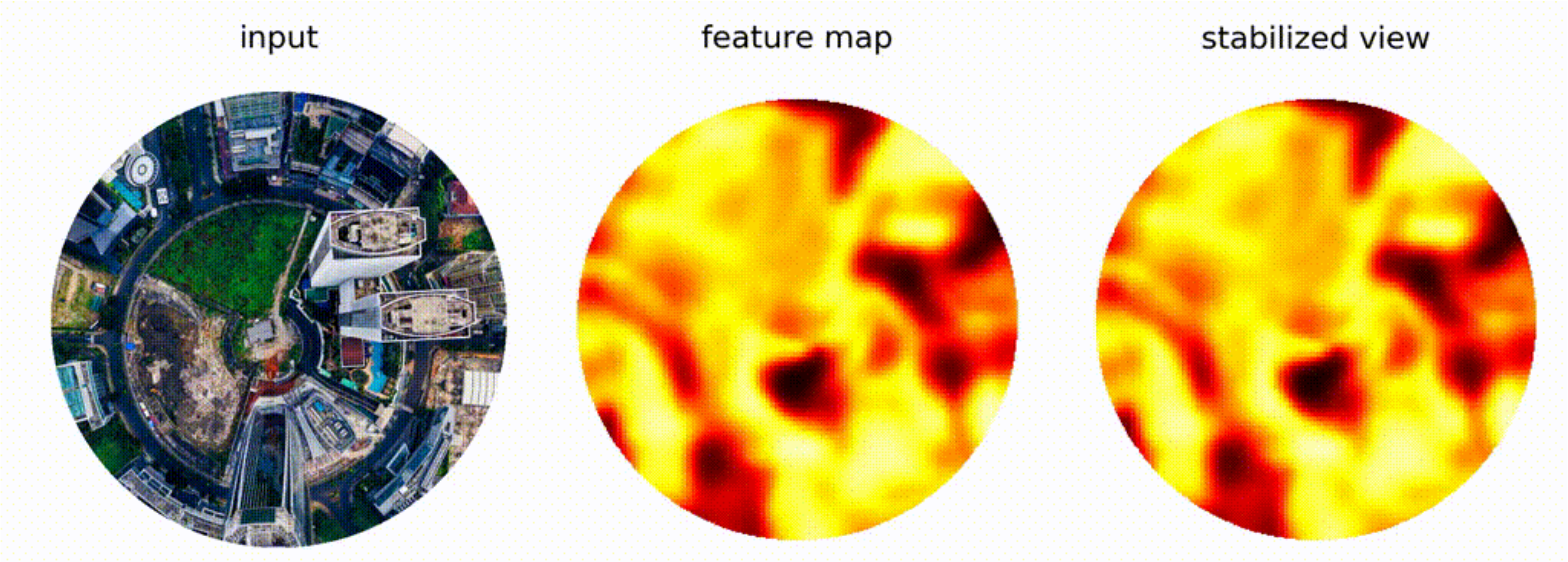
Normal CNN



Figures source:
https://github.com/QUVA-Lab/e2cnn

Geometric guarantees (equivariance)

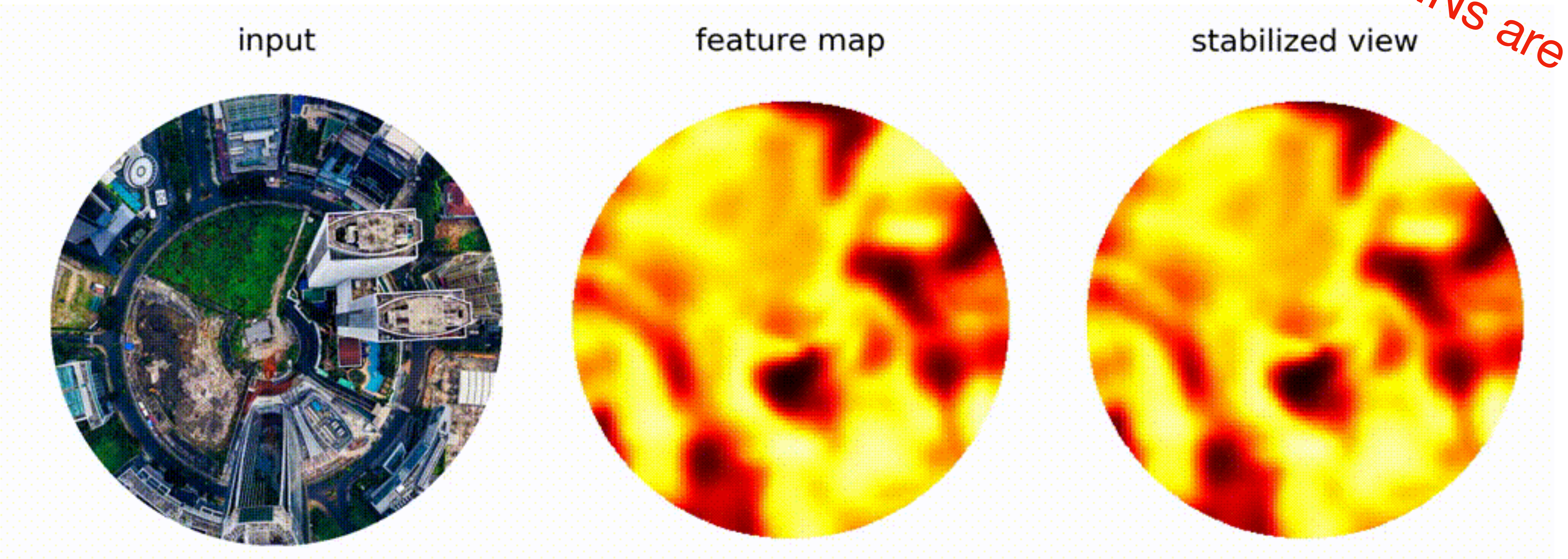
Normal CNN



Figures source:
https://github.com/QUVA-Lab/e2cnn

Geometric guarantees (equivariance)

Normal CNN



CNNs are not rotation equivariant

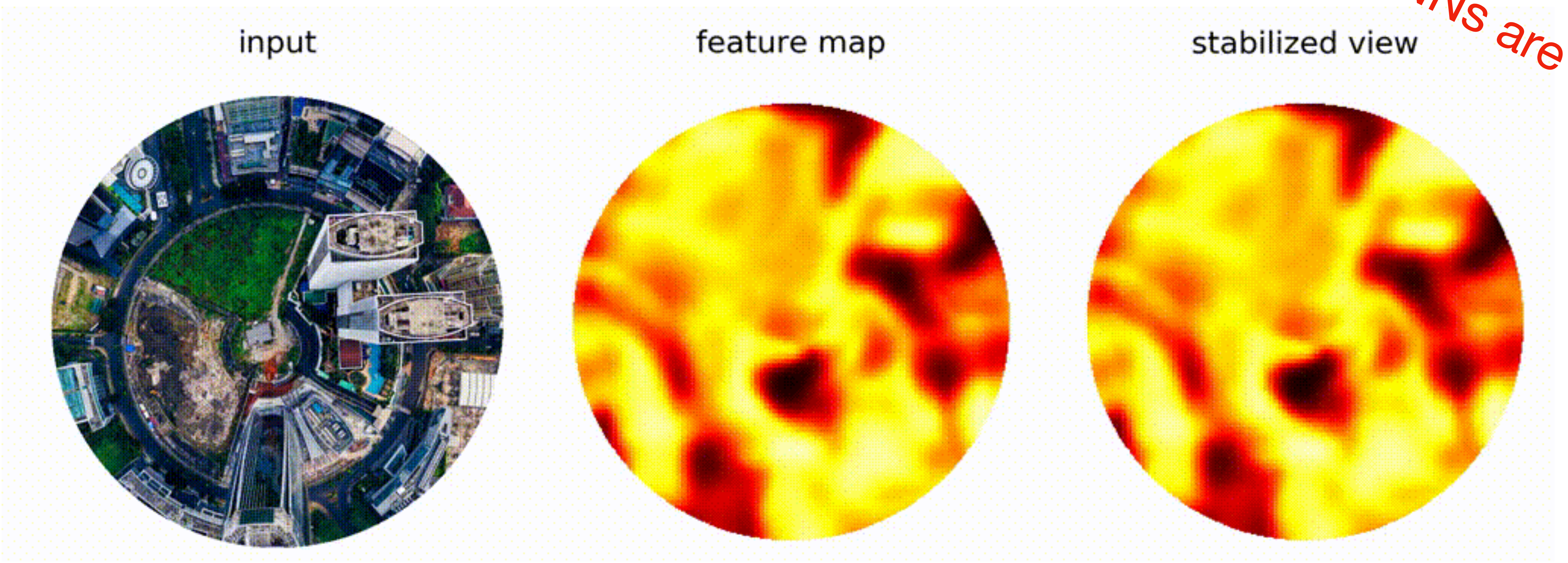
☹

Figures source:
https://github.com/QUVA-Lab/e2cnn

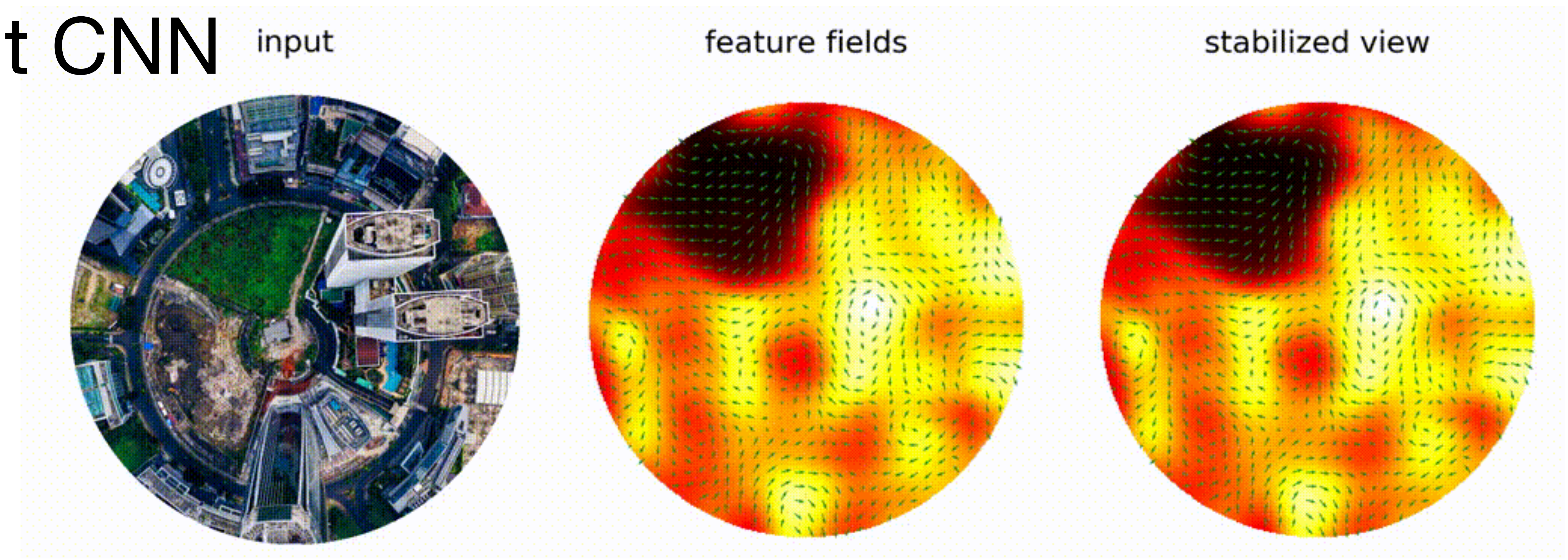
Geometric guarantees (equivariance)

CNNs are not rotation equivariant

Normal CNN



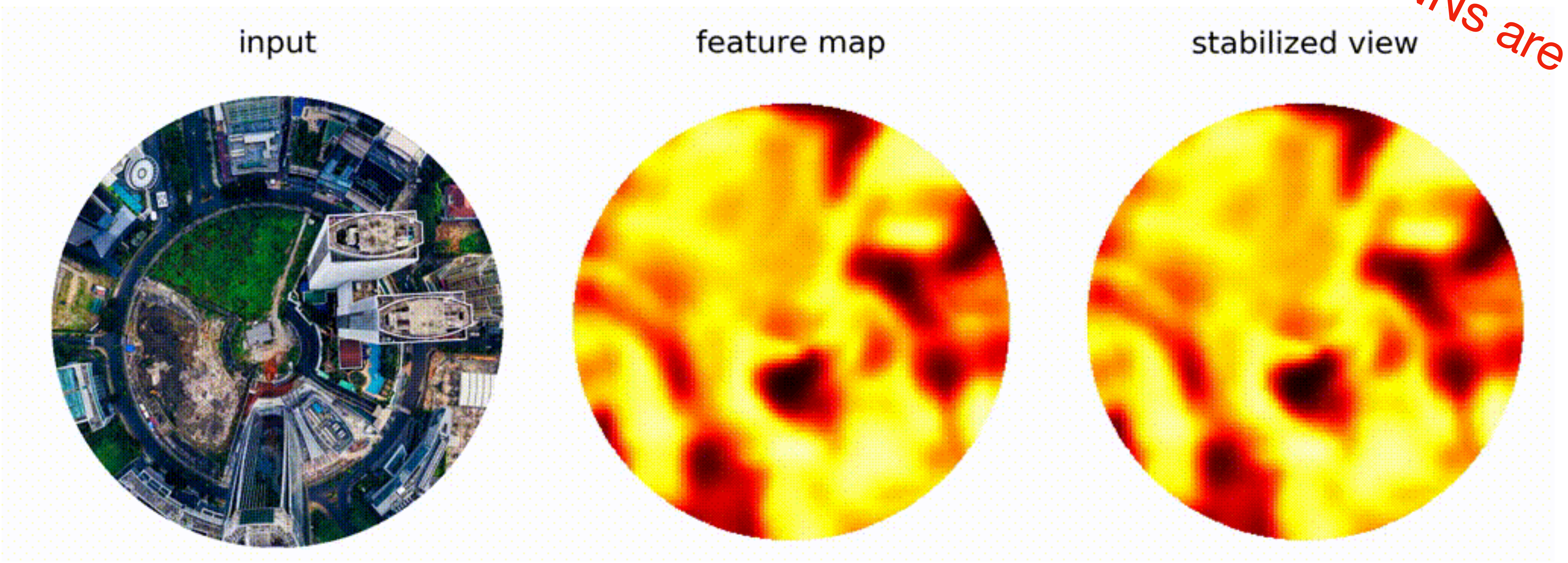
Group equivariant CNN



Figures source:
https://github.com/QUVA-Lab/e2cnn

Geometric guarantees (equivariance)

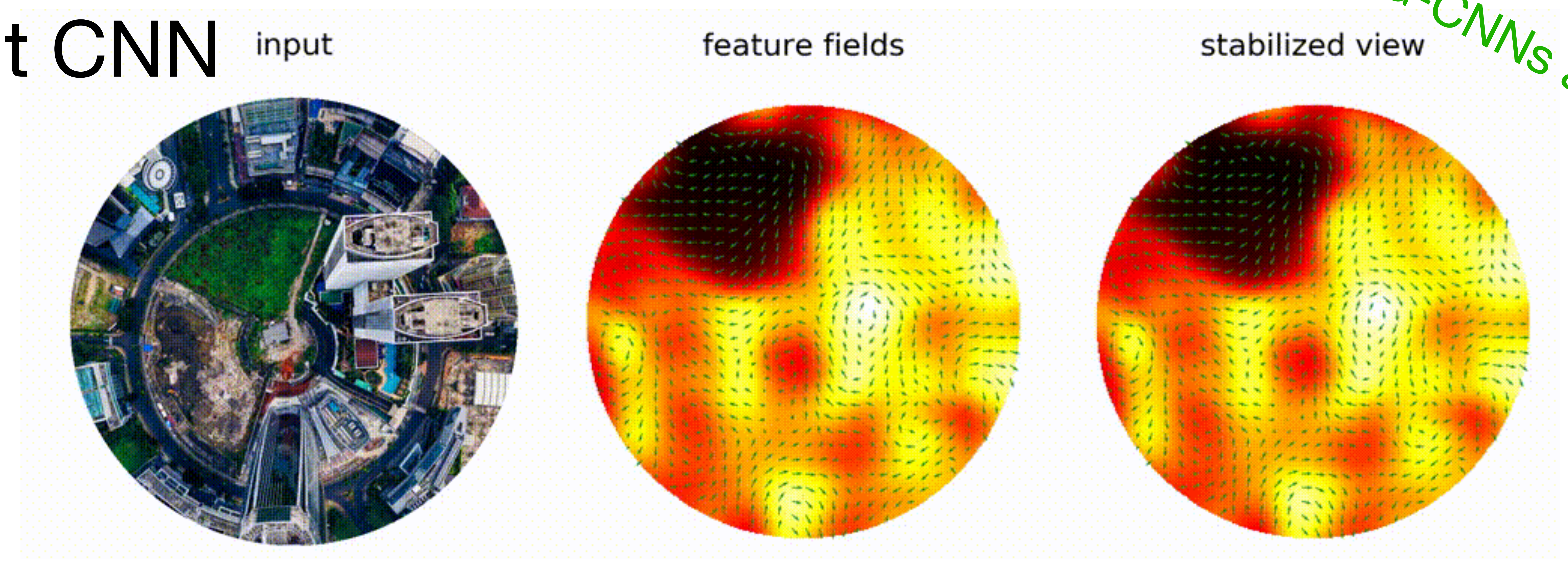
Normal CNN



CNNs are not rotation equivariant

☹

Group equivariant CNN

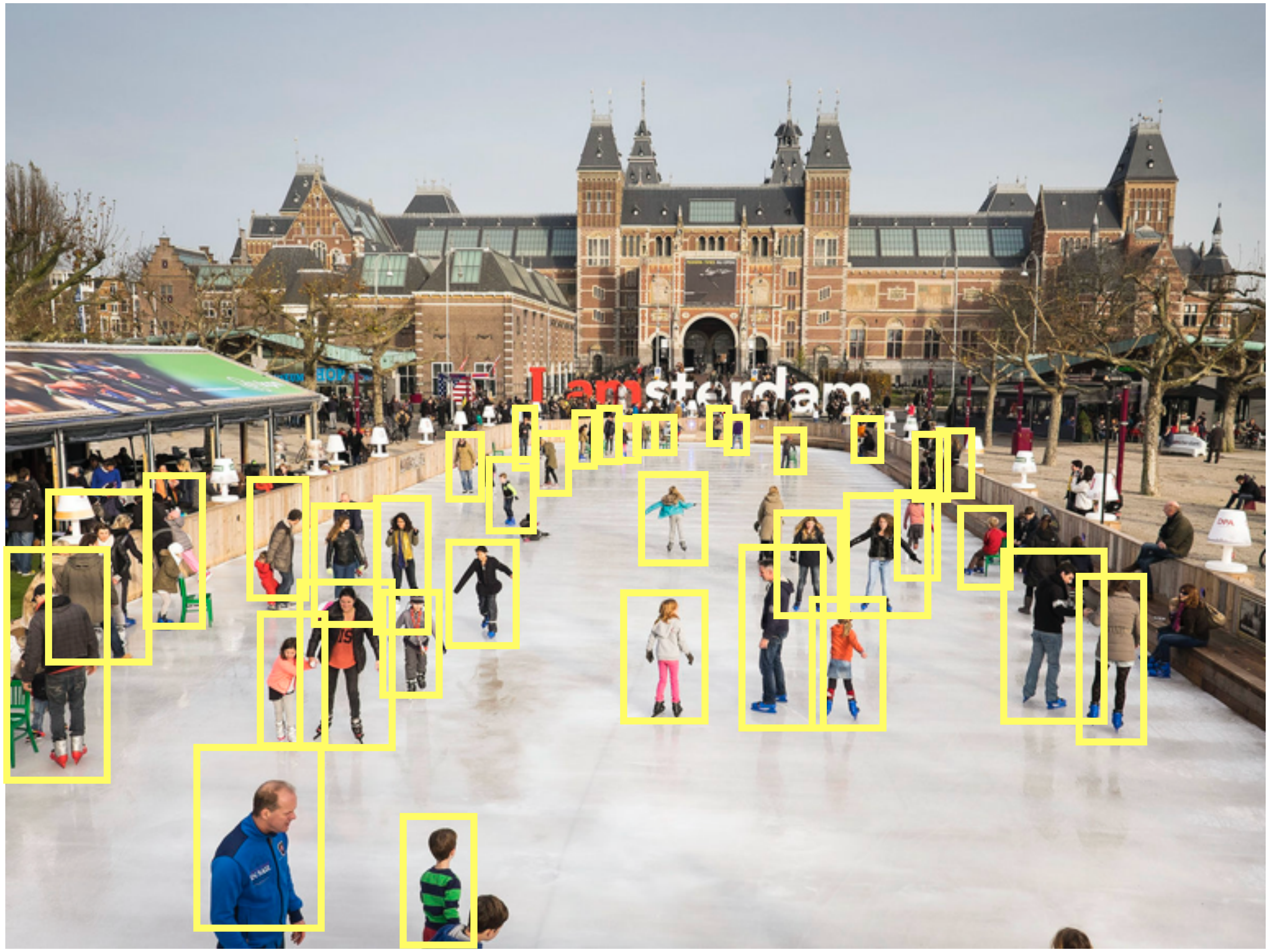


G-CNNs are rotation equivariant!

☺

Figures source:
https://github.com/QUVA-Lab/e2cnn

Geometric guarantees (equivariance)



Importance of equivariance:

- No information is lost when the input is transformed
- Guaranteed stability to (local + global) transformations

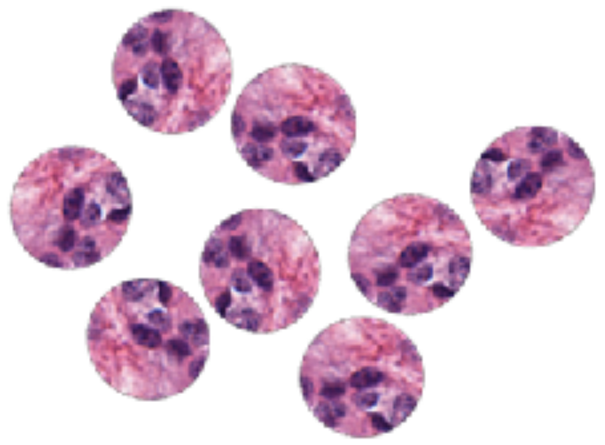
Group convolutions:

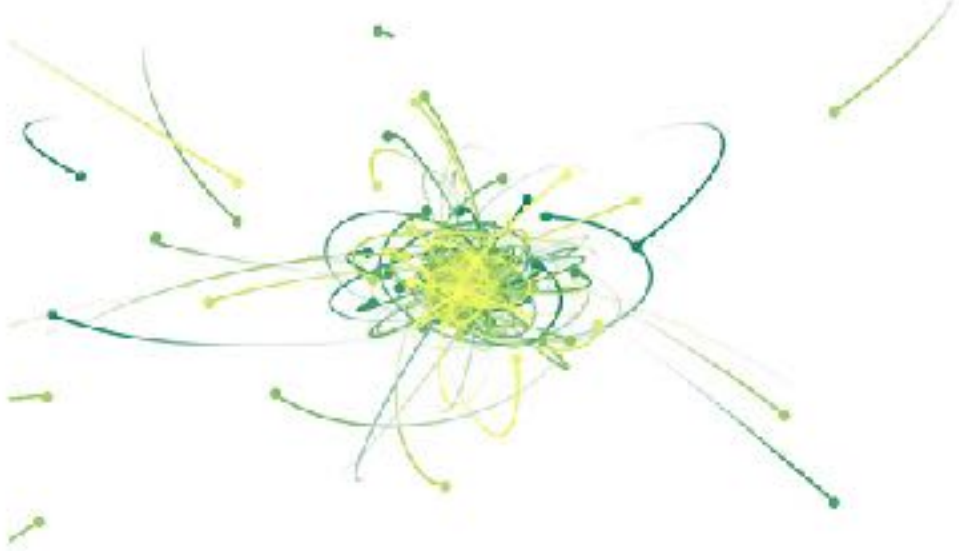
- Equivariance beyond translations
- Geometric guarantees
- Increased weight sharing

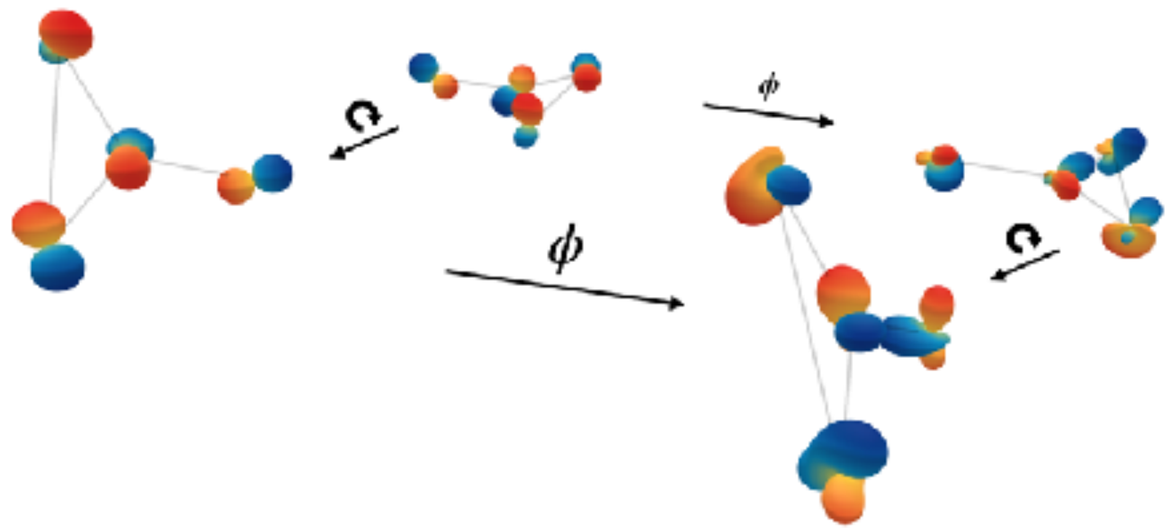
G-CNNs are not only relevant for invariant problems but for any type of structured data!

Group equivariant deep learning

Create architectures with guarantees of invariance or equivariance
(often demanded by problems)



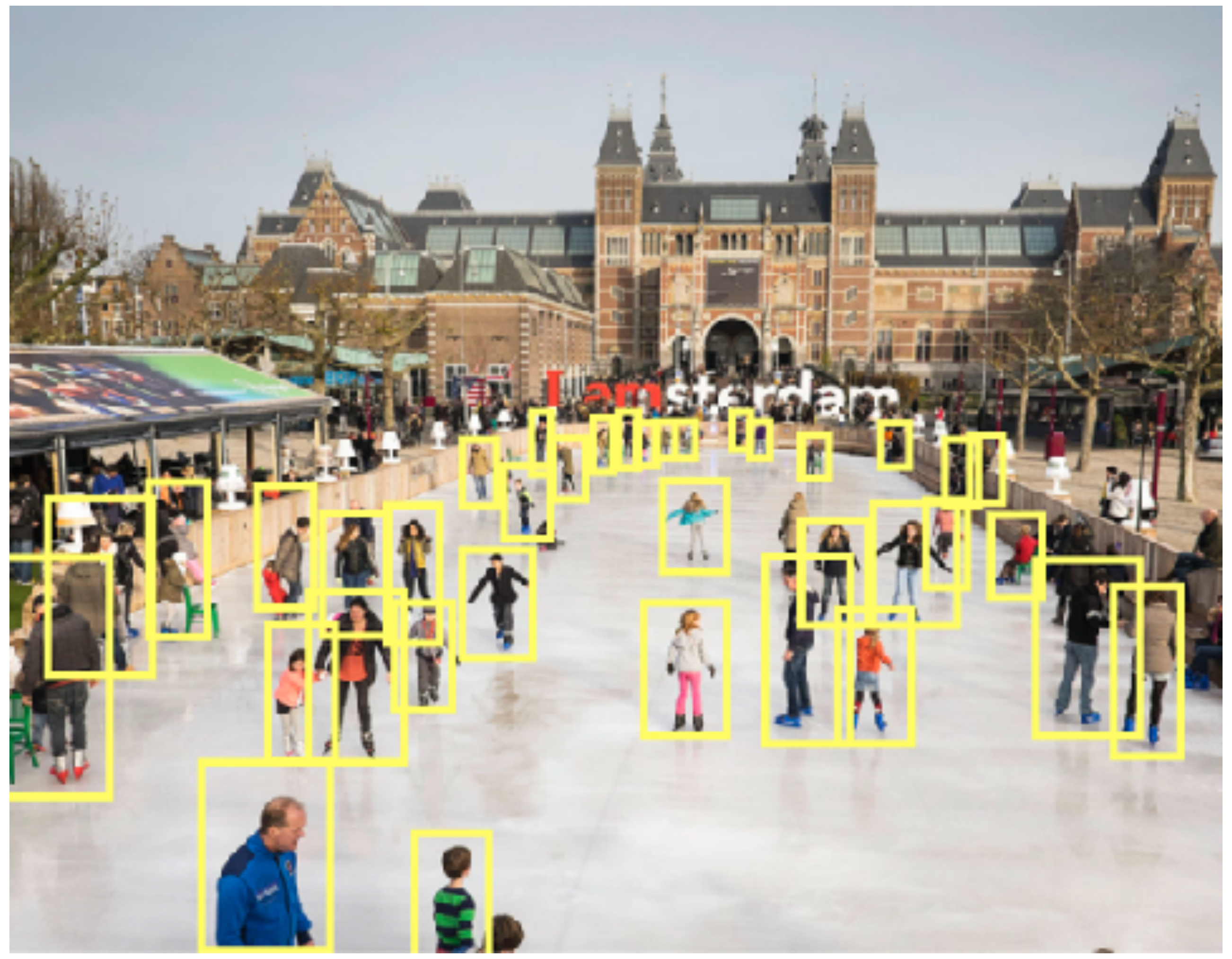


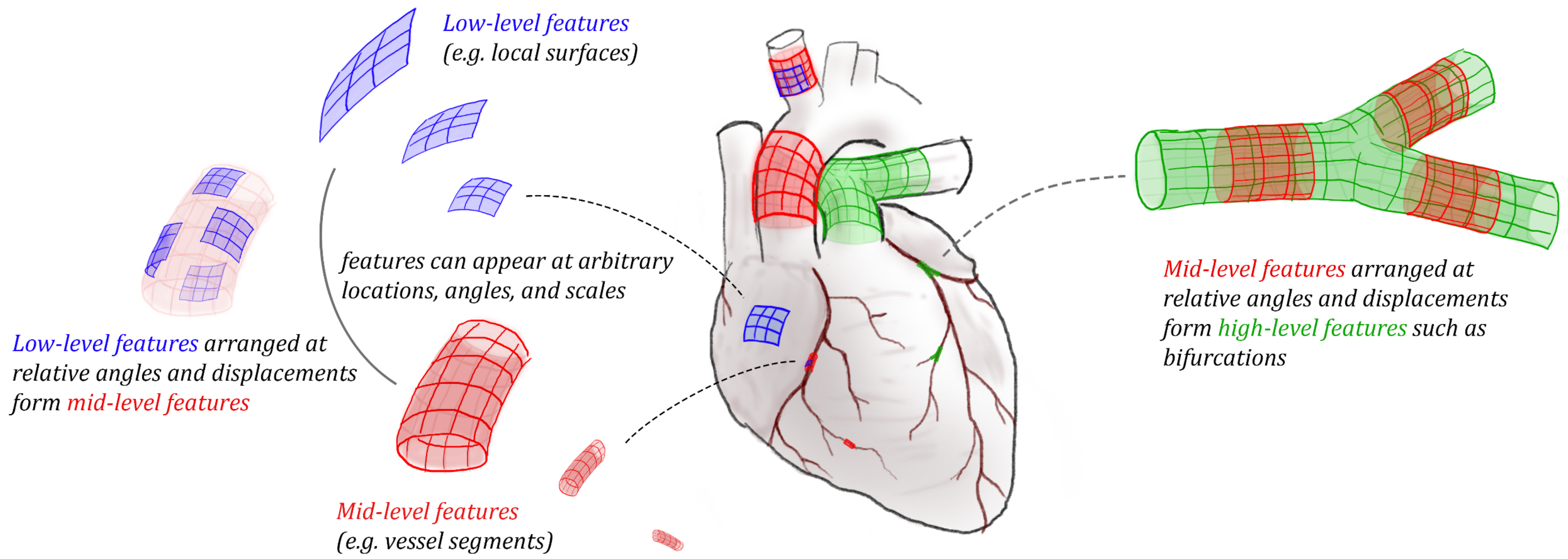


Equivariance allows for increased weight sharing

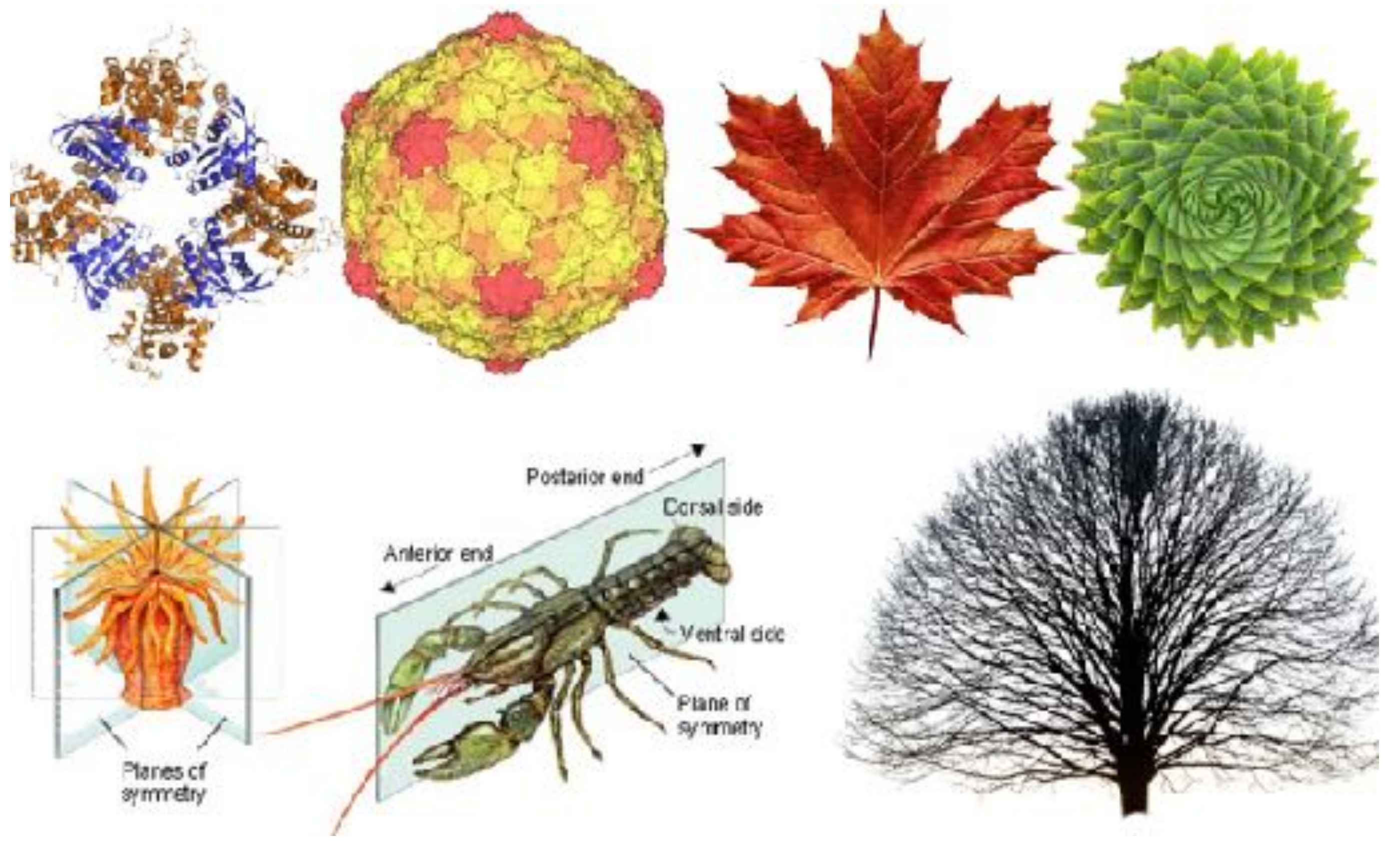
Psychology of vision (recognition by components)

Efficient representation learning (leverage symmetries)





Low-level features
(e.g. local surfaces)
features can appear at arbitrary locations, angles, and scales
Low-level features arranged at relative angles and displacements form mid-level features
Mid-level features
(e.g. vessel segments)
Mid-level features arranged at relative angles and displacements form high-level features such as bifurcations



1. Motivation — Equivariance → weight-sharing and generalization

2. Pattern matching using group theory

3. Group convolutions

  4. Example

  5. G-convs are all you need!

6. Steerable group convolutions

  7. Feature fields and escnn library

8. Equivariant tensor product layers

9. Equivariant graph NNs

1. Motivation — Equivariance → weight-sharing and generalization

2. **Pattern matching using group theory**

3. Group convolutions

    4. Example

    5. G-convs are all you need!

6. Steerable group convolutions

    7. Feature fields and escnn library

8. Equivariant tensor product layers

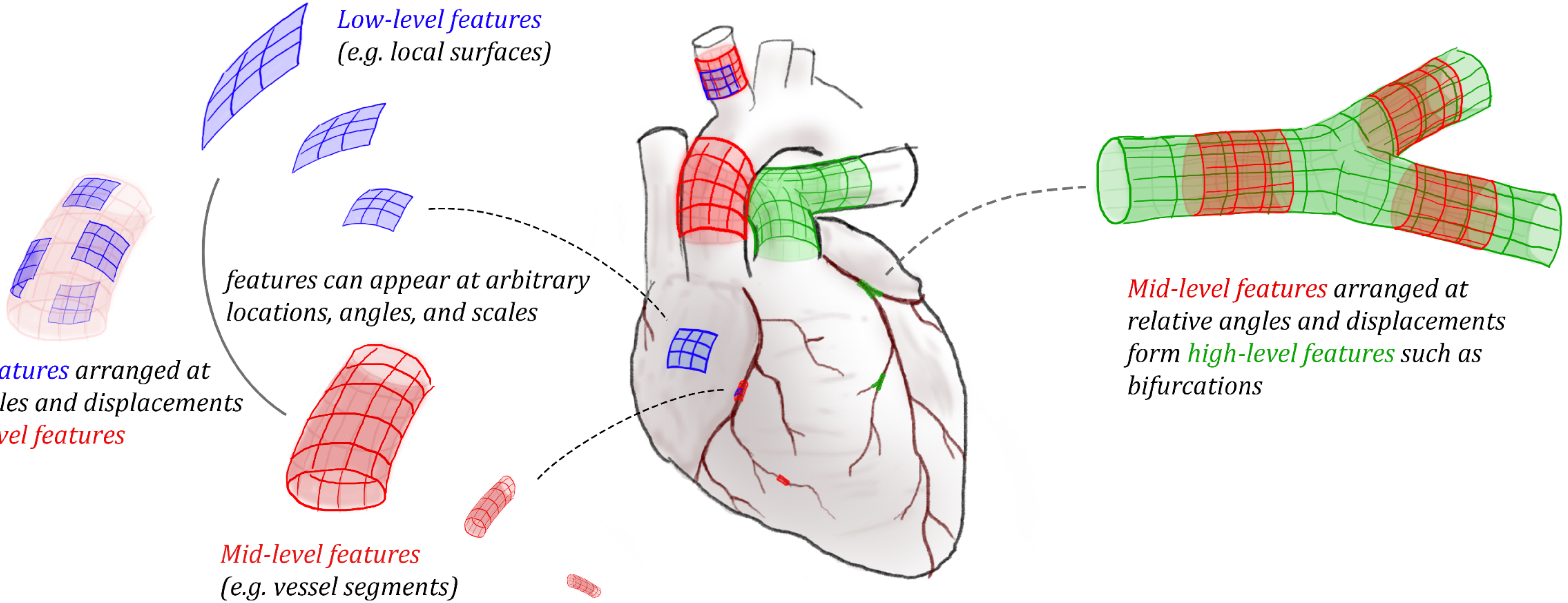
9. Equivariant graph NNs

What is a group?

A group $(G, \cdot\ )$ is a set of elements G equipped with a group product $\cdot$, a binary operator, that satisfies the following four axioms:

- **Closure**: Given two elements g and h of G, the product $g \cdot h$ is also in G.
- **Associativity**: For $g, h, i \in G$ the product $\cdot$ is associative, i.e., $g \cdot (h \cdot i) = (g \cdot h) \cdot i$.
- **Identity element**: There exists an identity element $e \in G$ such that $e \cdot g = g \cdot e = g$ for any $g \in G$.
- **Inverse element**: For each $g \in G$ there exists an inverse element $g^{-1} \in G$ s.t. $g^{-1} \cdot g = g \cdot g^{-1} = e$.

Psychology of vision: recognition by components



Low-level features
(e.g. local surfaces)

features can appear at arbitrary locations, angles, and scales

Low-level features arranged at relative angles and displacements form mid-level features

Mid-level features
(e.g. vessel segments)

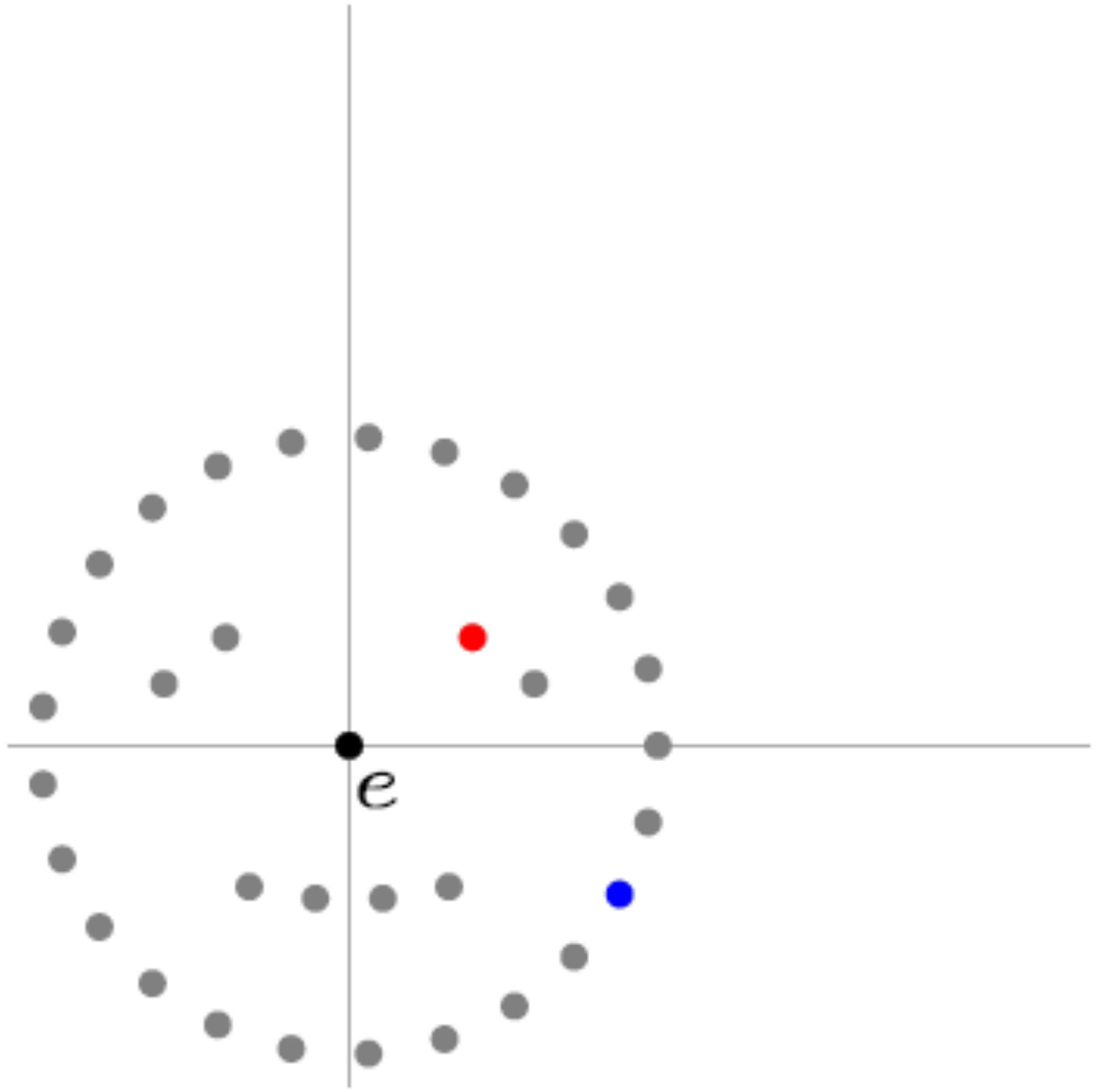
Mid-level features arranged at relative angles and displacements form high-level features such as bifurcations

Translation group $(\mathbb{R}^2, +)$

The translation group consists of all possible translations in $\mathbb{R}^2$ and is equipped with the group product and group inverse:

$$g \cdot g' = (\mathbf{x} + \mathbf{x}')$$
$$g^{-1} = (-\mathbf{x})$$

with $g = (\mathbf{x}), g' = (\mathbf{x}')$ and $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^2$.

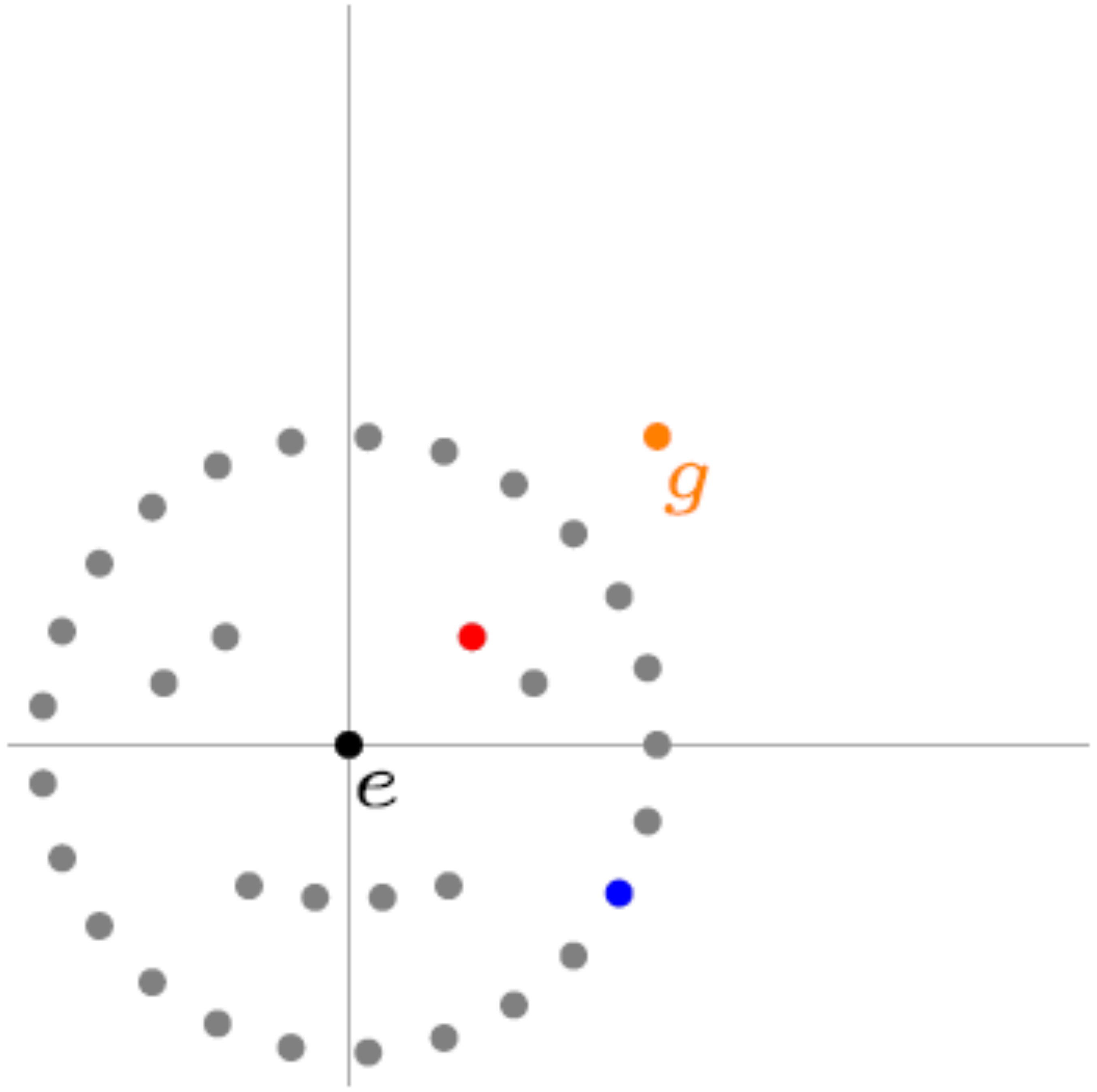


Translation group $(\mathbb{R}^2, +)$

The translation group consists of all possible translations in $\mathbb{R}^2$ and is equipped with the group product and group inverse:

$$
\begin{aligned}
g \cdot g' &= (\mathbf{x} + \mathbf{x}') \\
g^{-1} &= (-\mathbf{x})
\end{aligned}
$$

with $g = (\mathbf{x}), g' = (\mathbf{x}')$ and $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^2$.

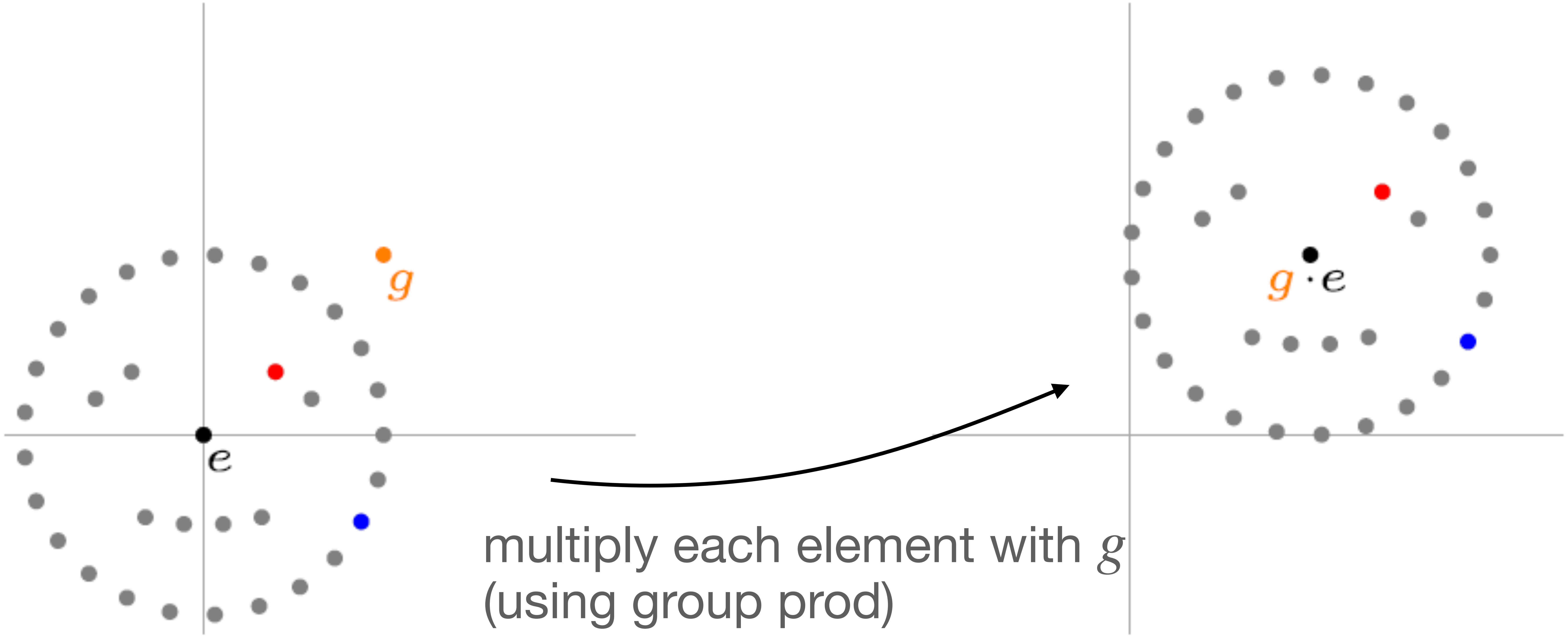


Translation group $(\mathbb{R}^2, +)$

The translation group consists of all possible translations in $\mathbb{R}^2$ and is equipped with the group product and group inverse:

$$g \cdot g' = (\mathbf{x} + \mathbf{x}')$$
$$g^{-1} = (-\mathbf{x})$$

with $g = (\mathbf{x}), g' = (\mathbf{x}')$ and $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^2$.



multiply each element with g
(using group prod)

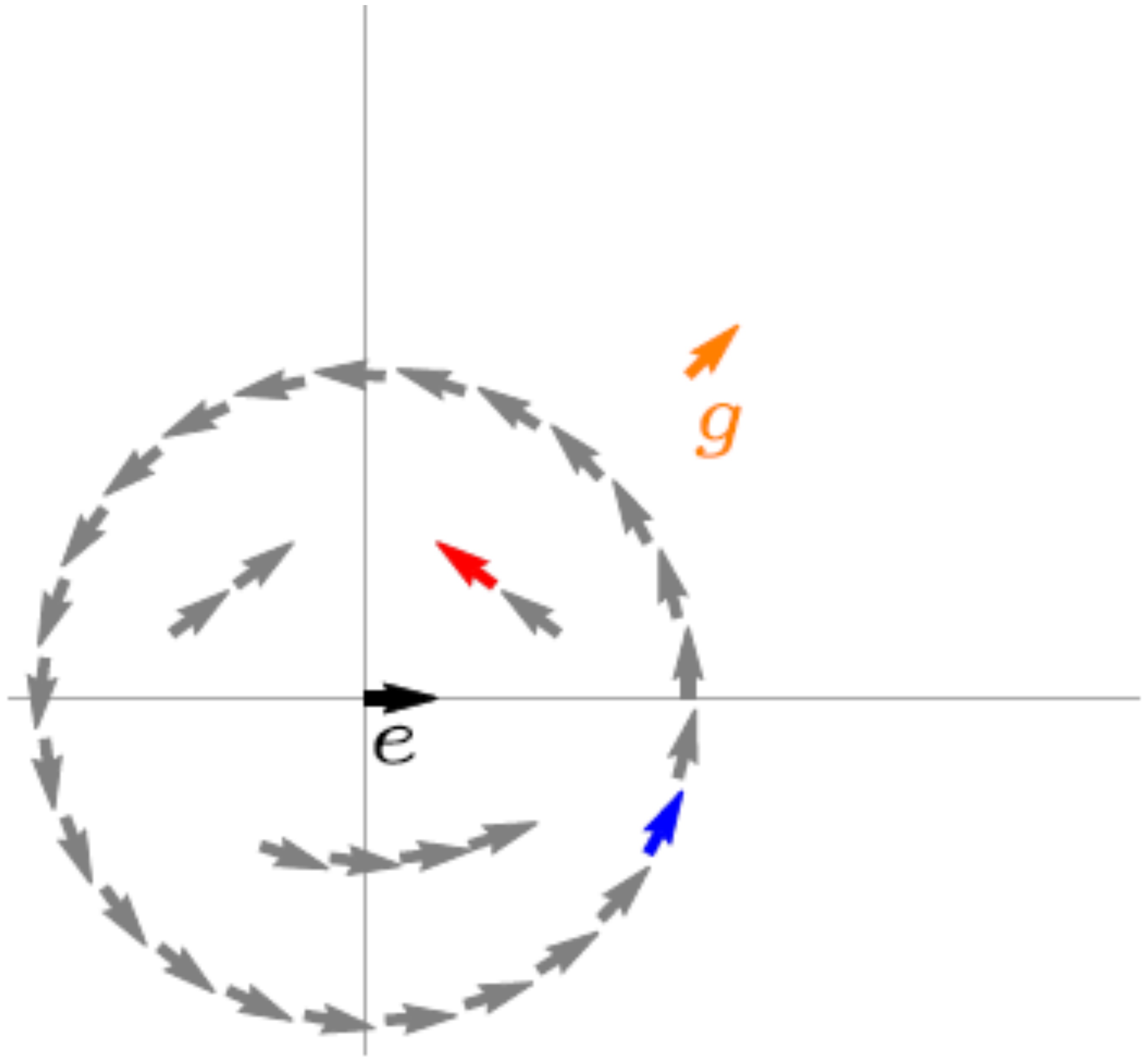
Roto-ranslation group $SE(2)$

***2**D **S**pecial **E**uclidean motion group*

The group $SE(2) = \mathbb{R}^2 \rtimes SO(2)$ consists of the coupled space $\mathbb{R}^2 \times S^1$ of translations vectors in $\mathbb{R}^2$, and rotations in $SO(2)$ (or equivalently orientations in S^1), and is equipped with the group product and group inverse:

$$g \cdot g' = (\mathbf{x}, \mathbf{R}_\theta) \cdot (\mathbf{x}', \mathbf{R}_{\theta'}) = (\mathbf{R}_\theta \mathbf{x}' + \mathbf{x}, \mathbf{R}_{\theta+\theta'})$$
$$g^{-1} = (-\mathbf{R}_\theta^{-1}\mathbf{x}, \mathbf{R}_\theta^{-1})$$

with $g = (\mathbf{x}, \mathbf{R}_\theta)$, $g' = (\mathbf{x}', \mathbf{R}_{\theta'})$.



Roto-ranslation group $SE(2)$

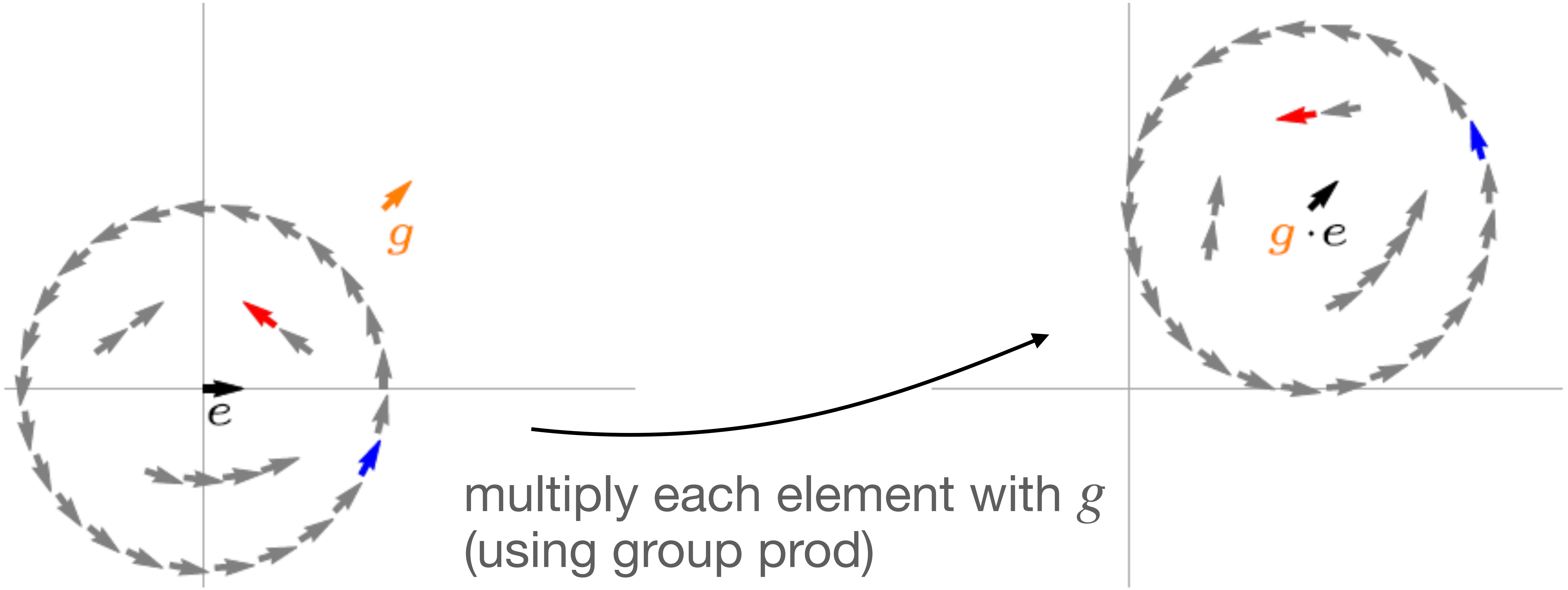
*2D **S**pecial **E**uclidean motion group*

The group $SE(2) = \mathbb{R}^2 \rtimes SO(2)$ consists of the coupled space $\mathbb{R}^2 \times S^1$ of translations vectors in $\mathbb{R}^2$, and rotations in $SO(2)$ (or equivalently orientations in S^1), and is equipped with the group product and group inverse:

$$g \cdot g' = (\mathbf{x}, \mathbf{R}_\theta) \cdot (\mathbf{x}', \mathbf{R}_{\theta'}) = (\mathbf{R}_\theta \mathbf{x}' + \mathbf{x}, \mathbf{R}_{\theta+\theta'})$$

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

with $g = (\mathbf{x}, \mathbf{R}_\theta)$, $g' = (\mathbf{x}', \mathbf{R}_{\theta'})$.



multiply each element with g
(using group prod)

Roto-ranslation group $SE(2)$

*2D **S**pecial **E**uclidean motion group*

Matrix representation: The group can also be represented by matrices

$$g = (\mathbf{x}, \mathbf{R}_\theta) \quad \leftrightarrow \quad \mathbf{G} = \begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_\theta & \mathbf{x} \\ \mathbf{0}^T & 1 \end{pmatrix}$$

with the group product and inverse simply given by the matrix product and matrix inverse.

In parametric form: $(\mathbf{x}, \theta) \cdot (\mathbf{x}', \theta') = (\mathbf{R}_\theta \mathbf{x}' + \mathbf{x},\ \theta + \theta' \operatorname{mod} 2\pi)$

$$\leftrightarrow$$

In matrix form: $\begin{pmatrix} \mathbf{R}_\theta & \mathbf{x} \\ \mathbf{0}^T & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}'_\theta & \mathbf{x}' \\ \mathbf{0}^T & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_{\theta+\theta'} & \mathbf{R}_\theta \mathbf{x}' + \mathbf{x} \\ \mathbf{0}^T & 1 \end{pmatrix}$

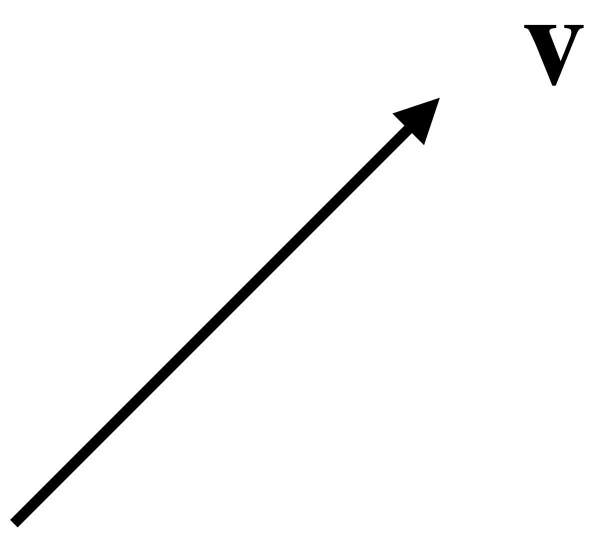
Representations

A representation $\rho : G \rightarrow GL(V)$ is a group homomorphism from G to the general linear group $GL(V)$.

That is $\rho(g)$ is a linear transformation that is parameterized by group elements $g \in G$ that transforms some vector $\mathbf{v} \in V$ (e.g. an image) such that

$$\rho(g') \circ \rho(g)[\mathbf{v}] = \rho(g' \cdot g)\,[\mathbf{v}]$$

Representations

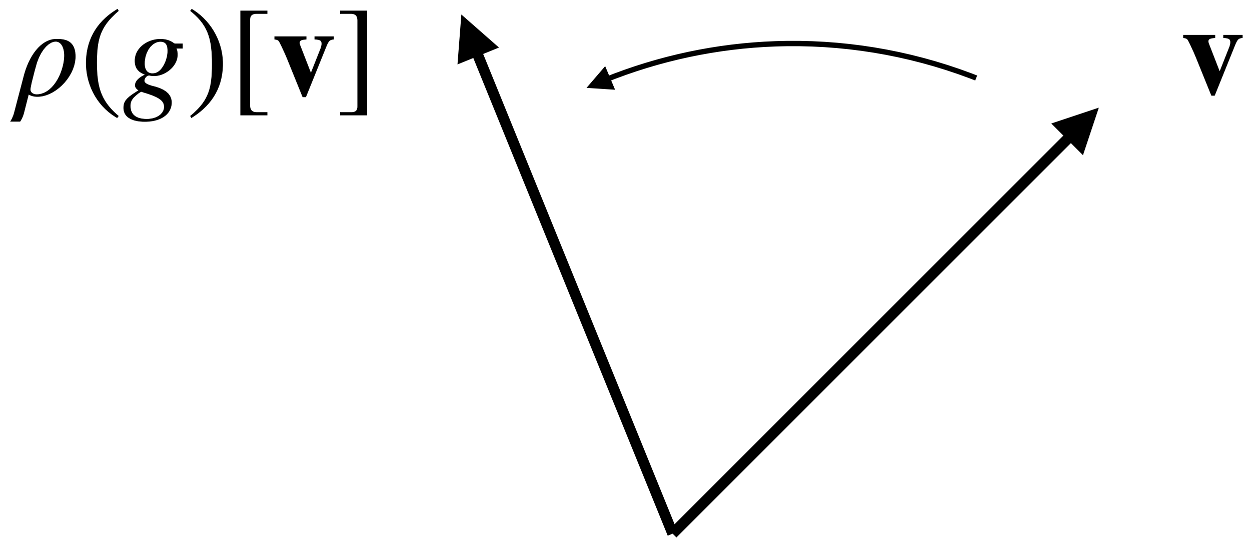


A representation $\rho : G \rightarrow GL(V)$ is a group homomorphism from G to the general linear group $GL(V)$.

That is $\rho(g)$ is a linear transformation that is parameterized by group elements $g \in G$ that transforms some vector $\mathbf{v} \in V$ (e.g. an image) such that

$$\rho(g') \circ \rho(g)[\mathbf{v}] = \rho(g' \cdot g)\,[\mathbf{v}]$$

Representations

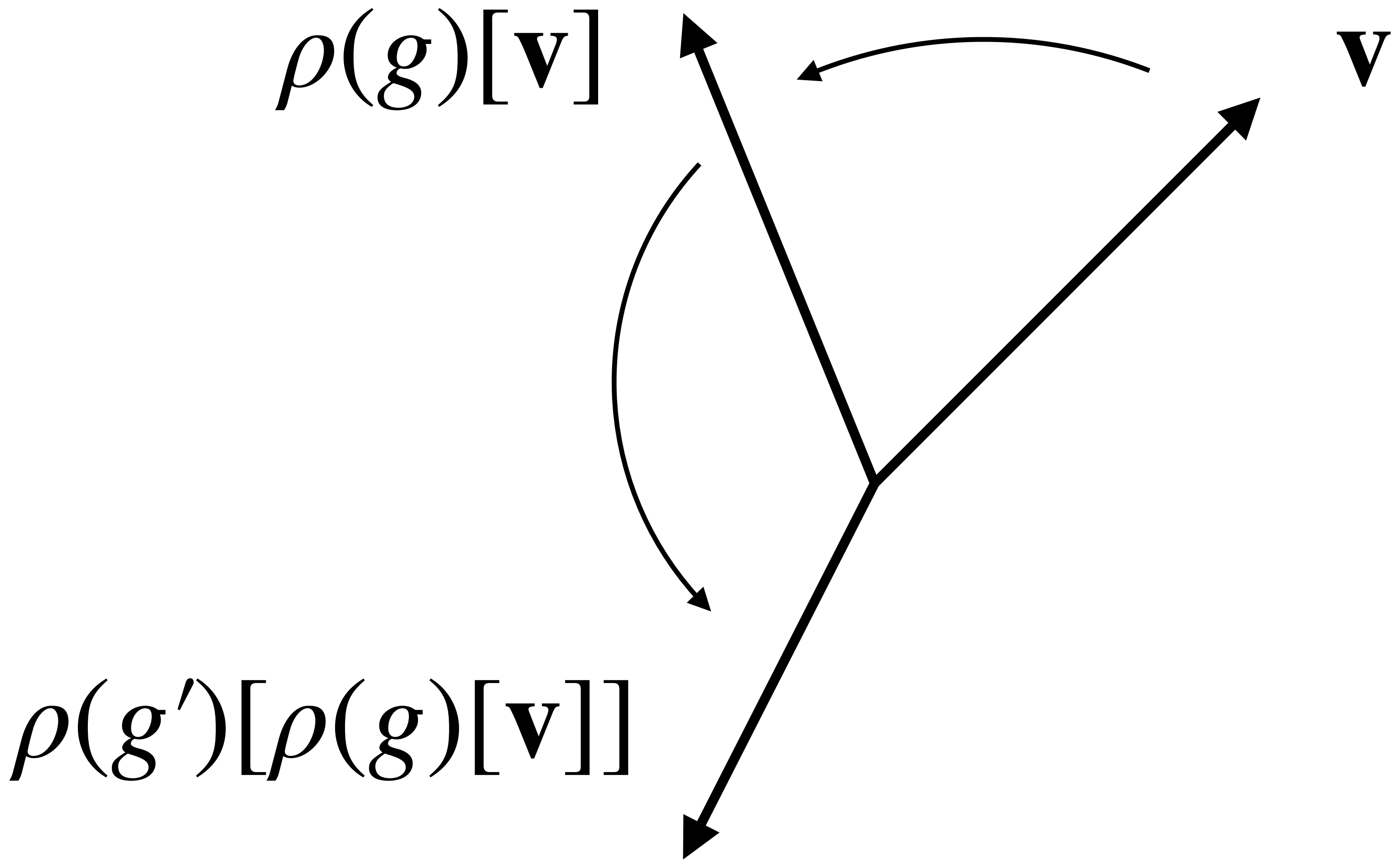


A representation $\rho : G \rightarrow GL(V)$ is a group homomorphism from G to the general linear group $GL(V)$.

That is $\rho(g)$ is a linear transformation that is parameterized by group elements $g \in G$ that transforms some vector $\mathbf{v} \in V$ (e.g. an image) such that

$$\rho(g') \circ \rho(g)[\mathbf{v}] = \rho(g' \cdot g)\,[\mathbf{v}]$$

Representations

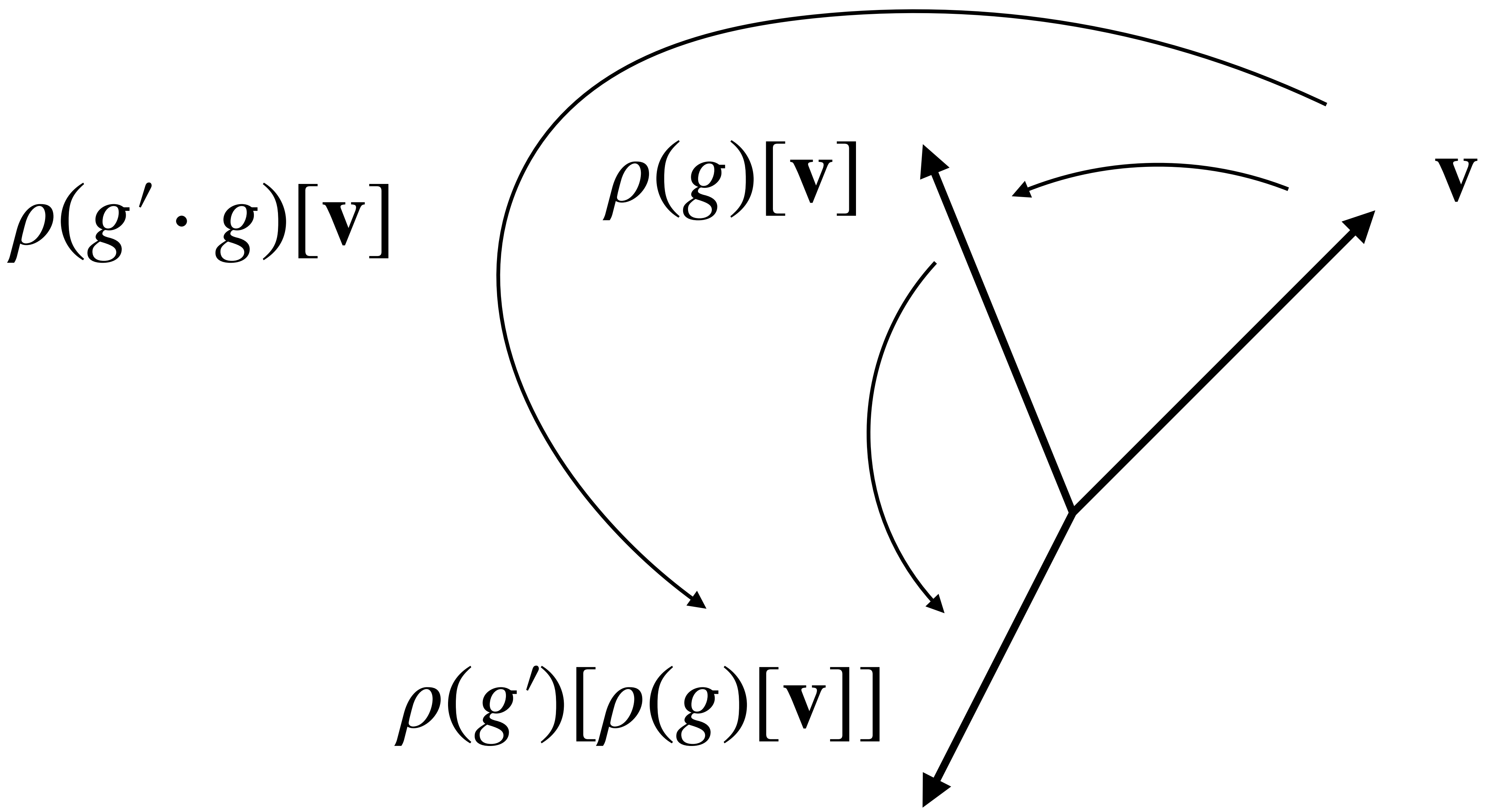


A representation $\rho : G \to GL(V)$ is a group homomorphism from G to the general linear group $GL(V)$.

That is $\rho(g)$ is a linear transformation that is parameterized by group elements $g \in G$ that transforms some vector $\mathbf{v} \in V$ (e.g. an image) such that

$$\rho(g') \circ \rho(g)[\mathbf{v}] = \rho(g' \cdot g)\,[\mathbf{v}]$$

Representations


$\rho(g' \cdot g)[\mathbf{v}]$
$\rho(g)[\mathbf{v}]$
$\mathbf{v}$
$\rho(g')[\rho(g)[\mathbf{v}]]$

A representation $\rho : G \rightarrow GL(V)$ is a group homomorphism from G to the general linear group $GL(V)$.

That is $\rho(g)$ is a linear transformation that is parameterized by group elements $g \in G$ that transforms some vector $\mathbf{v} \in V$ (e.g. an image) such that

$$\rho(g') \circ \rho(g)[\mathbf{v}] = \rho(g' \cdot g)\,[\mathbf{v}]$$

Left-regular Representations

A left-regular representation $\mathscr{L}_g$ is a representation that transforms functions f by transforming their domains via the inverse group action

$$\mathscr{L}_g[f](x) := f(g^{-1} \cdot x)$$

"group action" equals group product when domain is G

Left-regular Representations

Example:

$f \in \mathbb{L}_2(\mathbb{R}^2)$

- a 2D image

$G = SE(2)$

- the roto-translation group

$\mathscr{L}_g(f)(\mathbf{y}) = f(\mathbf{R}_\theta^{-1}(\mathbf{y} - \mathbf{x}))$

- a roto-translation of the image

A left-regular representation $\mathscr{L}_g$ is a representation that transforms functions f by transforming their domains via the inverse group action

$$\mathscr{L}_g[f](x) := f(g^{-1} \cdot x)$$

"group action" equals group product when domain is G

Left-regular Representations

Example:

$f \in \mathbb{L}_2(\mathbb{R}^2)$

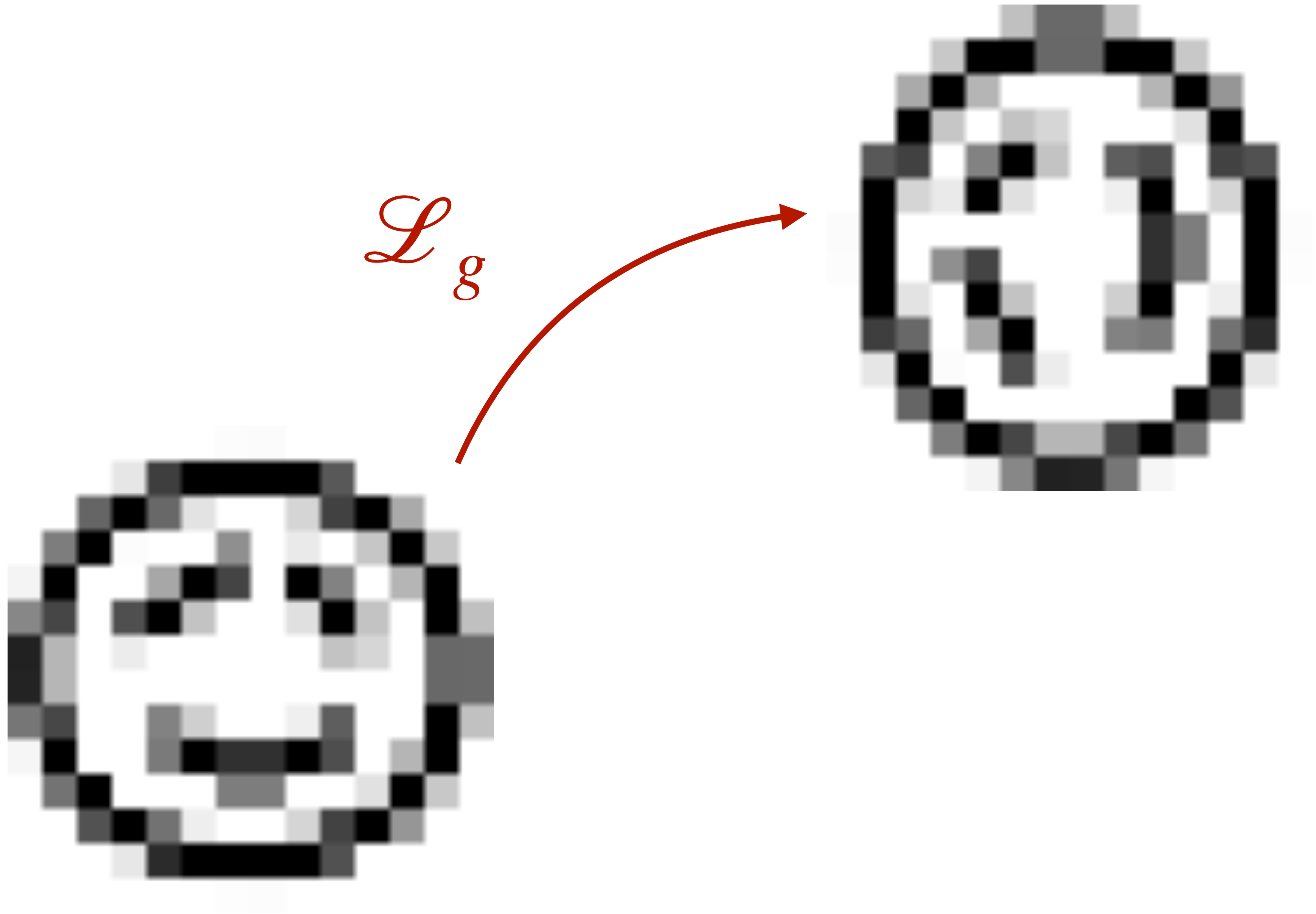
- a 2D image

$G = SE(2)$

- the roto-translation group

$\mathscr{L}_g(f)(\mathbf{y}) = f(\mathbf{R}_\theta^{-1}(\mathbf{y} - \mathbf{x}))$

- a roto-translation of the image



A left-regular representation $\mathscr{L}_g$ is a representation that transforms functions f by transforming their domains via the inverse group action

$$\mathscr{L}_g[f](x) := f(g^{-1} \cdot x)$$

"group action" equals group product when domain is G

Left-regular Representations

Example:

$f \in \mathbb{L}_2(\mathbb{R}^2)$

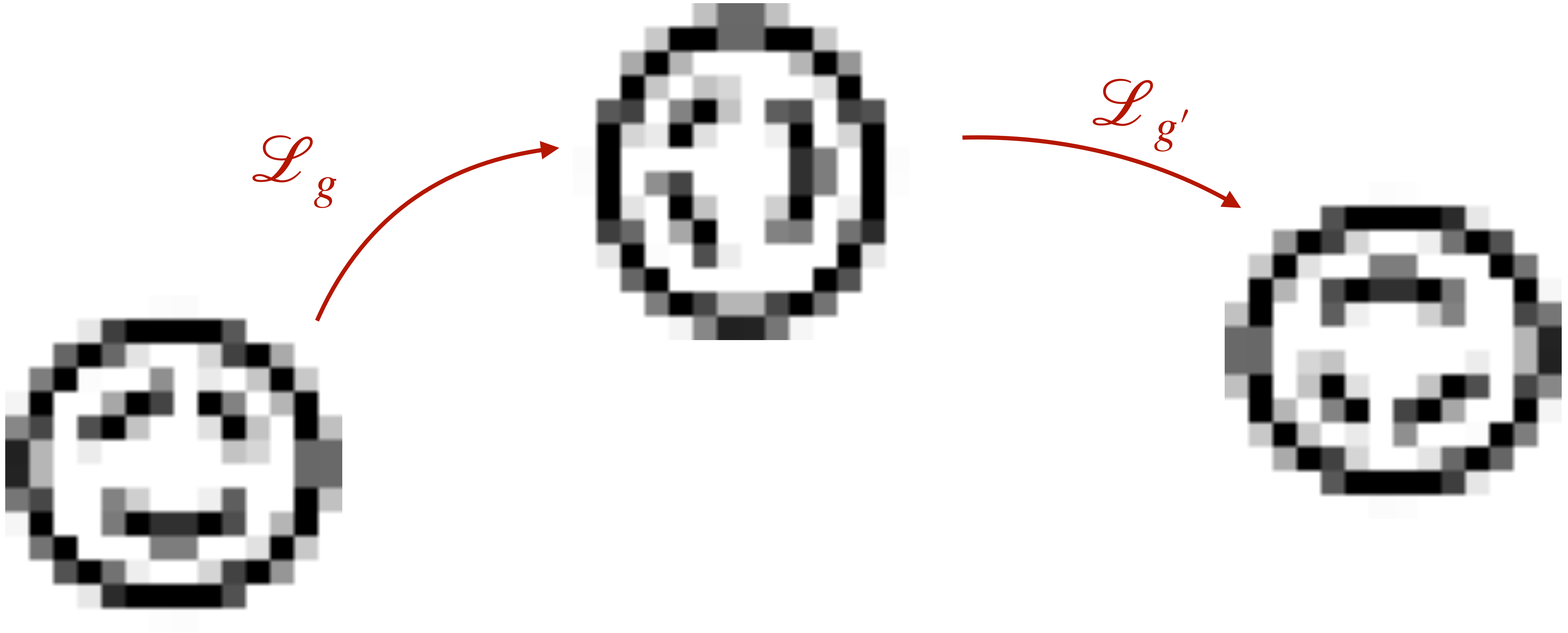
- a 2D image

$G = SE(2)$

- the roto-translation group

$\mathscr{L}_g(f)(\mathbf{y}) = f(\mathbf{R}_\theta^{-1}(\mathbf{y} - \mathbf{x}))$

- a roto-translation of the image



A left-regular representation $\mathscr{L}_g$ is a representation that transforms functions f by transforming their domains via the inverse group action

$$\mathscr{L}_g[f](x) := f(g^{-1} \cdot x)$$

“group action” equals group product when domain is G

Left-regular Representations

Example:

$f \in \mathbb{L}_2(\mathbb{R}^2)$

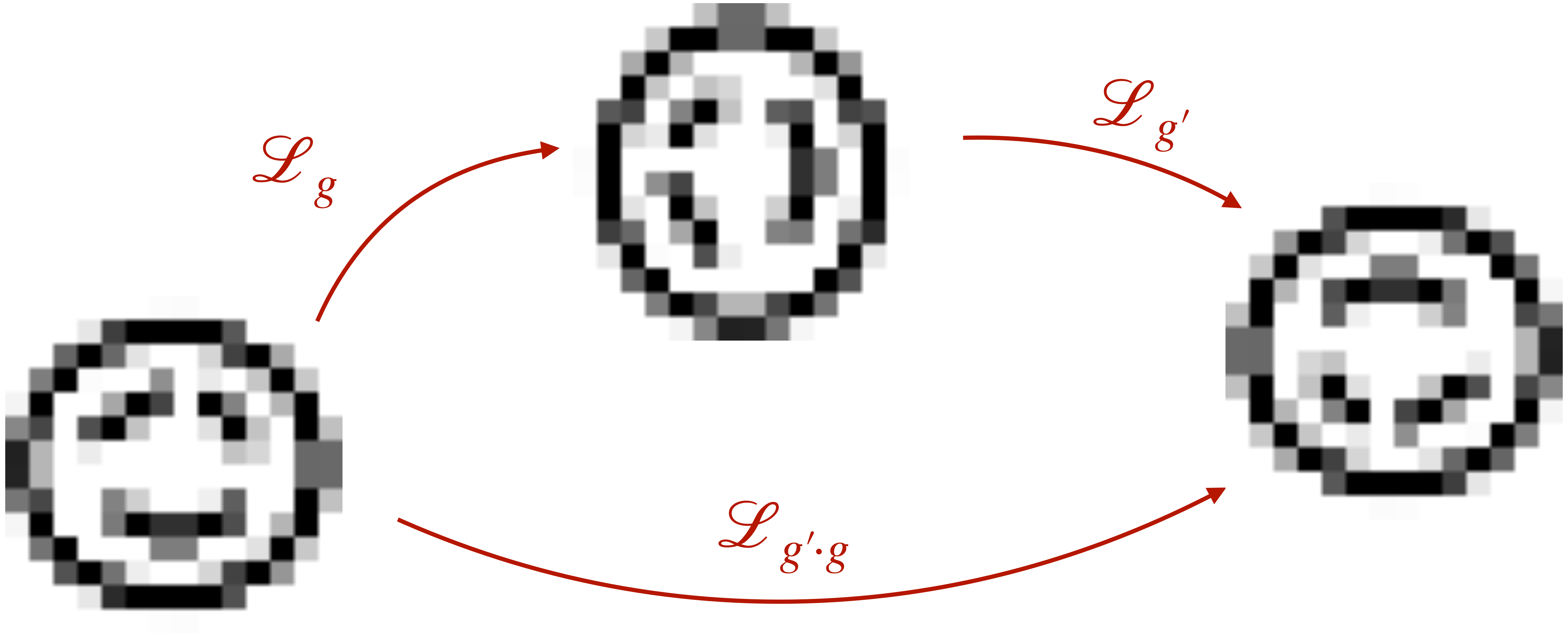
- a 2D image

$G = SE(2)$

- the roto-translation group

$\mathscr{L}_g(f)(\mathbf{y}) = f(\mathbf{R}_\theta^{-1}(\mathbf{y} - \mathbf{x}))$

- a roto-translation of the image



A left-regular representation $\mathscr{L}_g$ is a representation that transforms functions f by transforming their domains via the inverse group action

$$\mathscr{L}_g[f](x) := f(g^{-1} \cdot x)$$

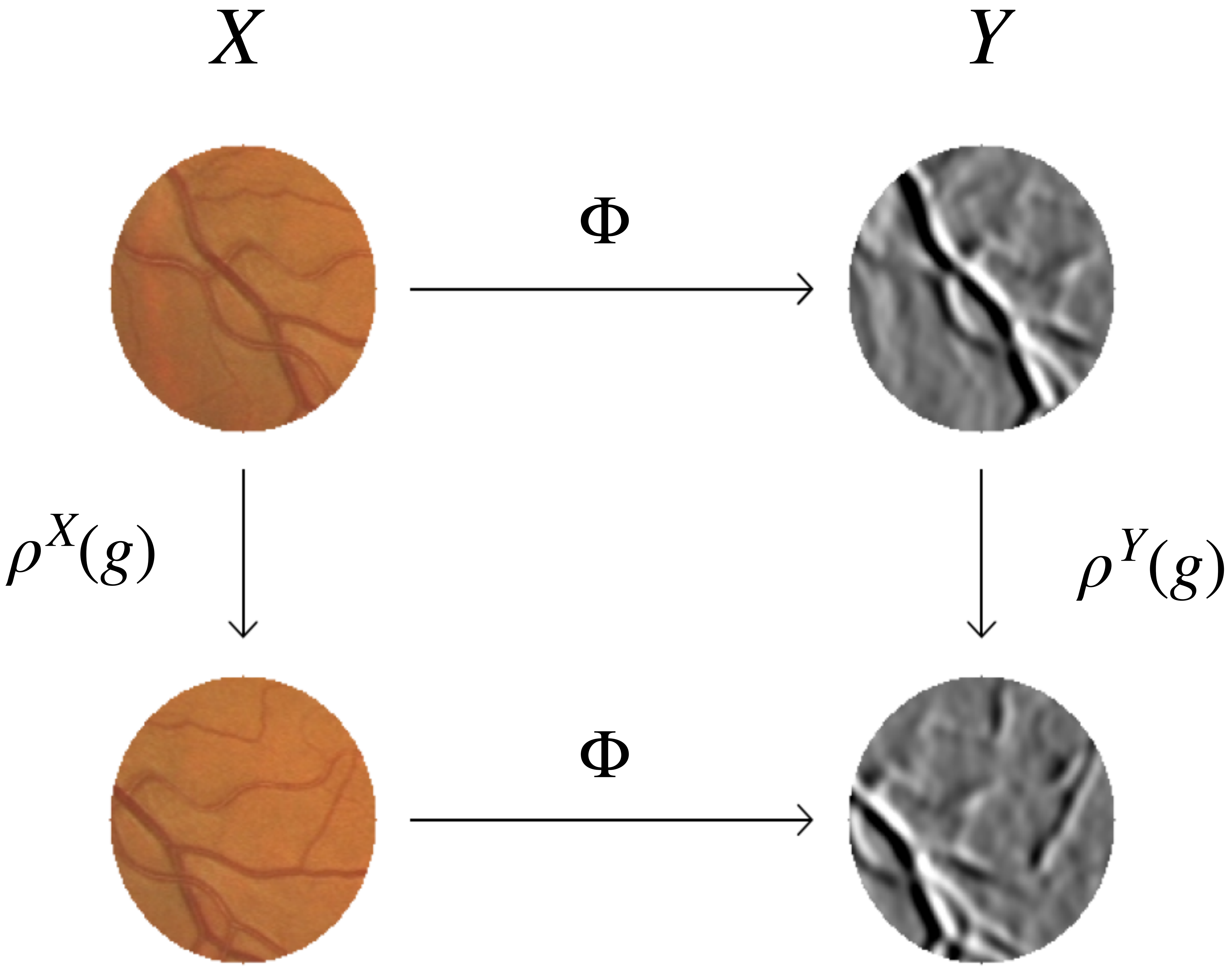
"group action" equals group product when domain is G

Equivariance

Equivariance is a property of an operator $\Phi : X \to Y$ (such as a neural network layer) by which it commutes with the group action:

$$\Phi \circ \rho^X(g) = \rho^Y(g) \circ \Phi$$

group representation action on X



1. Motivation — Equivariance → weight-sharing and generalization

2. Pattern matching using group theory — Group theory: symmetries & recognition by components

(features have “poses”)

3. Group convolutions

4. Example

5. G-convs are all you need!

6. Steerable group convolutions

7. Feature fields and escnn library

8. Equivariant tensor product layers

9. Equivariant graph NNs

1. Motivation — Equivariance → weight-sharing and generalization

2. Pattern matching using group theory — Group theory: symmetries & recognition by components
   (features have “poses”)

3. **Group convolutions**

    4. Example

    5. G-convs are all you need!

6. Steerable group convolutions

    7. Feature fields and escnn library

8. Equivariant tensor product layers

9. Equivariant graph NNs

Are convolutions with reflected conv kernels (and vice versa)

Cross-correlations

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

Are convolutions with reflected conv kernels (and vice versa)

Cross-correlations

Representation of the translation group!

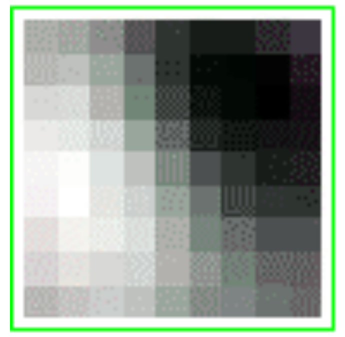
$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = \int_{\mathbb{R}^2} k(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}' = (\mathscr{L}_g\, k\,,\, f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

Are convolutions with reflected conv kernels (and vice versa)

Cross-correlations

Representation of the translation group!

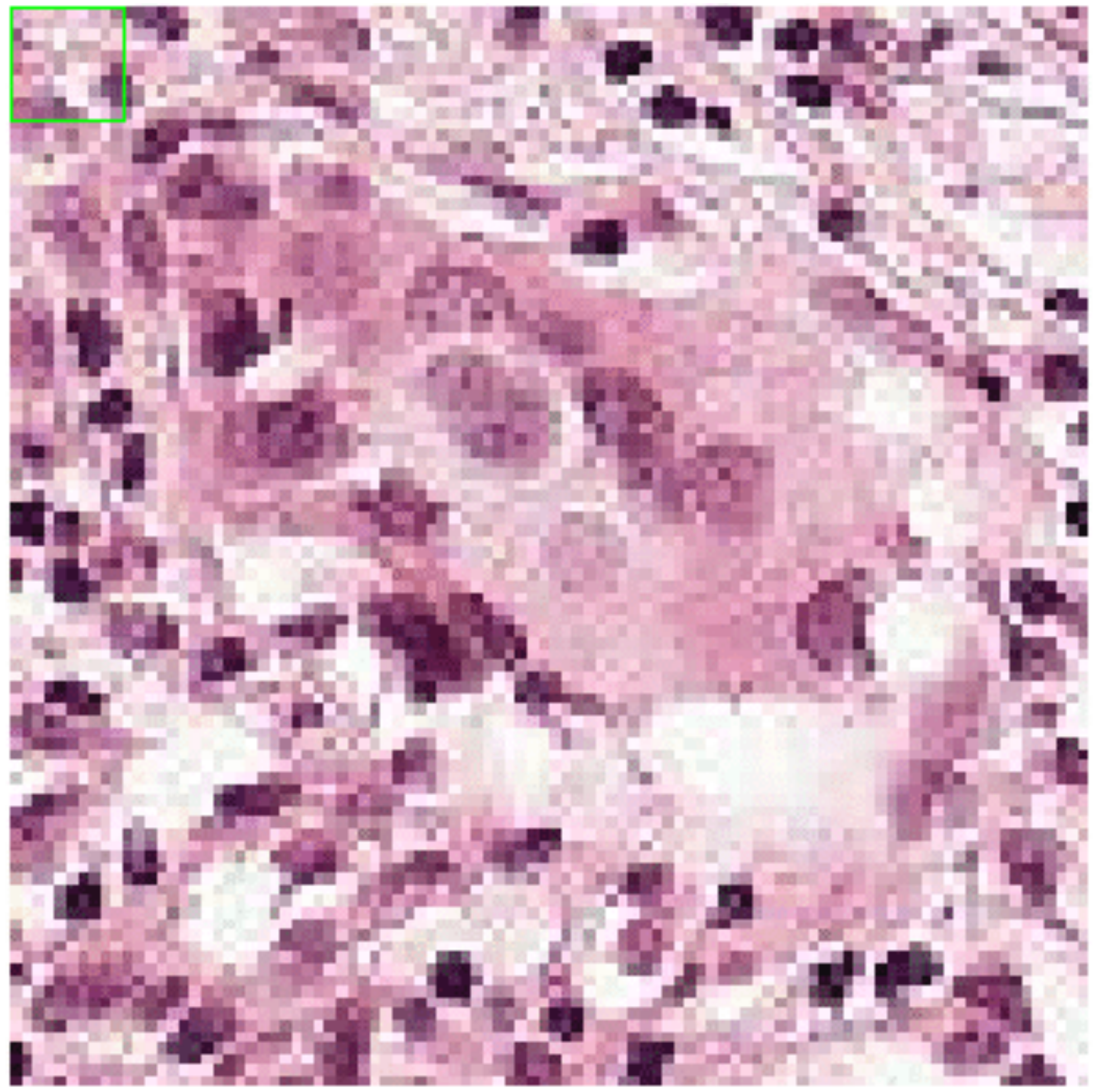
$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = \int_{\mathbb{R}^2} k(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}' = (\mathscr{L}_g \, k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$$



k

2D convolution kernel

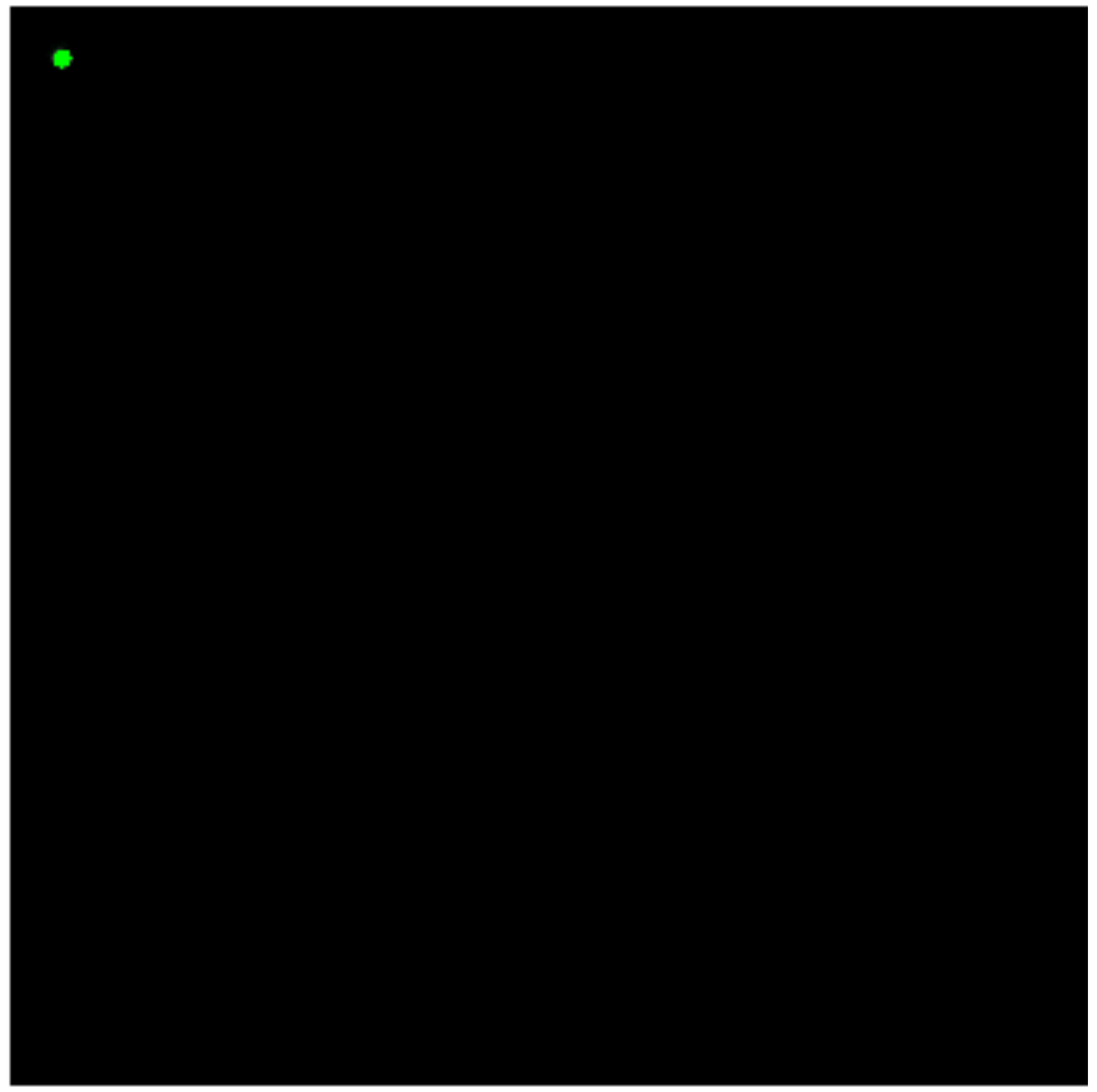
$\star_{\mathbb{R}^2}$



f^{in}

2D feature map

$=$



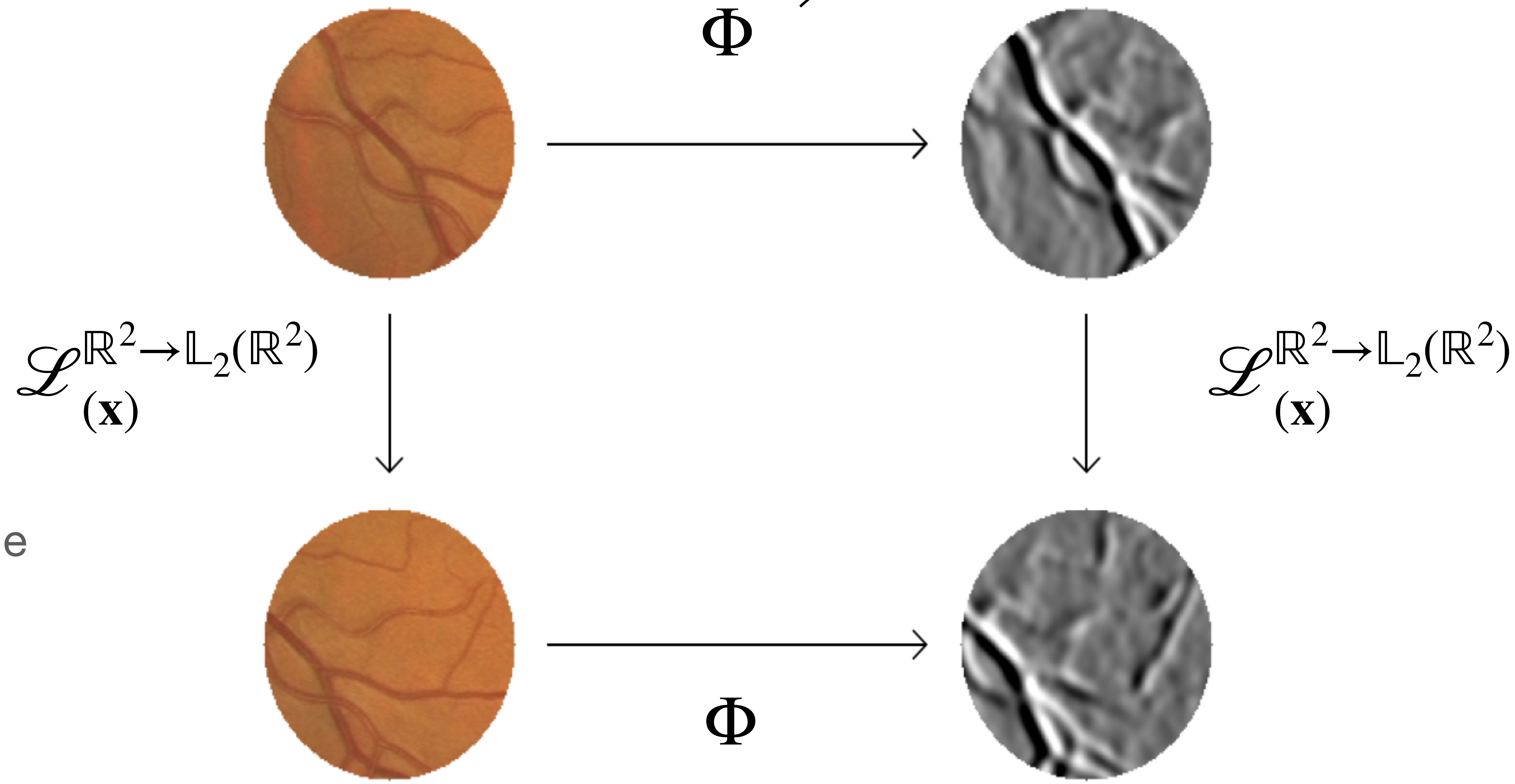
f^{out}

2D feature map (after ReLU)

Equivariance

Convolutions/cross-correlations are translation equivariant

$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = (\mathscr{L}^{\mathbb{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)}_{(\mathbf{x})} k \, , f)_{\mathbb{L}_2(\mathbb{R}^2)}$$


Φ
$\mathscr{L}^{\mathbb{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)}_{(\mathbf{x})}$
$\mathscr{L}^{\mathbb{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)}_{(\mathbf{x})}$
Φ

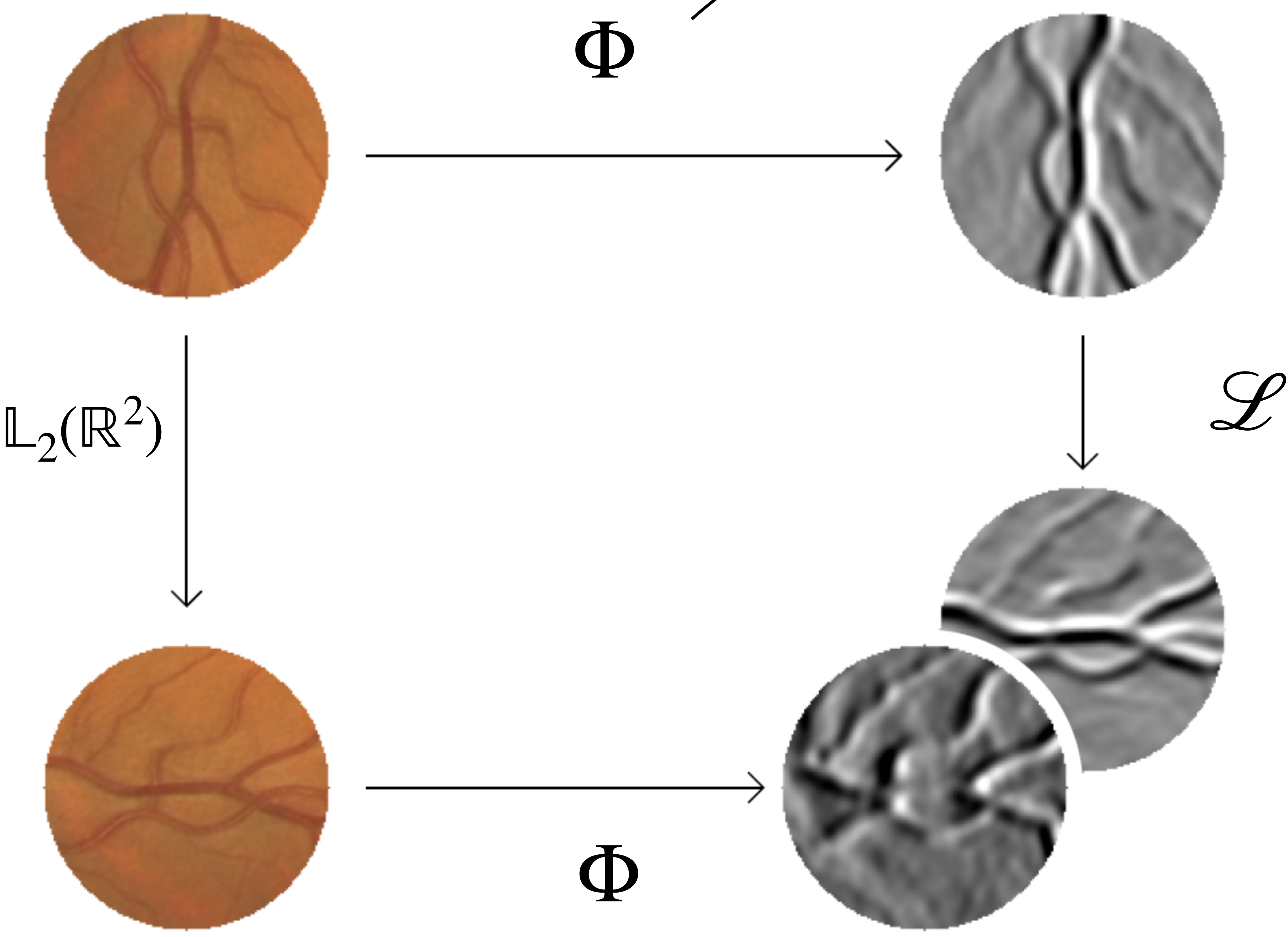
Representation of the translation group

Equivariance

Convolutions are generally **not equivariant to roto-translations**

$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = (\mathscr{L}^{\mathbb{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)}_{(\mathbf{x})}\, k \,, f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

Representation of the translation group



$$\mathscr{L}_{\theta}^{SO(2) \to \mathbb{L}_2(\mathbb{R}^2)}$$

$$\mathscr{L}_{g}^{SO(2) \to \mathbb{L}_2(\mathbb{R}^2)}$$

Representation of the rotation group

SE(2) equivariant cross-correlations

Representation of the roto-translation group!

Lifting correlations: $(k \tilde{\star} f)(\mathbf{x}, \theta) = (\mathscr{L}_g^{SE(2) \to \mathbb{L}_2(\mathbb{R}^2)}\, k\ ,\ f)_{\mathbb{L}_2(\mathbb{R}^2)}$

SE(2) equivariant cross-correlations

Representation of the roto-translation group!

Lifting correlations: $(k \tilde{\star} f)(\mathbf{x}, \theta) = (\mathscr{L}_g^{SE(2) \to \mathbb{L}_2(\mathbb{R}^2)}\, k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^2)} = (\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2 \to \mathbb{L}_2(\mathbb{R}^2)} \mathscr{L}_{\theta}^{SO(2) \to \mathbb{L}_2(\mathbb{R}^2)}\, k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^2)}$

translation *rotation*

SE(2) equivariant cross-correlations

Representation of the roto-translation group!

Lifting correlations: $(k \,\tilde{\star}\, f)(\mathbf{x}, \theta) = (\mathscr{L}_g^{SE(2)\to\mathbb{L}_2(\mathbb{R}^2)}\, k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^2)} = (\overbrace{\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(\mathbb{R}^2)}\,\mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)}\, k}^{k(\mathbf{R}_\theta^{-1}(\mathbf{x}'-\mathbf{x}))} \,,\, f)_{\mathbb{L}_2(\mathbb{R}^2)}$

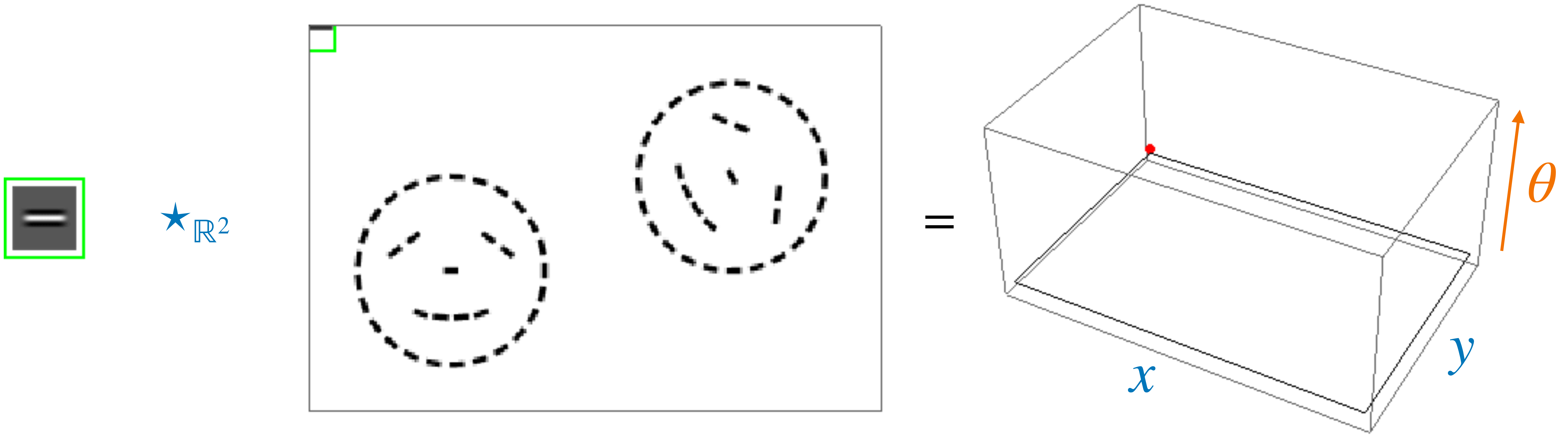
translation *rotation*

SE(2) equivariant cross-correlations

Representation of the roto-translation group!

Lifting correlations: $$(k \,\tilde{\star}\, f)(\mathbf{x},\theta) = (\mathcal{L}_g^{SE(2)\to\mathbb{L}_2(\mathbb{R}^2)}\, k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^2)} = (\overbrace{\mathcal{L}_{\mathbf{x}}^{\mathbb{R}^2\to\mathbb{L}_2(\mathbb{R}^2)}\,\mathcal{L}_{\theta}^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)}\, k}^{k(\mathbf{R}_\theta^{-1}(\mathbf{x}'-\mathbf{x}))} \,,\, f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

translation *rotation*



$\mathcal{L}_{\theta}^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)} k$
Rotated 2D convolution kernel

f^{in}
2D feature map

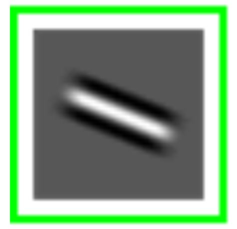
f^{out}
3D (*SE(2)*) feature map (after ReLU)

SE(2) equivariant cross-correlations

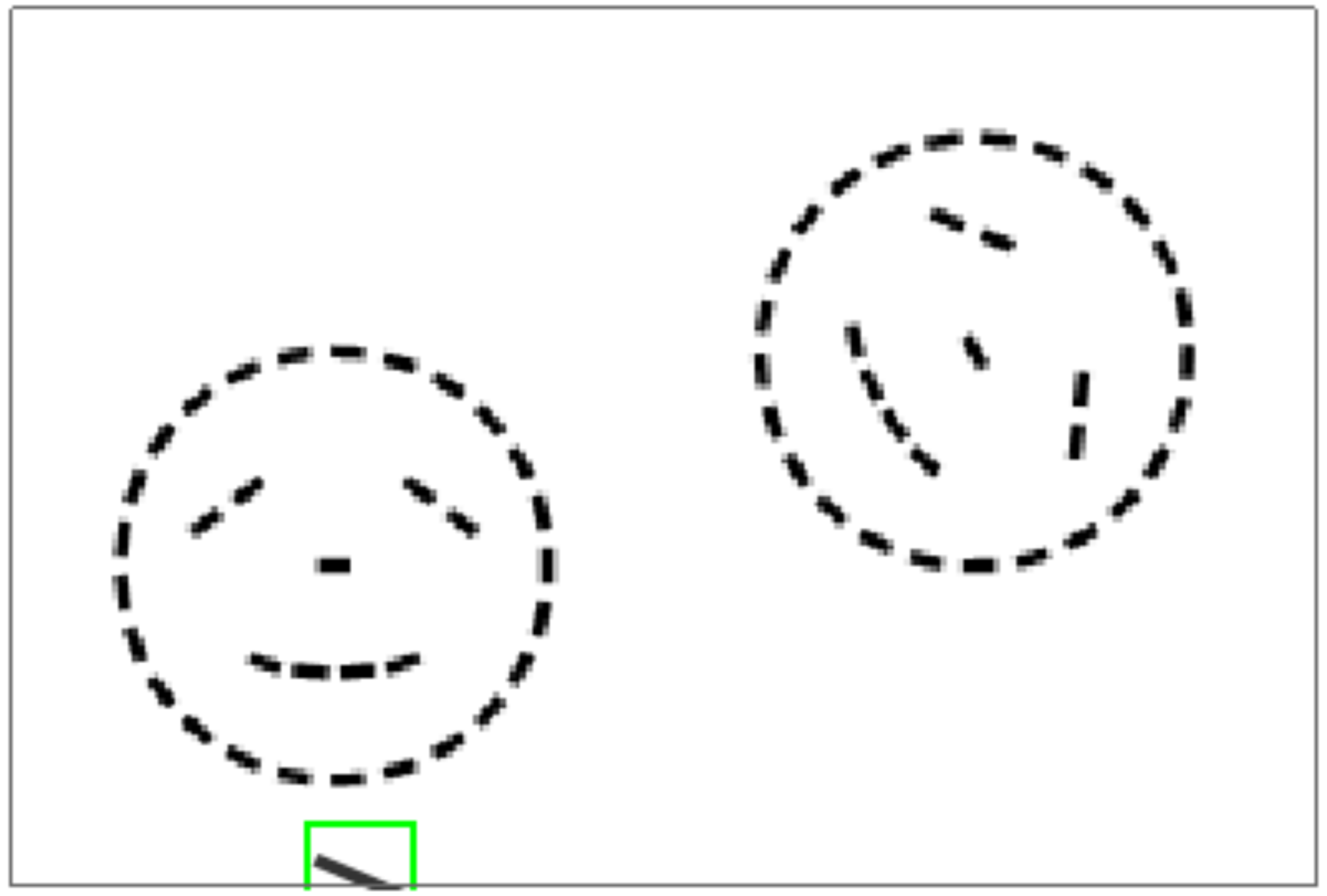
Representation of the roto-translation group!

Lifting correlations: $(k \,\tilde{\star}\, f)(\mathbf{x},\theta) = (\mathscr{L}_g^{SE(2)\to\mathbb{L}_2(\mathbb{R}^2)}\, k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^2)} = (\overbrace{\mathscr{L}_{\mathbf{x}}^{\mathbb{R}^2\to\mathbb{L}_2(\mathbb{R}^2)}\, \mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)}\, k}^{k(\mathbf{R}_\theta^{-1}(\mathbf{x}'-\mathbf{x}))} \,,\, f)_{\mathbb{L}_2(\mathbb{R}^2)}$

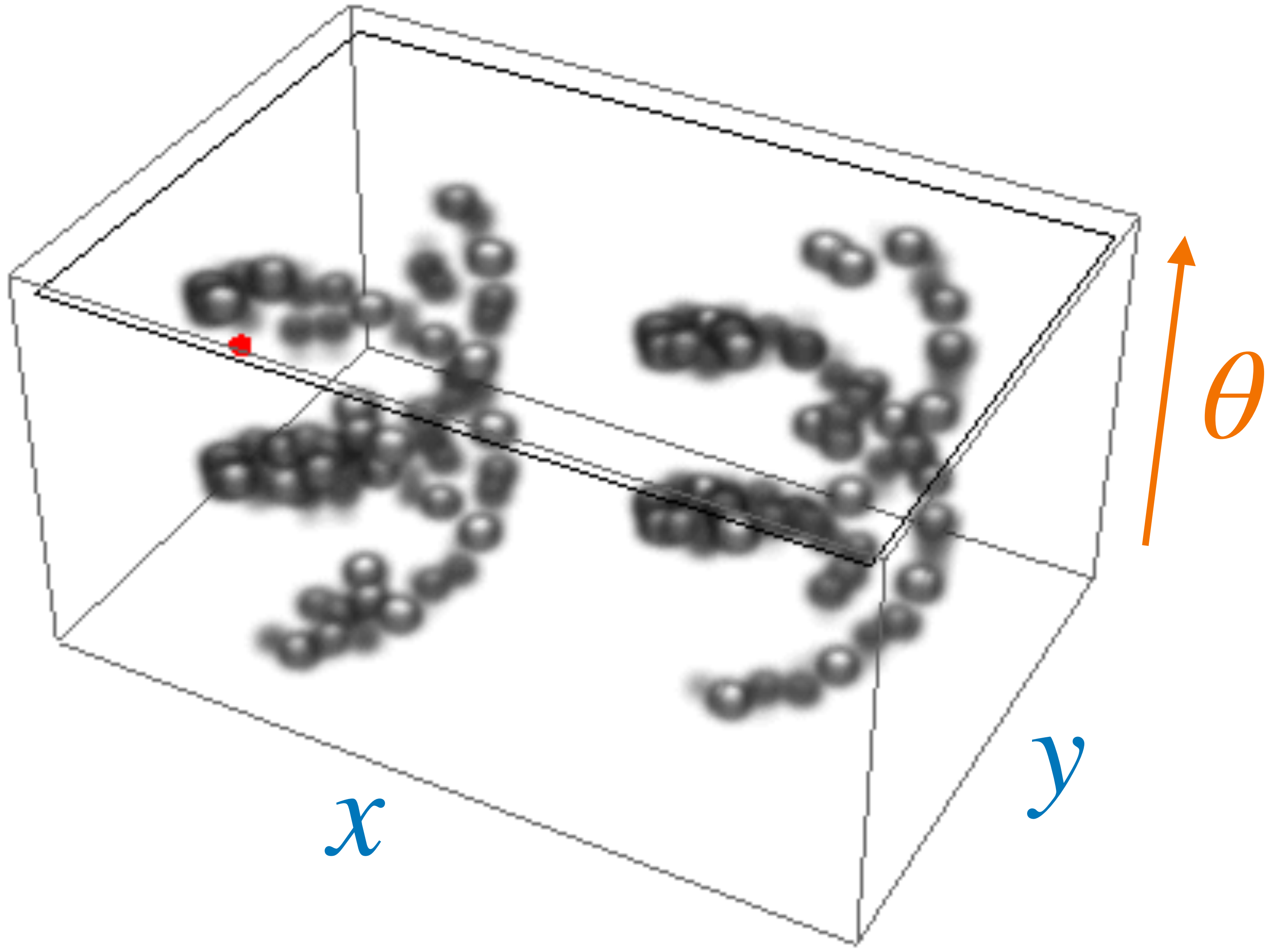
translation *rotation*



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



$=$



$\mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)} k$

Rotated 2D convolution kernel

f^{in}

2D feature map

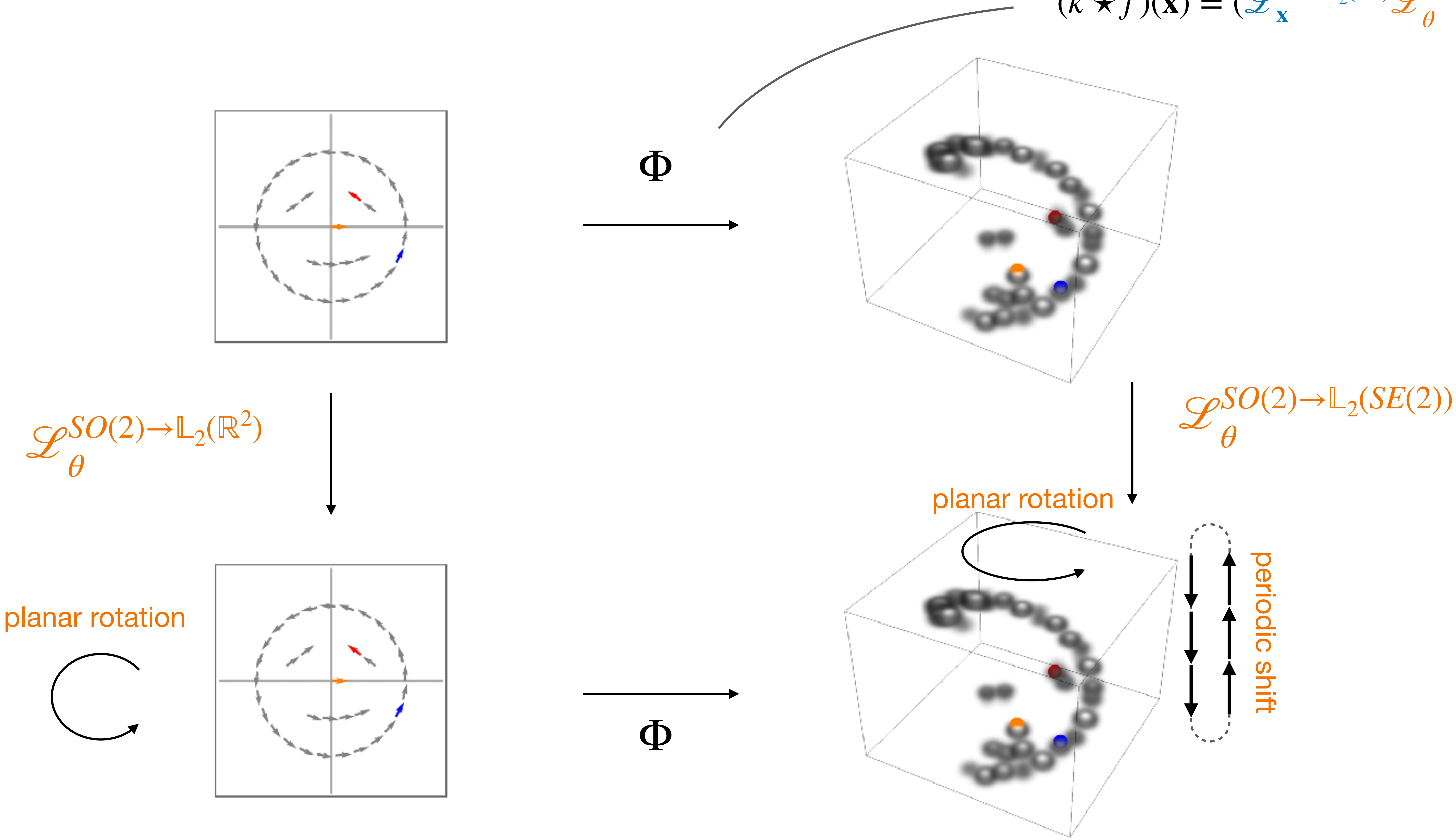
f^{out}

3D (*SE(2)*) feature map (after ReLU)

Equivariance

SE(2) group **lifting convolutions** are roto-translation equivariant

$$(k \,\tilde{\star}\, f)(\mathbf{x}) = (\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(\mathbb{R}^2)} \mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

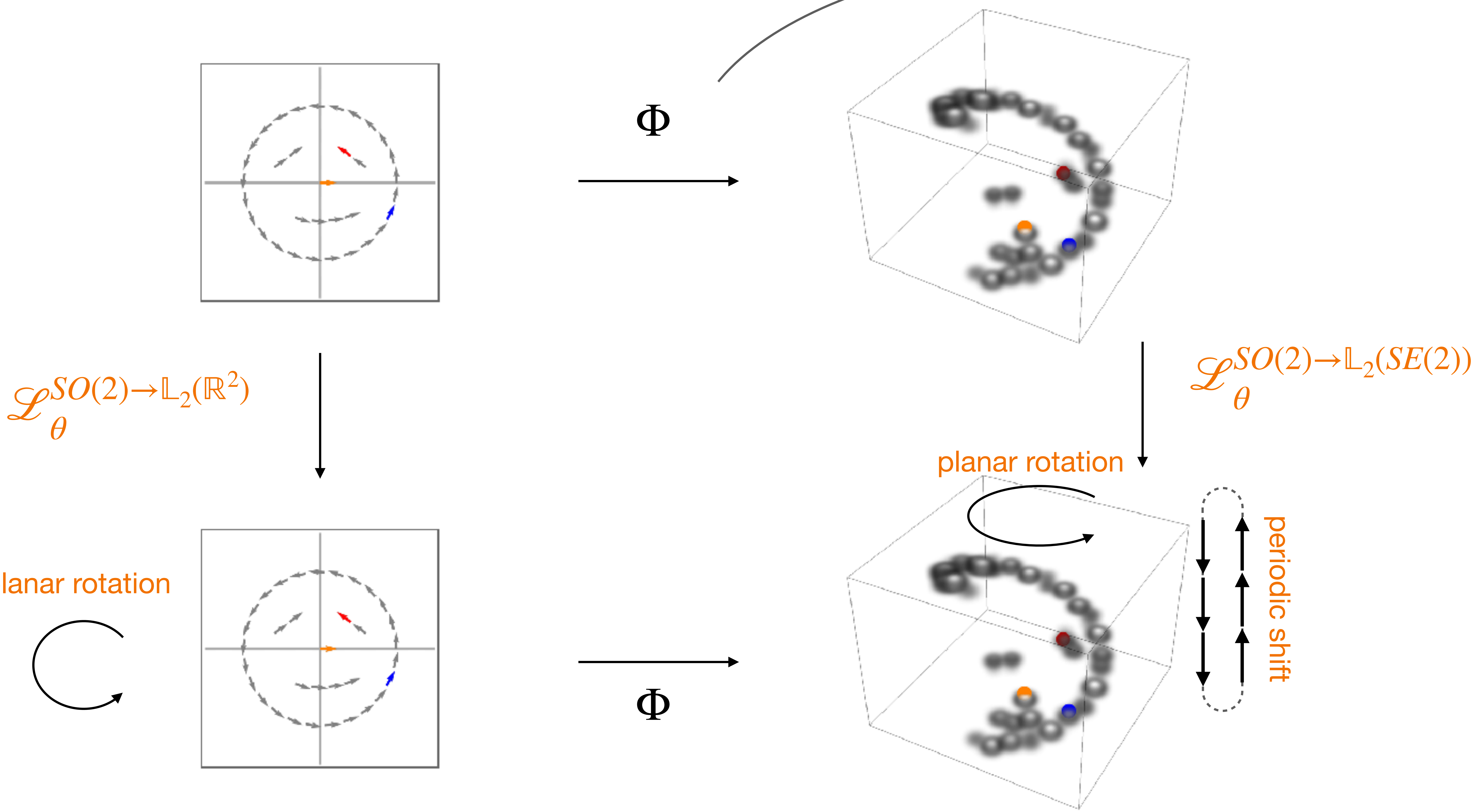


Φ
$\mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)}$
$\mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2))}$
planar rotation
planar rotation
periodic shift
Φ

Equivariance

SE(2) group **lifting convolutions** are roto-translation equivariant

$$(k \,\tilde{\star}\, f)(\mathbf{x}) = (\mathcal{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(\mathbb{R}^2)} \mathcal{L}_{\theta}^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^2)}$$



Φ

$\mathcal{L}_{\theta}^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)}$

$\mathcal{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2))}$

planar rotation

planar rotation

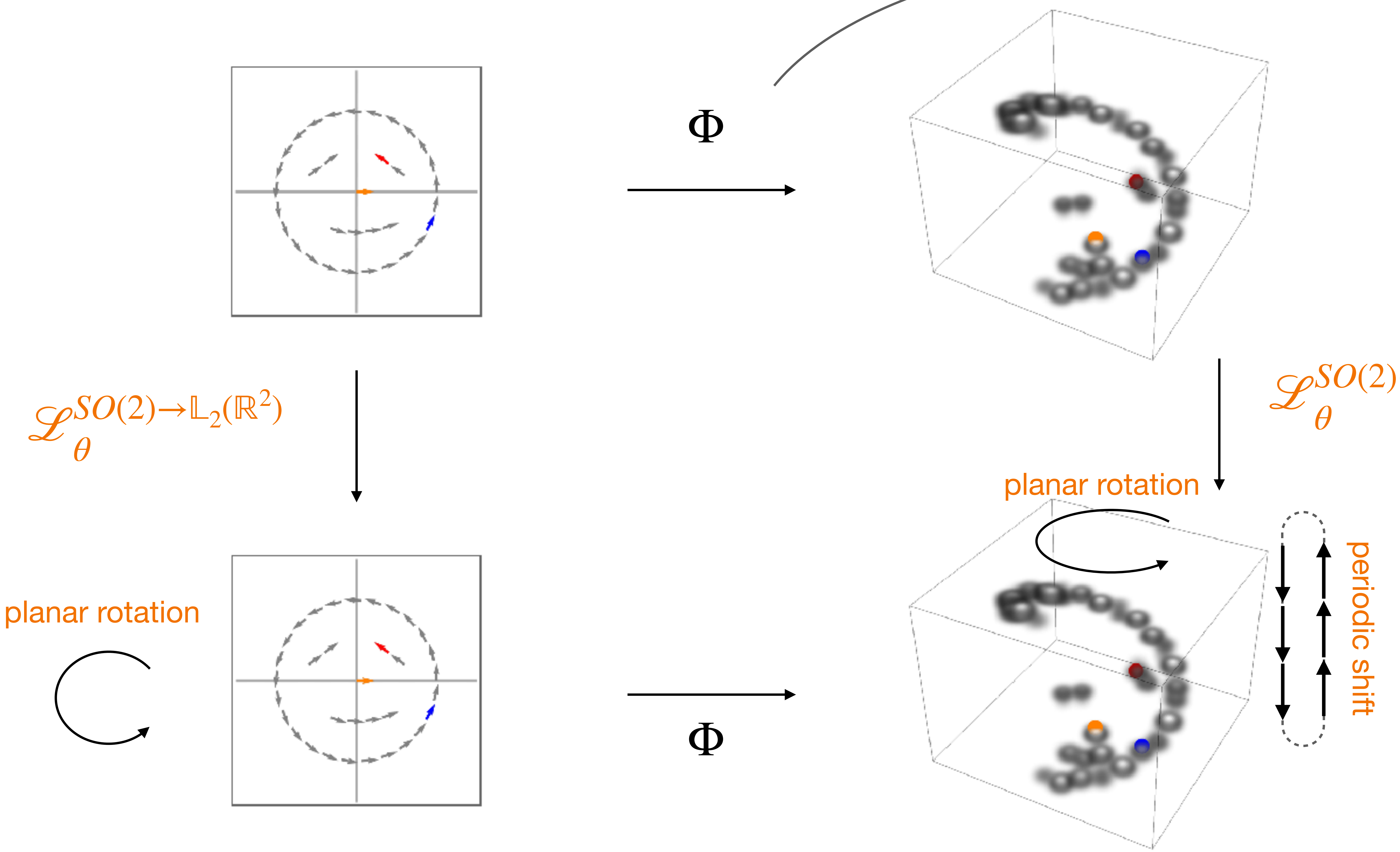
periodic shift

Φ

Equivariance

SE(2) group **lifting convolutions** are roto-translation equivariant

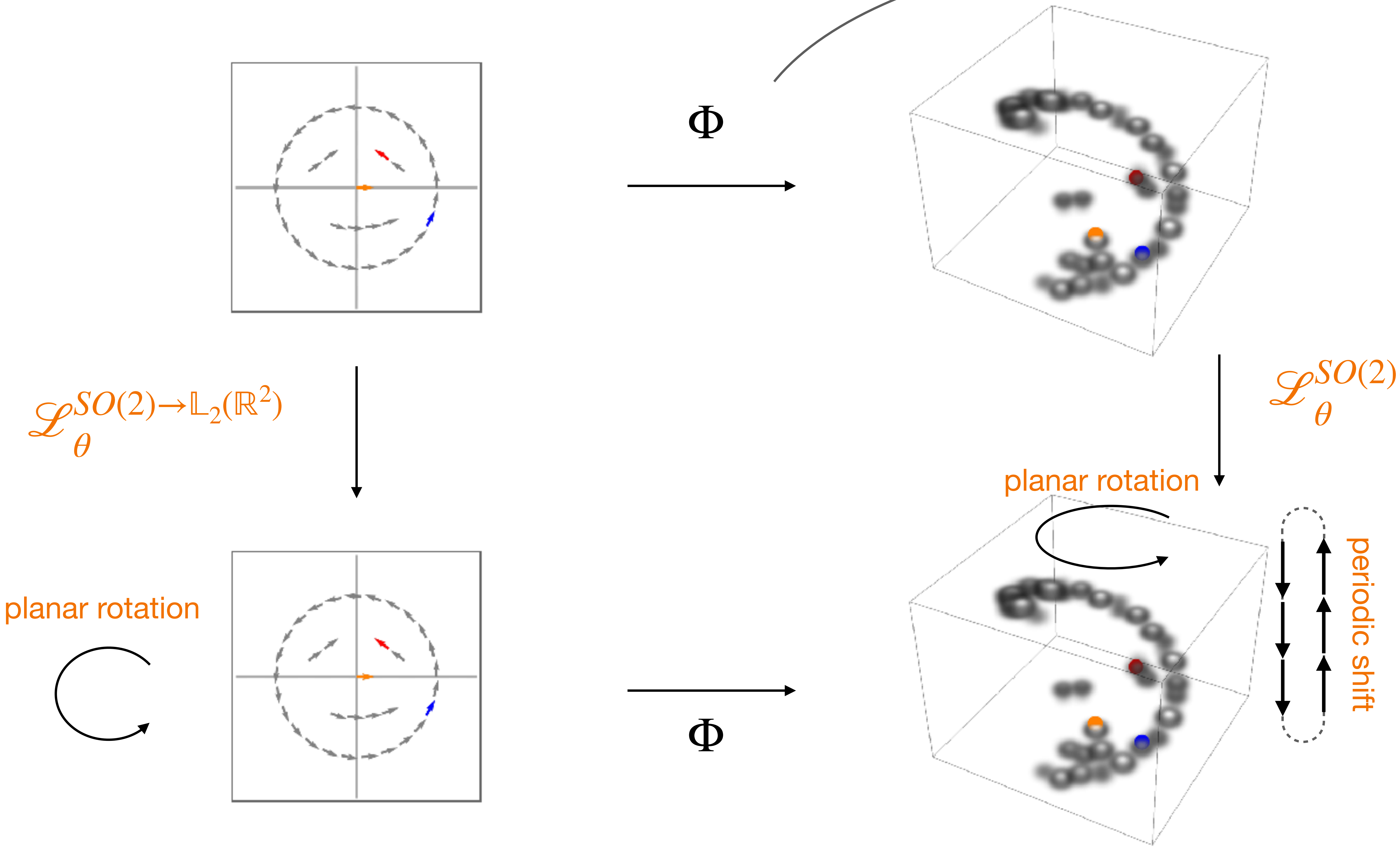
$$(k \,\tilde{\star}\, f)(\mathbf{x}) = (\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(\mathbb{R}^2)} \mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^2)}$$


Φ
$\mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)}$
$\mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2))}$
planar rotation
planar rotation
periodic shift
Φ

Equivariance

SE(2) group **lifting convolutions** are roto-translation equivariant

$$(k \,\tilde{\star}\, f)(\mathbf{x}) = (\mathcal{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(\mathbb{R}^2)} \mathcal{L}_{\theta}^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^2)}$$



Φ

$\mathcal{L}_{\theta}^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)}$

$\mathcal{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2))}$

planar rotation

planar rotation

periodic shift

Φ

What about subsequent layers?

SE(2) equivariant cross-correlations

Group correlations:

$$(k \star f)(\mathbf{x}, \theta) = (\mathscr{L}_g^{SE(2) \to \mathbb{L}_2(SE(2))} \, k \, , f)_{\mathbb{L}_2(SE(2))}$$

SE(2) equivariant cross-correlations

Group correlations:

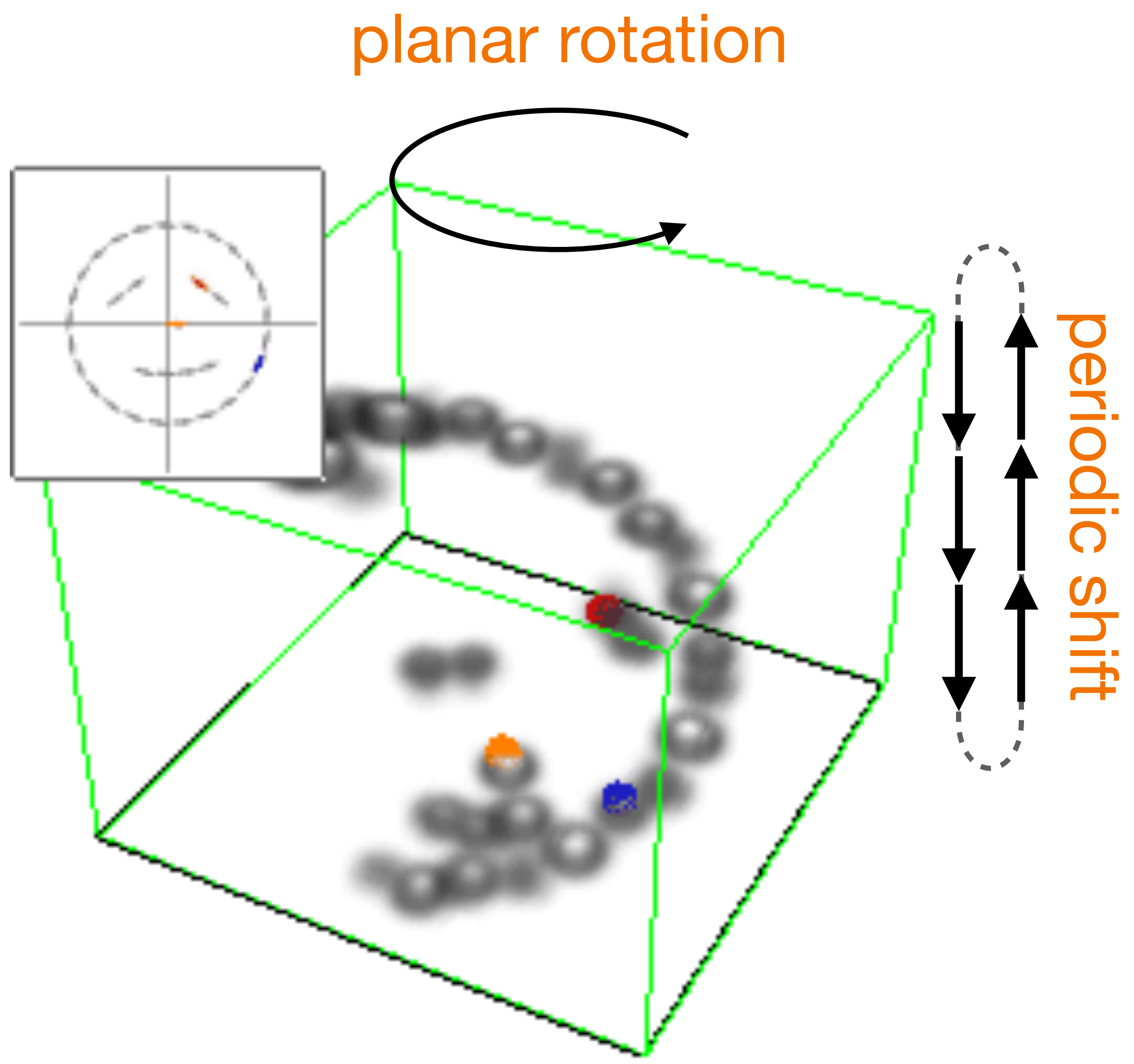
$$(k \star f)(\mathbf{x}, \theta) = (\mathscr{L}_g^{SE(2)\to\mathbb{L}_2(SE(2))}\ k\ ,\, f)_{\mathbb{L}_2(SE(2))} = (\overbrace{\underbrace{\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(SE(2))}}_{\textit{translation}}\ \underbrace{\mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2))}}_{\textit{rotation}}\ k}^{k(\mathbf{R}_\theta^{-1}(\mathbf{x}'-\mathbf{x}),\mathbf{R}_{\theta'-\theta})}\ ,\, f)_{\mathbb{L}_2(SE(2)}$$

SE(2) equivariant cross-correlations

Group correlations:

$$(k \star f)(\mathbf{x},\theta) = (\mathscr{L}_g^{SE(2)\to\mathbb{L}_2(SE(2))}\, k\,,\, f)_{\mathbb{L}_2(SE(2))} = (\overbrace{\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(SE(2))}\,\mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2))}\, k}^{k(\mathbf{R}_\theta^{-1}(\mathbf{x}'-\mathbf{x}),\mathbf{R}_{\theta'-\theta})}\,,\, f)_{\mathbb{L}_2(SE(2)}$$

translation *rotation*



$\mathscr{L}_\theta^{SO(2)\to\mathbb{L}_2(SE(2)} k$

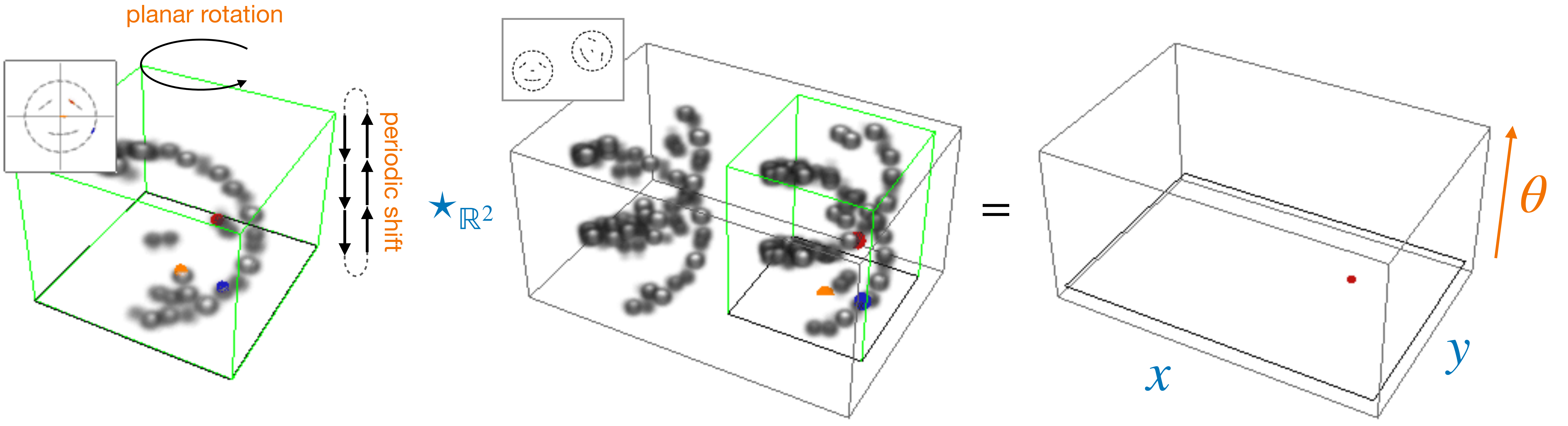
Rotated SE(2) convolution kernel

SE(2) equivariant cross-correlations

Group correlations:

$$(k \star f)(\mathbf{x},\theta) = (\mathscr{L}_g^{SE(2)\to\mathbb{L}_2(SE(2))}\, k\,,\, f)_{\mathbb{L}_2(SE(2))} = (\overbrace{\mathscr{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(SE(2))}\,\mathscr{L}_{\theta}^{SO(2)\to\mathbb{L}_2(SE(2))}\, k}^{k(\mathbf{R}_\theta^{-1}(\mathbf{x}'-\mathbf{x}),\mathbf{R}_{\theta'-\theta})}\,,\, f)_{\mathbb{L}_2(SE(2)}$$

translation *rotation*



$\mathscr{L}_\theta^{SO(2)\to\mathbb{L}_2(SE(2)}k$
Rotated SE(2) convolution kernel

f^{in}
SE(2) feature map

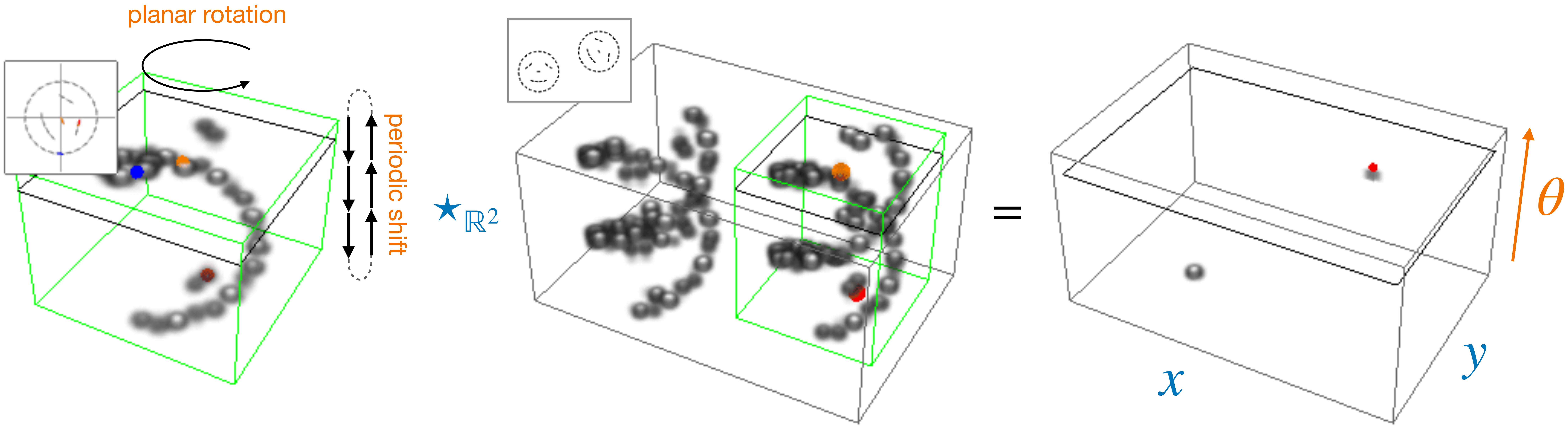
f^{out}
SE(2) feature map (after ReLU)

SE(2) equivariant cross-correlations

Group correlations:

$$(k \star f)(\mathbf{x},\theta) = (\mathcal{L}_g^{SE(2)\to\mathbb{L}_2(SE(2))}\, k\, ,\, f)_{\mathbb{L}_2(SE(2))} = (\overbrace{\mathcal{L}_{\mathbf{x}}^{\mathbf{R}^2\to\mathbb{L}_2(SE(2))}\,\mathcal{L}_\theta^{SO(2)\to\mathbb{L}_2(SE(2))}\, k}^{k(\mathbf{R}_\theta^{-1}(\mathbf{x}'-\mathbf{x}),\mathbf{R}_{\theta'-\theta})}\, ,\, f)_{\mathbb{L}_2(SE(2)}$$

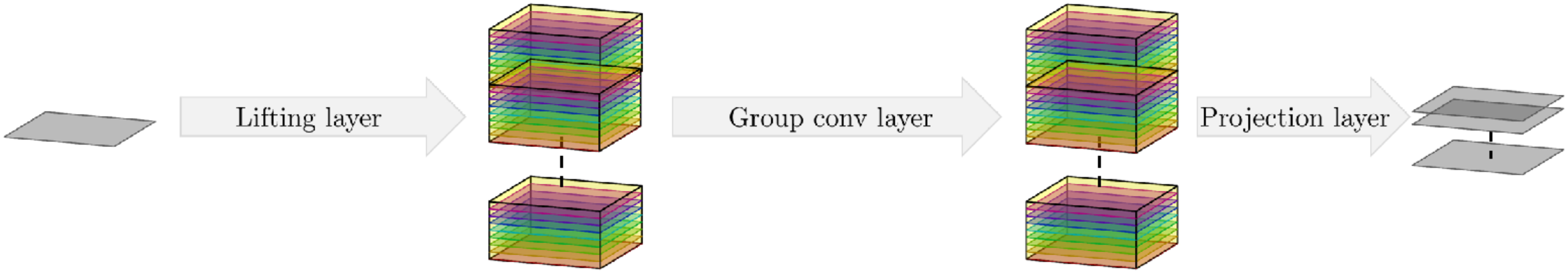
translation *rotation*

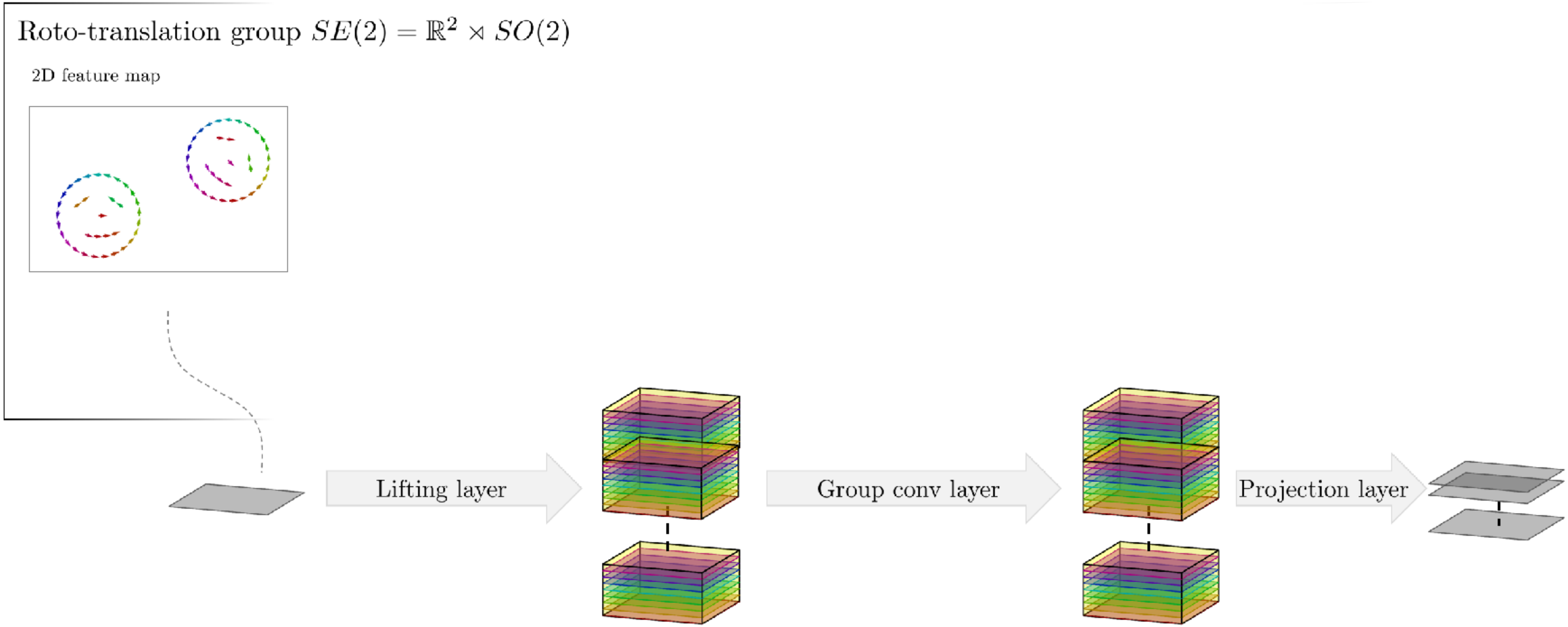


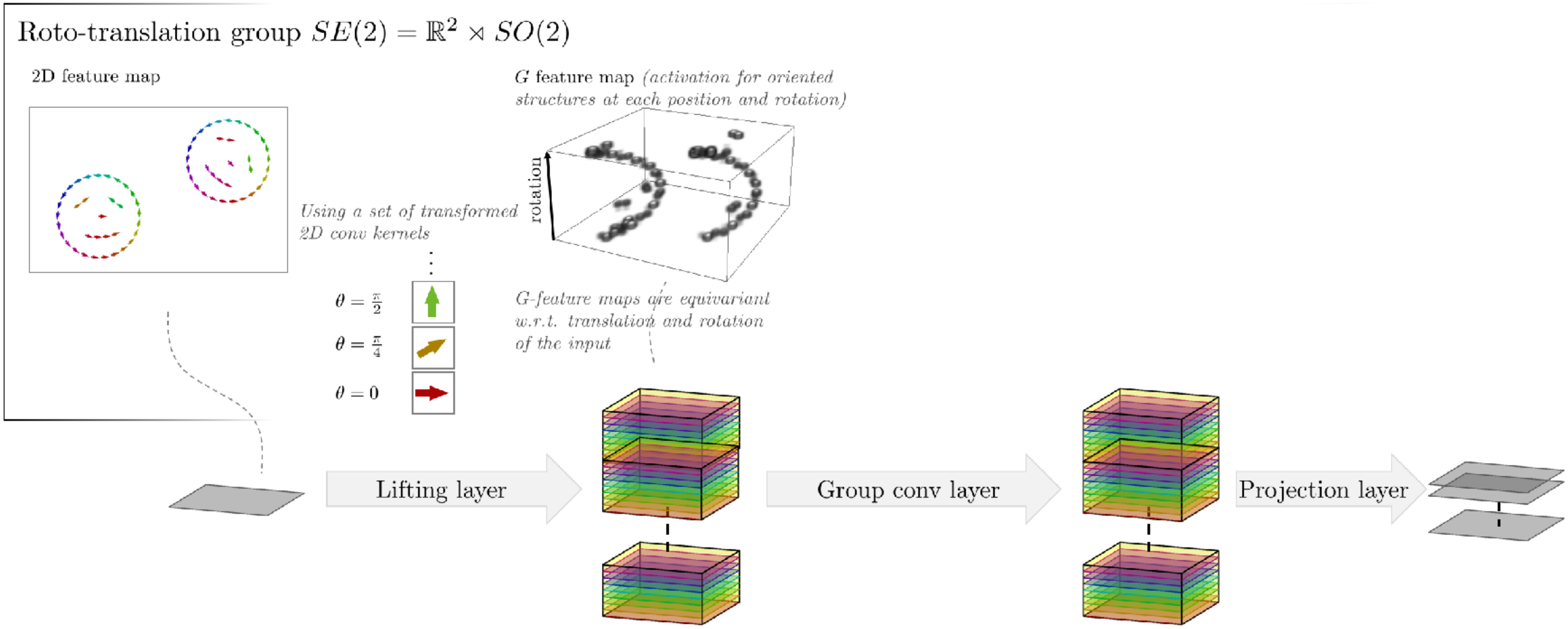
$\mathcal{L}_\theta^{SO(2)\to\mathbb{L}_2(SE(2)}k$
Rotated SE(2) convolution kernel

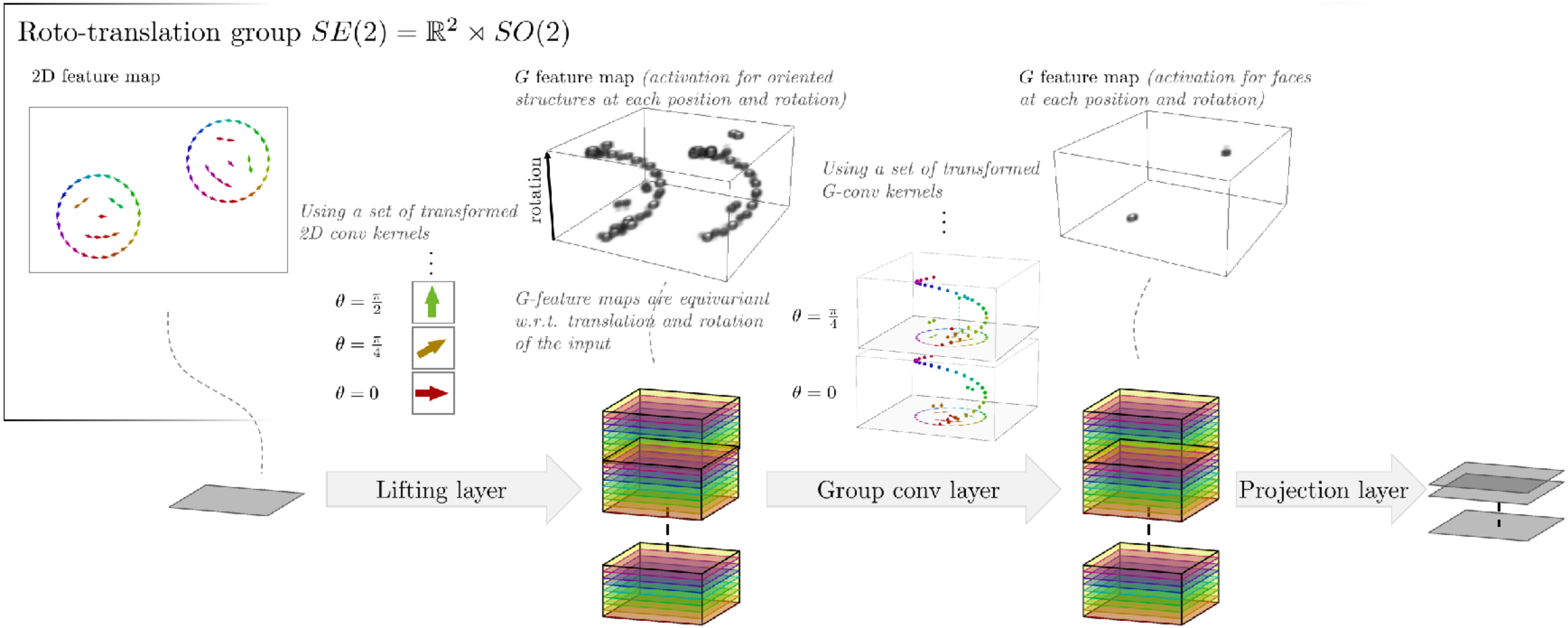
f^{in}
SE(2) feature map

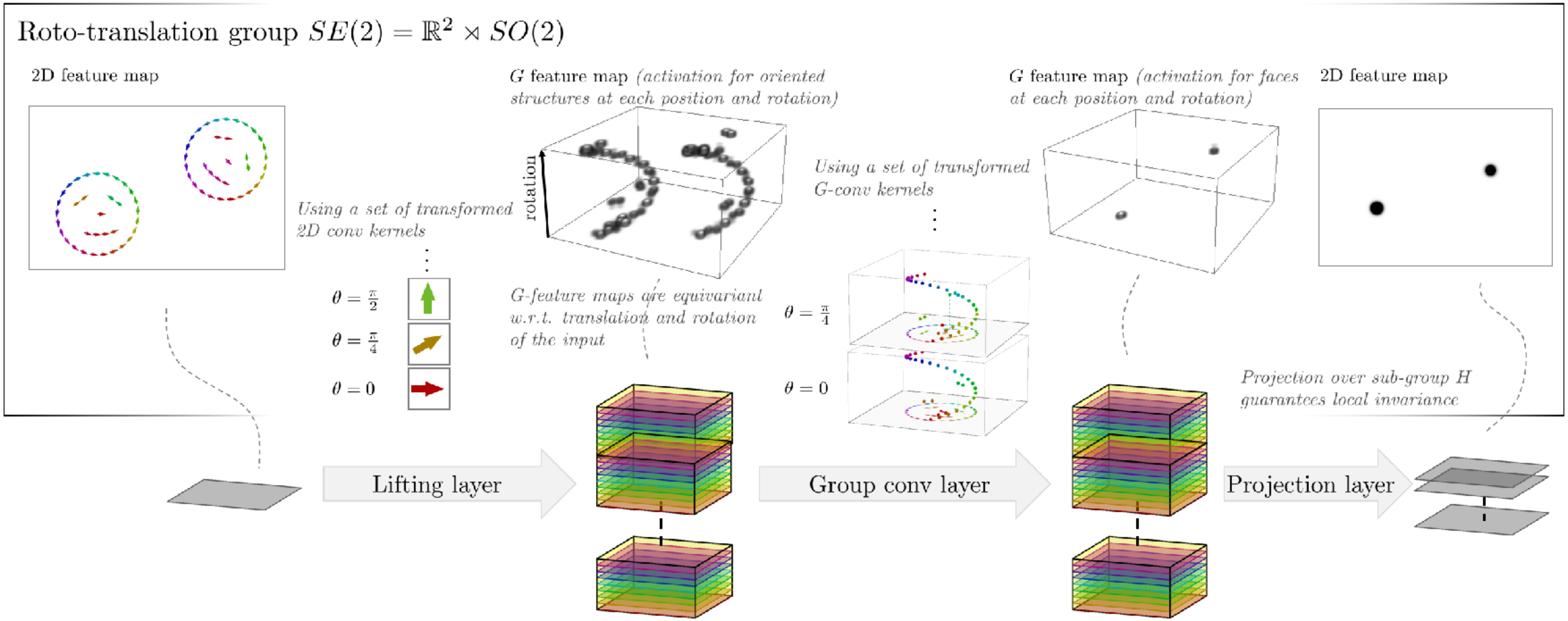
f^{out}
SE(2) feature map (after ReLU)

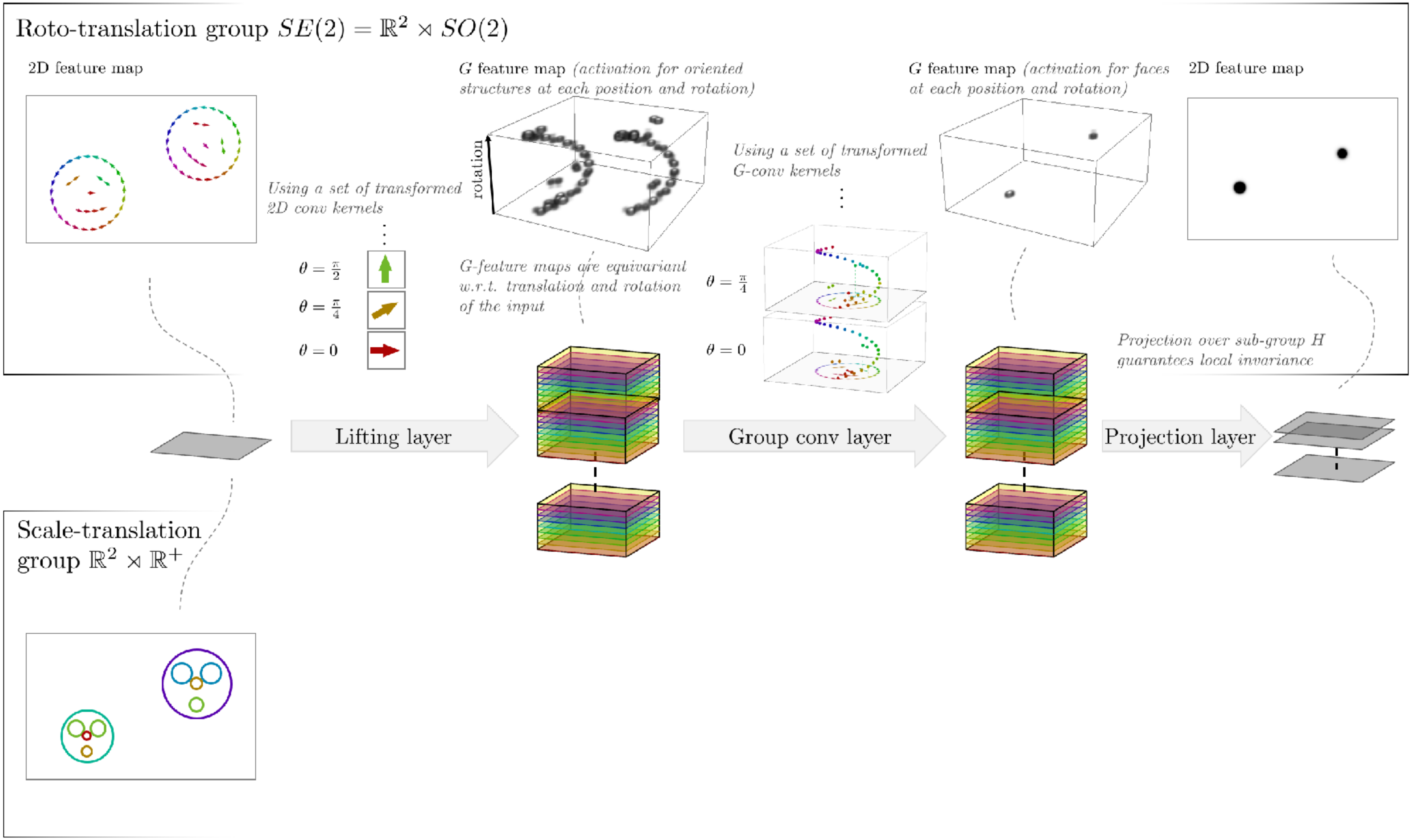

Lifting layer
Group conv layer
Projection layer

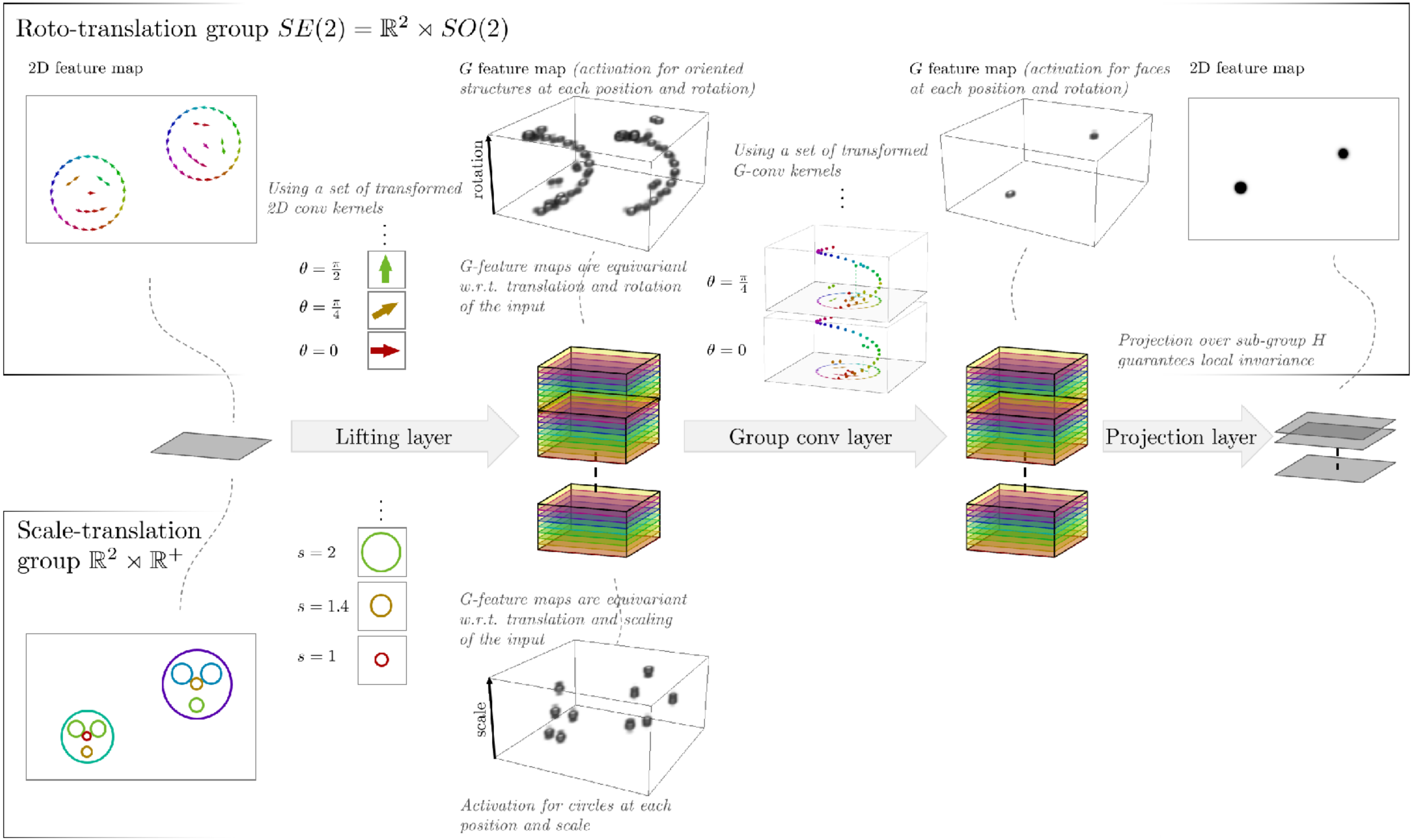

Roto-translation group $SE(2) = \mathbb{R}^2 \rtimes SO(2)$
2D feature map
Lifting layer
Group conv layer
Projection layer


Roto-translation group $SE(2) = \mathbb{R}^2 \rtimes SO(2)$
2D feature map
G feature map *(activation for oriented structures at each position and rotation)*
rotation
Using a set of transformed 2D conv kernels
$\theta = \frac{\pi}{2}$
$\theta = \frac{\pi}{4}$
$\theta = 0$
G-feature maps are equivariant w.r.t. translation and rotation of the input
Lifting layer
Group conv layer
Projection layer


Roto-translation group $SE(2) = \mathbb{R}^2 \rtimes SO(2)$
2D feature map
G feature map *(activation for oriented structures at each position and rotation)*
G feature map *(activation for faces at each position and rotation)*
rotation
Using a set of transformed 2D conv kernels
Using a set of transformed G-conv kernels
G-feature maps are equivariant w.r.t. translation and rotation of the input
$\theta = \frac{\pi}{2}$
$\theta = \frac{\pi}{4}$
$\theta = 0$
$\theta = \frac{\pi}{4}$
$\theta = 0$
Lifting layer
Group conv layer
Projection layer


Roto-translation group $SE(2) = \mathbb{R}^2 \rtimes SO(2)$
2D feature map
G feature map *(activation for oriented structures at each position and rotation)*
rotation
G feature map *(activation for faces at each position and rotation)*
2D feature map
Using a set of transformed 2D conv kernels
$\theta = \frac{\pi}{2}$
$\theta = \frac{\pi}{4}$
$\theta = 0$
G-feature maps are equivariant w.r.t. translation and rotation of the input
Using a set of transformed G-conv kernels
$\theta = \frac{\pi}{4}$
$\theta = 0$
Projection over sub-group H guarantees local invariance
Lifting layer
Group conv layer
Projection layer


Roto-translation group $SE(2) = \mathbb{R}^2 \rtimes SO(2)$
2D feature map
Using a set of transformed 2D conv kernels
$\theta = \frac{\pi}{2}$
$\theta = \frac{\pi}{4}$
$\theta = 0$
G feature map *(activation for oriented structures at each position and rotation)*
rotation
G-feature maps are equivariant w.r.t. translation and rotation of the input
Using a set of transformed G-conv kernels
$\theta = \frac{\pi}{4}$
$\theta = 0$
G feature map *(activation for faces at each position and rotation)*
2D feature map
Projection over sub-group H guarantees local invariance
Lifting layer
Group conv layer
Projection layer
Scale-translation group $\mathbb{R}^2 \rtimes \mathbb{R}^+$



Roto-translation group $SE(2) = \mathbb{R}^2 \rtimes SO(2)$

Using a set of transformed 2D conv kernels

G-feature maps are equivariant w.r.t. translation and rotation of the input

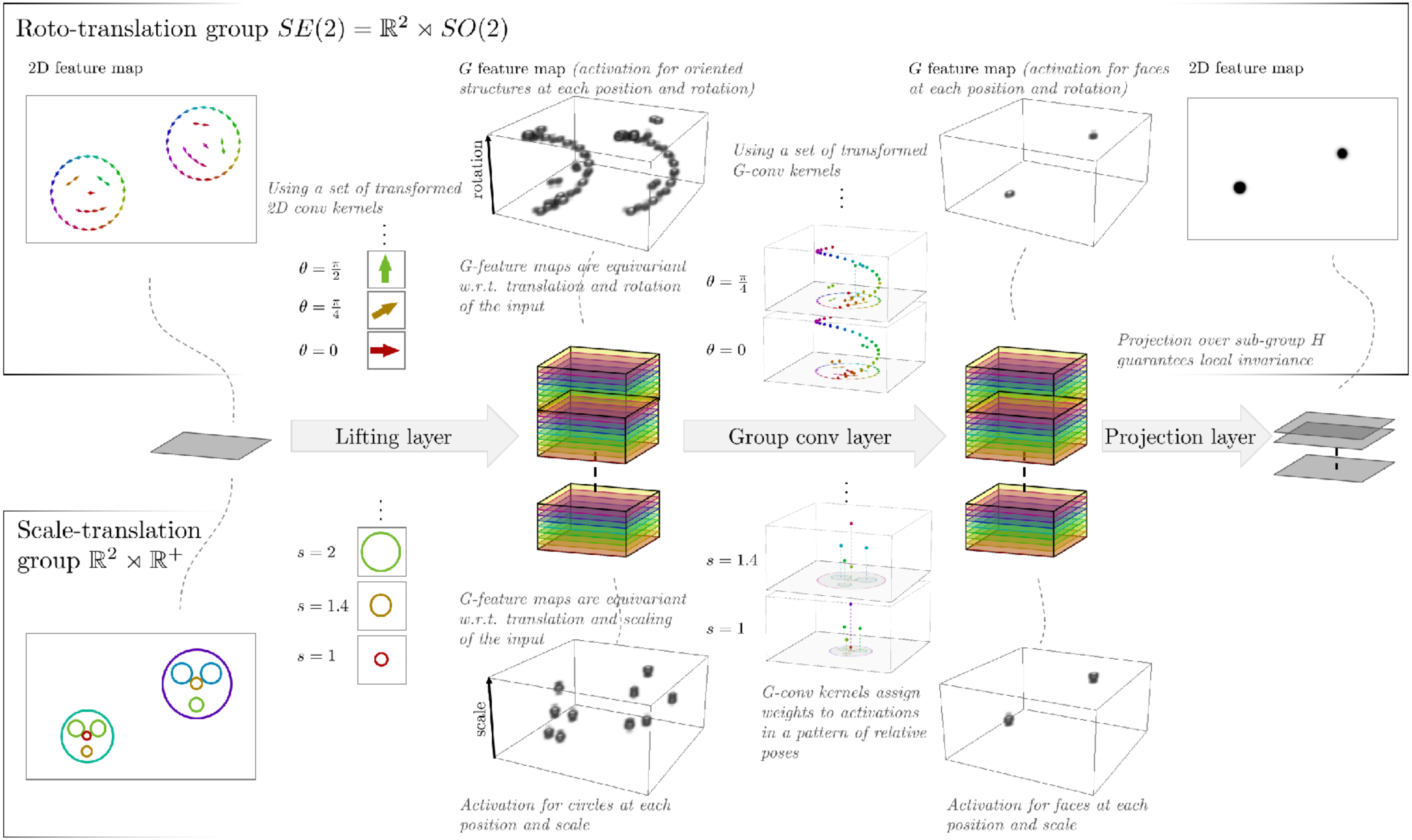
Using a set of transformed G-conv kernels

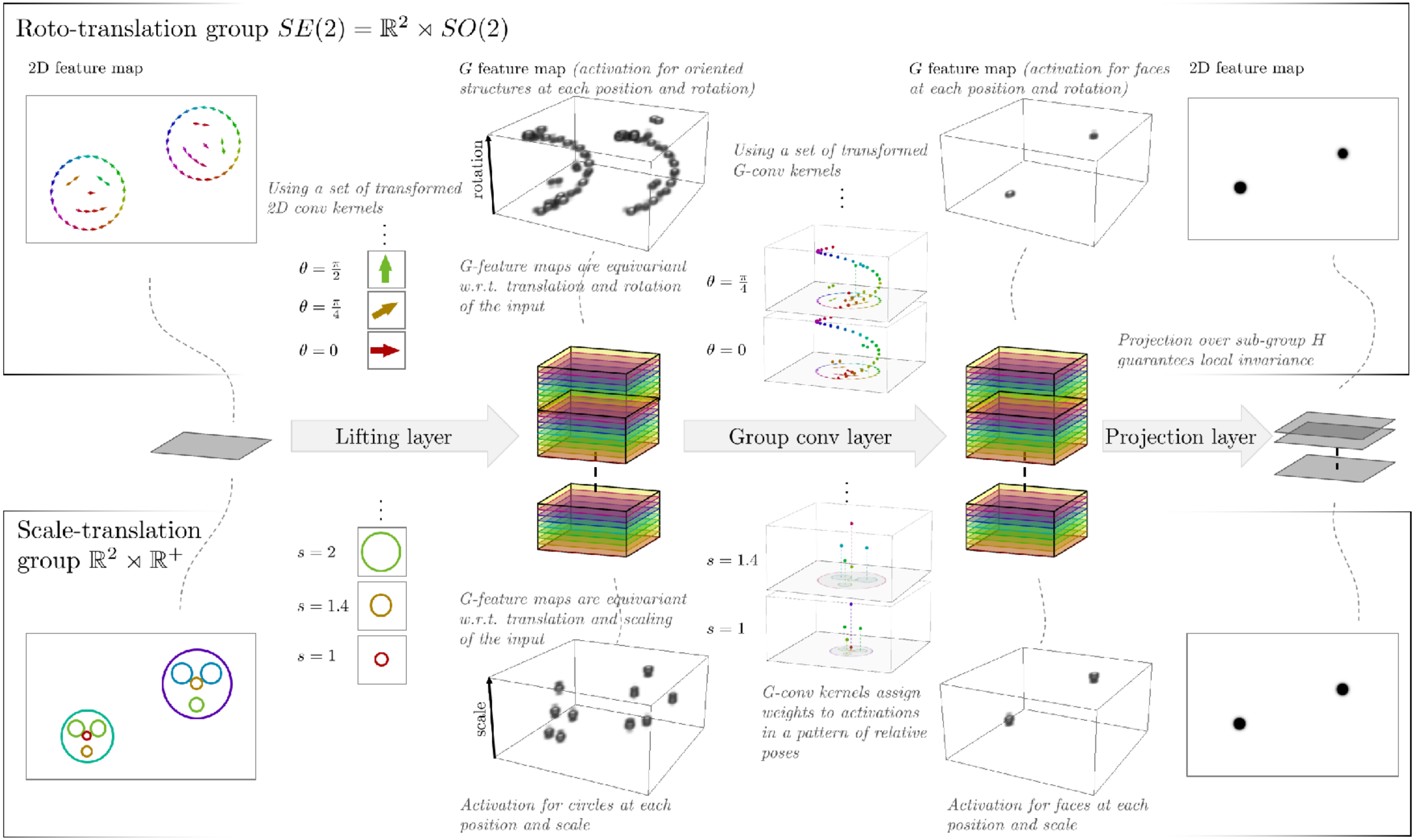
Projection over sub-group H guarantees local invariance

Scale-translation group $\mathbb{R}^2 \rtimes \mathbb{R}^+$

G-feature maps are equivariant w.r.t. translation and scaling of the input

Activation for circles at each position and scale


Roto-translation group $SE(2) = \mathbb{R}^2 \rtimes SO(2)$
2D feature map
Using a set of transformed 2D conv kernels
$\theta = \frac{\pi}{2}$
$\theta = \frac{\pi}{4}$
$\theta = 0$
G feature map *(activation for oriented structures at each position and rotation)*
rotation
G-feature maps are equivariant w.r.t. translation and rotation of the input
Using a set of transformed G-conv kernels
$\theta = \frac{\pi}{4}$
$\theta = 0$
G feature map *(activation for faces at each position and rotation)*
2D feature map
Projection over sub-group H guarantees local invariance
Lifting layer
Group conv layer
Projection layer
Scale-translation group $\mathbb{R}^2 \rtimes \mathbb{R}^+$
$s = 2$
$s = 1.4$
$s = 1$
G-feature maps are equivariant w.r.t. translation and scaling of the input
scale
Activation for circles at each position and scale
$s = 1.4$
$s = 1$
G-conv kernels assign weights to activations in a pattern of relative poses
Activation for faces at each position and scale


Roto-translation group $SE(2) = \mathbb{R}^2 \rtimes SO(2)$
2D feature map
Using a set of transformed 2D conv kernels
$\theta = \frac{\pi}{2}$
$\theta = \frac{\pi}{4}$
$\theta = 0$
G feature map (activation for oriented structures at each position and rotation)
rotation
G-feature maps are equivariant w.r.t. translation and rotation of the input
Using a set of transformed G-conv kernels
$\theta = \frac{\pi}{4}$
$\theta = 0$
G feature map (activation for faces at each position and rotation)
2D feature map
Projection over sub-group H guarantees local invariance
Lifting layer
Group conv layer
Projection layer
Scale-translation group $\mathbb{R}^2 \rtimes \mathbb{R}^+$
$s = 2$
$s = 1.4$
$s = 1$
G-feature maps are equivariant w.r.t. translation and scaling of the input
scale
Activation for circles at each position and scale
$s = 1.4$
$s = 1$
G-conv kernels assign weights to activations in a pattern of relative poses
Activation for faces at each position and scale

1. Motivation	Equivariance → weight-sharing and generalization
2. Pattern matching using group theory	Group theory: symmetries & recognition by components (features have “poses”)
3. Group convolutions	Template matching over groups
4. Example	
5. G-convs are all you need!	
6. Steerable group convolutions	
7. Feature fields and escnn library	
8. Equivariant tensor product layers	
9. Equivariant graph NNs	

1. Motivation — Equivariance → weight-sharing and generalization

2. Pattern matching using group theory — Group theory: symmetries & recognition by components
(features have “poses”)

3. Group convolutions — Template matching over groups

4. Example

5. G-convs are all you need!

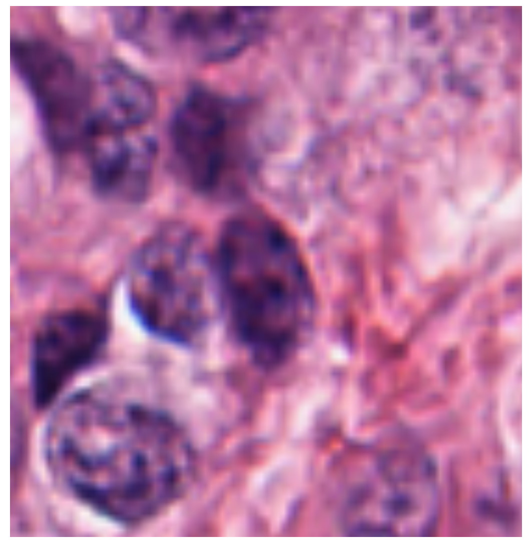
6. Steerable group convolutions

7. Feature fields and escnn library

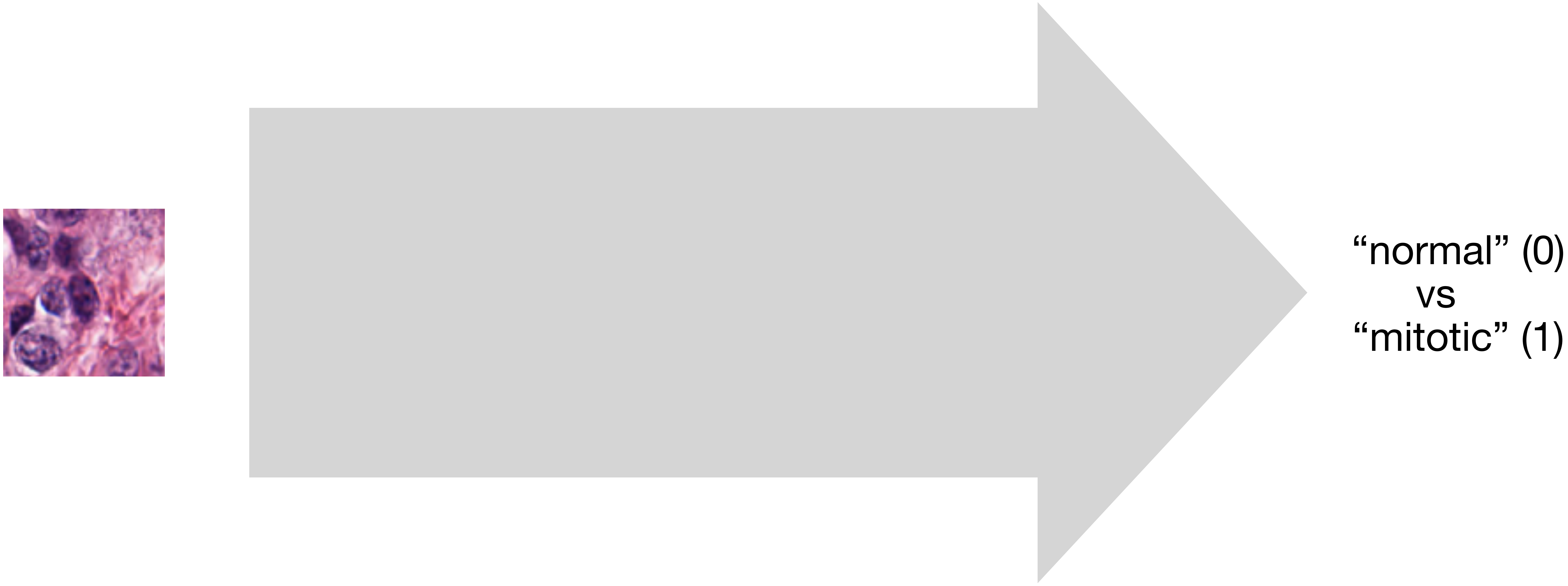
8. Equivariant tensor product layers

9. Equivariant graph NNs

Architecture for rotation invariant mitotic cell detection

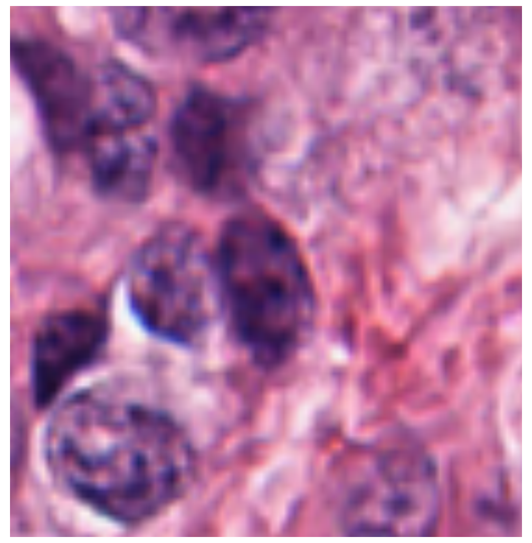


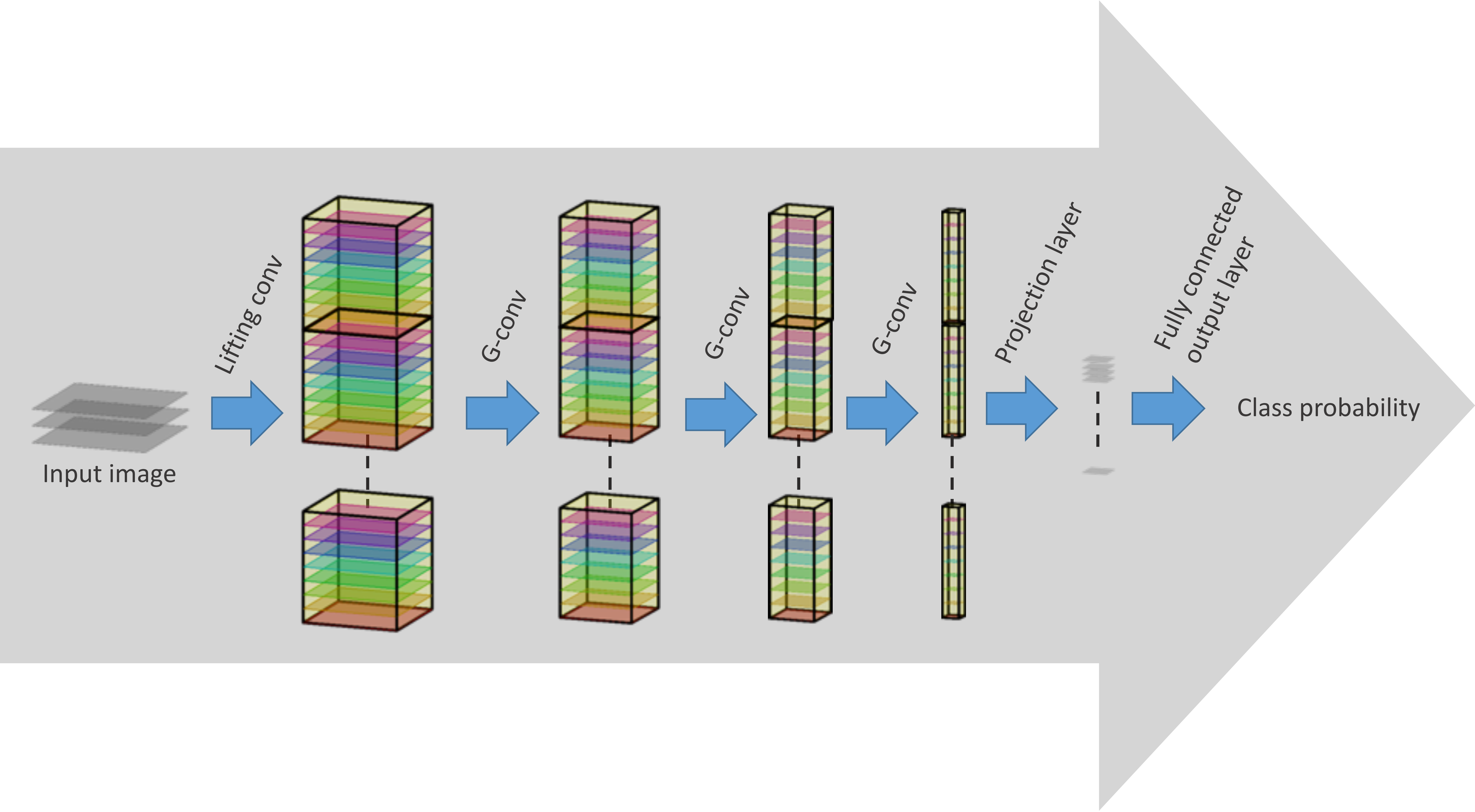
Architecture for rotation invariant mitotic cell detection



“normal” (0)
vs
“mitotic” (1)

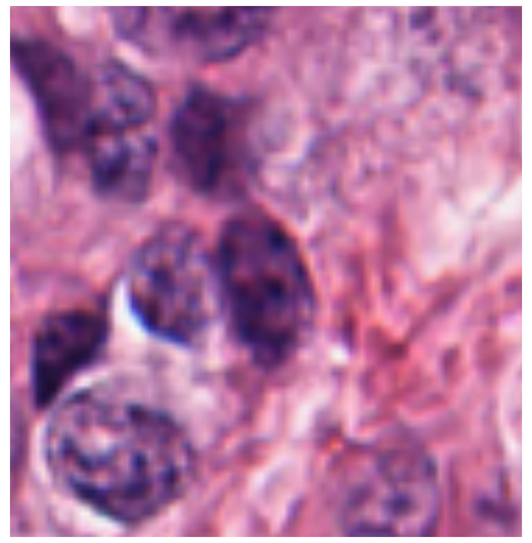
Architecture for rotation invariant mitotic cell detection

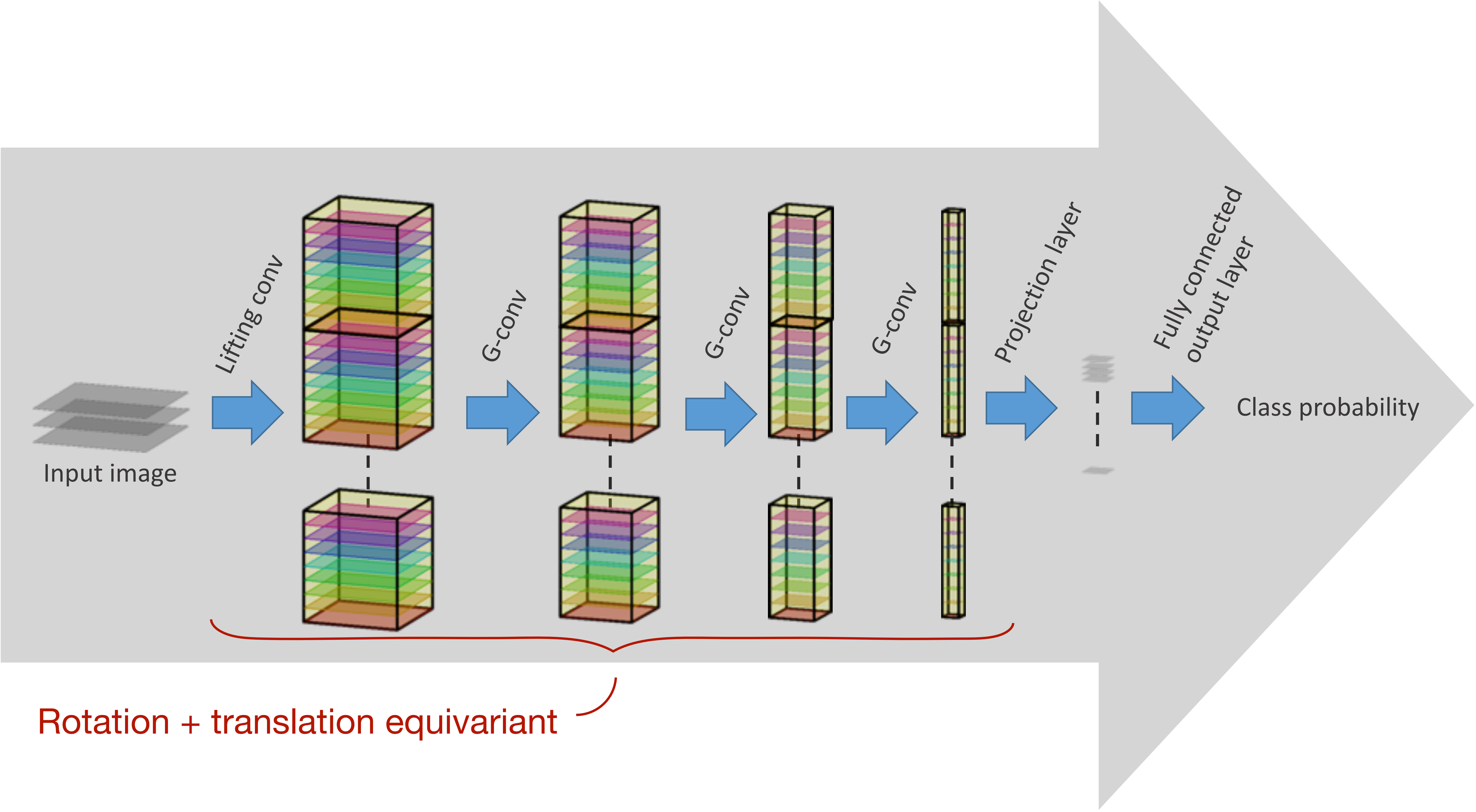




“normal” (0)
vs
“mitotic” (1)

Architecture for rotation invariant mitotic cell detection

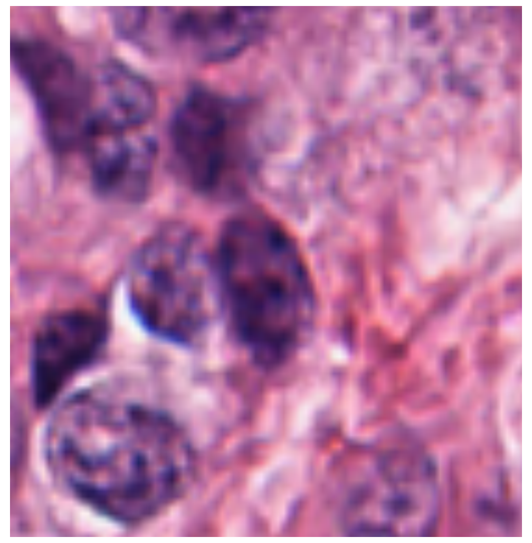


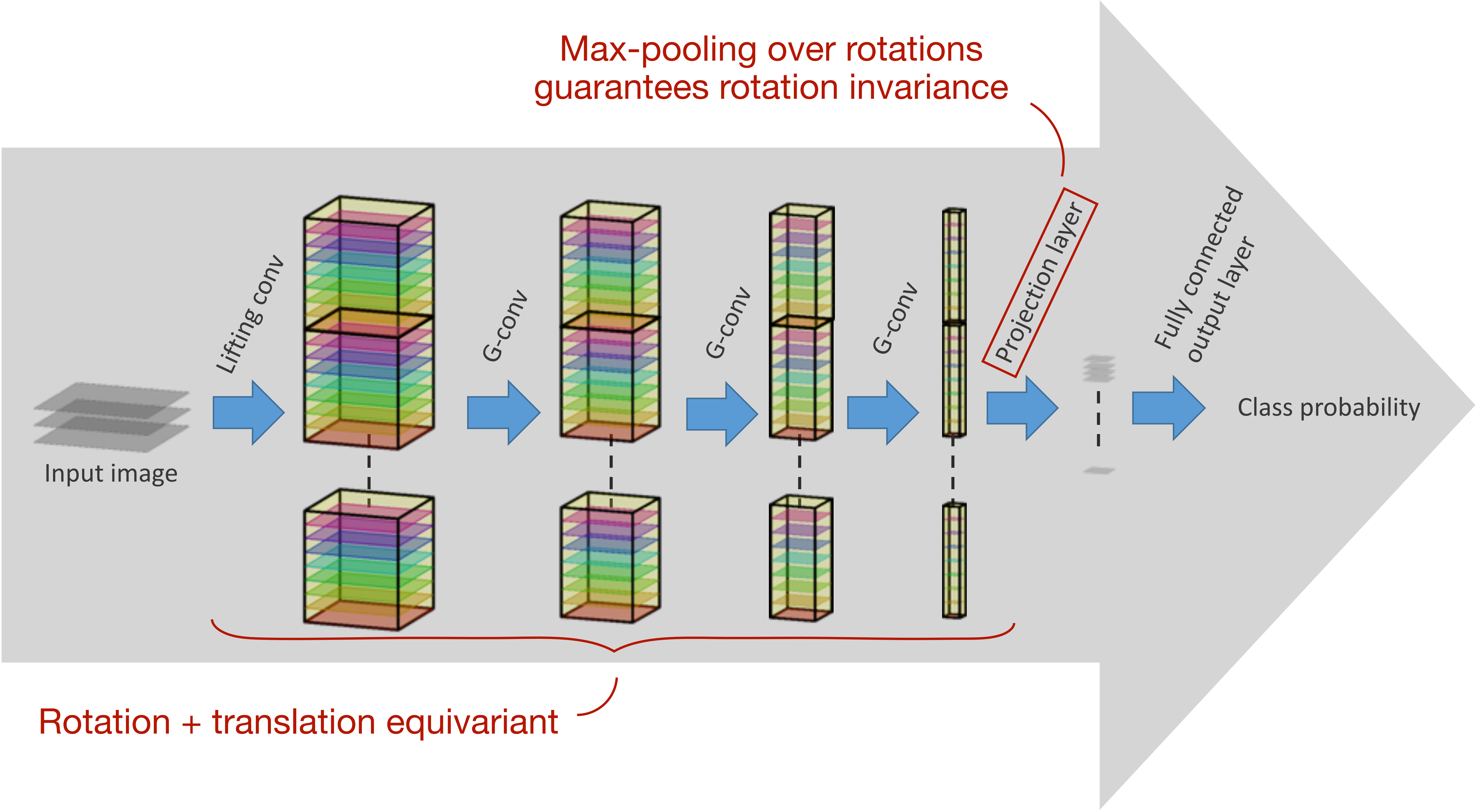


Rotation + translation equivariant

“normal” (0)
vs
“mitotic” (1)

Architecture for rotation invariant mitotic cell detection

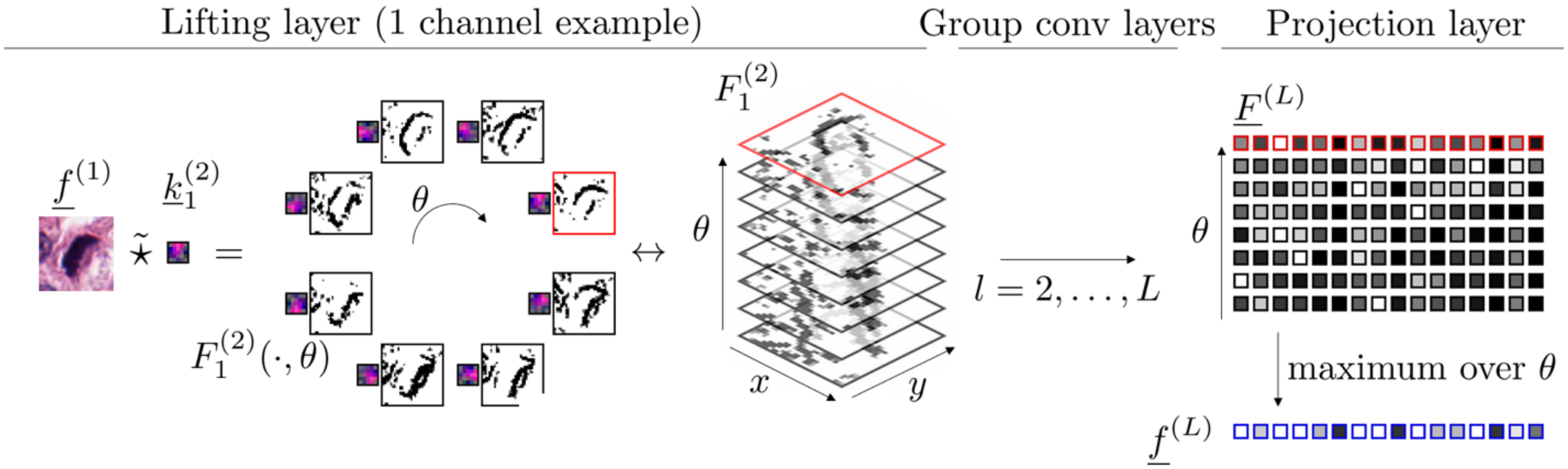




“normal” (0)
vs
“mitotic” (1)

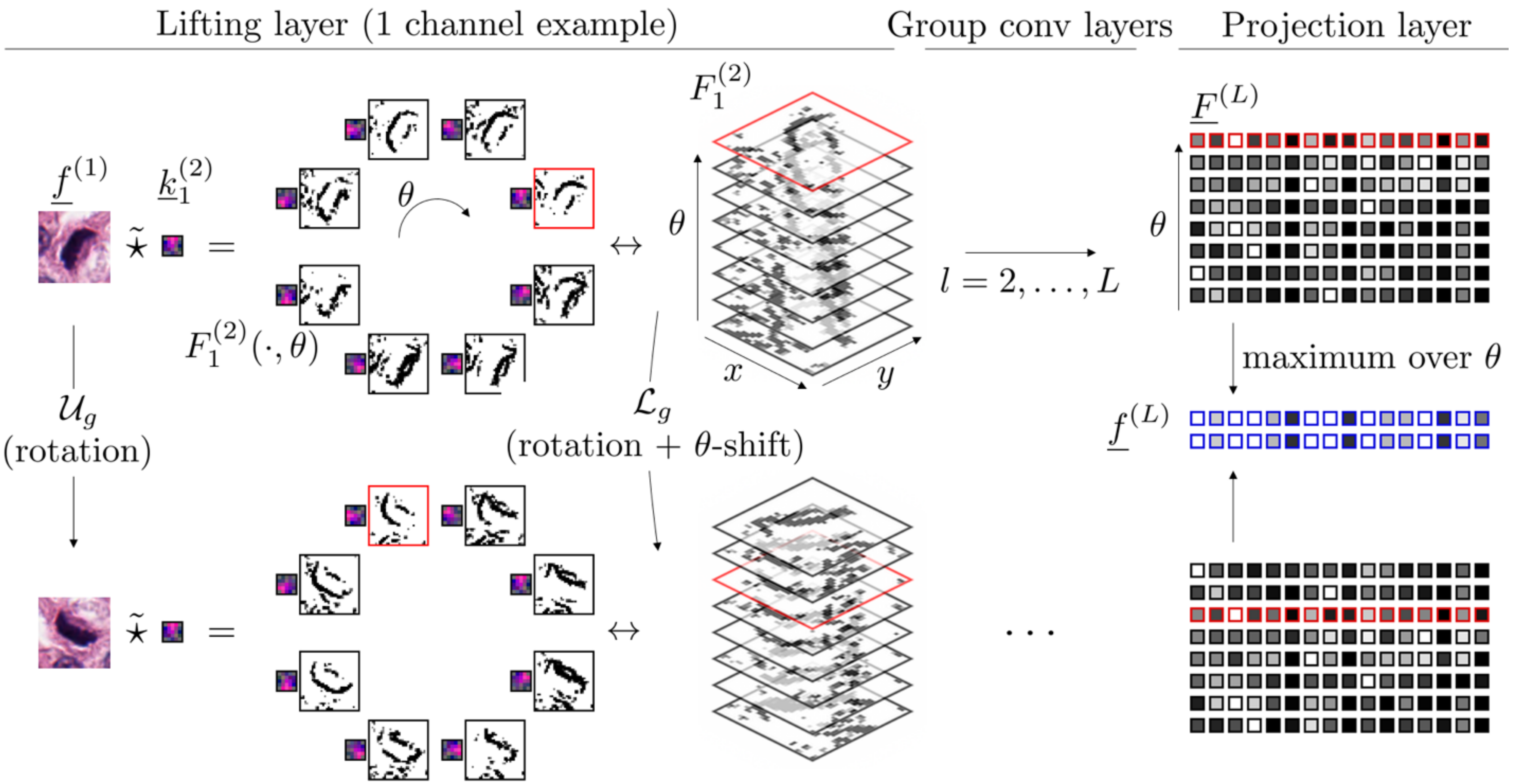
Bekkers & Lafarge et al. MICCAI 2018

Architecture for rotation invariant mitotic cell detection



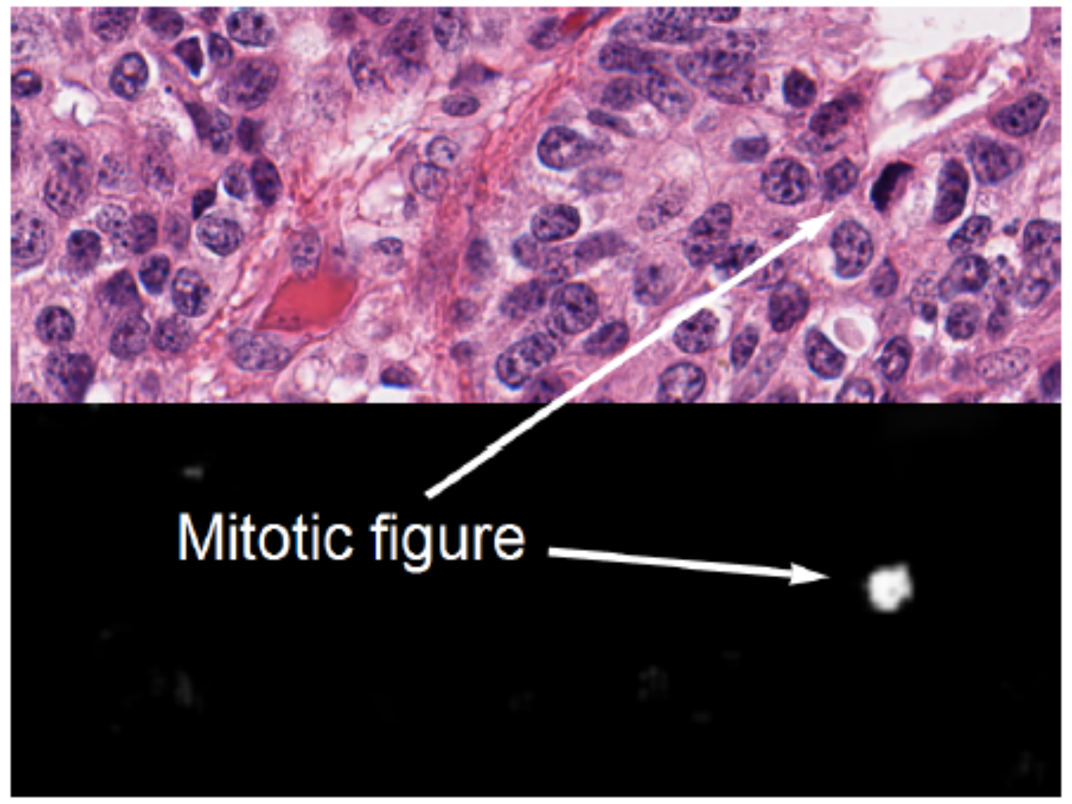
Bekkers & Lafarge et al. MICCAI 2018

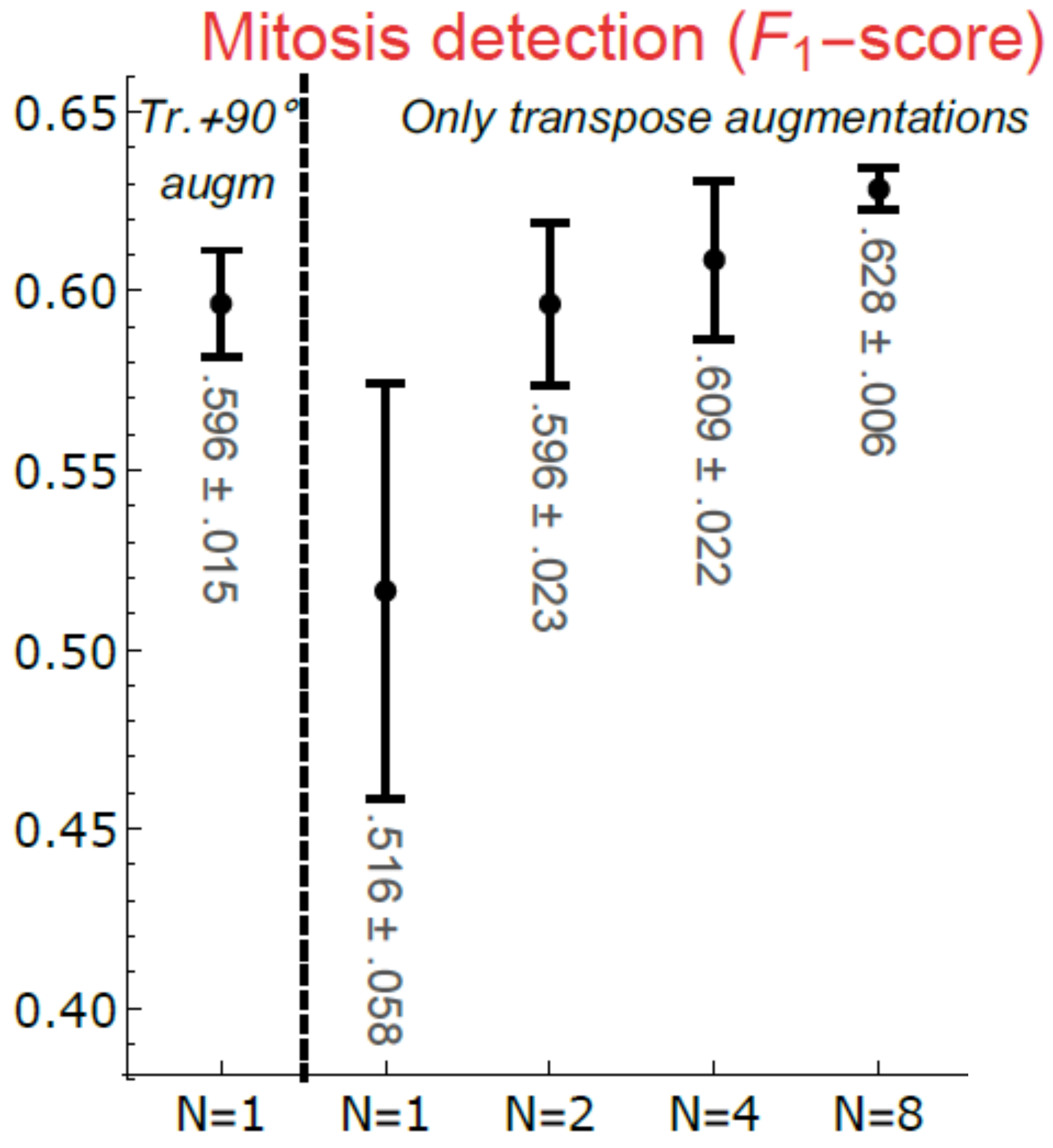
Architecture for rotation invariant mitotic cell detection



Architecture for rotation invariant mitotic cell detection

Bekkers & Lafarge et al. MICCAI 2018

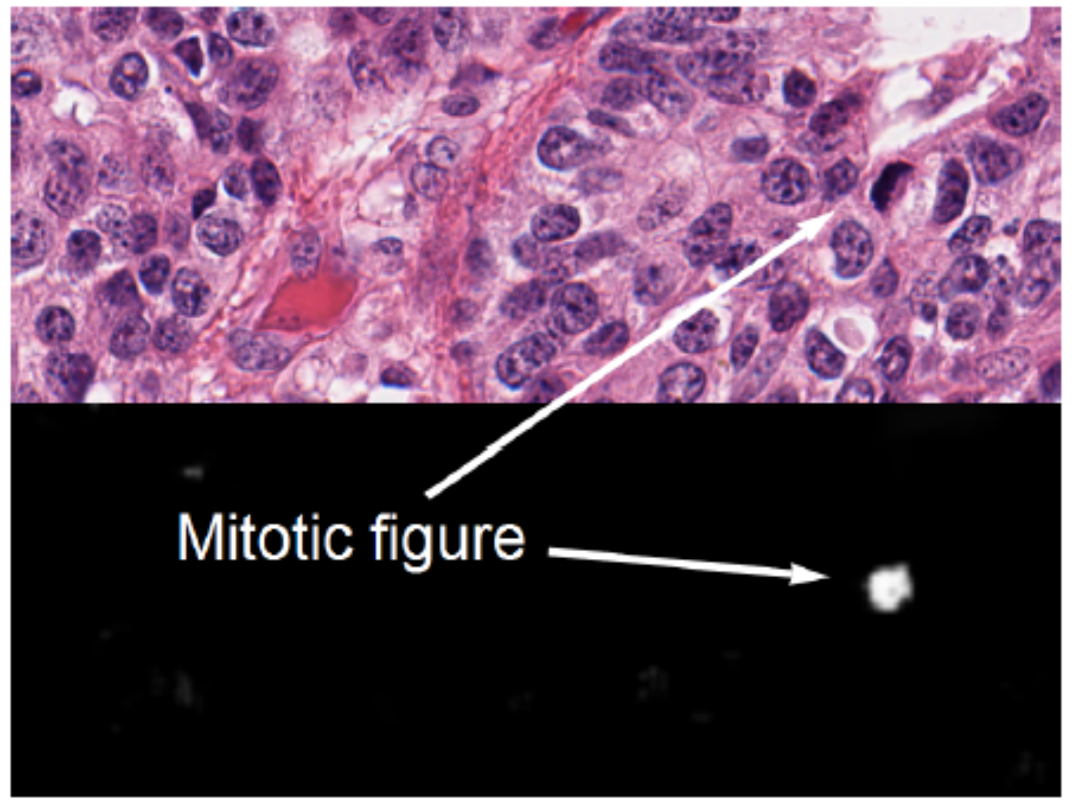


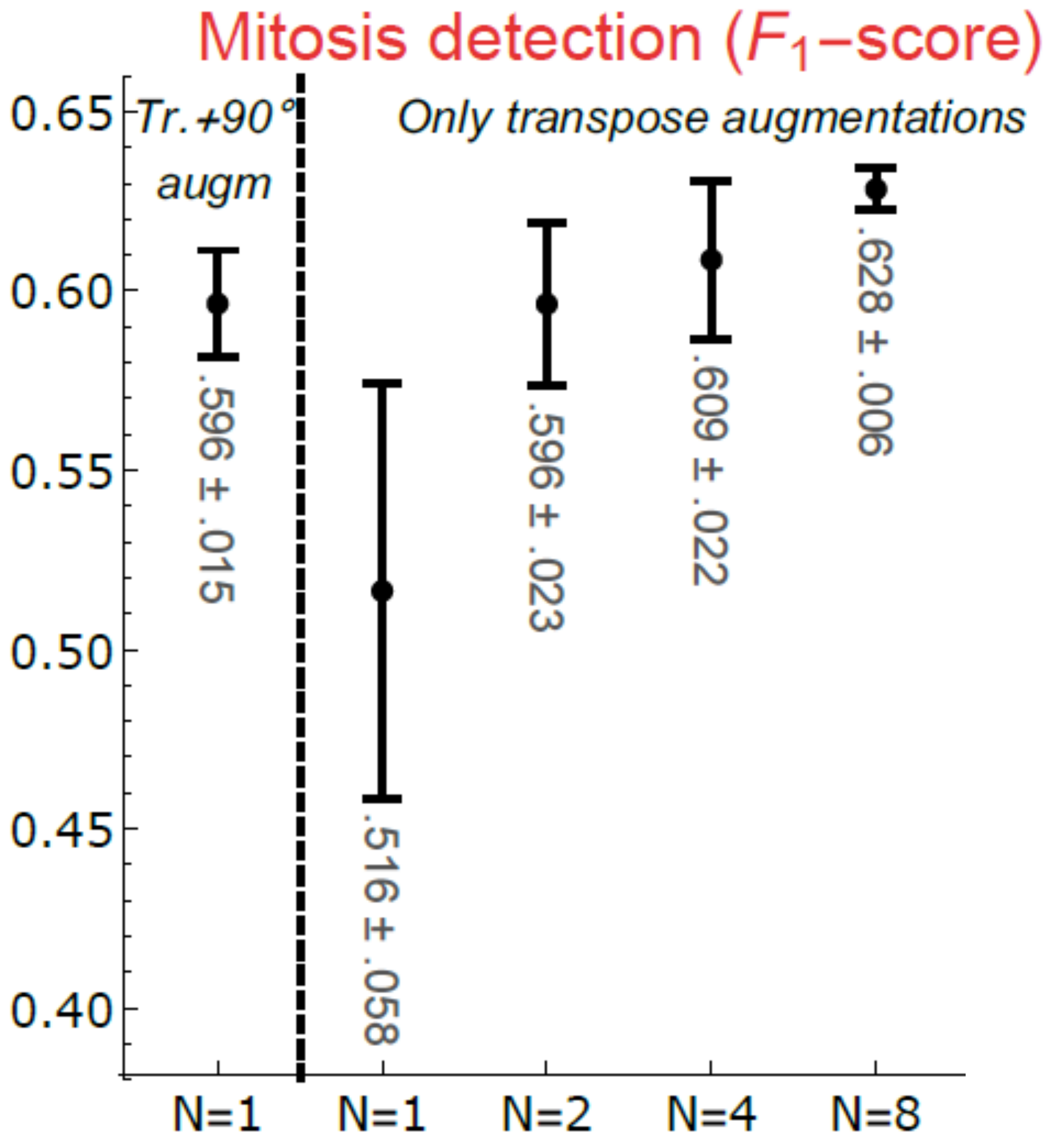


G-CNNs without data-augmentation *outperform*
CNNs with data-augmentation

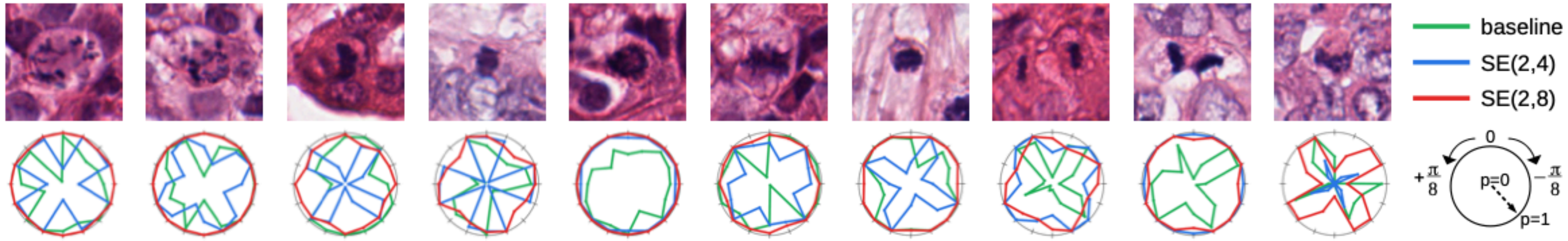
Architecture for rotation invariant mitotic cell detection

Bekkers & Lafarge et al. MICCAI 2018





Lafarge et al. MedIA 2020

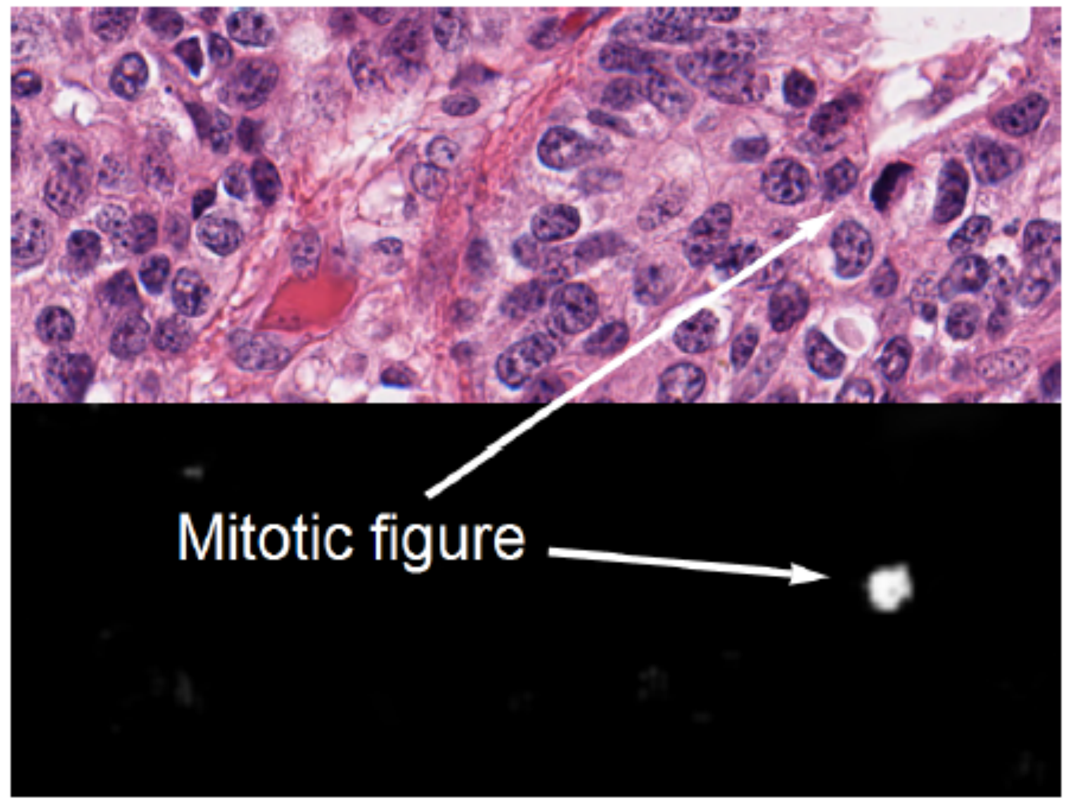


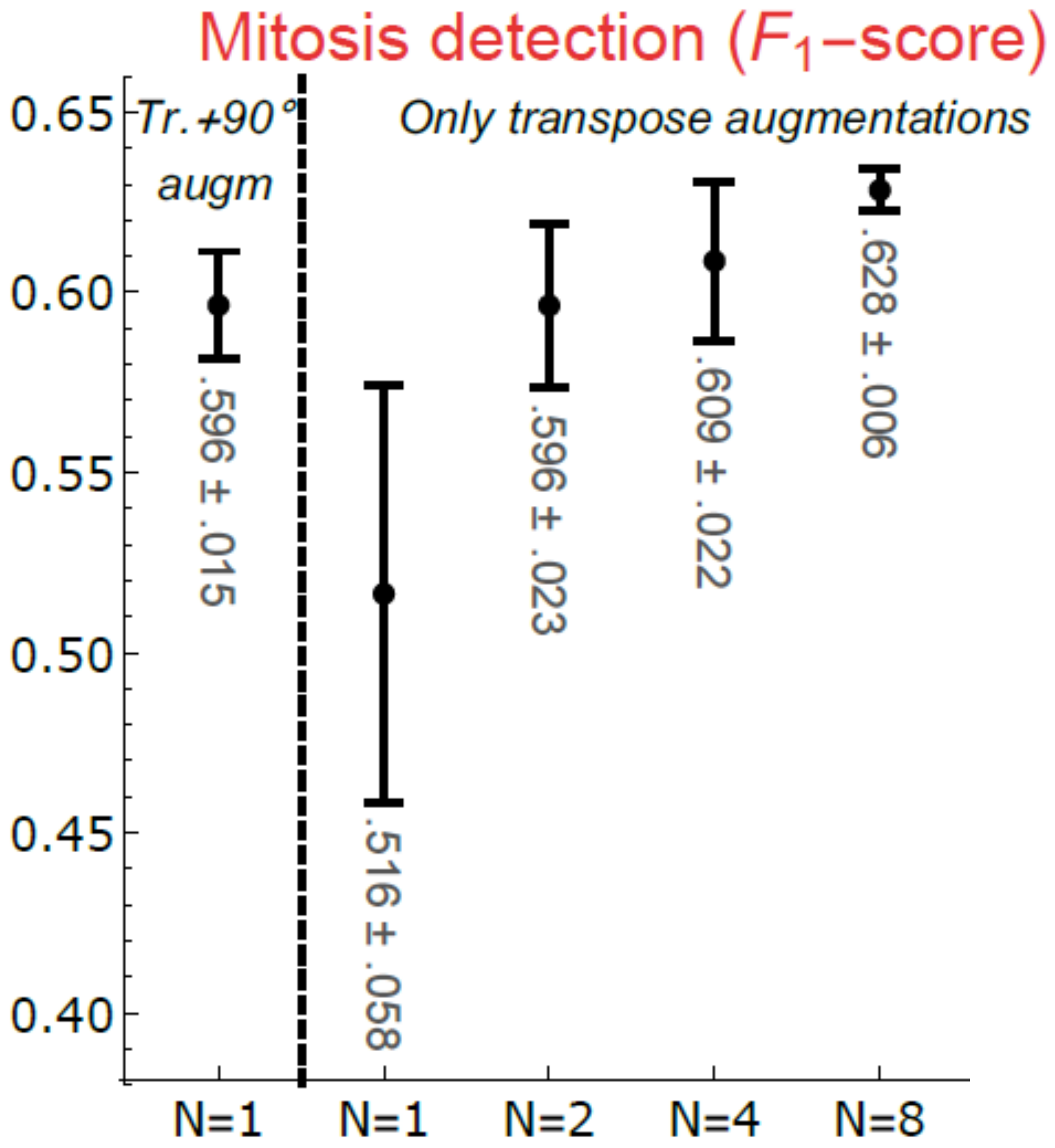
G-CNNs guarantee geometric stability. They are ***robust to input distortions****, regular CNNs aren't…*

G-CNNs without data-augmentation
outperform
CNNs with data-augmentation

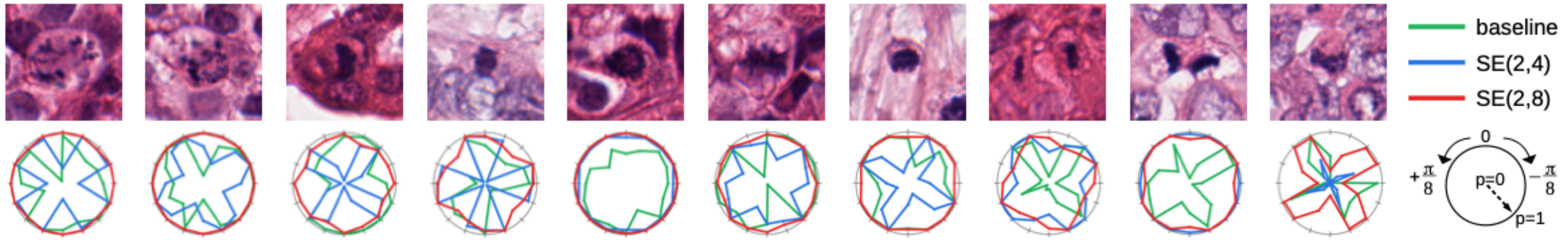
Architecture for rotation invariant mitotic cell detection

Bekkers & Lafarge et al. MICCAI 2018





Lafarge et al. MedIA 2020



*G-CNNs guarantee geometric stability. They are **robust to input distortions**, regular CNNs aren't…*

*G-CNNs are more **sample efficient**!*
G-CNNs (25% data) > CNNs (100% data)

***G-CNNs without data-augmentation** outperform CNNs with data-augmentation*

Lafarge et al. ArXiv/MedIA 2020

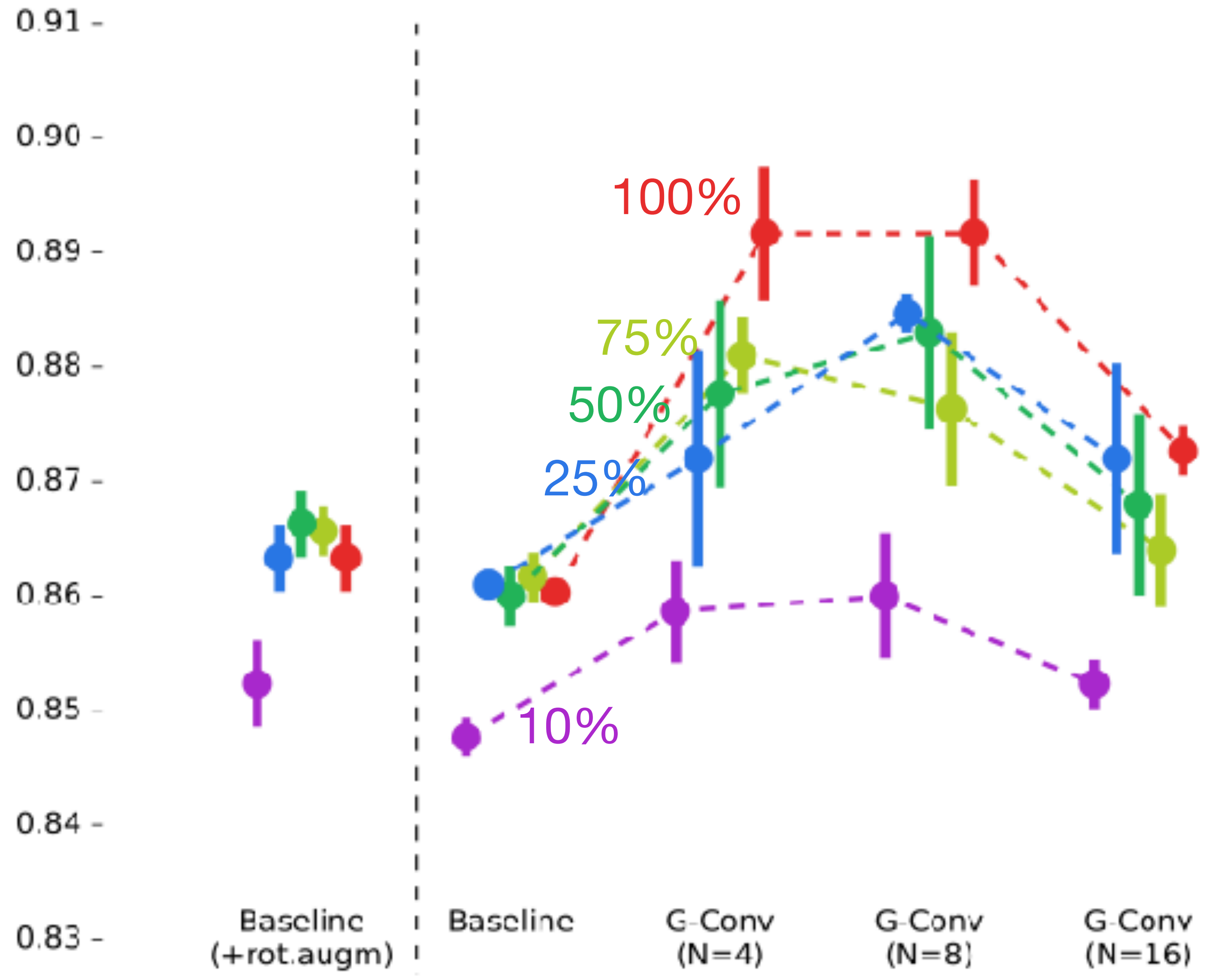


Figure 7: Mean and Standard Deviation plots summarizing the accuracy of the tumor classification models. Mean ± standard deviation is indicated. Color identifies the different data regime (red: 100%; lime: 75%; green: 50%; blue: 25%; purple: 10%).

G-CNNs rule!

- The right inductive bias: guaranteed equivariance
  (no loss of information)
- Performance gains that can’t be obtained by data-augmentation alone
  (both local and global equivariance/invariance)
- Increased sample efficiency
  (increased weight sharing, no geometric augmentation necessary)

G-CNNs rule!

- The right inductive bias: guaranteed equivariance
  (no loss of information)
- Performance gains that can’t be obtained by data-augmentation alone
  (both local and global equivariance/invariance)
- Increased sample efficiency
  (increased weight sharing, no geometric augmentation necessary)

Oriented Response Networks

Learning SO(3) Equivariant
with Spherical C

On the Generalization of Equivariance and Convolution in Neural N
to the Action of Compact Groups

Risi Kondor Shubhendu Trivedi

Published as a conference paper at ICLR 2020

B-Spline CNNs on Lie Gr

Erik J. Bekkers

Roto-Translation Covariant Convolut
Networks for Medical Image Analy

Tensor field networks:
Rotation- and translation-equivariant neural
networks for 3D point clouds

Nathaniel Thomas
Stanford University
Stanford, California, US

Roto-Translation Equivariant Conv
Application to Histopathology

General E(2) - Equivariant Steerable CNNs

Maurice Weiler
University of Amsterdam, QUVA Lab

Gabriele Cesa
University of Amsterdam

Group Equivariant Convolutional Networks

Taco S. Cohen
University of Amsterdam

Max Welling
University of Amsterdam
University of California Irvine
Canadian Institute for Advanced Research

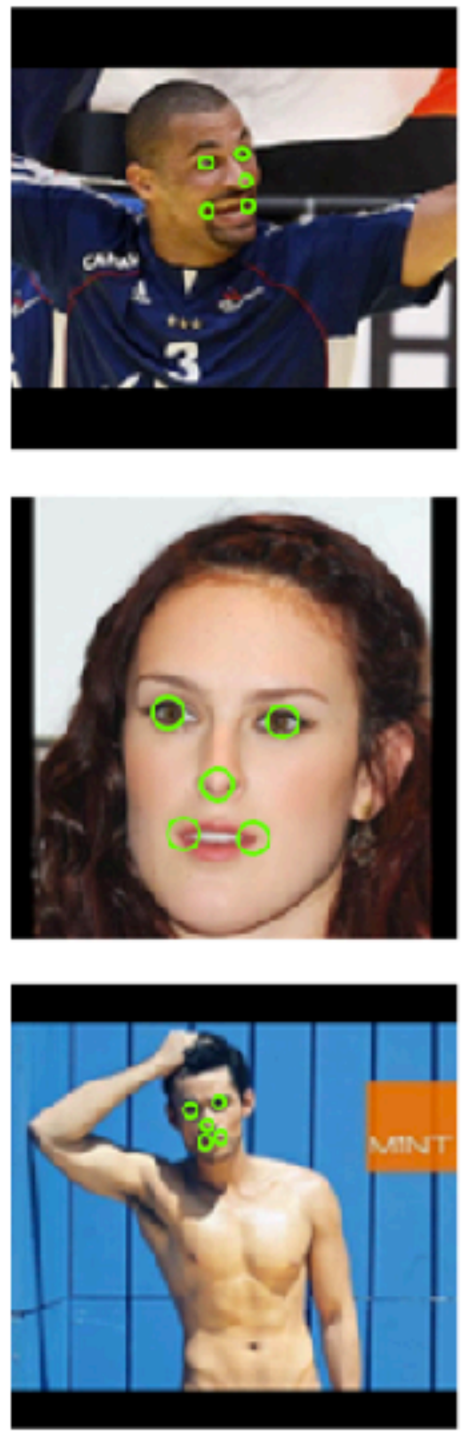
Generalizing Convolutional Neural Networks for Equivarian
to Lie Groups on Arbitrary Continuous Data

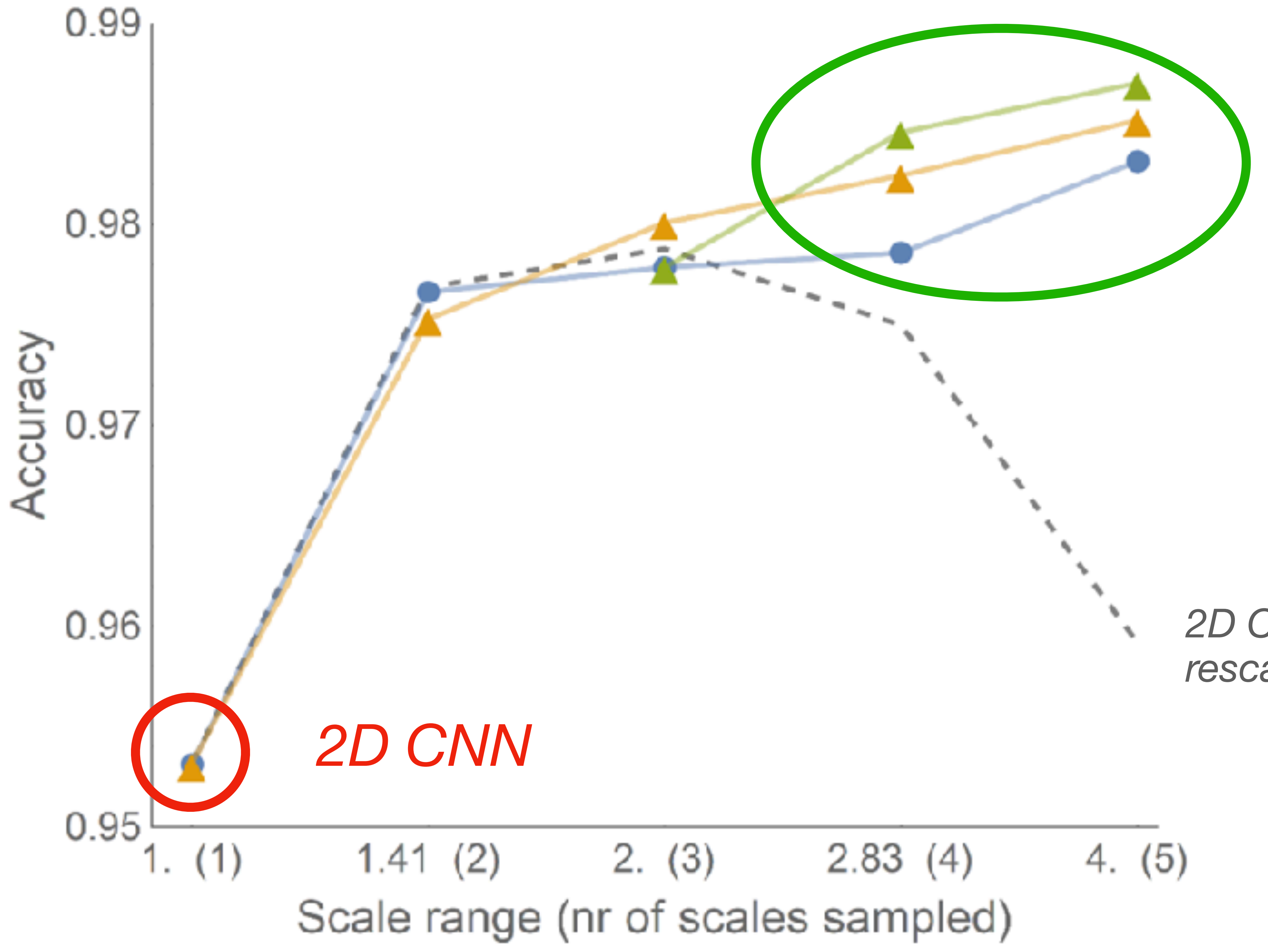
Marc Finzi Samuel Stanton Pavel Izmailov Andrew Gordon Wilson
New York University

From rotation to scale equivariant CNNs

Bekkers ICLR 2020

Translation + scale equivariant G-CNNs



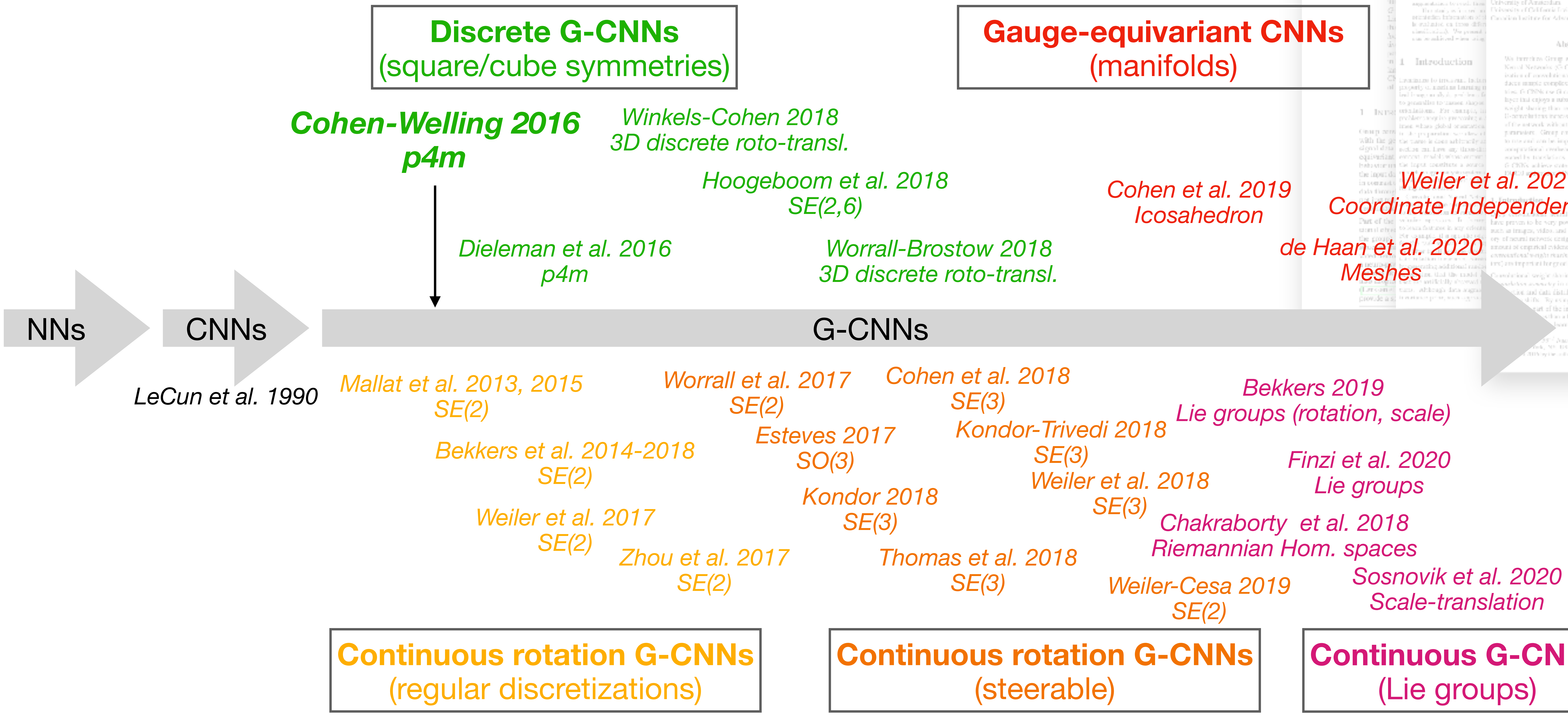


2D CNN

2D CNN with rescaled input

A brief history of G-CNNs

https://github.com/Chen-Cai-OSU/awesome-equivariant-network



Discrete G-CNNs
(square/cube symmetries)

Gauge-equivariant CNNs
(manifolds)

Cohen-Welling 2016
p4m

Winkels-Cohen 2018
3D discrete roto-transl.

Hoogeboom et al. 2018
SE(2,6)

Dieleman et al. 2016
p4m

Worrall-Brostow 2018
3D discrete roto-transl.

Cohen et al. 2019
Icosahedron

Weiler et al. 2021
Coordinate Independent CNNs

de Haan et al. 2020
Meshes

LeCun et al. 1990

Mallat et al. 2013, 2015
SE(2)

Bekkers et al. 2014-2018
SE(2)

Weiler et al. 2017
SE(2)

Zhou et al. 2017
SE(2)

Worrall et al. 2017
SE(2)

Esteves 2017
SO(3)

Kondor 2018
SE(3)

Cohen et al. 2018
SE(3)

Kondor-Trivedi 2018
SE(3)

Weiler et al. 2018
SE(3)

Thomas et al. 2018
SE(3)

Weiler-Cesa 2019
SE(2)

Bekkers 2019
Lie groups (rotation, scale)

Finzi et al. 2020
Lie groups

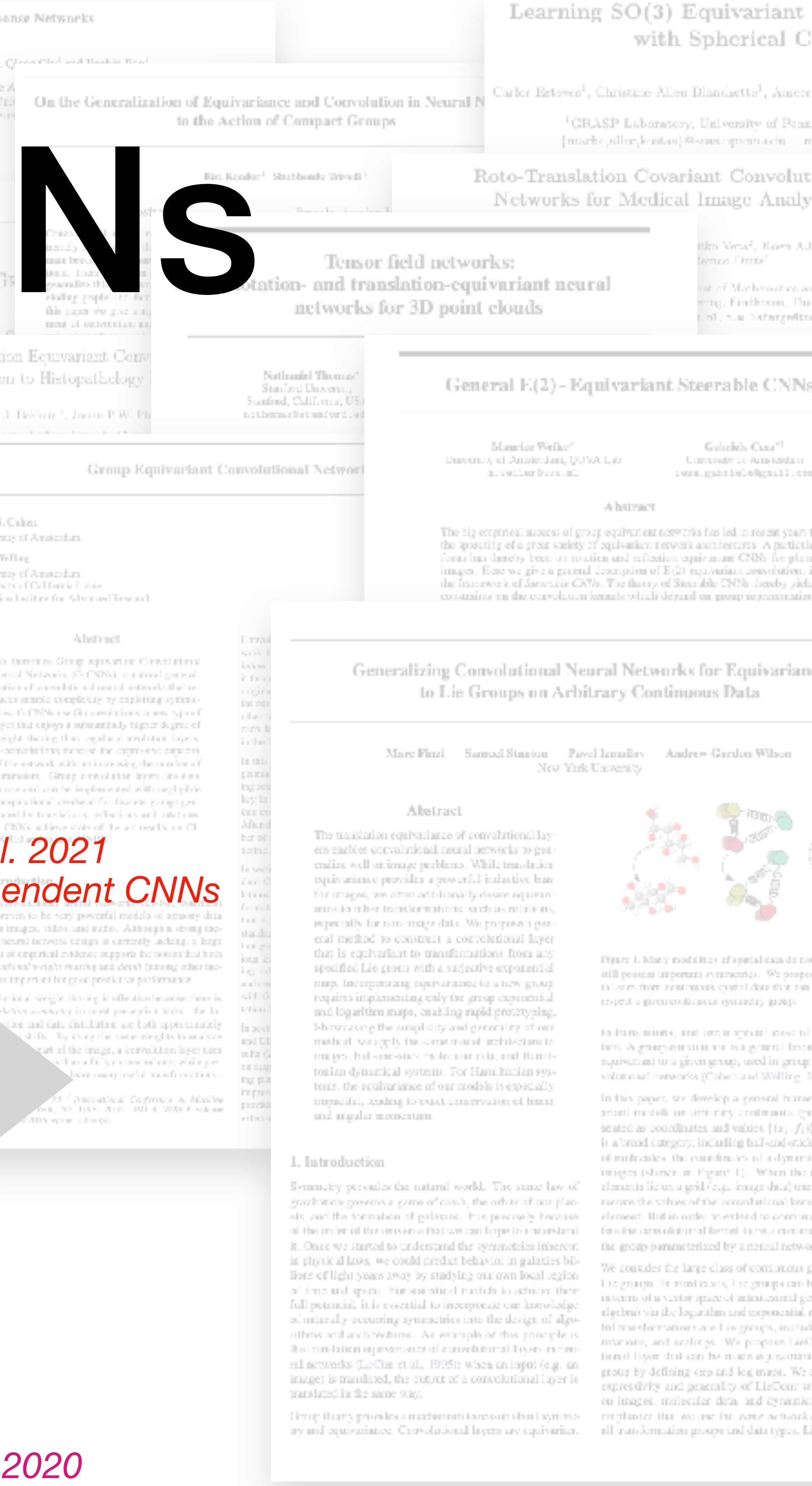
Chakraborty et al. 2018
Riemannian Hom. spaces

Sosnovik et al. 2020
Scale-translation

Continuous rotation G-CNNs
(regular discretizations)

Continuous rotation G-CNNs
(steerable)

Continuous G-CNNs
(Lie groups)



Cesa-Lang-Weiler 2022
$G = \mathbb{R}^d \rtimes H$ *with* H *compact*

https://quva-lab.github.io/escnn/

1. Motivation	Equivariance → weight-sharing and generalization
2. Pattern matching using group theory	Group theory: symmetries & recognition by components (features have “poses”)
3. Group convolutions	Template matching over groups
4. Example	Effective representation learning and generalization
5. G-convs are all you need!	
6. Steerable group convolutions	
7. Feature fields and escnn library	
8. Equivariant tensor product layers	
9. Equivariant graph NNs	

1. Motivation	Equivariance → weight-sharing and generalization
2. Pattern matching using group theory	Group theory: symmetries & recognition by components (features have “poses”)
3. Group convolutions	Template matching over groups
4. Example	Effective representation learning and generalization
5. G-convs are all you need!	
6. Steerable group convolutions	
7. Feature fields and escnn library	
8. Equivariant tensor product layers	
9. Equivariant graph NNs	

Theorem (G-convs are all you need)

Bekkers ICLR 2020, Thm. 1*

Let $\mathscr{K} : \mathbb{L}_2(X) \to \mathbb{L}_2(Y)$ map between signals on homogeneous spaces of G.

Let homogeneous space $Y \equiv G/H$ such that $H = \mathrm{Stab}_G(y_0)$ for some chosen origin $y_0 \in Y$ and let $g_y \in G$ such that $\forall_{y \in Y} : y = g_y y_0$.

Then $\mathscr{K}$ is equivariant to group G if and only if:

1. It is a group convolution: $[\mathscr{K} f](y) = \int_X \frac{1}{|g_y|} k(g_y^{-1} x) f(x) \mathrm{d}x$

2. The kernel satisfies a symmetry constraint: $\forall_{h \in H} : \frac{1}{|g_y|} k(hx) = k(x)$

*Work with Remco Duits at TU/e.

See also: Duits 2005 – Thm 25, Cohen, Geiger, Weiler 2018 - Thm 6.1, Kondor, Trivedi 2018 - Thm 1

1. Motivation

Equivariance → weight-sharing and generalization

2. Pattern matching using group theory

Group theory: symmetries & recognition by components
(features have “poses”)

3. Group convolutions

Template matching over groups

  4. Example

  Effective representation learning and generalization

  5. G-convs are all you need!

  Any equivariant linear layer is a group convolution

6. Steerable group convolutions

  7. Feature fields and escnn library

8. Equivariant tensor product layers

9. Equivariant graph NNs

1. Motivation — Equivariance → weight-sharing and generalization

2. Pattern matching using group theory — Group theory: symmetries & recognition by components
(features have “poses”)

3. Group convolutions — Template matching over groups

4. Example — Effective representation learning and generalization

5. G-convs are all you need! — Any equivariant linear layer is a group convolution

6. Steerable group convolutions

7. Feature fields and escnn library

8. Equivariant tensor product layers

9. Equivariant graph NNs

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

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

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

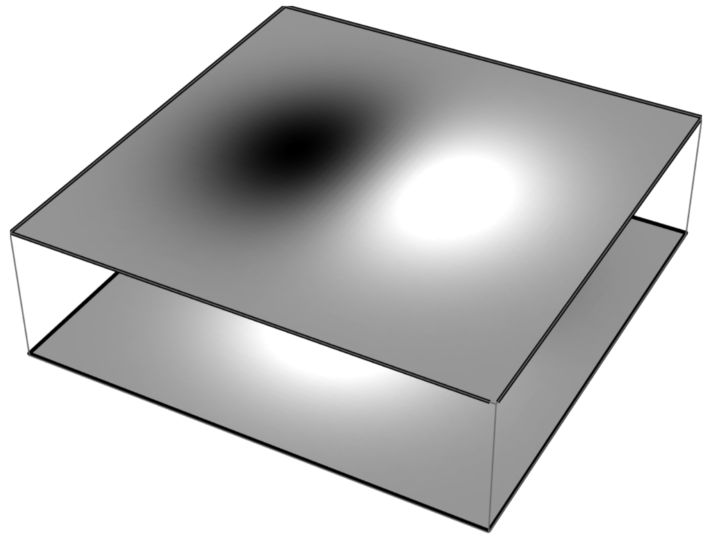
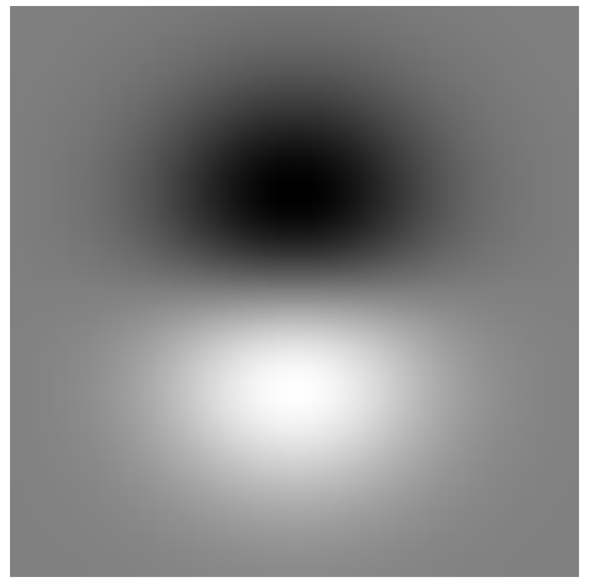
I.e., we can transform all basis functions simply by taking a linear combination of the original basis functions.

Function in steerable basis

Let $f(x \,|\, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^{\dagger} Y(x)$ ($Y(x)$ a steerable basis)

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

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

$(\rho(h)\hat{\mathbf{w}})^{\dagger}$ $\Bigg($  $\Bigg)$ $=$ 

|| $Y(x)$

|| $f(x \,|\, \rho(h)\hat{\mathbf{w}})$

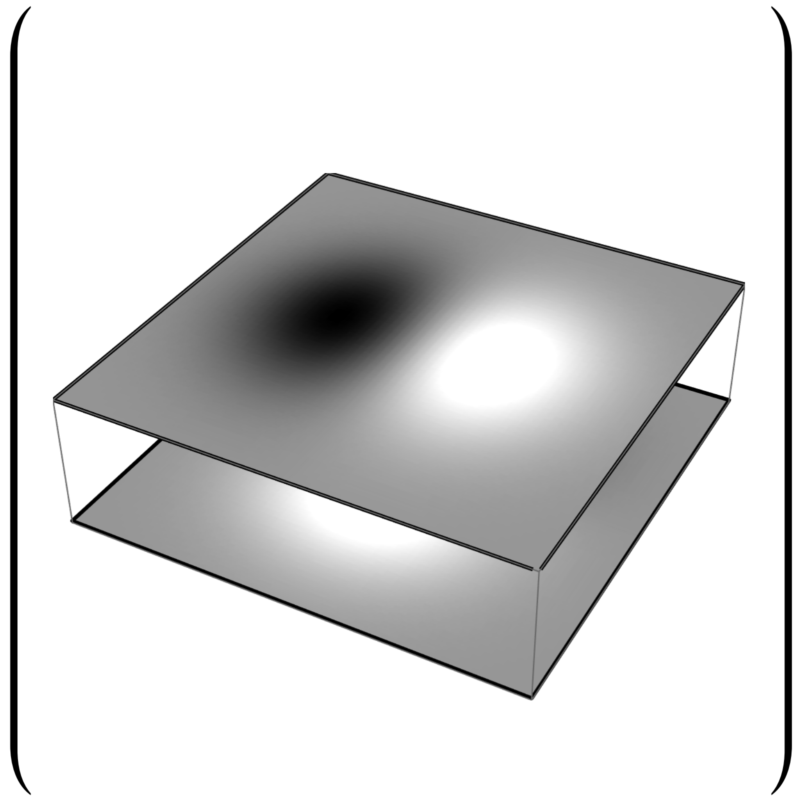
Function in steerable basis

Let $$f(x \mid \hat{\mathbf{w}}) = \hat{\mathbf{w}}^\dagger Y(x)$$ ($Y(x)$ a steerable basis)

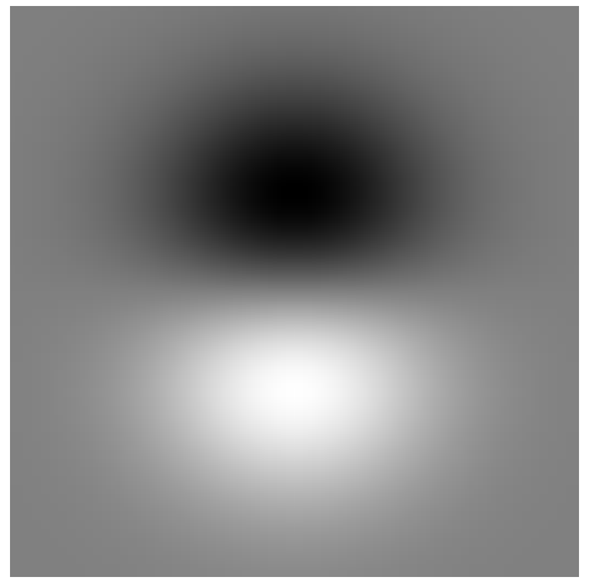
Then we can steer/shift this function by transforming the weights $\hat{\mathbf{w}}$

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

$(\rho(h)\hat{\mathbf{w}})^\dagger$



$=$



$\shortparallel$

$Y(x)$

$\shortparallel$

$f(x \mid \rho(h)\hat{\mathbf{w}})$

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: $$\begin{aligned} Y_l(\alpha - \theta) &= e^{i\,l\,(\alpha-\theta)} \\ &= e^{-i\,l\,\theta}\, e^{i\,l\,\alpha} \\ &= \rho_l(-\theta)\, Y_l(\alpha) \end{aligned}$$

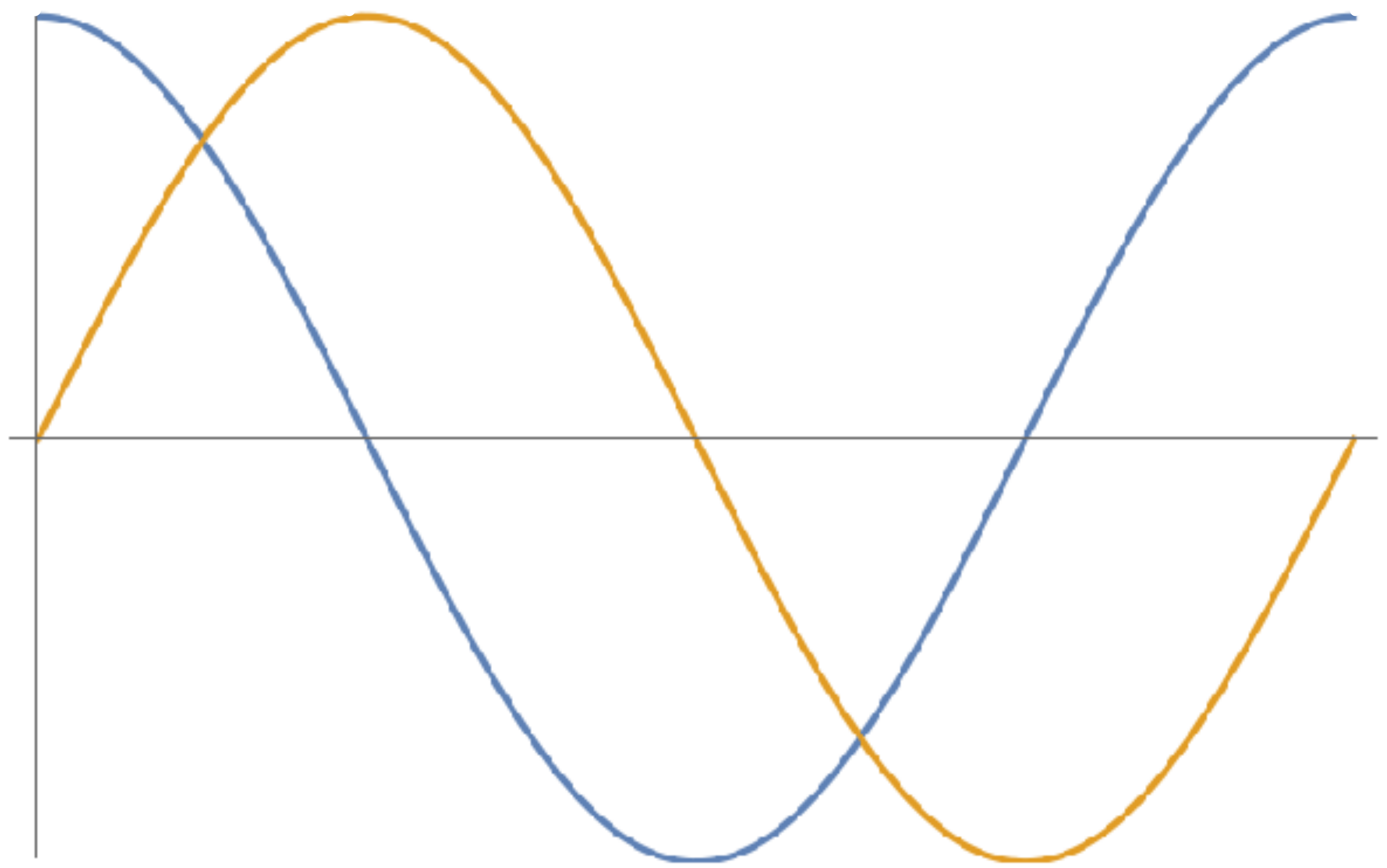
Example: Steerable basis on S^1 (circular harmonics)

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

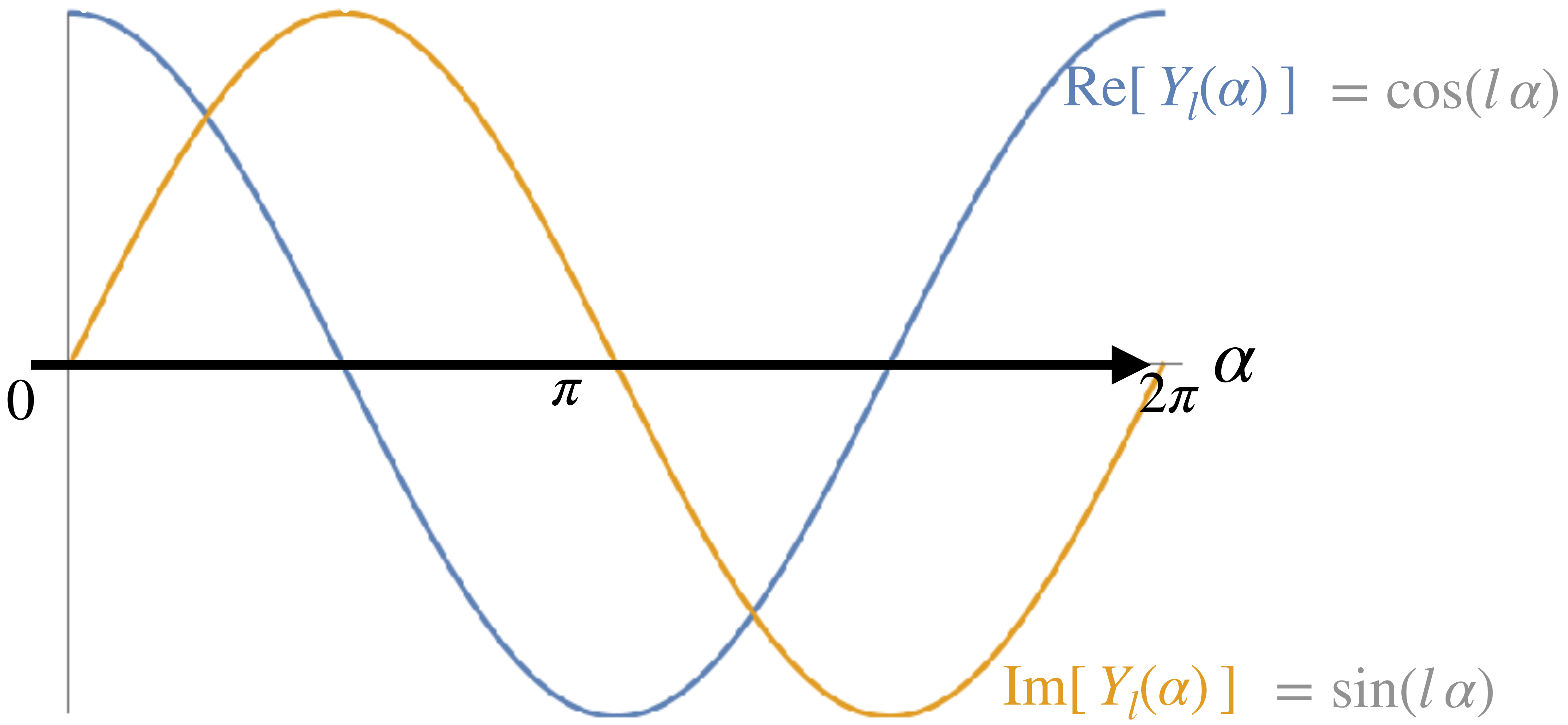
Are steered by representations: $\rho_l(\theta) = e^{i\,l\,\theta}$

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

$$\overbrace{}^{Y_l(\alpha-\theta)} = \overbrace{}^{\rho_l(-\theta)} \overbrace{}^{Y_l(\alpha)}$$

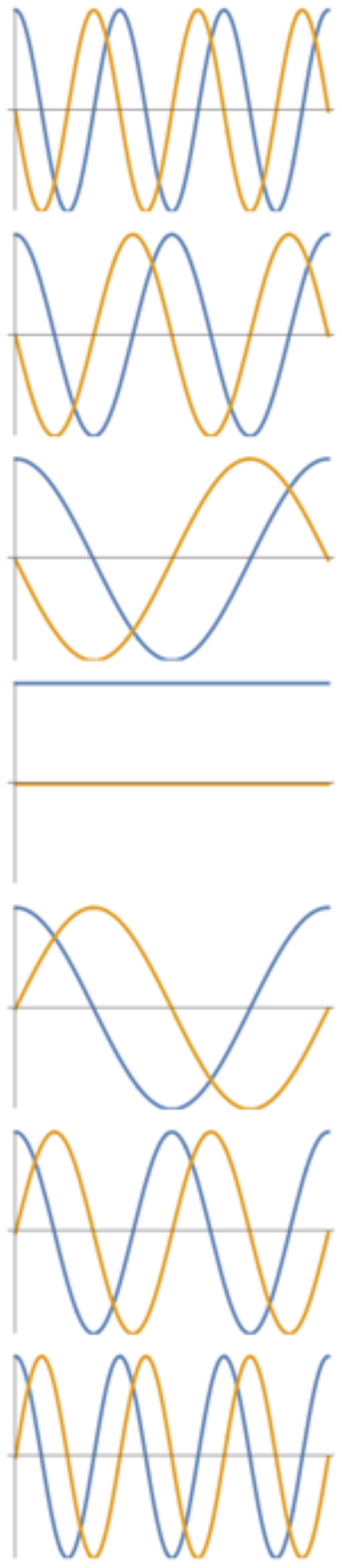


$= \quad e^{-i\theta}$



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

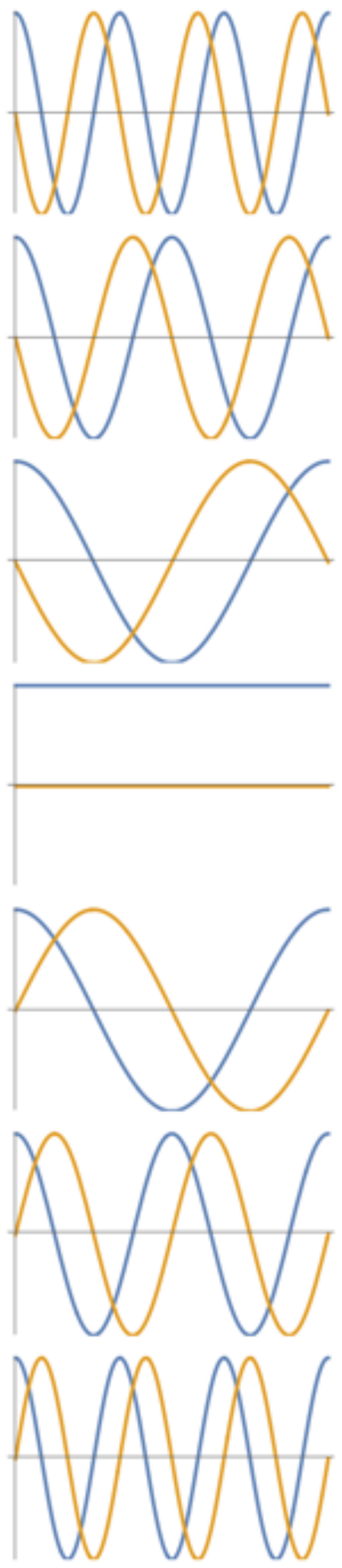
$Y(\alpha - \theta)$



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

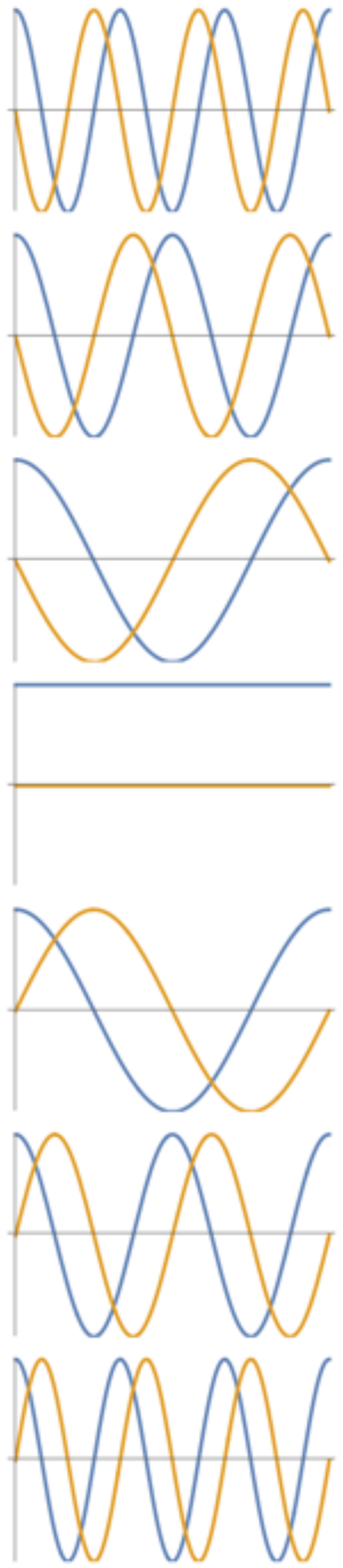
$$= \begin{pmatrix} e^{i3\theta} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{i2\theta} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{i1\theta} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{-i1\theta} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-i2\theta} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-i3\theta} \end{pmatrix}$$

$Y(\alpha)$



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

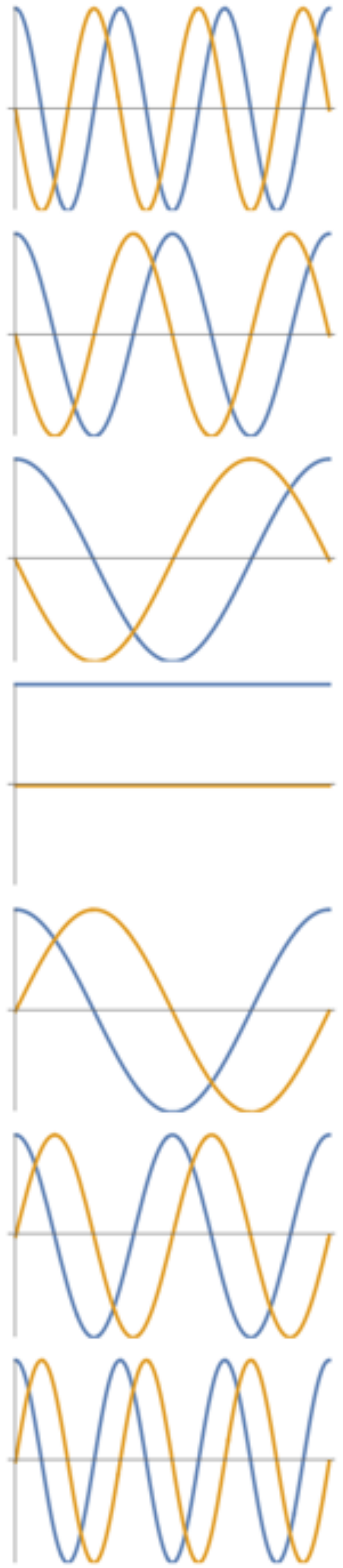
$Y(\alpha - \theta)$



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

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

$Y(\alpha)$



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

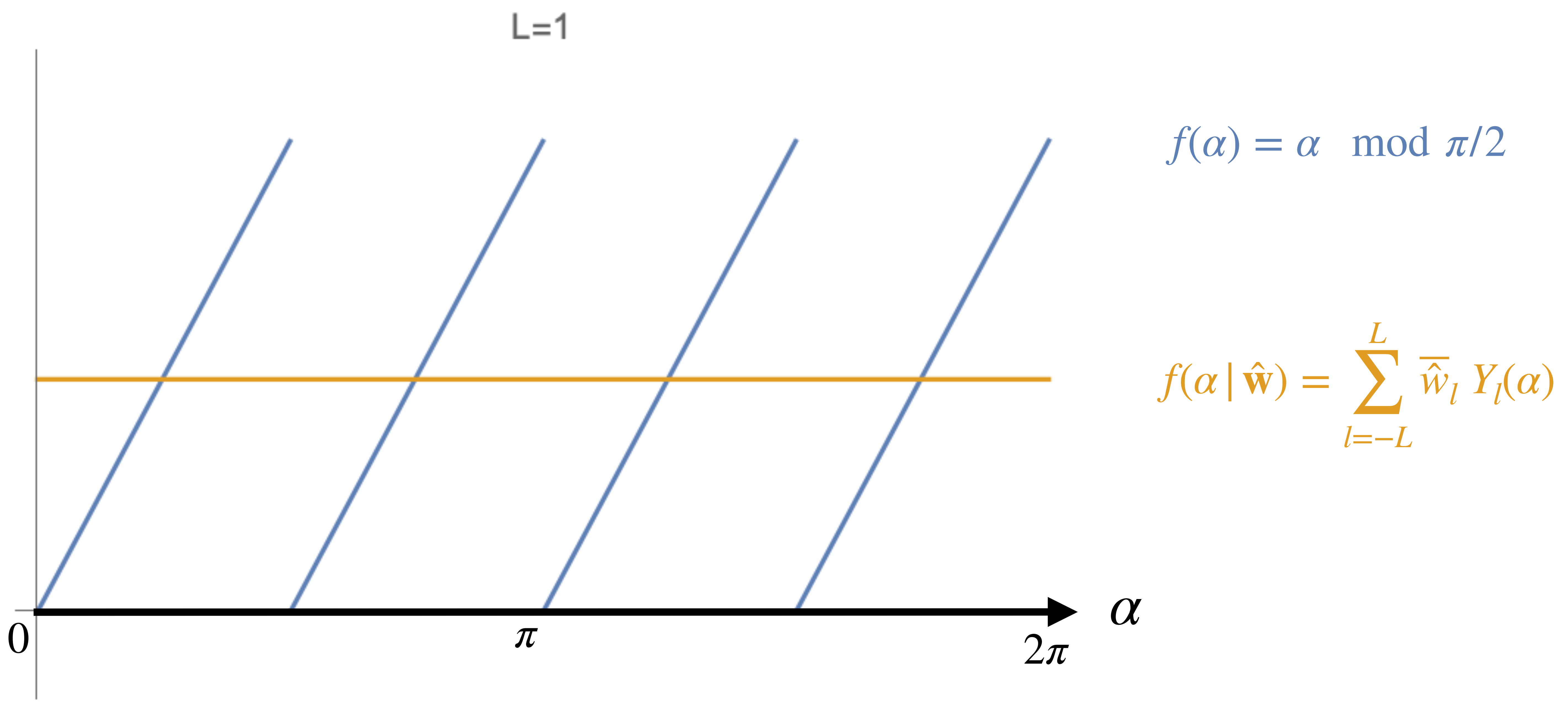
Basis functions (for $\mathbb{L}_2(S^1)$):

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

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)

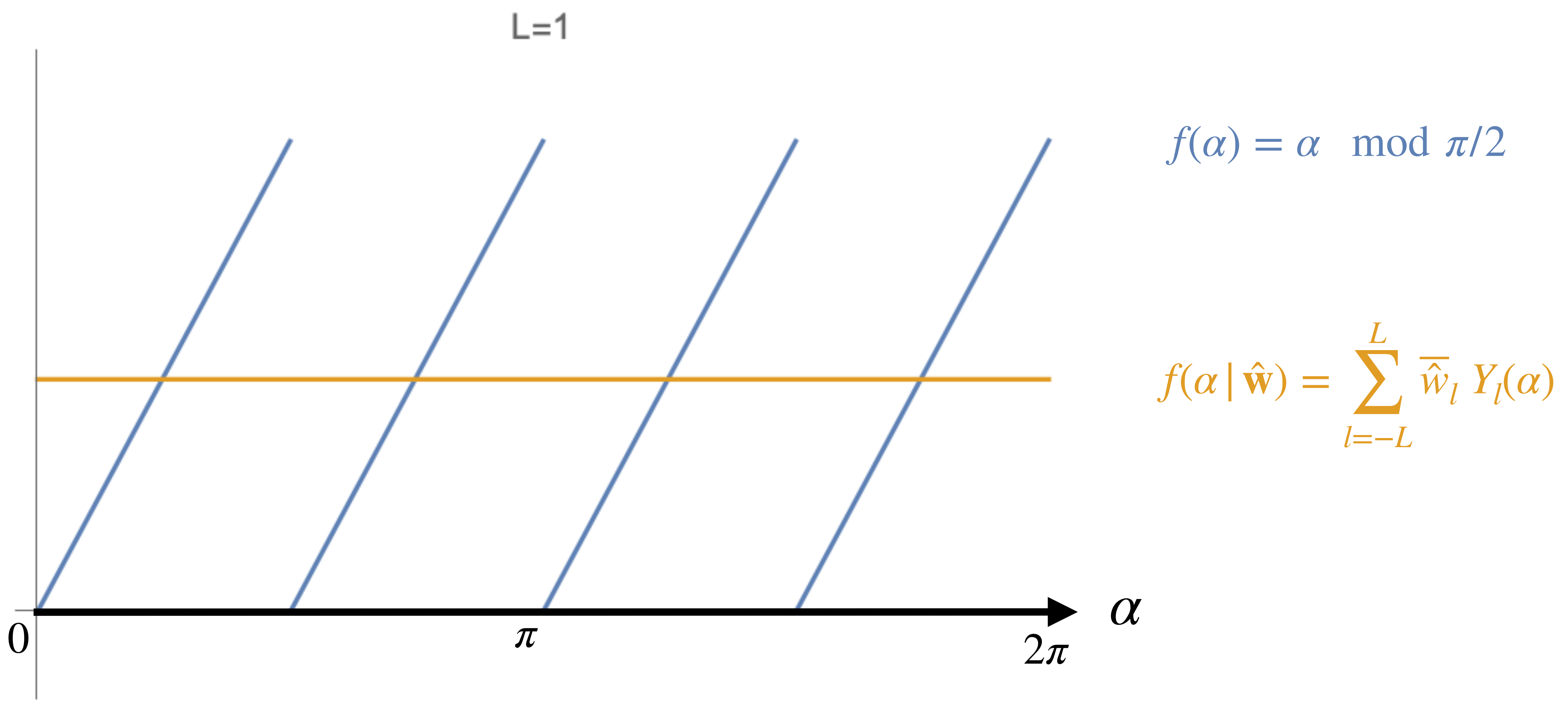


Example: Steerable basis on S^1 (circular harmonics)

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)

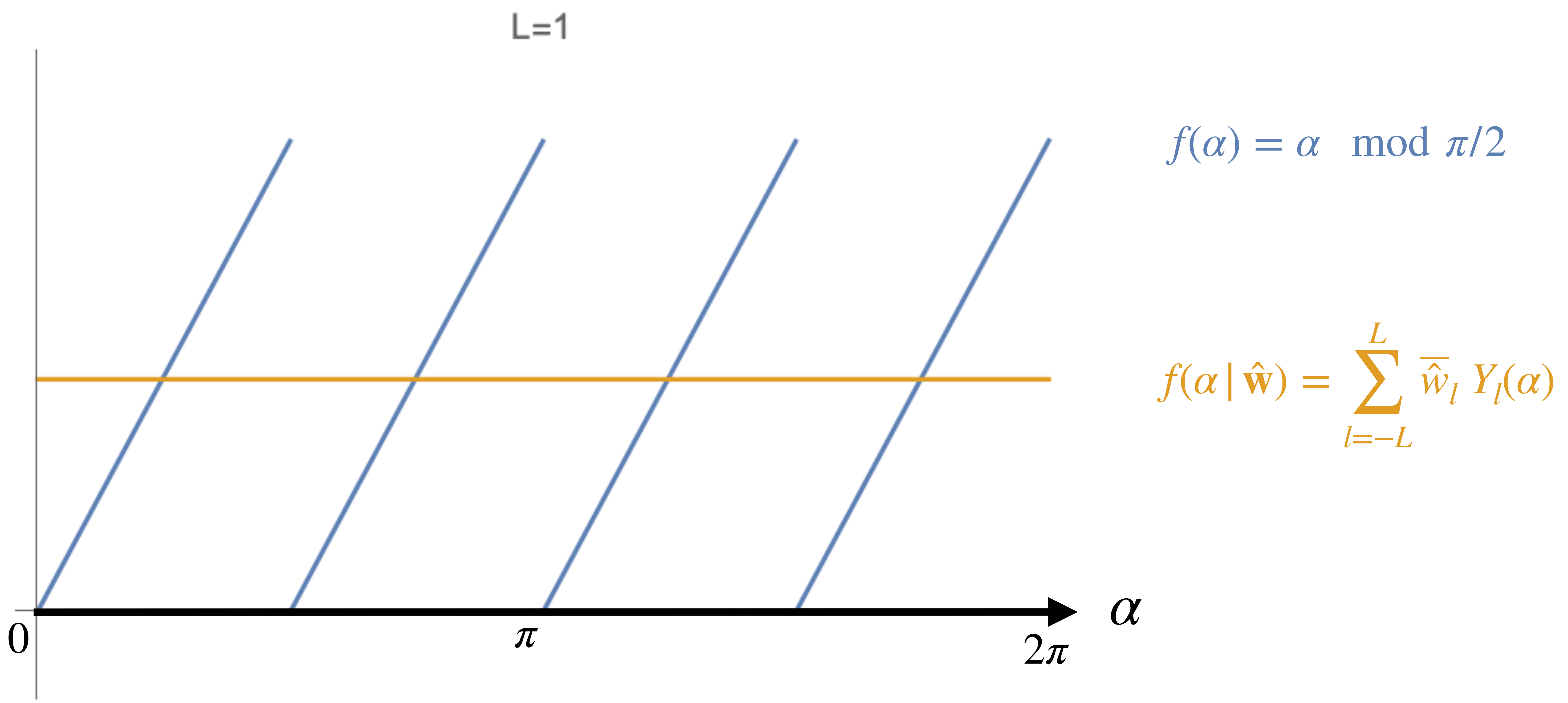


Example: Steerable basis on S^1 (circular harmonics)

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)

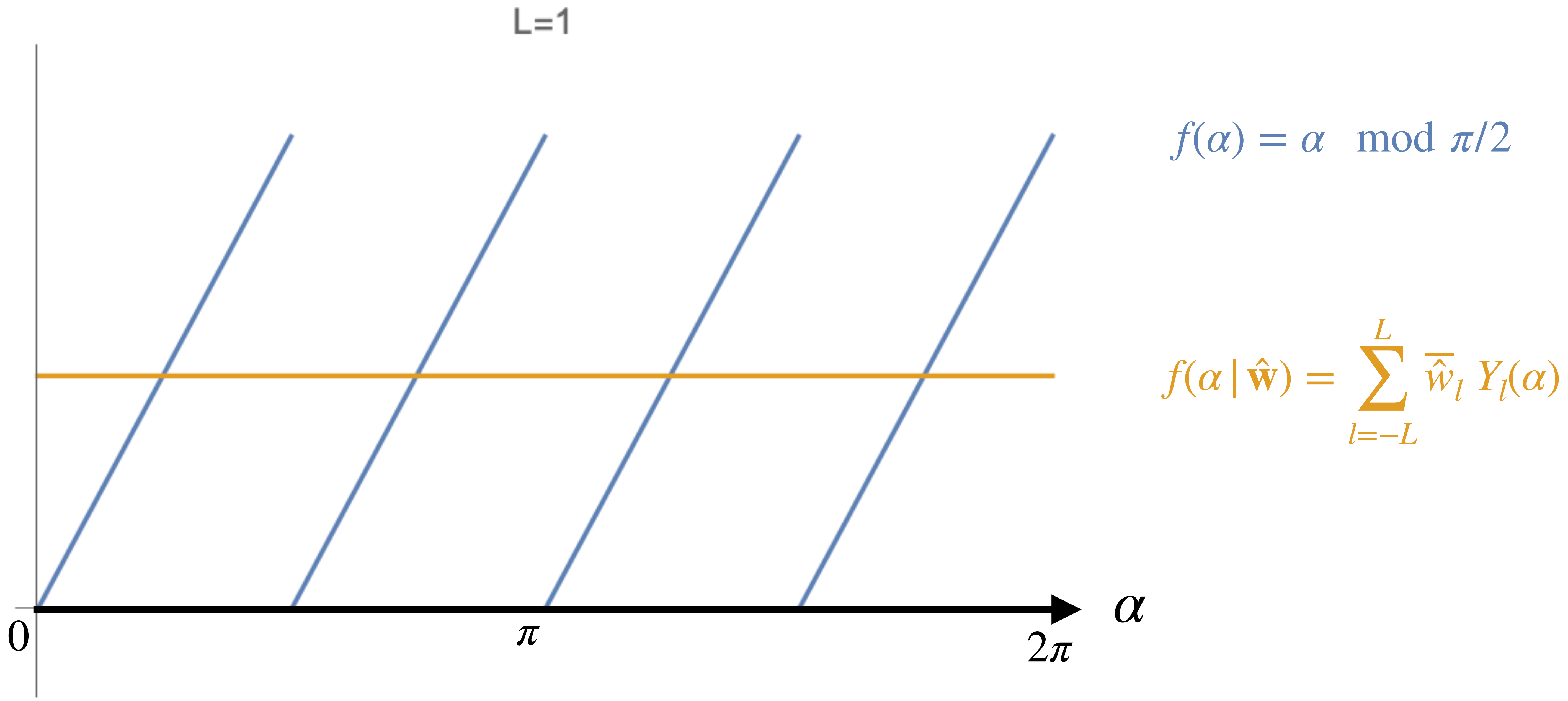


Example: Steerable basis on S^1 (circular harmonics)

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)



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

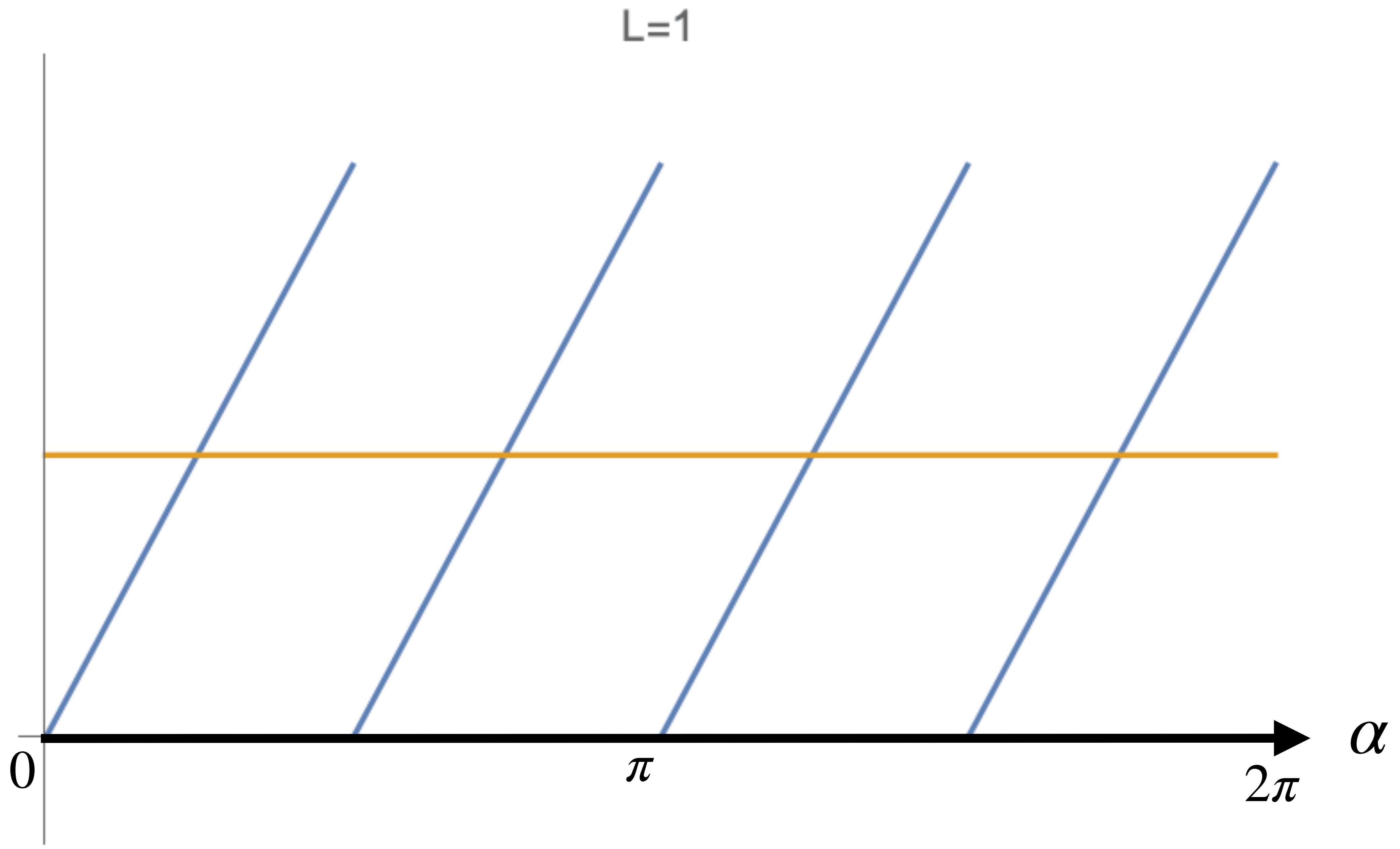
Basis functions (for $\mathbb{L}_2(S^1)$):

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

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)



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

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

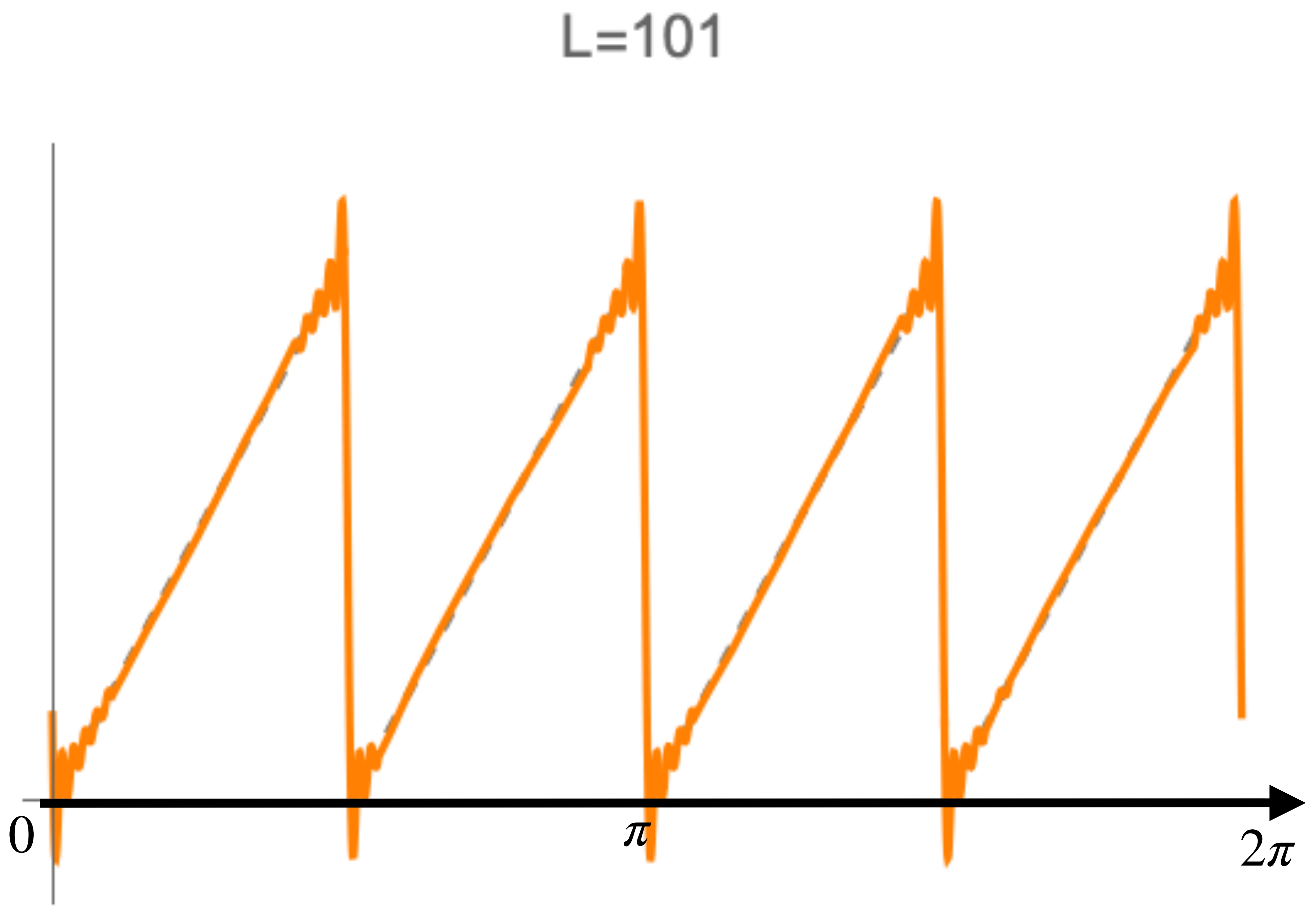
Example: Steerable basis on S^1 (circular harmonics)

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



α

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

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

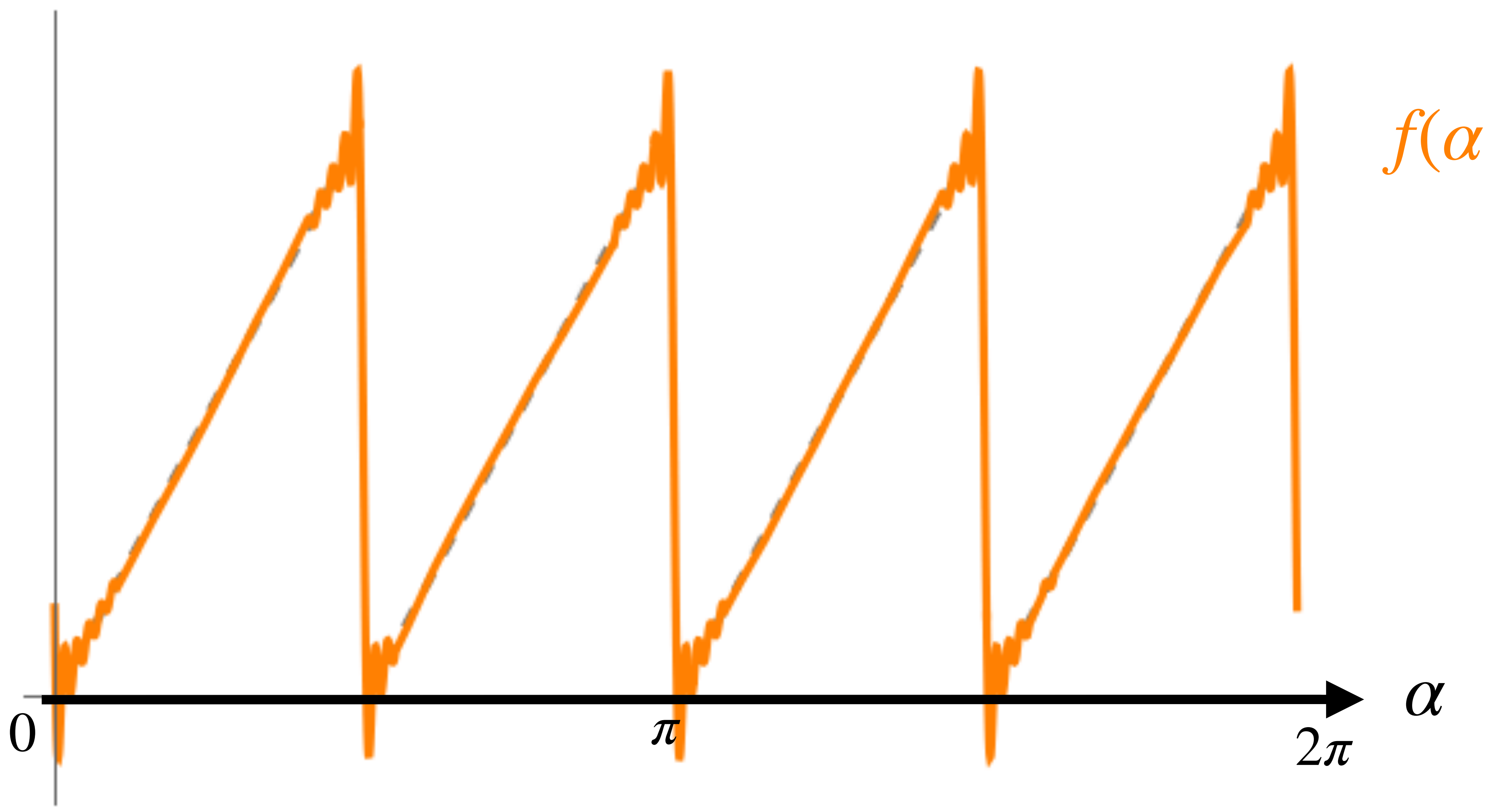
Let $$f(\alpha \mid \hat{\mathbf{w}}) = \hat{\mathbf{w}}^\dagger Y(\alpha)$$

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

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

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

L=101

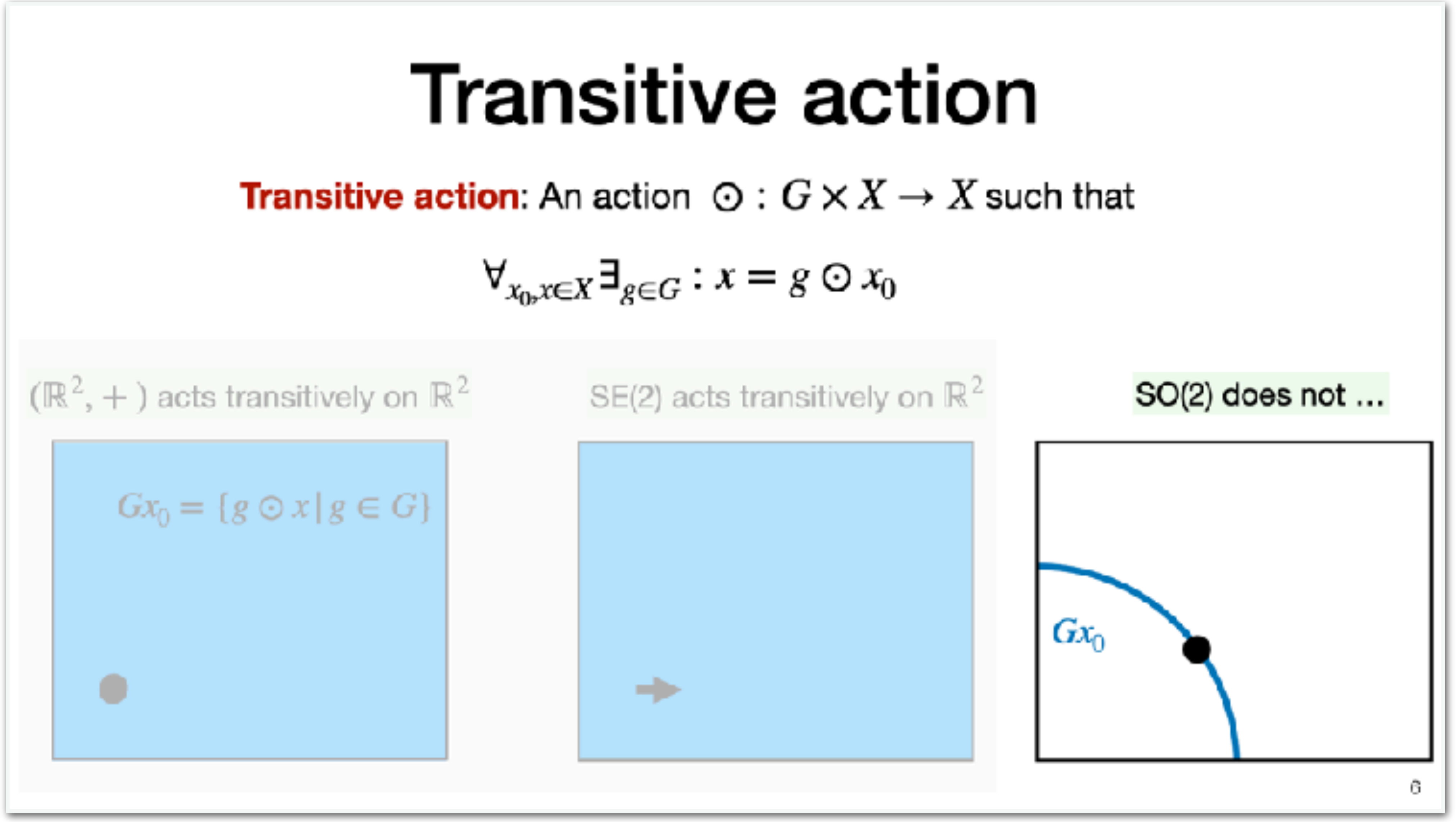


Two dimensional rotation-steerable functions

- 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)



Transitive action

Transitive action: An action $\odot : G \times X \to X$ such that

$$\forall_{x_0, x \in X} \exists_{g \in G} : x = g \odot x_0$$

$(\mathbb{R}^2, +)$ acts transitively on $\mathbb{R}^2$

SE(2) acts transitively on $\mathbb{R}^2$

SO(2) does not …

- 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

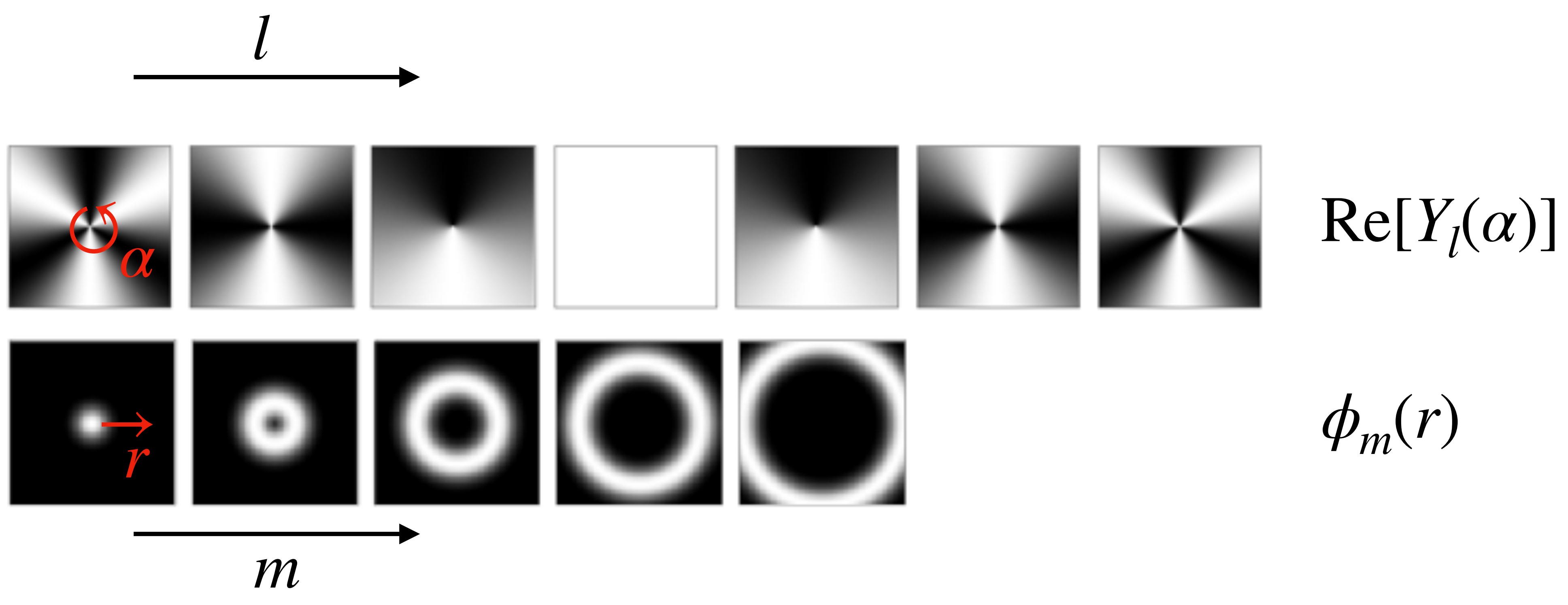
- Consider polar-separable convolution kernel:

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

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

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

$$k^{\rightarrow}(r \mid \mathbf{w}) = \sum_{m} w_m \phi_m(r)$$



- Then we may as well write it as

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

- Then such kernel is clearly rotation steerable!

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

Two dimensional rotation-steerable functions

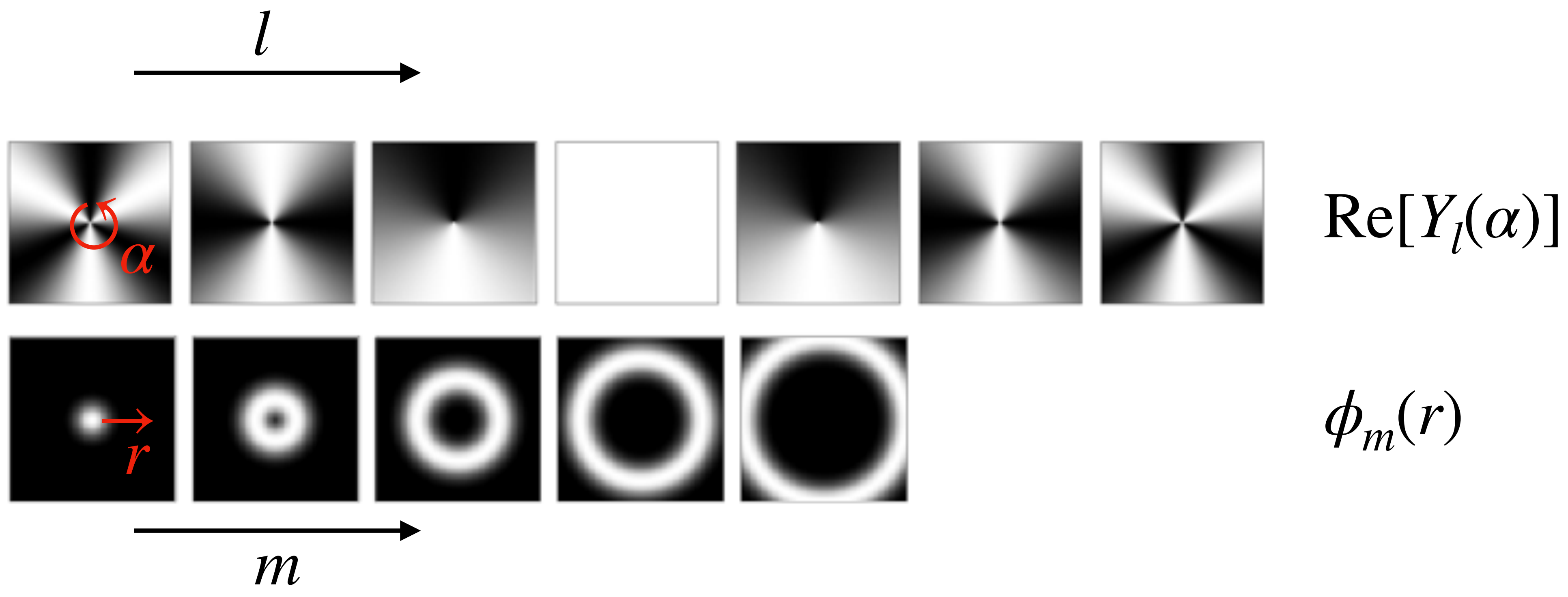
- Consider polar-separable convolution kernel:

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

- 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

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

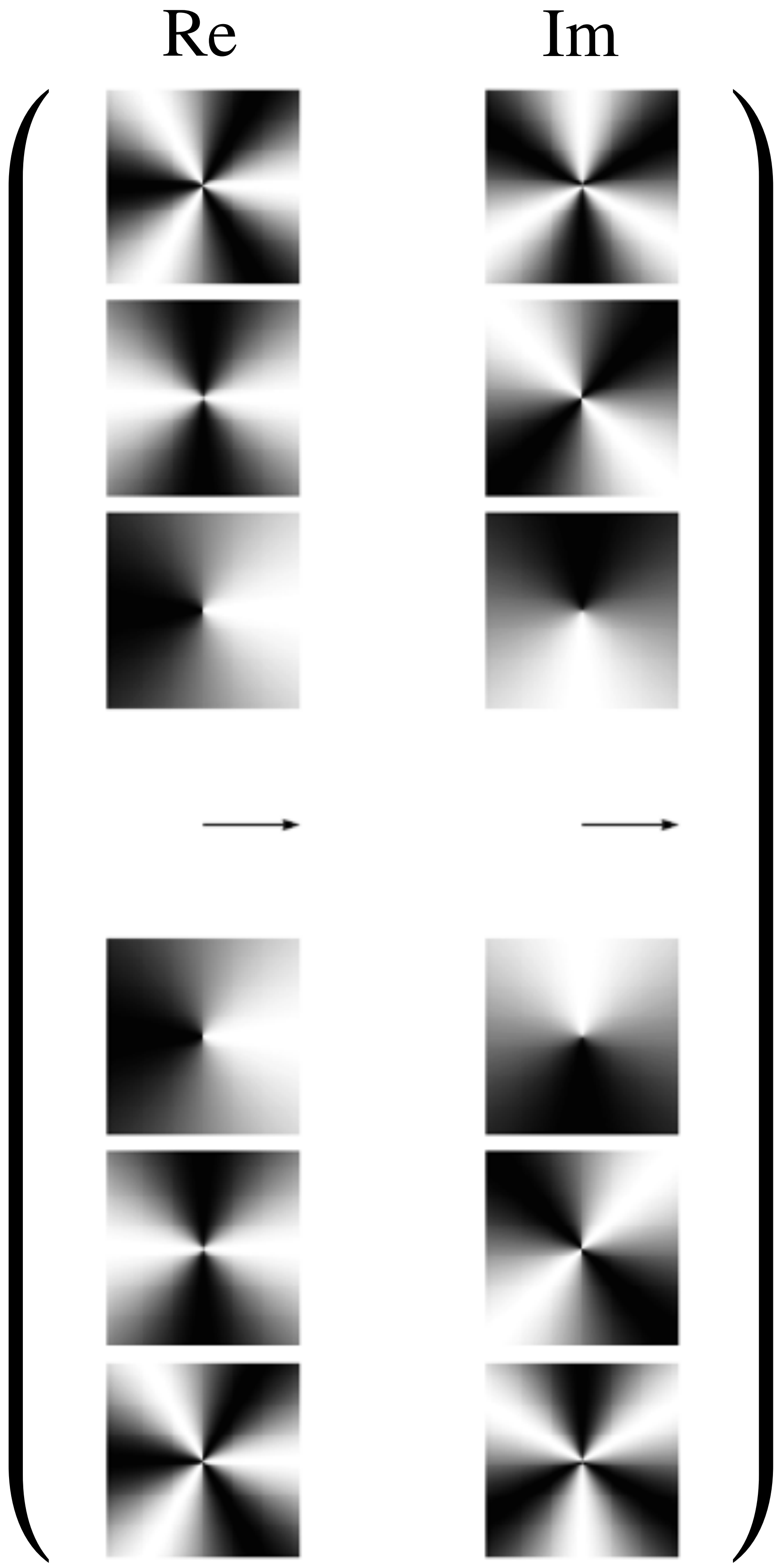
- Then such kernel is clearly rotation steerable!

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

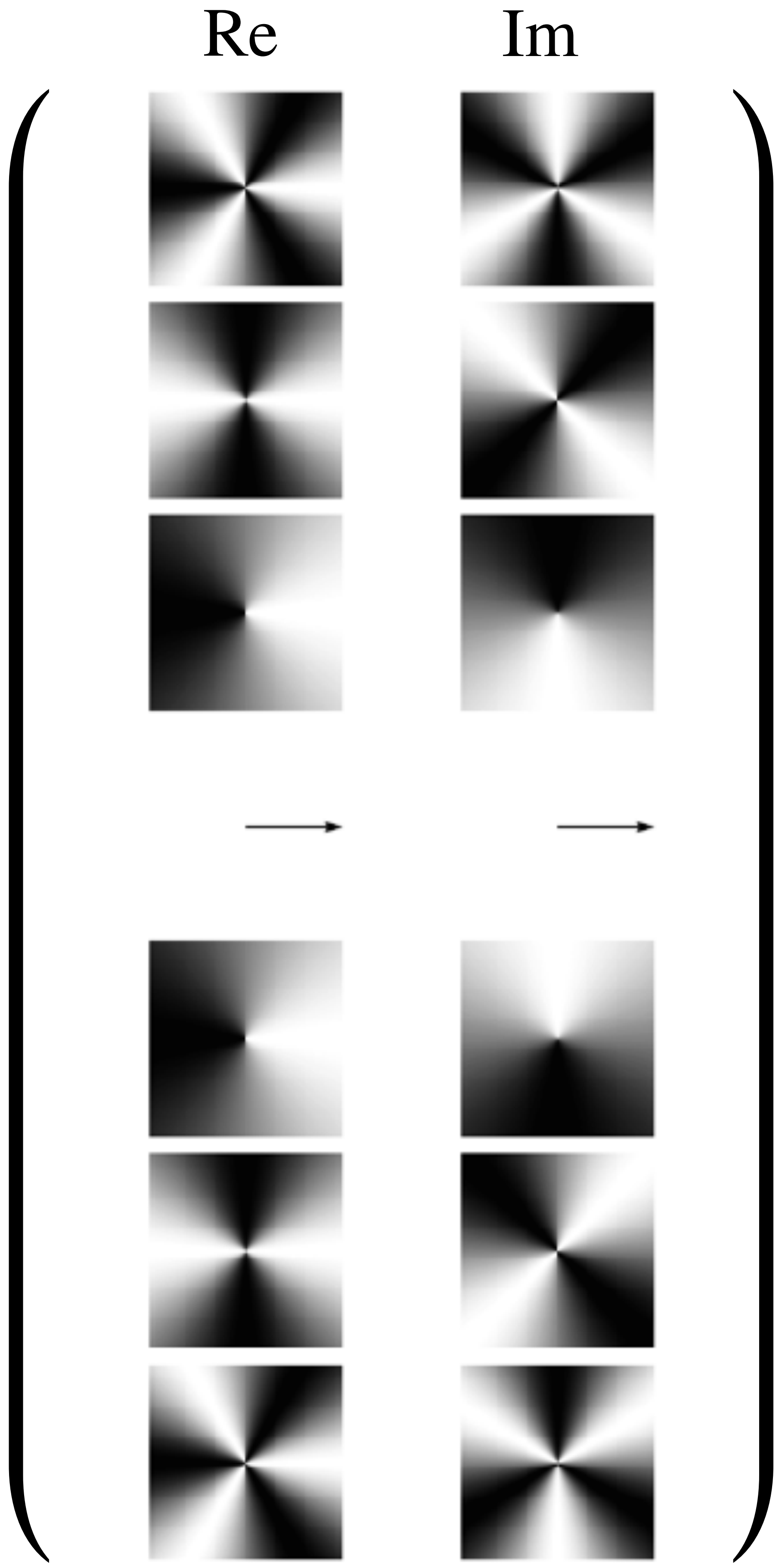
Or directly parametrize as $\hat{\mathbf{w}}(r) = \mathrm{MLP}(r\,|\,\mathbf{w})$!

Complex (irreducible) representations

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

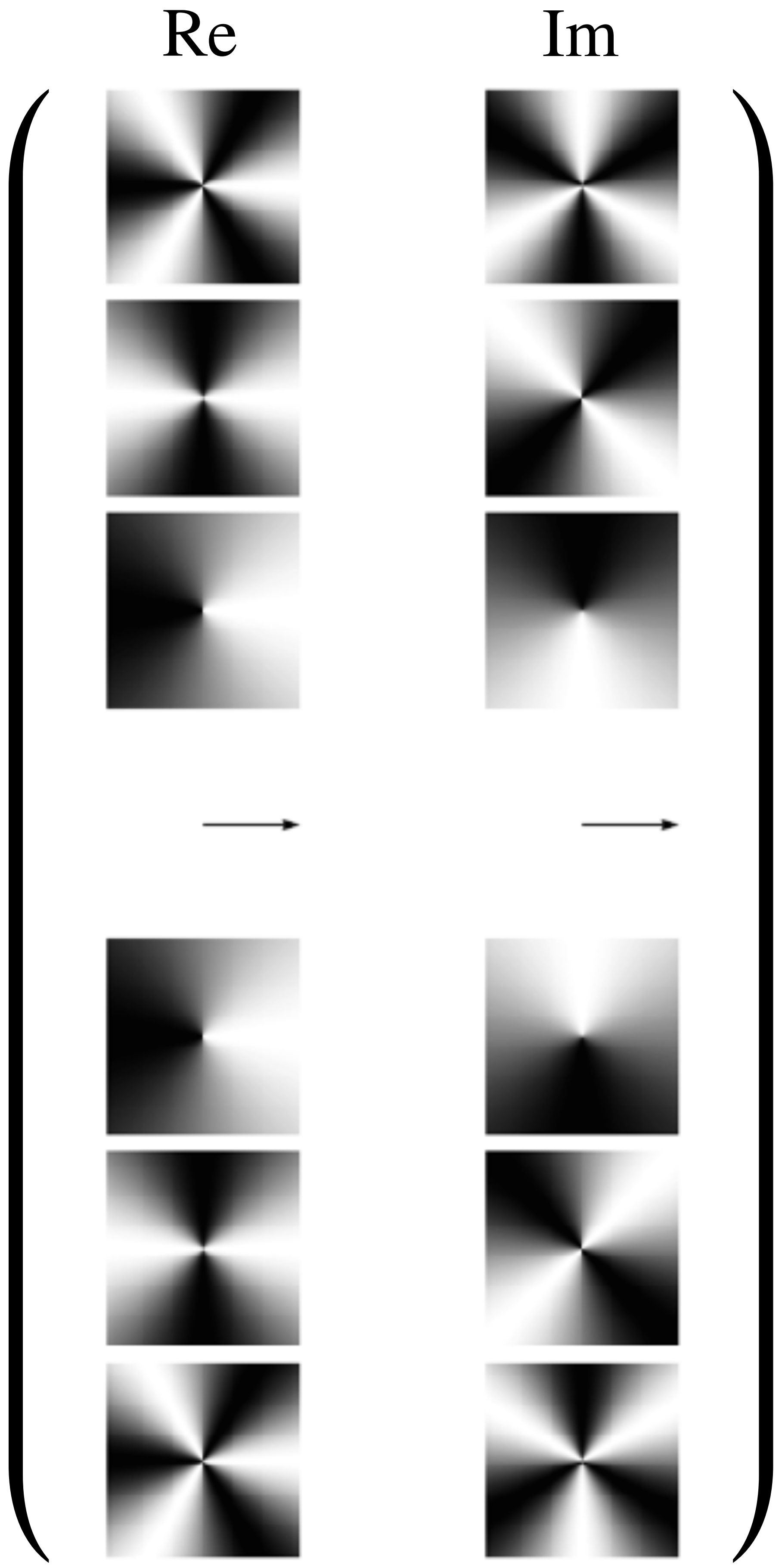


$$= \begin{pmatrix} e^{3i\theta} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{2i\theta} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{1i\theta} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{-1i\theta} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-2i\theta} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-3i\theta} \end{pmatrix}$$

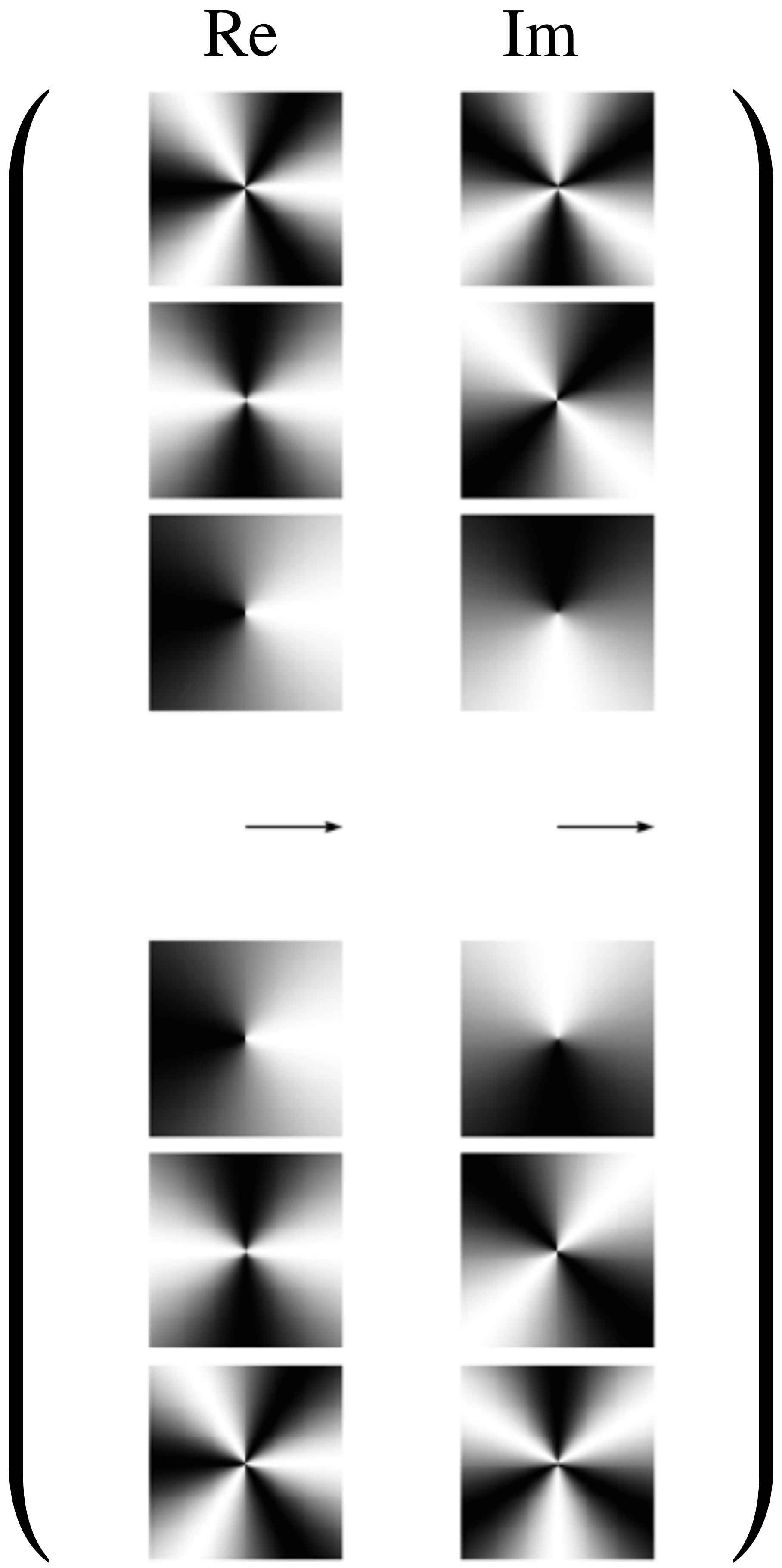


Complex (irreducible) representations

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

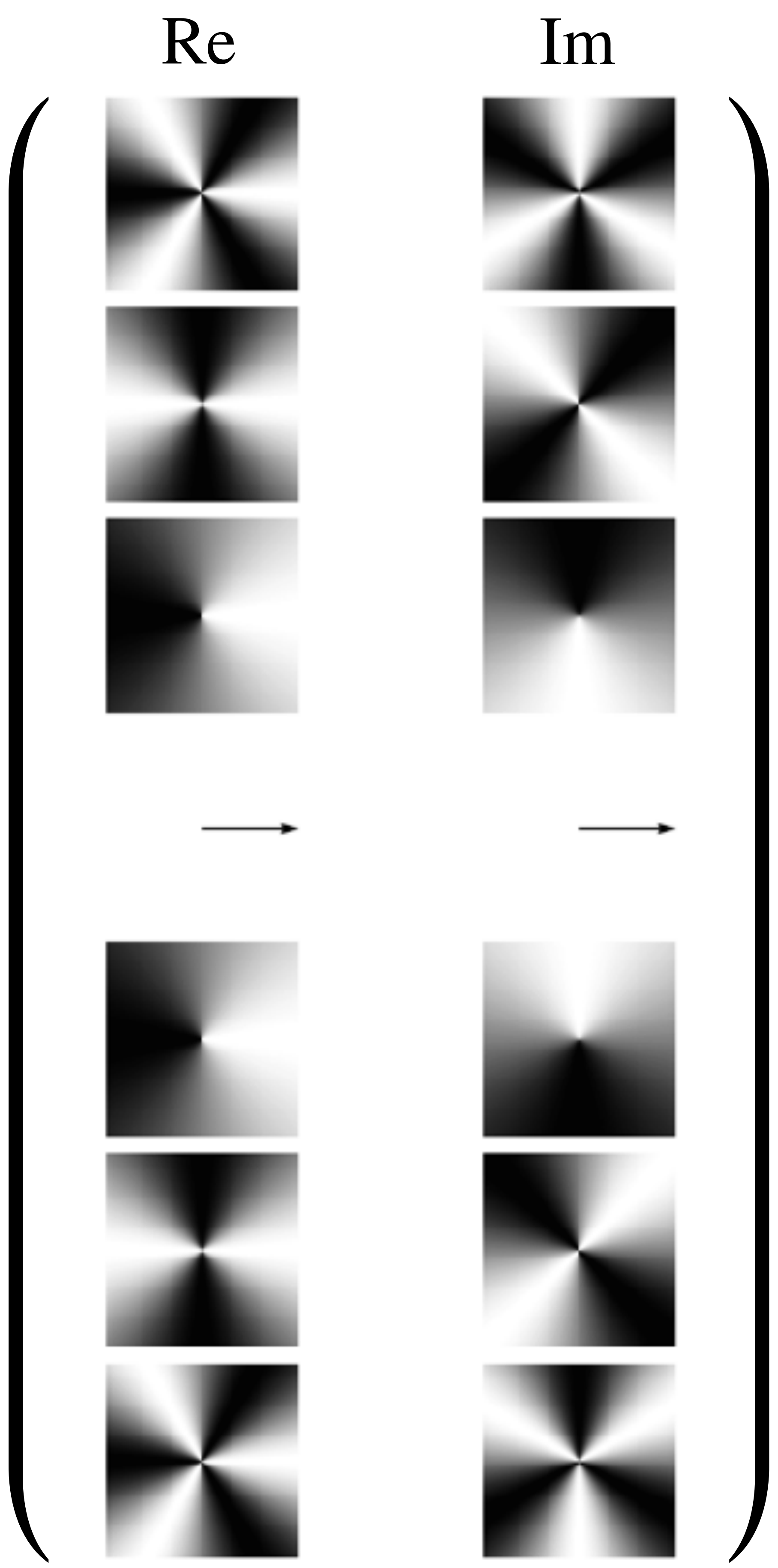


$$= \begin{pmatrix} e^{3\,i\,\theta} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{2\,i\,\theta} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{1\,i\,\theta} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{-1\,i\,\theta} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-2\,i\,\theta} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-3\,i\,\theta}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

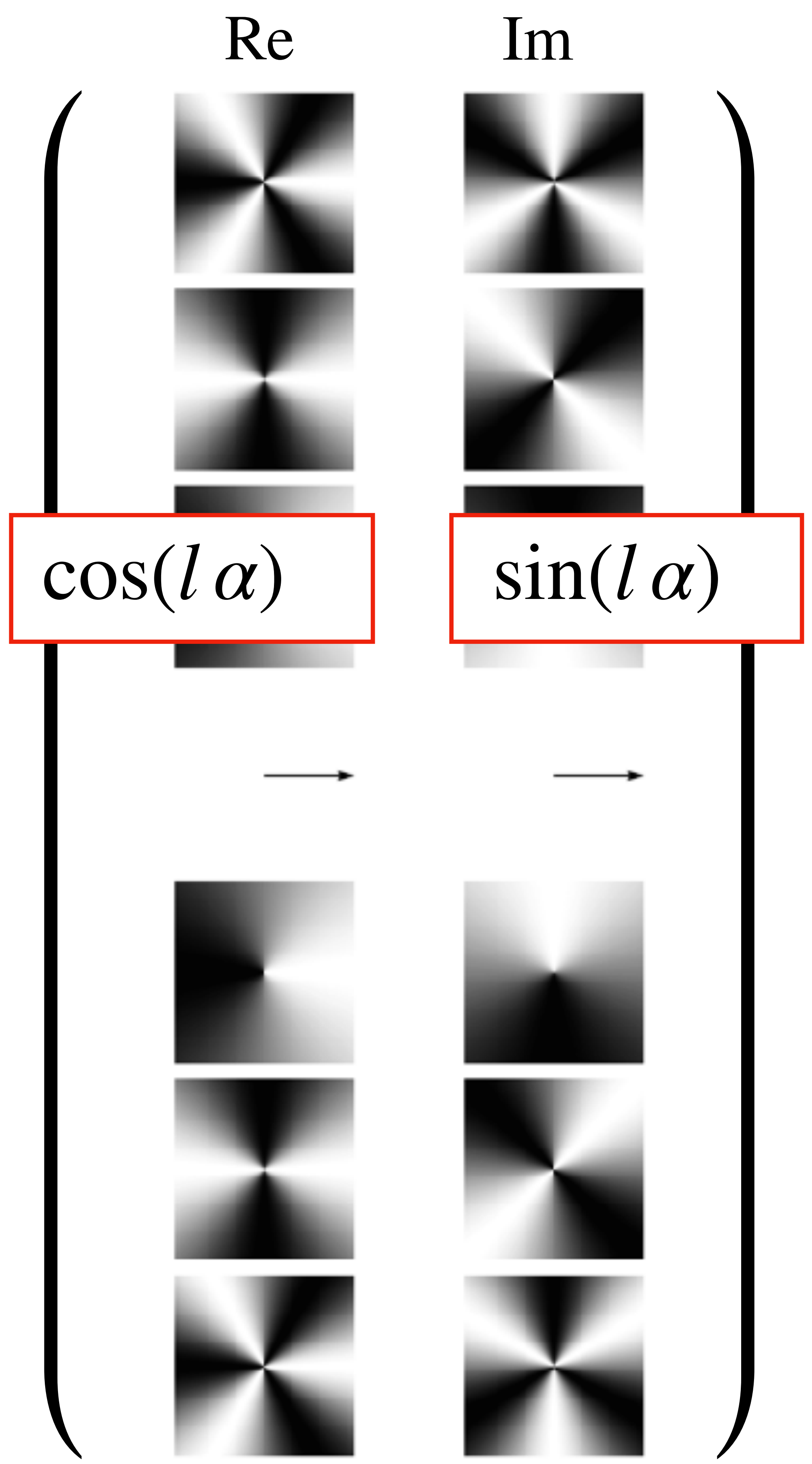


Complex (irreducible) representations

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



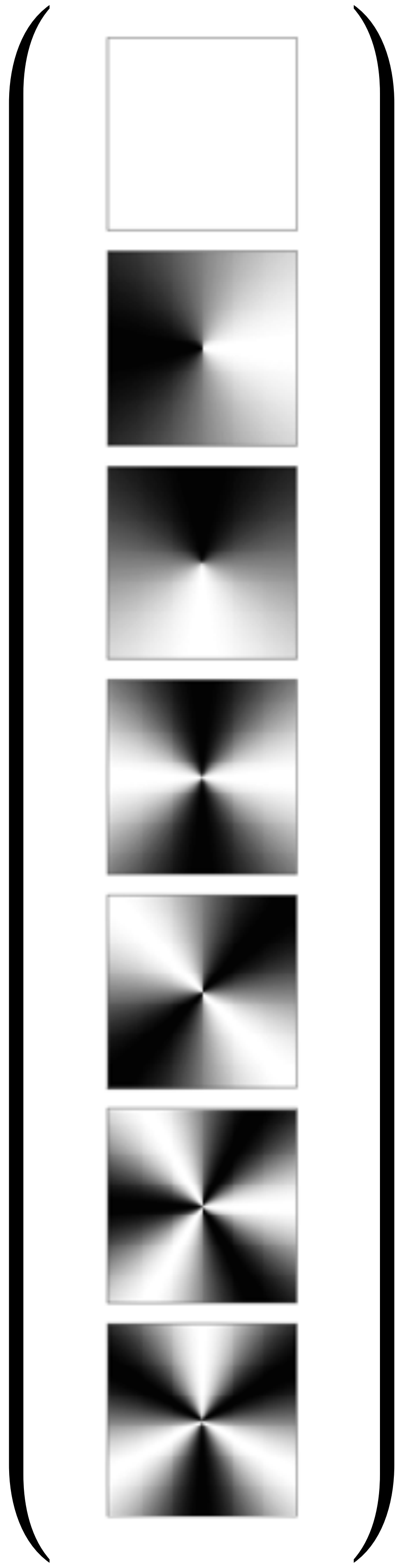
$$=\begin{pmatrix} e^{3i\theta} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{2i\theta} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{1i\theta} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{-1i\theta} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-2i\theta} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-3i\theta} \end{pmatrix}$$



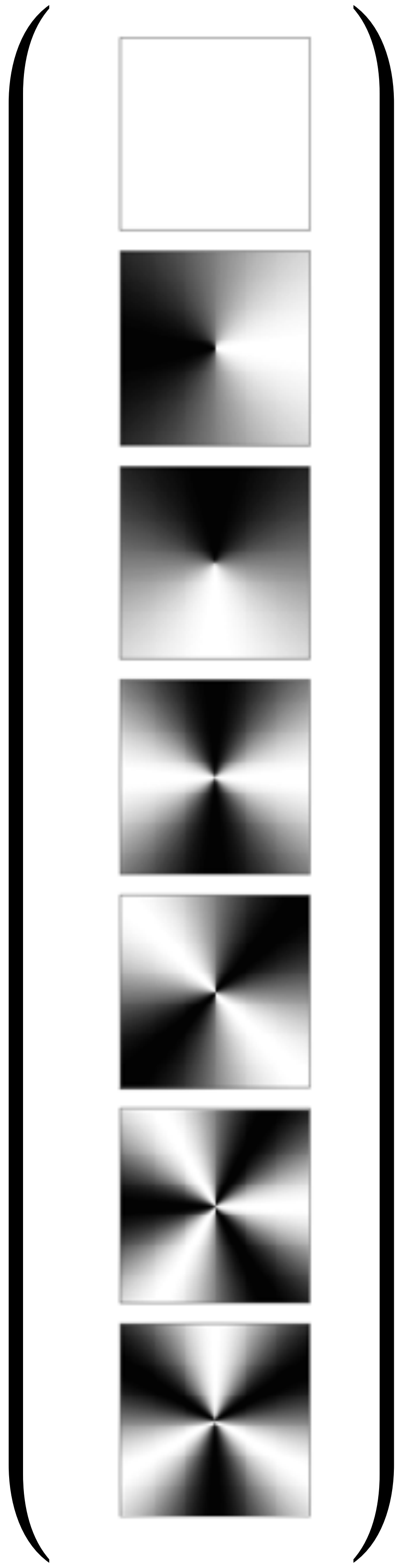
$\cos(l\,\alpha)$ $\sin(l\,\alpha)$

Real (irreducible) representations

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

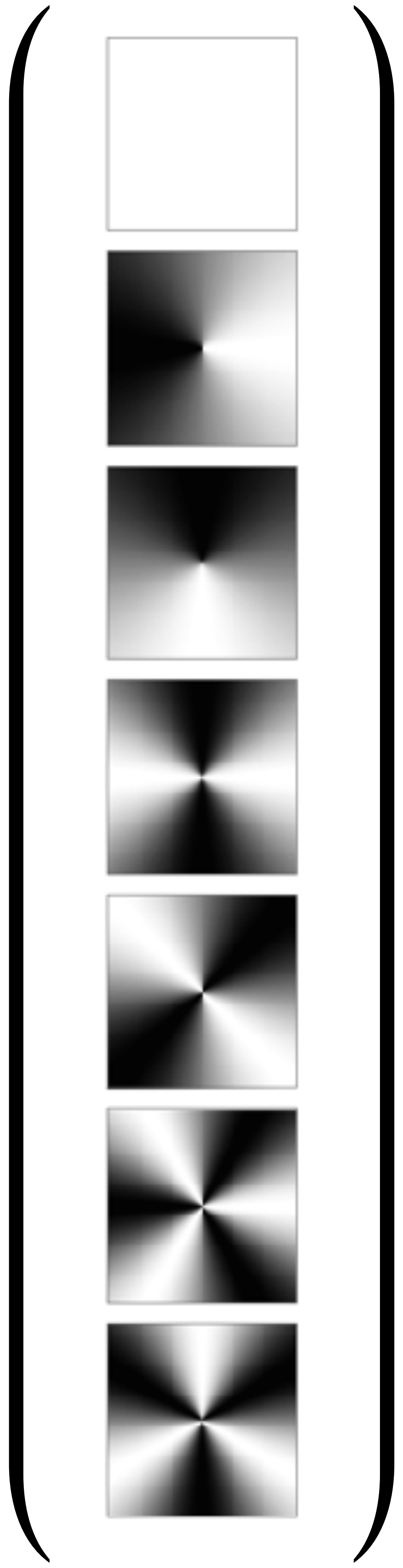


$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 & 0 & 0 & 0 \\ 0 & -\sin\theta & \cos\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos 2\theta & \sin 2\theta & 0 & 0 \\ 0 & 0 & 0 & -\sin 2\theta & \cos 2\theta & 0 & 0 \\ & 0 & 0 & 0 & 0 & \cos 3\theta & \sin 3\theta \\ & 0 & 0 & 0 & 0 & -\sin 3\theta & \cos 3\theta \end{pmatrix}$$

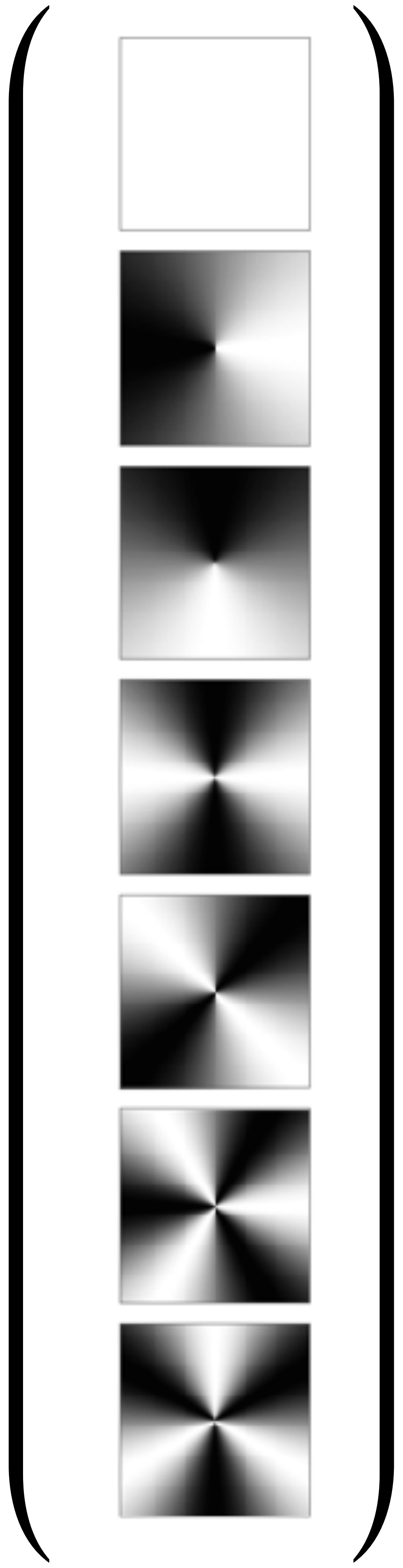


Real (irreducible) representations

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



$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 & 0 & 0 & 0 \\ 0 & -\sin\theta & \cos\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos 2\theta & \sin 2\theta & 0 & 0 \\ 0 & 0 & 0 & -\sin 2\theta & \cos 2\theta & 0 & 0 \\ & 0 & 0 & 0 & 0 & \cos 3\theta & \sin 3\theta \\ & 0 & 0 & 0 & 0 & -\sin 3\theta & \cos 3\theta \end{pmatrix}$$

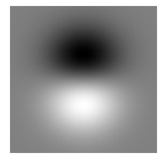


Regular group convolutions revisited

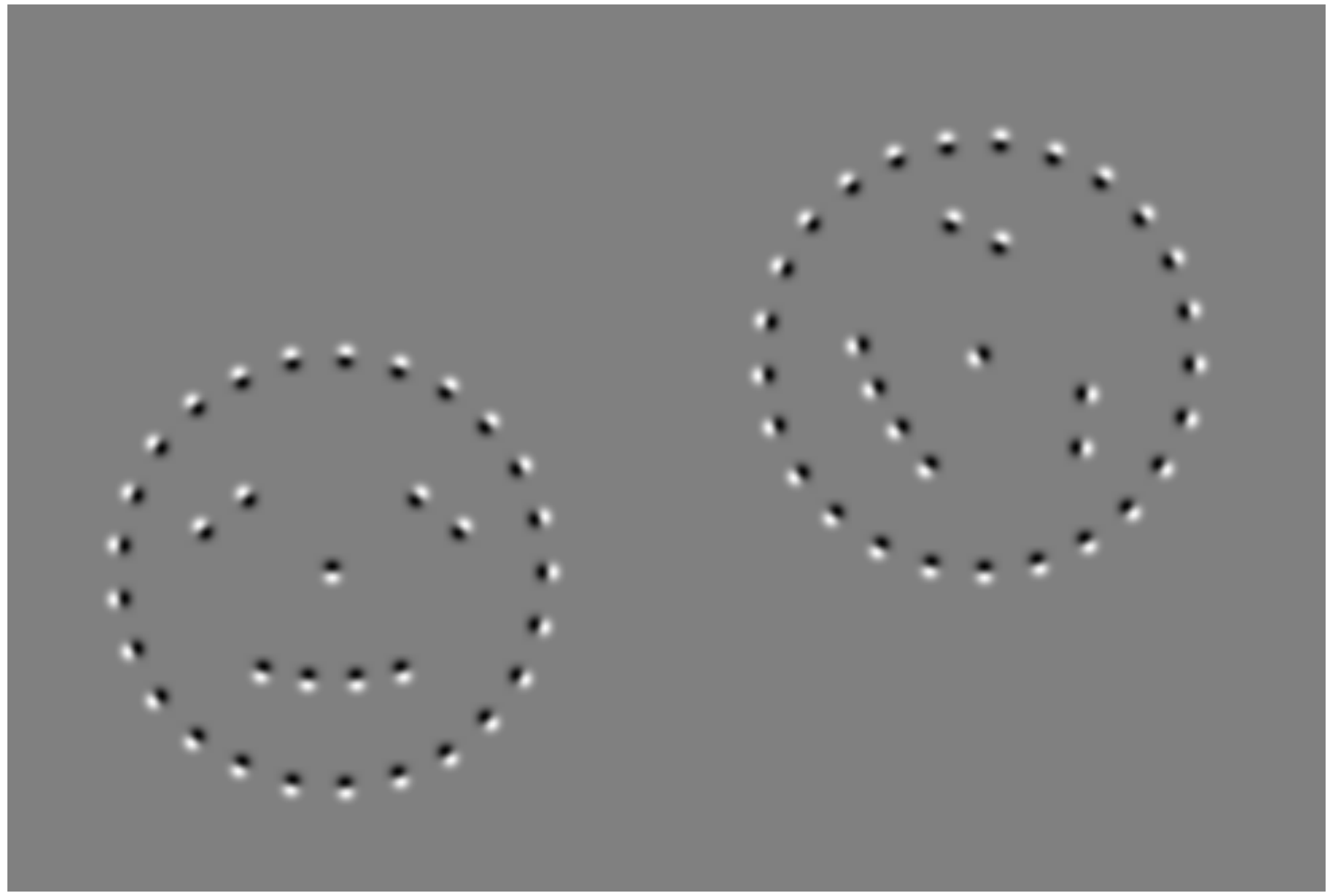
Group convolution ($G = \mathbb{R}^d \rtimes H$):

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

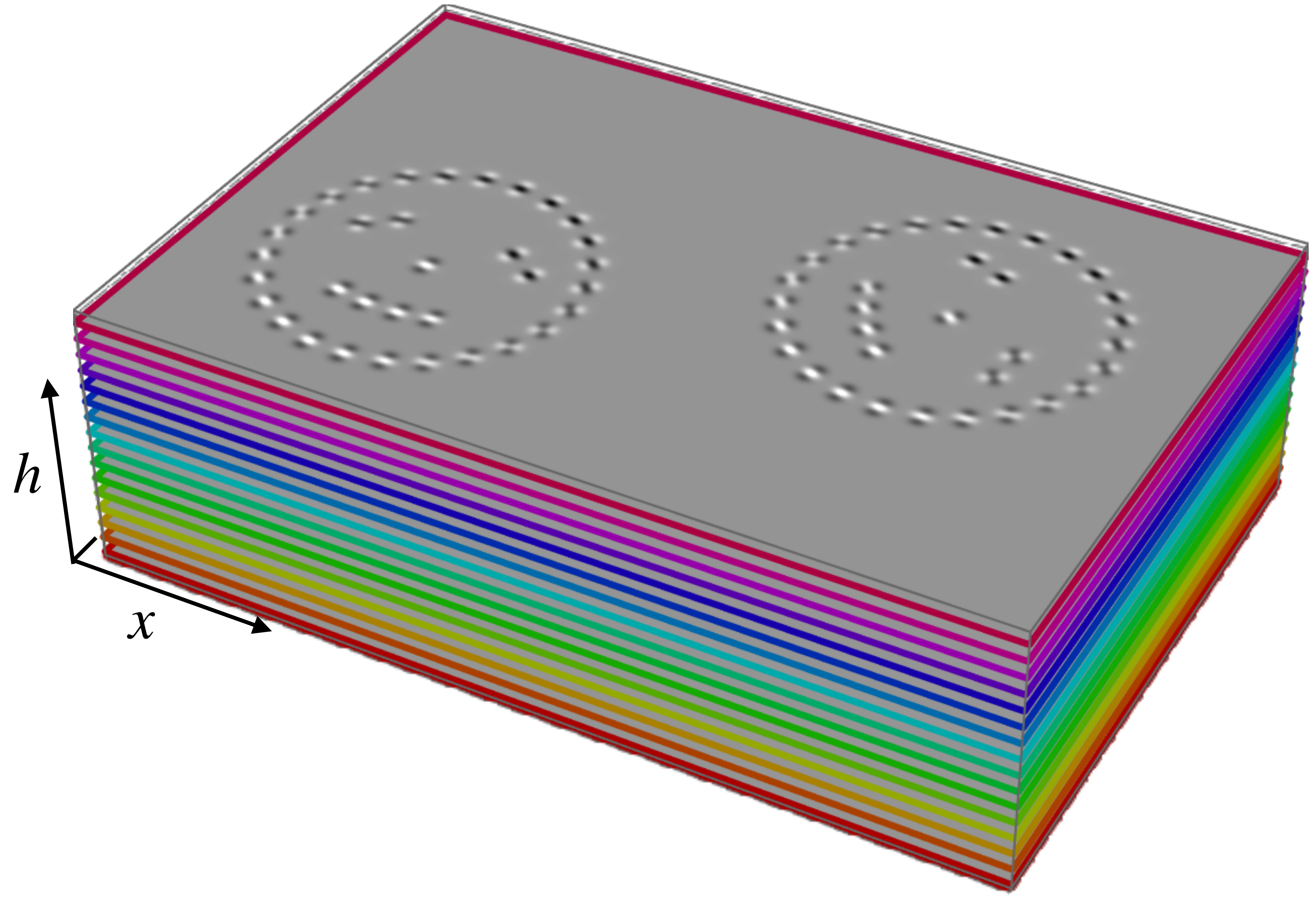
(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$



$\tilde{\star}$



$=$



2D convolution kernel

2D input feature map

SE(2) output feature map

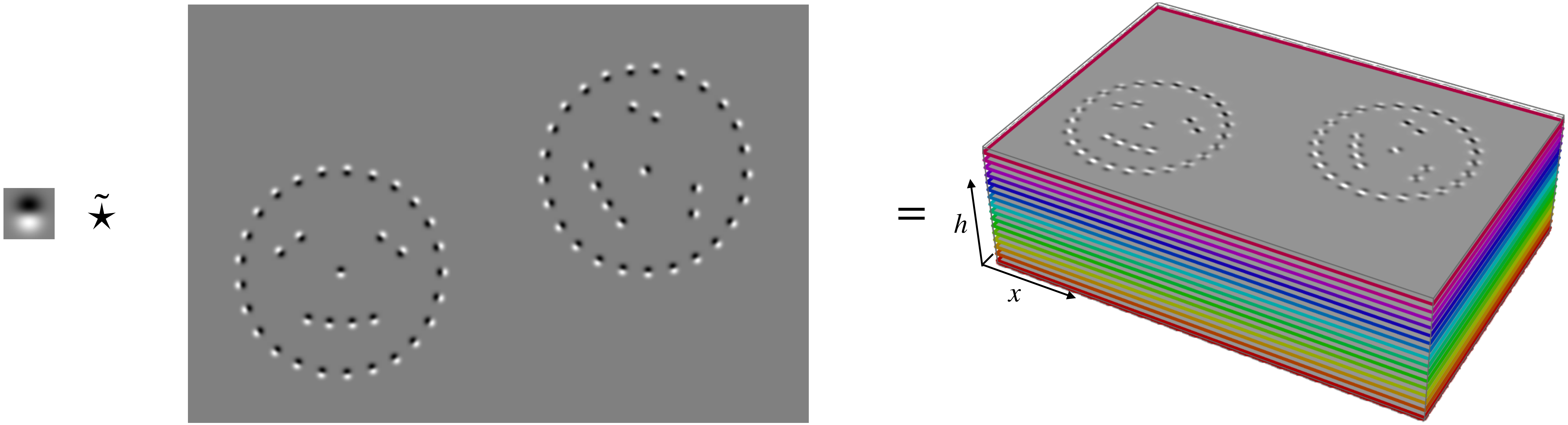
Regular group convolutions revisited

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

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

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

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



2D convolution kernel

2D input feature map

SE(2) output feature map

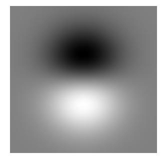
Regular group convolutions revisited

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

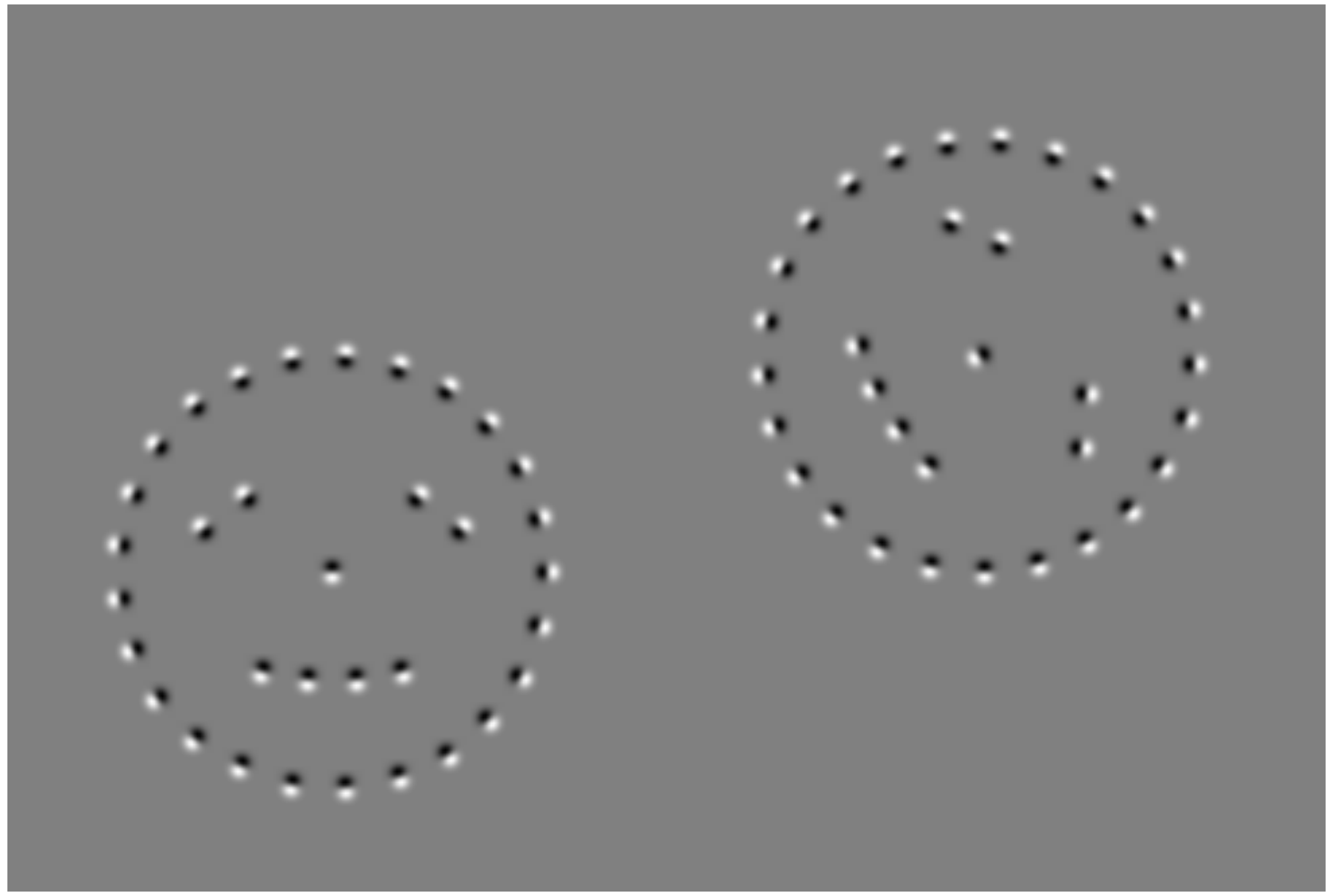
(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

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

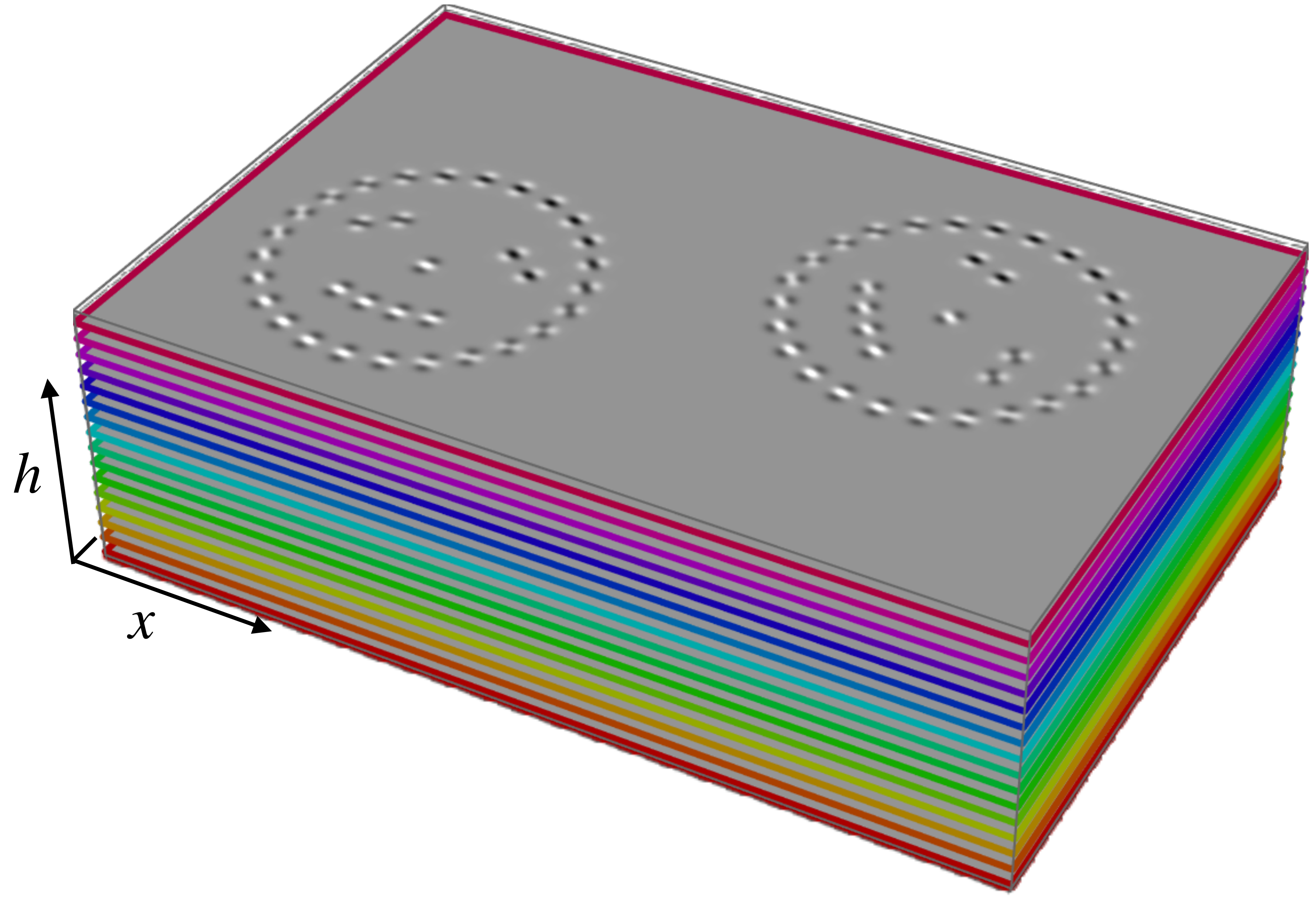
$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}')f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$



$\tilde{\star}$



$=$



2D convolution kernel

2D input feature map

SE(2) output feature map

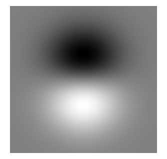
Regular group convolutions revisited

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

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

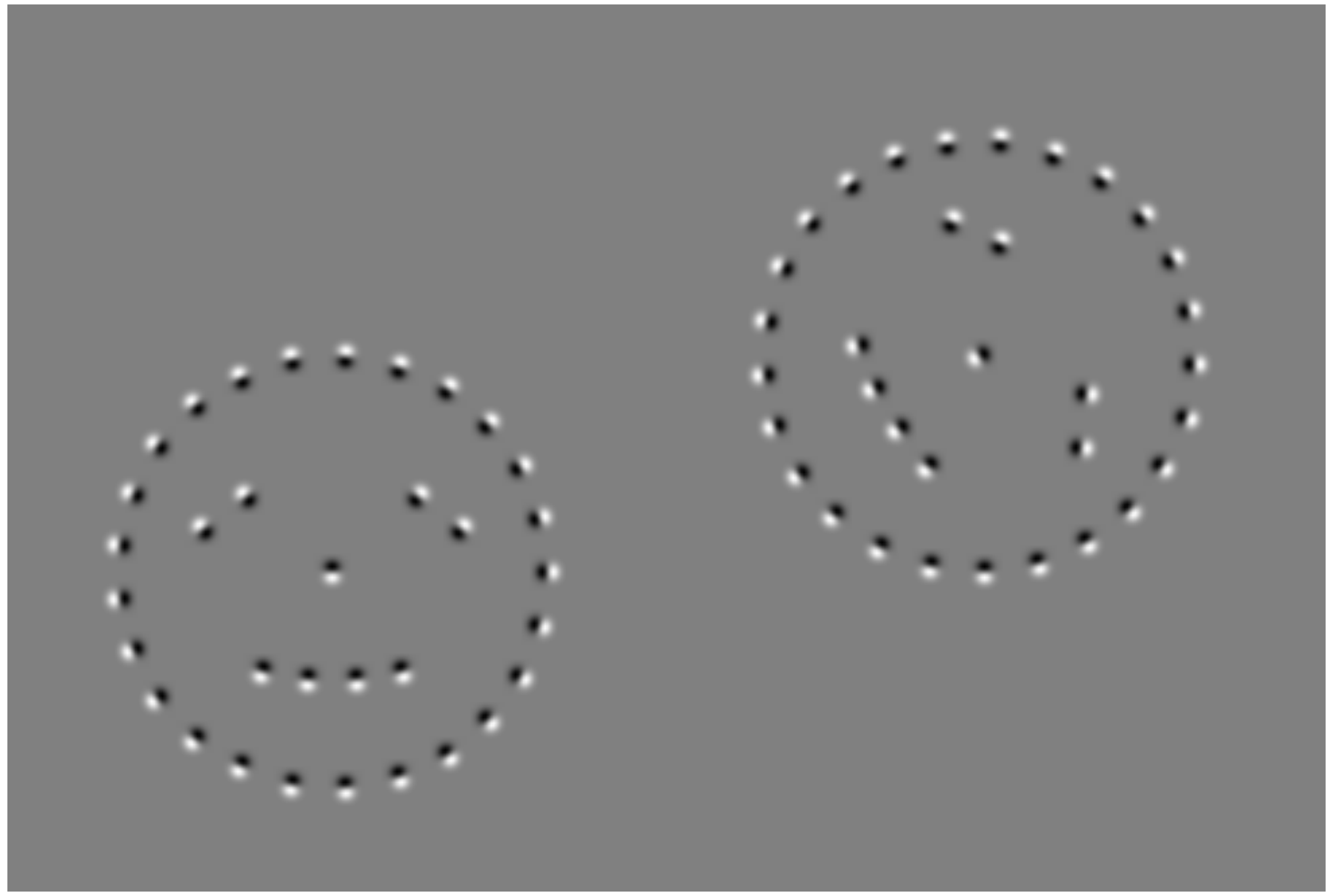
$$(k \,\tilde{\star}\, f)(g) = (\mathcal{L}_g^{G\to\mathbb{L}_2(X)}\, k \,,\, f)_{\mathbb{L}_2(X)} \quad = (\mathcal{L}_{\mathbf{x}}^{(\mathbf{R}^d,+)\to\mathbb{L}_2(\mathbb{R}^d)}\, \mathcal{L}_h^{H\to\mathbb{L}_2(\mathbb{R}^d)}\, k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

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



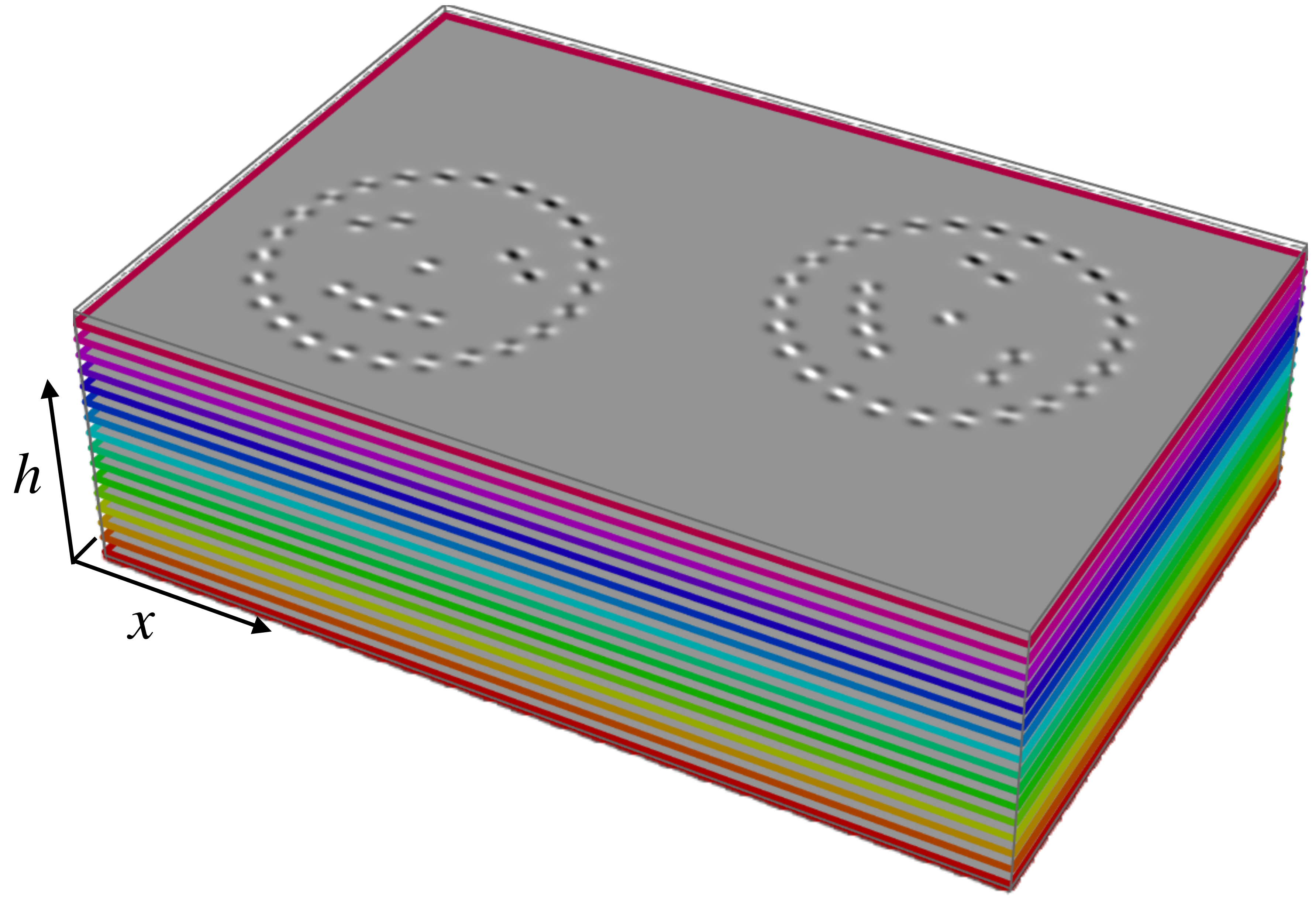
2D convolution kernel

$\tilde{\star}$



2D input feature map

$=$



SE(2) output feature map

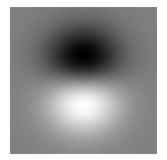
Regular group convolutions revisited

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

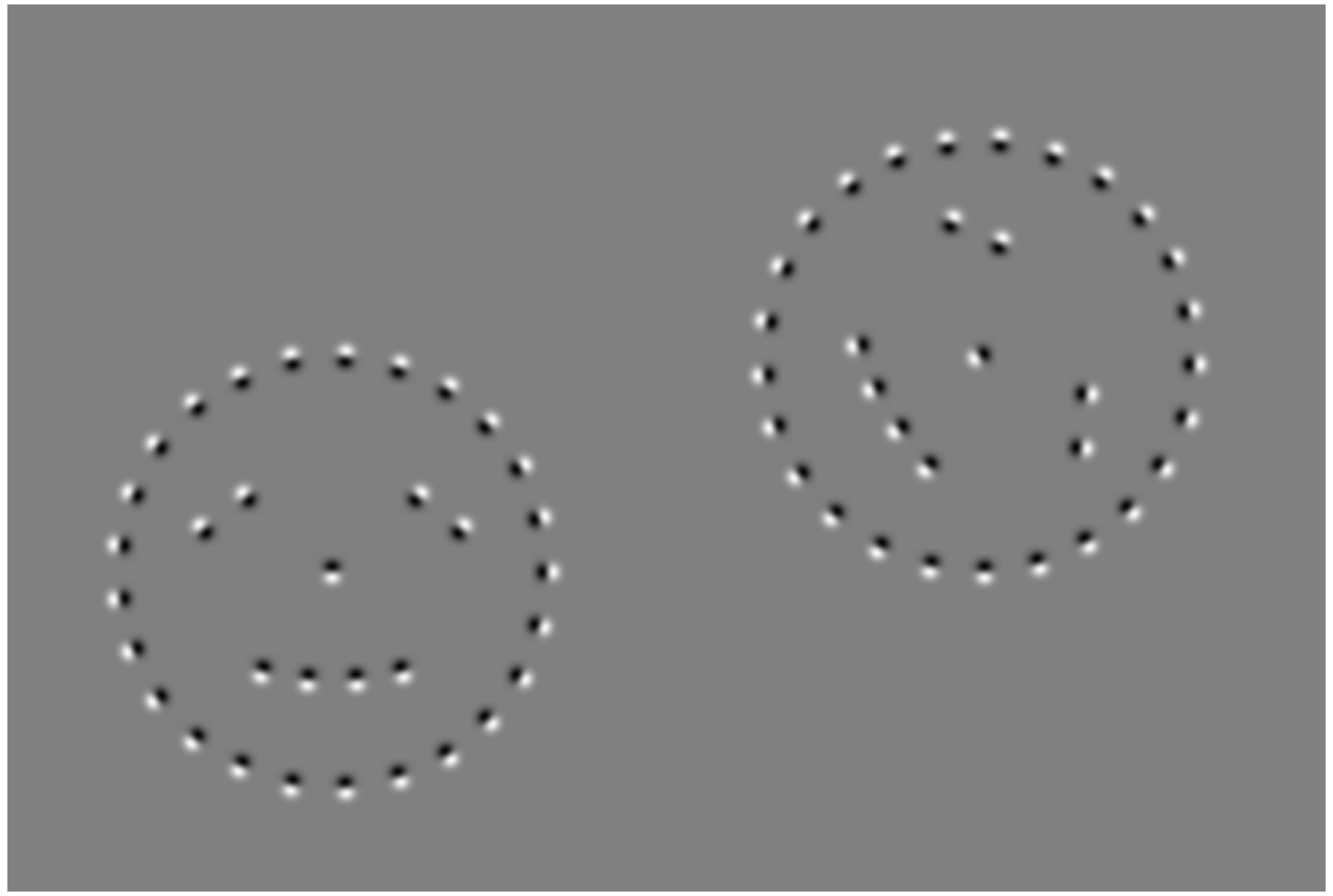
(e.g. $G = SE(2) = \mathbb{R}^2 \rtimes SO(2)$)
$X = \mathbb{R}^2$

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

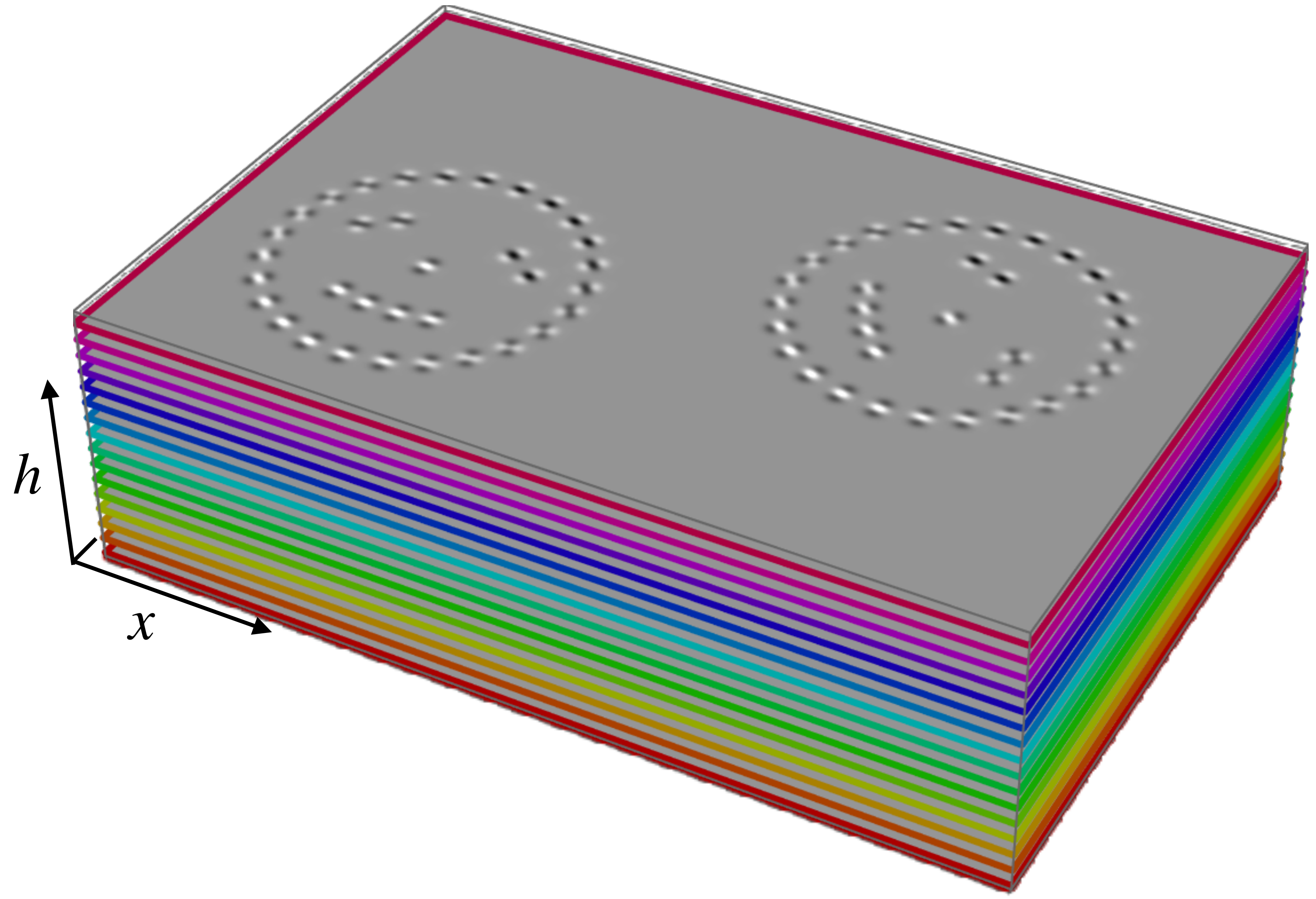
$$= \int_{\mathbb{R}^d} k(g^{-1}\mathbf{x}') f(\mathbf{x}') \mathrm{d}\mathbf{x}' \quad = \int_{\mathbb{R}^2} k_h(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$



$\tilde{\star}$



$=$



2D convolution kernel

2D input feature map

SE(2) output feature map

Regular group convolutions revisited

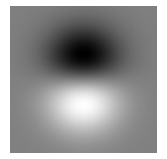
Group convolution ($G = \mathbb{R}^d \rtimes H$):

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

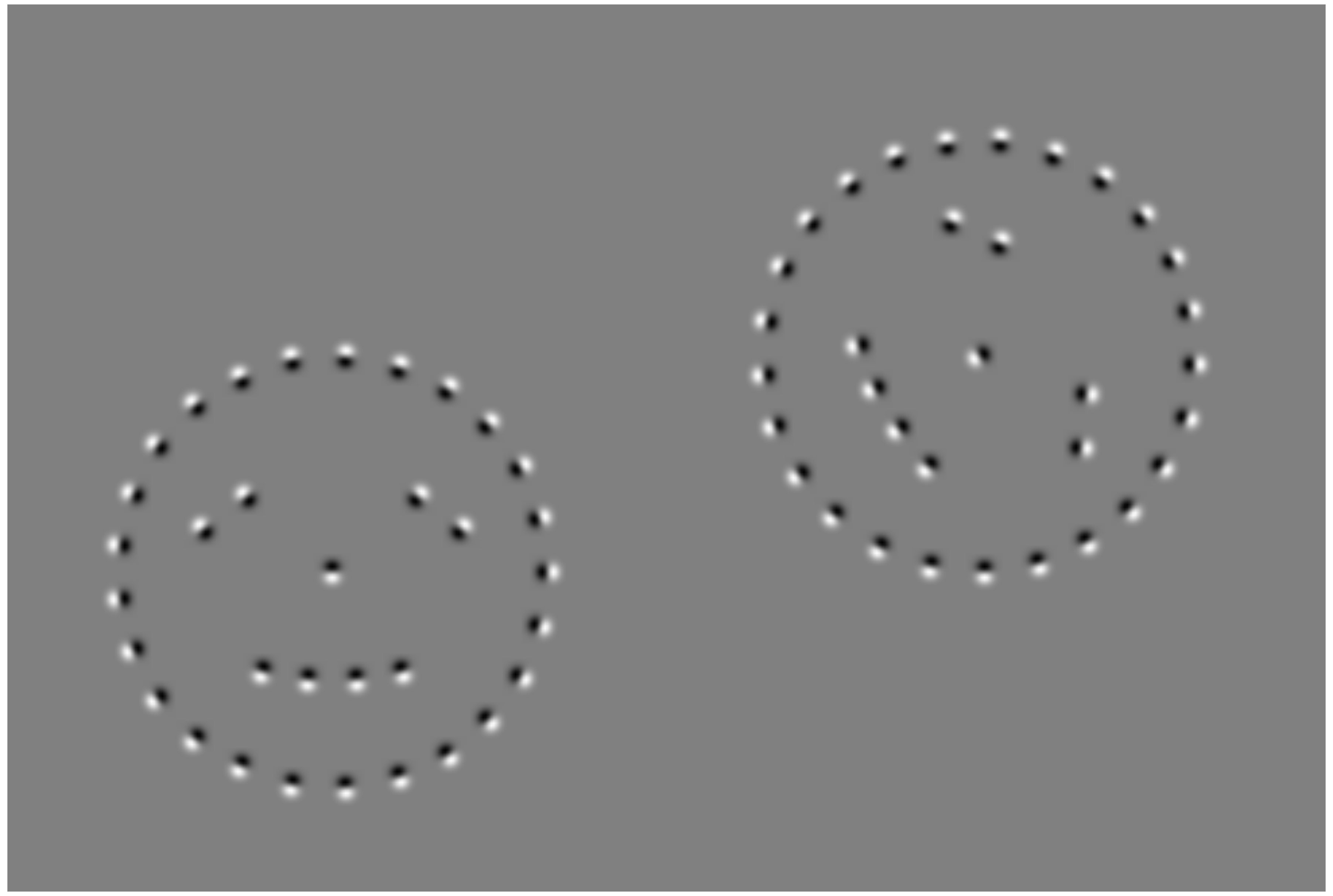
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)}\, k \,,\, f)_{\mathbb{L}_2(X)} \quad = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)}\, k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

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

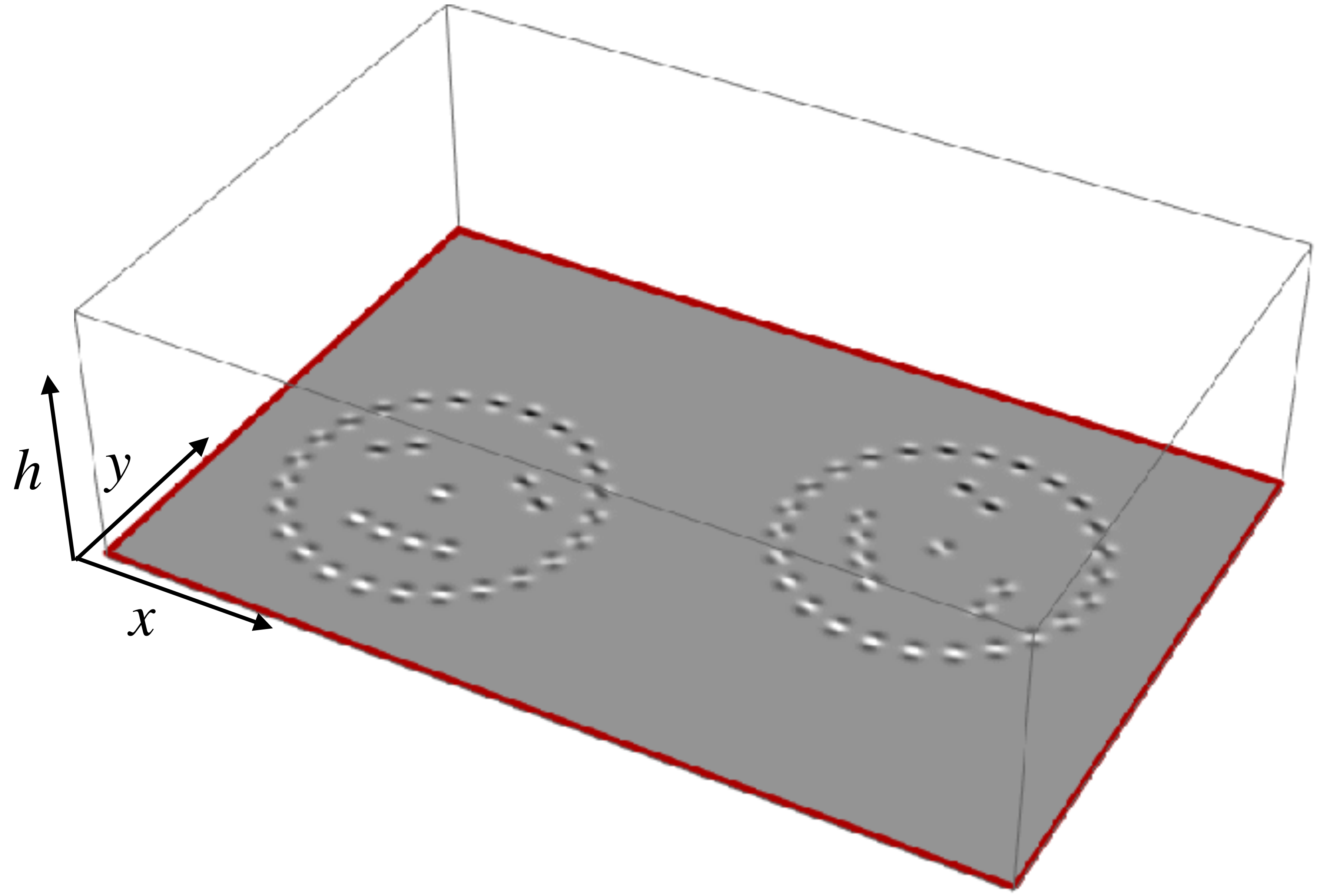
translation
“template matching”



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



=



Rotated
2D convolution kernel

2D input feature map

SE(2) output feature map

Regular group convolutions revisited

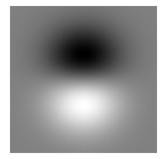
Group convolution ($G = \mathbb{R}^d \rtimes H$):

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

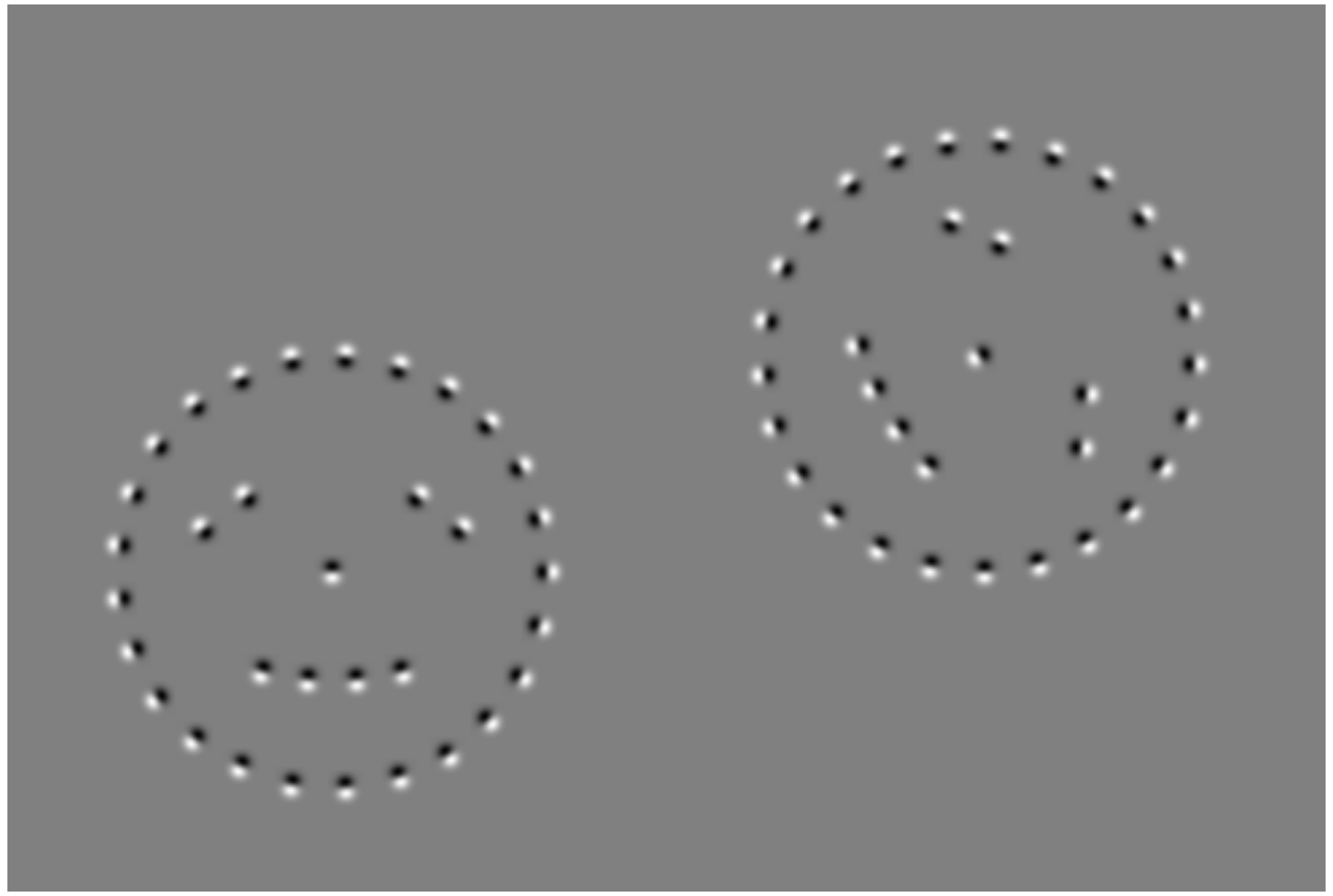
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G \to \mathbb{L}_2(X)}\, k \,,\, f)_{\mathbb{L}_2(X)} \quad = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+) \to \mathbb{L}_2(\mathbb{R}^d)}\, \mathscr{L}_h^{H \to \mathbb{L}_2(\mathbb{R}^d)}\, k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

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

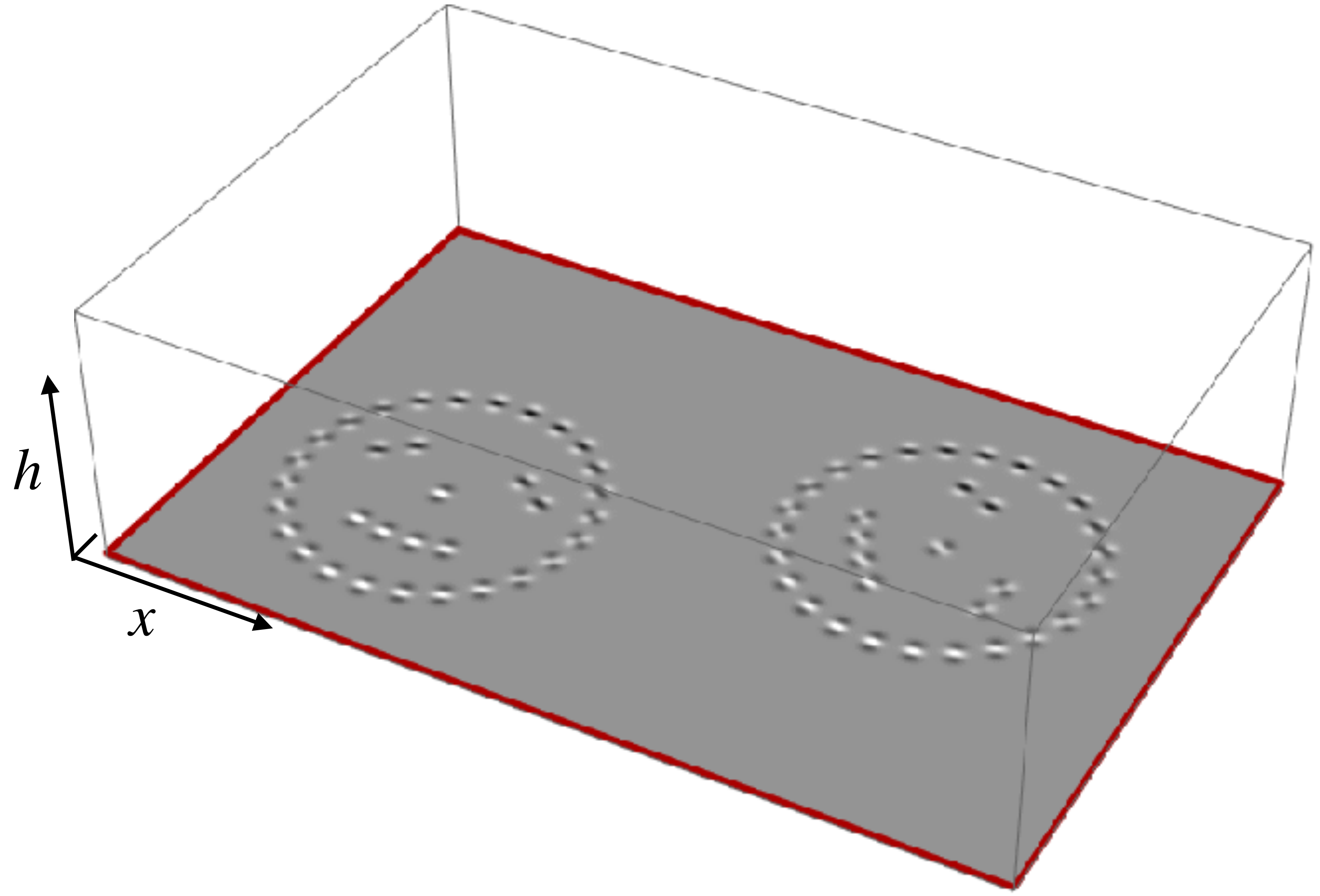
translation
“template matching”



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



$=$



Rotated
2D convolution kernel

2D input feature map

SE(2) output feature map

Regular group convolutions revisited

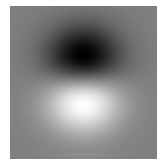
Group convolution ($G = \mathbb{R}^d \rtimes H$):

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

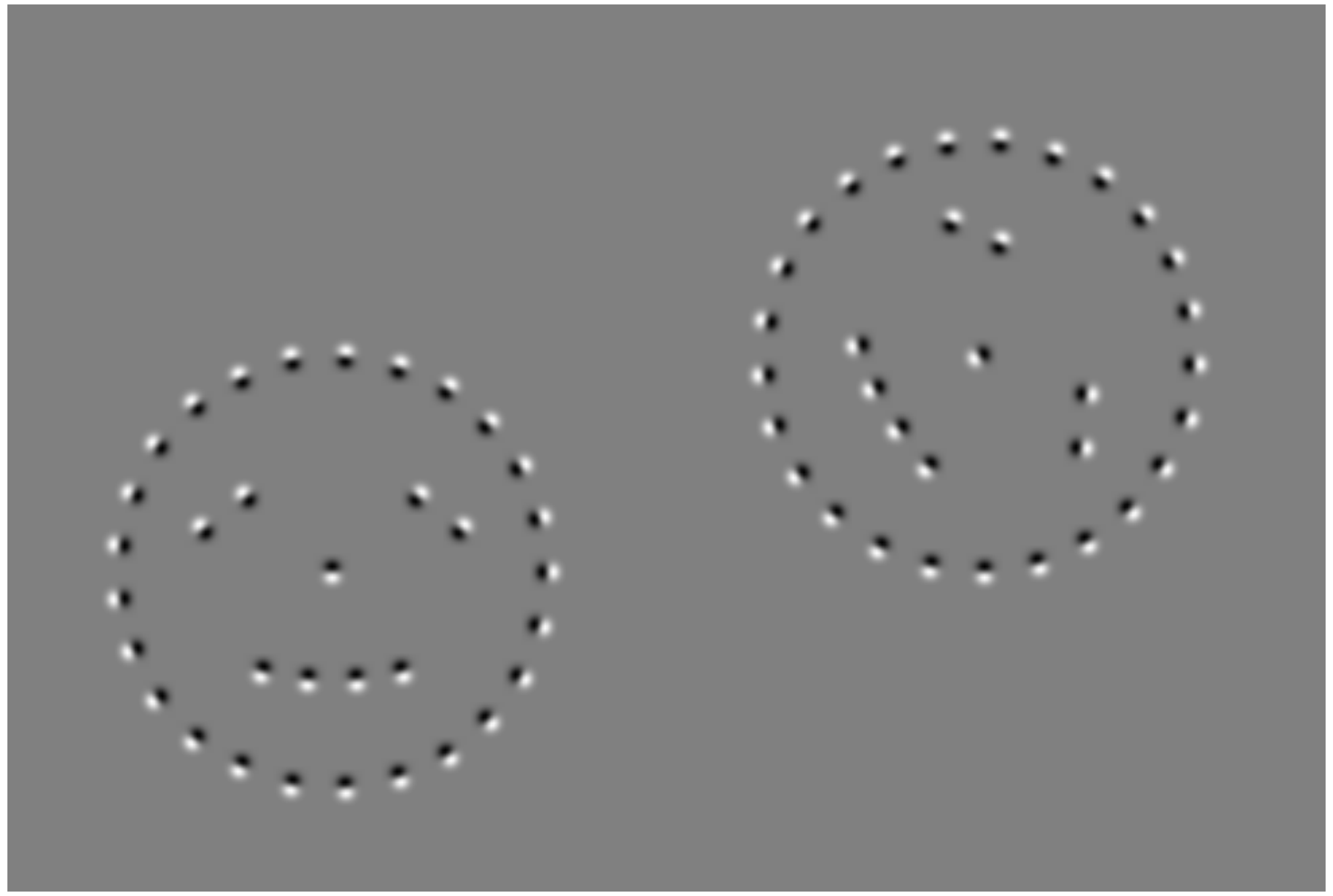
$$(k \,\tilde{\star}\, f)(g) = (\mathscr{L}_g^{G\to\mathbb{L}_2(X)} k \,,\, f)_{\mathbb{L}_2(X)} = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+)\to\mathbb{L}_2(\mathbb{R}^d)} \mathscr{L}_h^{H\to\mathbb{L}_2(\mathbb{R}^d)} k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

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

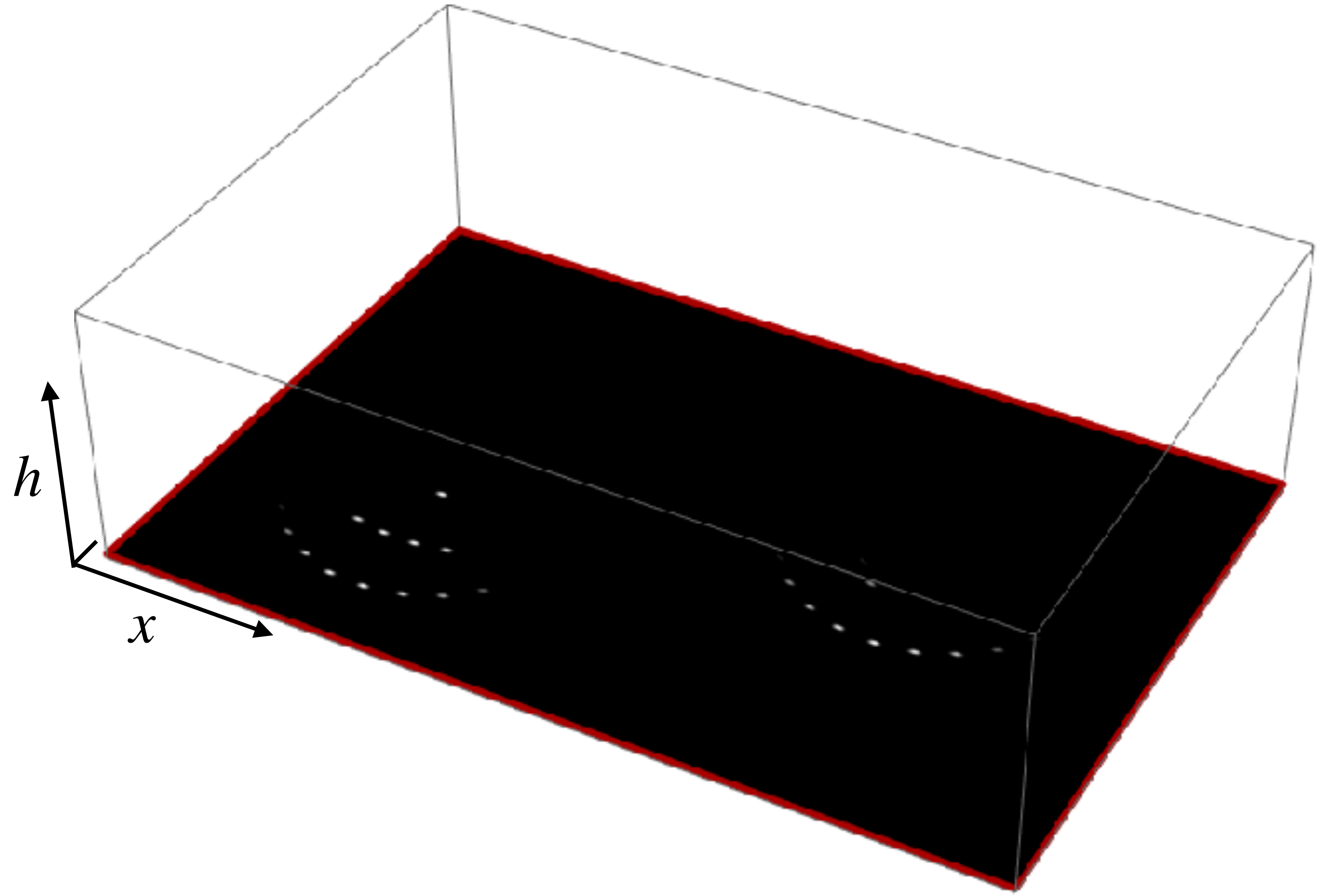
translation
“template matching”



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



=



Rotated
2D convolution kernel

2D input feature map

SE(2) output feature map (after ReLU)

Regular group convolutions revisited

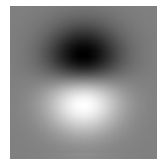
Group convolution ($G = \mathbb{R}^d \rtimes H$):

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

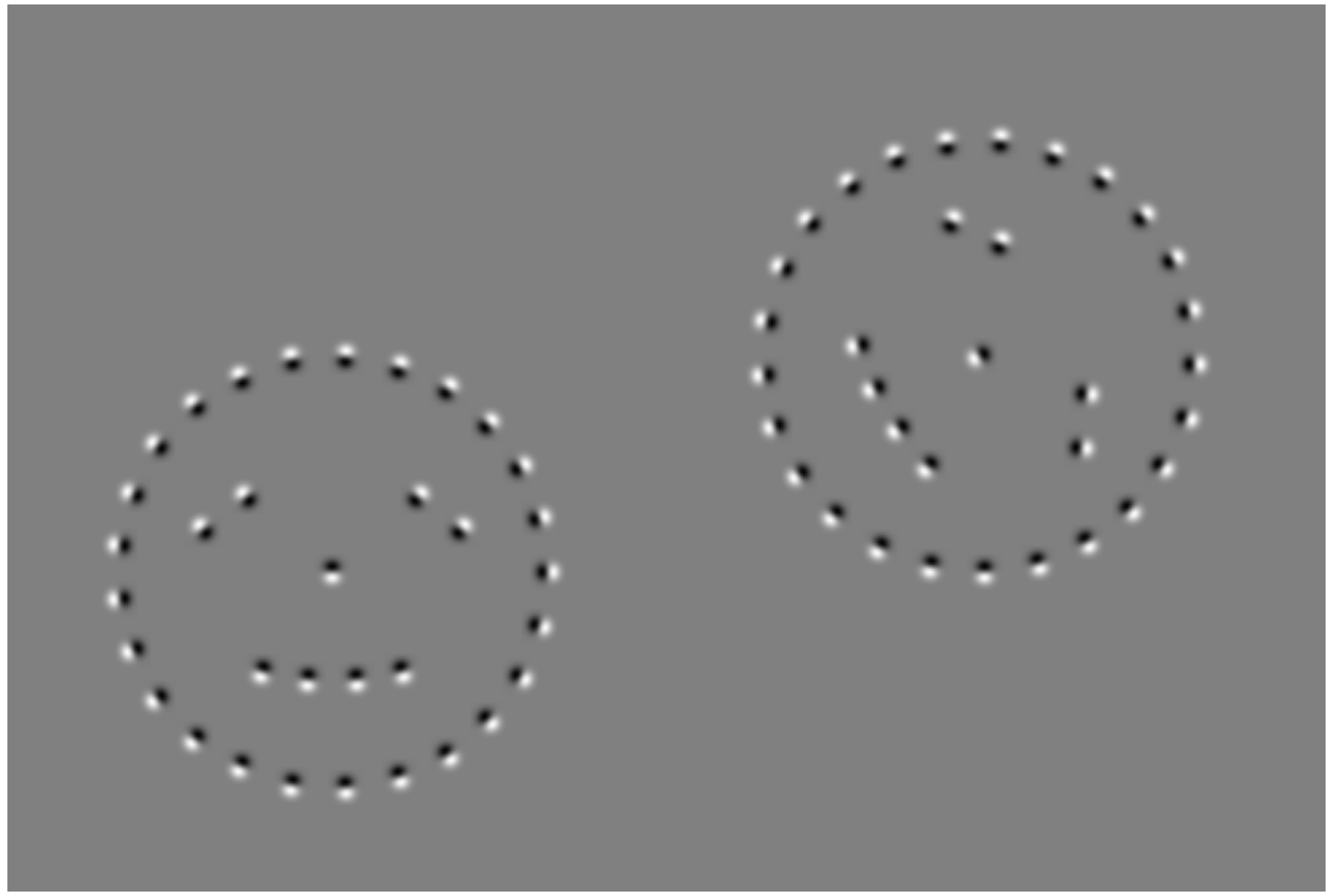
$$(k \tilde{\star} f)(g) = (\mathscr{L}_g^{G\to\mathbb{L}_2(X)}\, k \,,\, f)_{\mathbb{L}_2(X)} \quad = (\mathscr{L}_{\mathbf{x}}^{(\mathbf{R}^d,+)\to\mathbb{L}_2(\mathbb{R}^d)}\, \mathscr{L}_h^{H\to\mathbb{L}_2(\mathbb{R}^d)}\, k \,,\, f)_{\mathbb{L}_2(\mathbb{R}^d)}$$

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

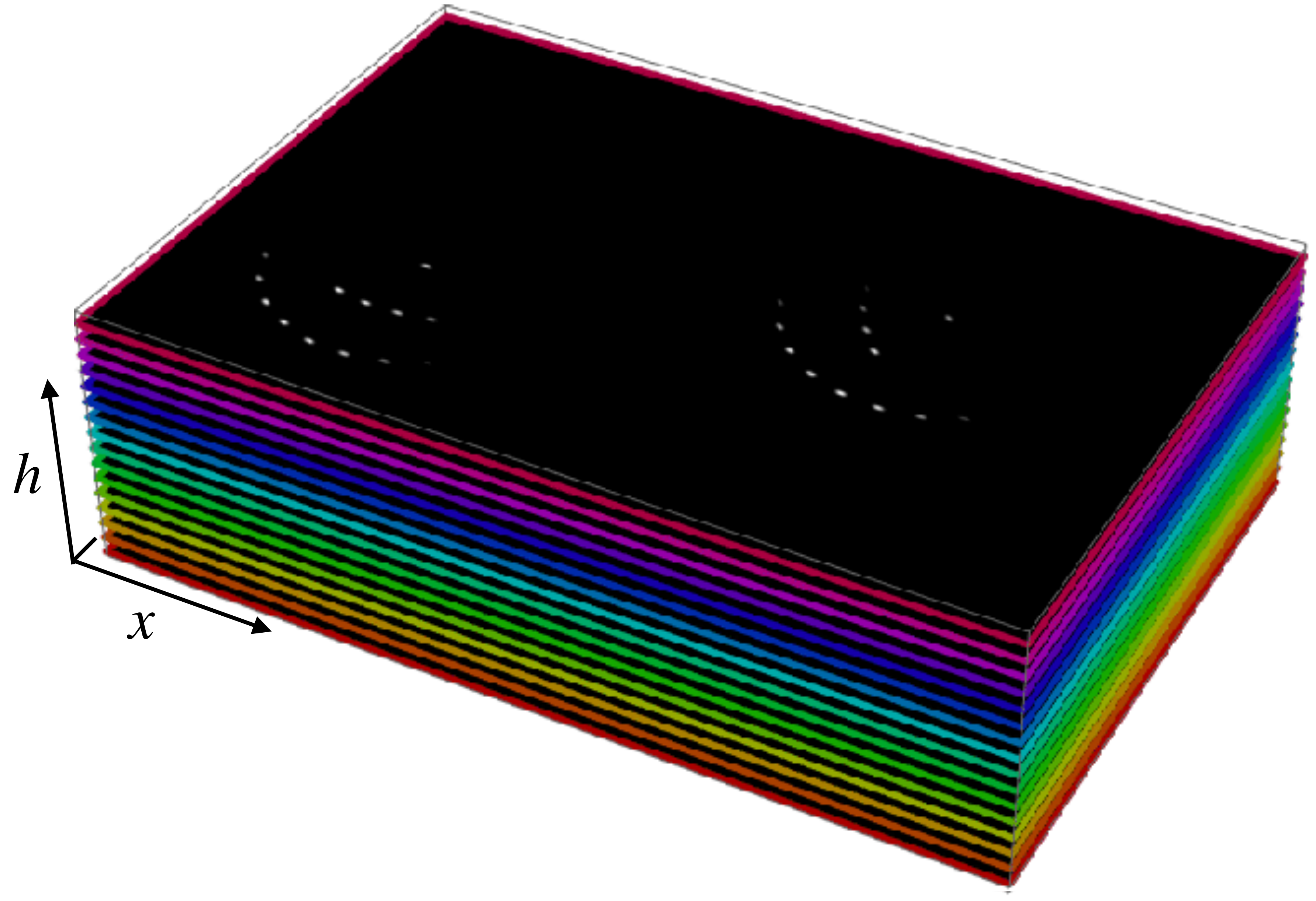
translation
“template matching”



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



=



Rotated
2D convolution kernel

2D input feature map

SE(2) output feature map (after ReLU)

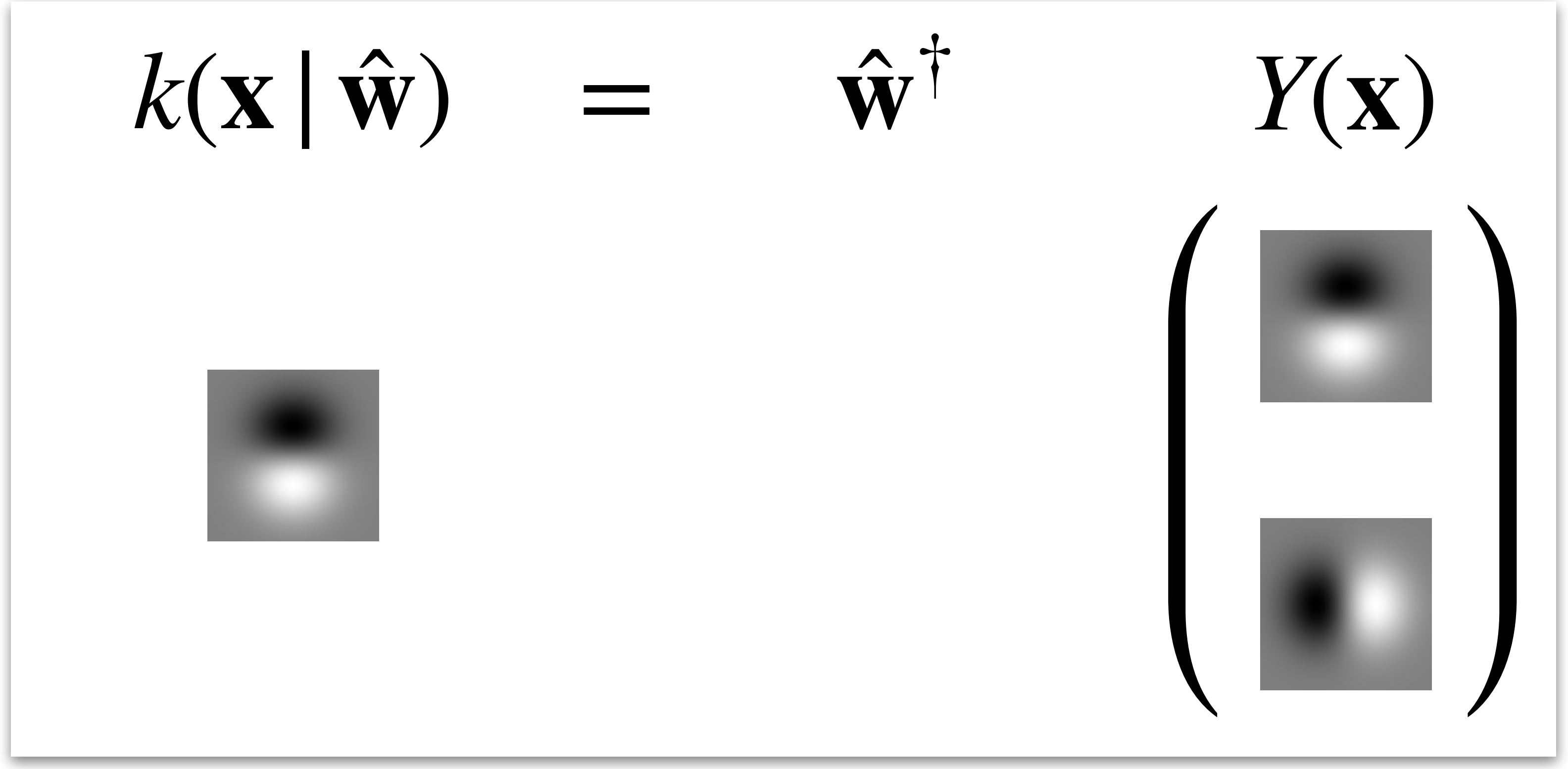
Lifting convolution with steerable kernel

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

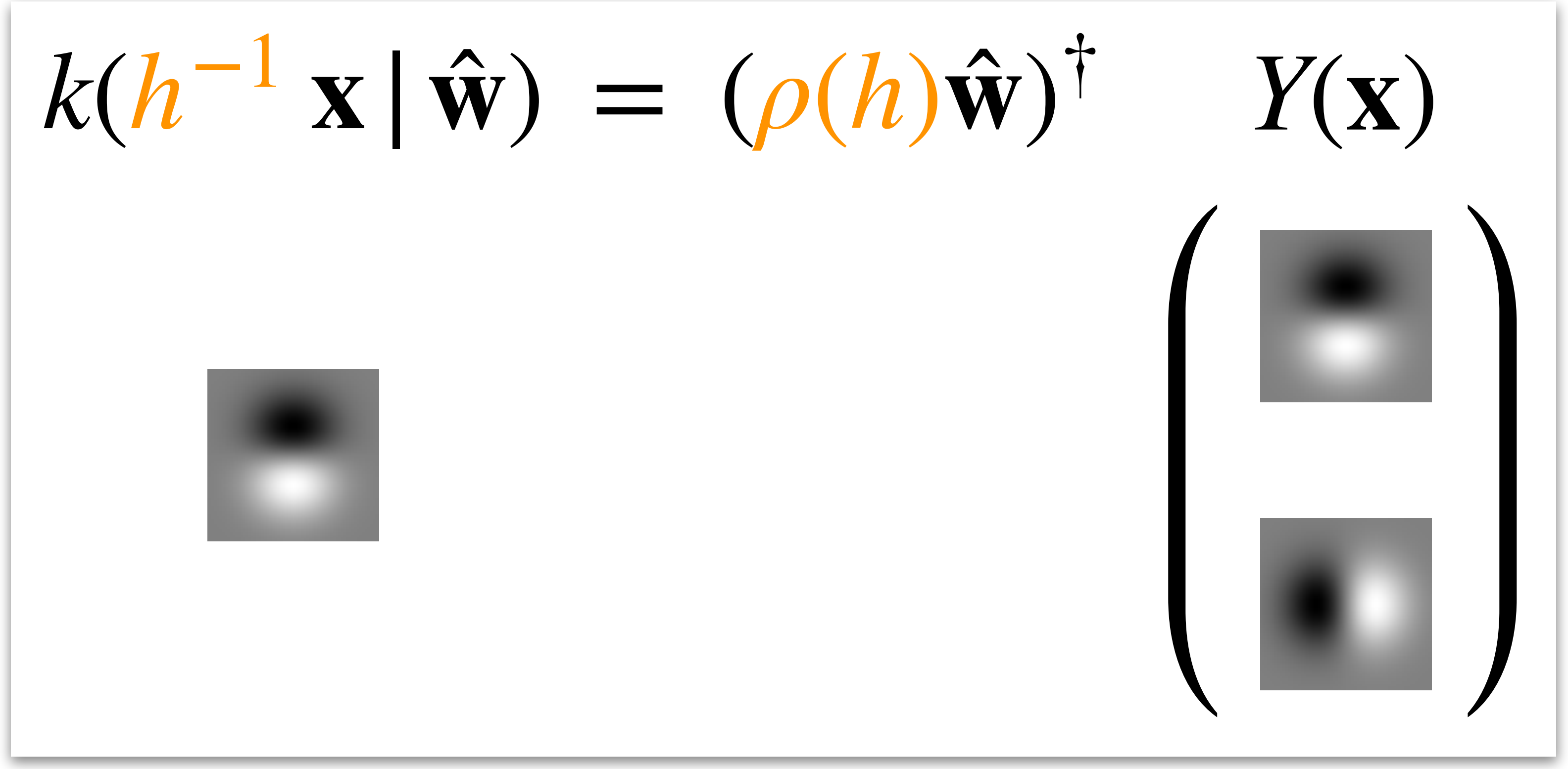


$$k(\mathbf{x} \,|\, \hat{\mathbf{w}}) \quad = \quad \hat{\mathbf{w}}^{\dagger} \quad Y(\mathbf{x})$$

Lifting convolution with steerable kernel

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

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



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

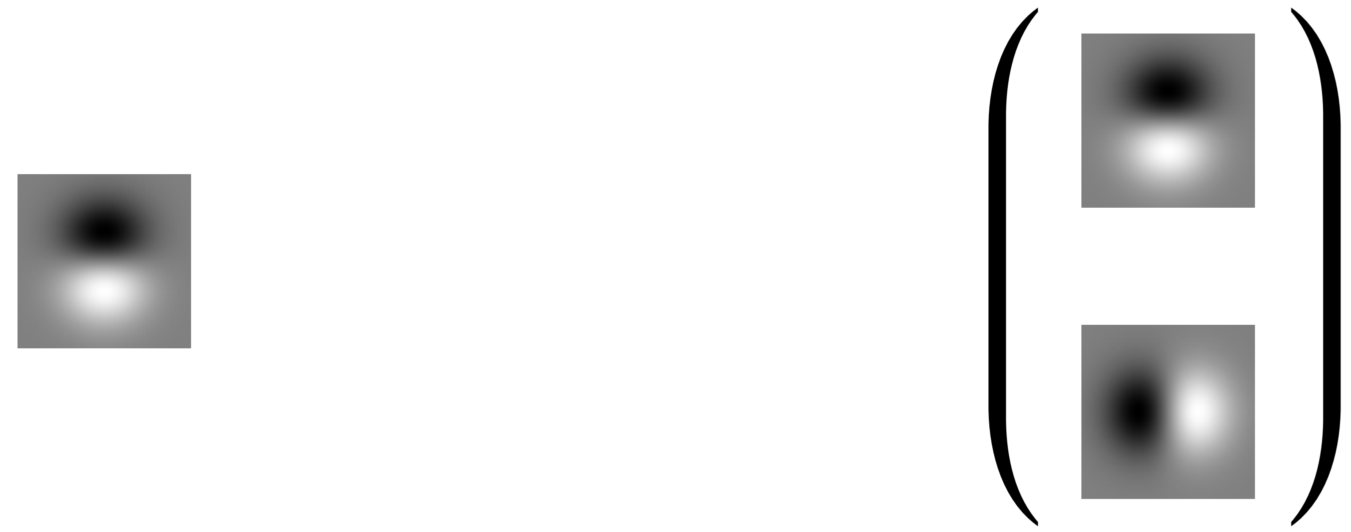
Lifting convolution with steerable kernel

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

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

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

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



Lifting convolution with steerable kernel

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

$$
\begin{aligned}
(k \,\tilde{\star}\, f)(\mathbf{x},h) &= \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}'-\mathbf{x}) \mid \hat{\mathbf{w}}) f(\mathbf{x}')\mathrm{d}\mathbf{x}' \\
&= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}'-\mathbf{x}) f(\mathbf{x}')\mathrm{d}\mathbf{x}' \\
&= (\rho(h)\,\hat{\mathbf{w}})^\dagger \int_{\mathbb{R}^d} Y(\mathbf{x}'-\mathbf{x}) f(\mathbf{x}')\mathrm{d}\mathbf{x}'
\end{aligned}
$$

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

Lifting convolution with steerable kernel

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

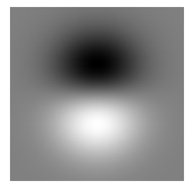
$$(k \,\tilde{\star}\, f)(\mathbf{x},h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}'-\mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

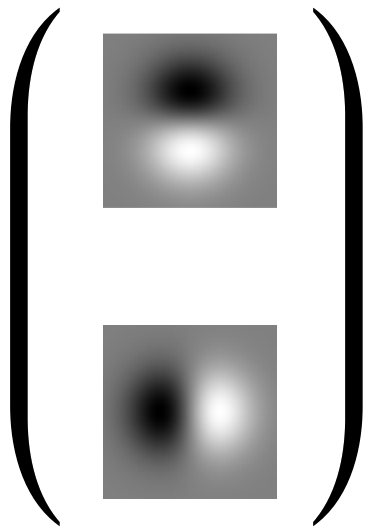
$$= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^{\dagger} Y(\mathbf{x}'-\mathbf{x}) f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

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

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

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





Lifting convolution with steerable kernel

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

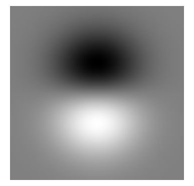
$$(k \,\tilde{\star}\, f)(\mathbf{x}, h) = \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}' - \mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

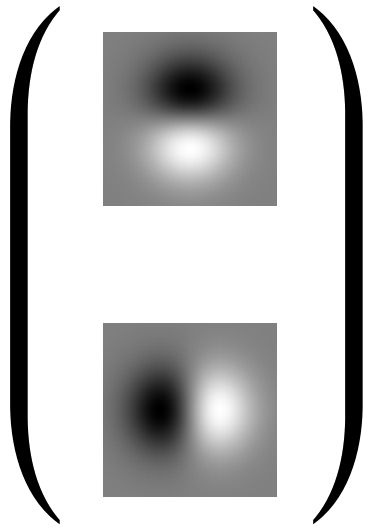
$$= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$$

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

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

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





! 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.

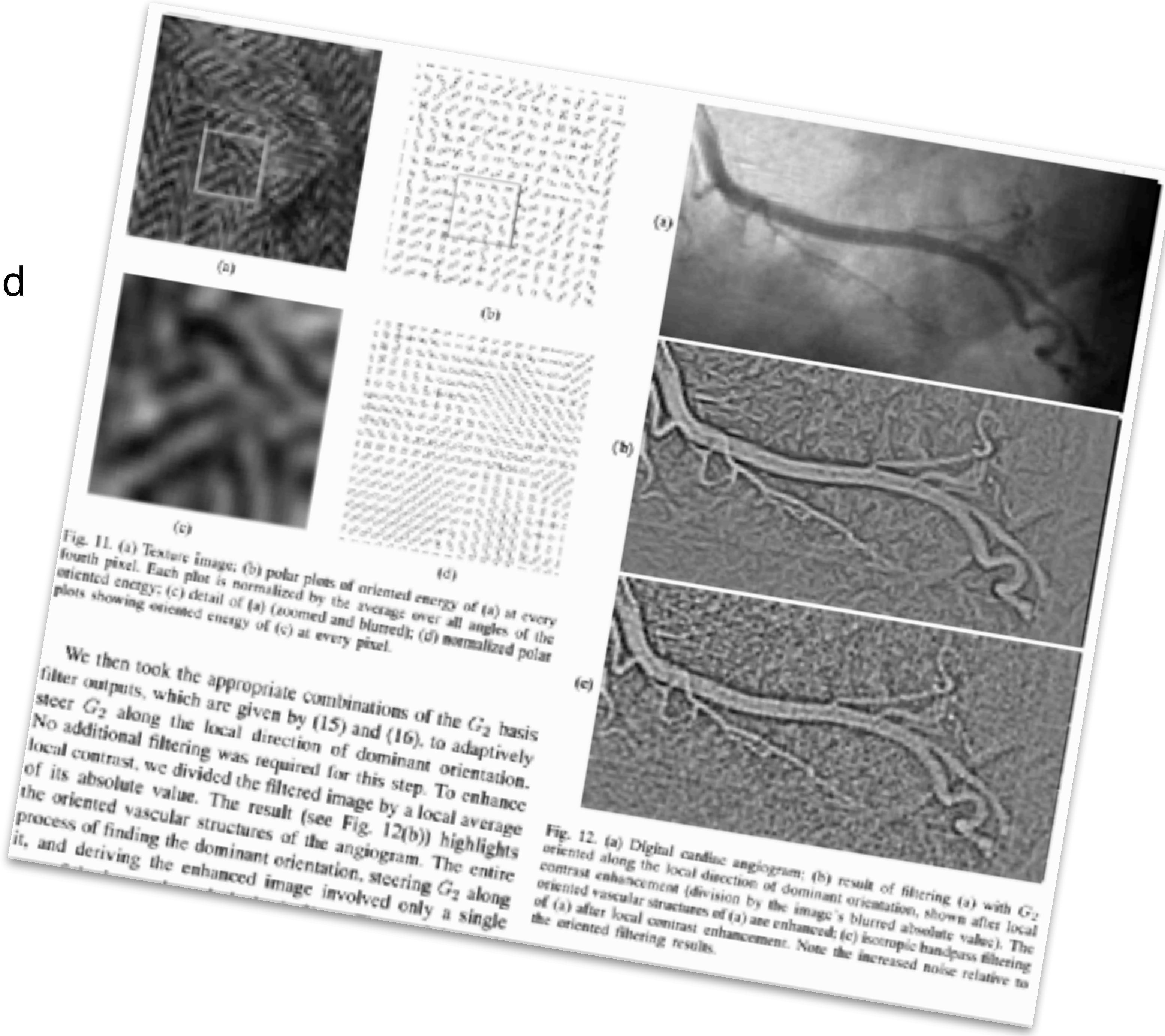


Fig. 11. (a) Texture image; (b) polar plots of oriented energy of (a) at every fourth pixel. Each plot is normalized by the average over all angles of the oriented energy; (c) detail of (a) (zoomed and blurred); (d) normalized polar plots showing oriented energy of (c) at every pixel.

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

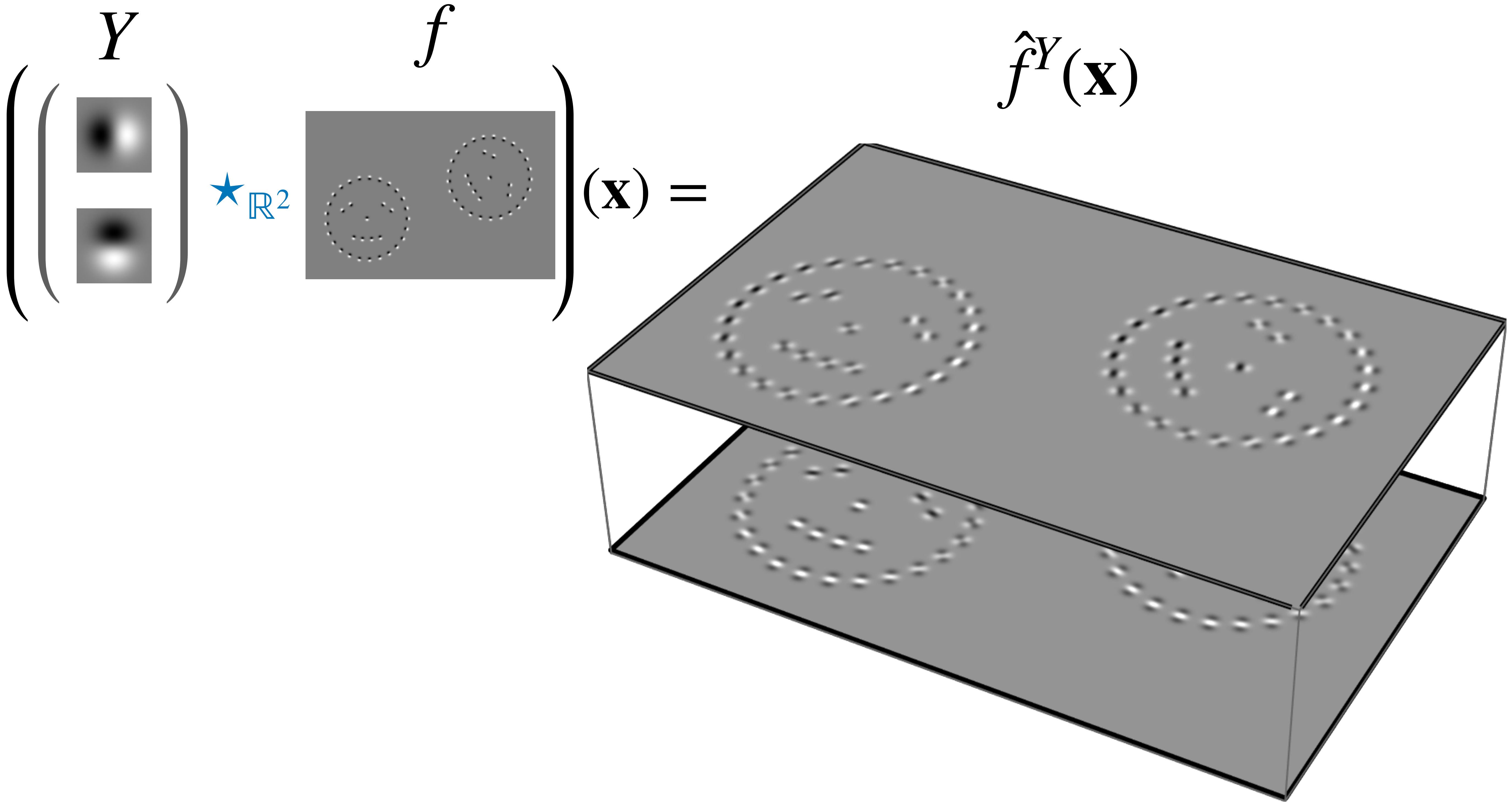
$$(k \,\tilde{\star}\, f)(\mathbf{x}, \theta) = (\rho(\mathbf{R}_\theta)\hat{\mathbf{w}})^\dagger \, \hat{f}^Y(\mathbf{x})$$

Lifting convolution with steerable kernel

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

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

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

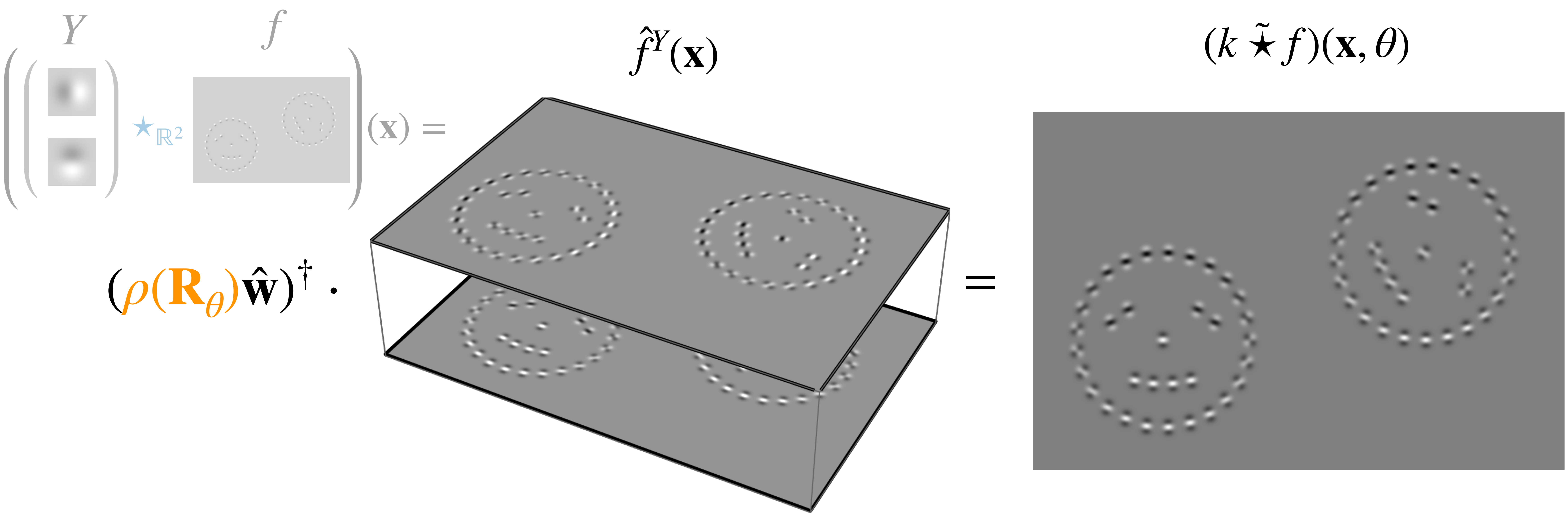


Lifting convolution with steerable kernel

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

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

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


Y
f
$\left(\left(\; \right) \star_{\mathbb{R}^2} \; \right)(\mathbf{x}) =$
$\hat{f}^Y(\mathbf{x})$
$(k \,\tilde{\star}\, f)(\mathbf{x}, \theta)$
$(\rho(\mathbf{R}_\theta)\hat{\mathbf{w}})^\dagger \cdot$
$=$

Lifting convolution with steerable kernel

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

$$
\begin{aligned}
(k \,\tilde{\star}\, f)(\mathbf{x},h) &= \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}'-\mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}') d\mathbf{x}' \\
&= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}'-\mathbf{x}) f(\mathbf{x}') d\mathbf{x}' \\
&= (\rho(h)\,\hat{\mathbf{w}})^\dagger \int_{\mathbb{R}^d} Y(\mathbf{x}'-\mathbf{x}) f(\mathbf{x}') d\mathbf{x}' \\
&= (\rho(h)\,\hat{\mathbf{w}})^\dagger\, \hat{f}^Y(\mathbf{x})
\end{aligned}
$$

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

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.

Lifting convolution with steerable kernel

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

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

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

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

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

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

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

$$
\begin{aligned}
(k \,\tilde{\star}\, f)(\mathbf{x},h) &= \int_{\mathbb{R}^d} k(h^{-1}(\mathbf{x}'-\mathbf{x}) \,|\, \hat{\mathbf{w}}) f(\mathbf{x}')d\mathbf{x}' \\
&= \int_{\mathbb{R}^d} (\rho(h)\hat{\mathbf{w}})^\dagger Y(\mathbf{x}'-\mathbf{x}) f(\mathbf{x}')d\mathbf{x}' \\
&= (\rho(h)\,\hat{\mathbf{w}})^\dagger \int_{\mathbb{R}^d} Y(\mathbf{x}'-\mathbf{x}) f(\mathbf{x}')d\mathbf{x}' \\
&= (\rho(h)\,\hat{\mathbf{w}})^\dagger \,\hat{f}^Y(\mathbf{x}) \\
&= \mathrm{tr}(\, \hat{f}^Y(\mathbf{x})\, \hat{\mathbf{w}}^\dagger\, \rho(h^{-1})\,)
\end{aligned}
$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

! 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.

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

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

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

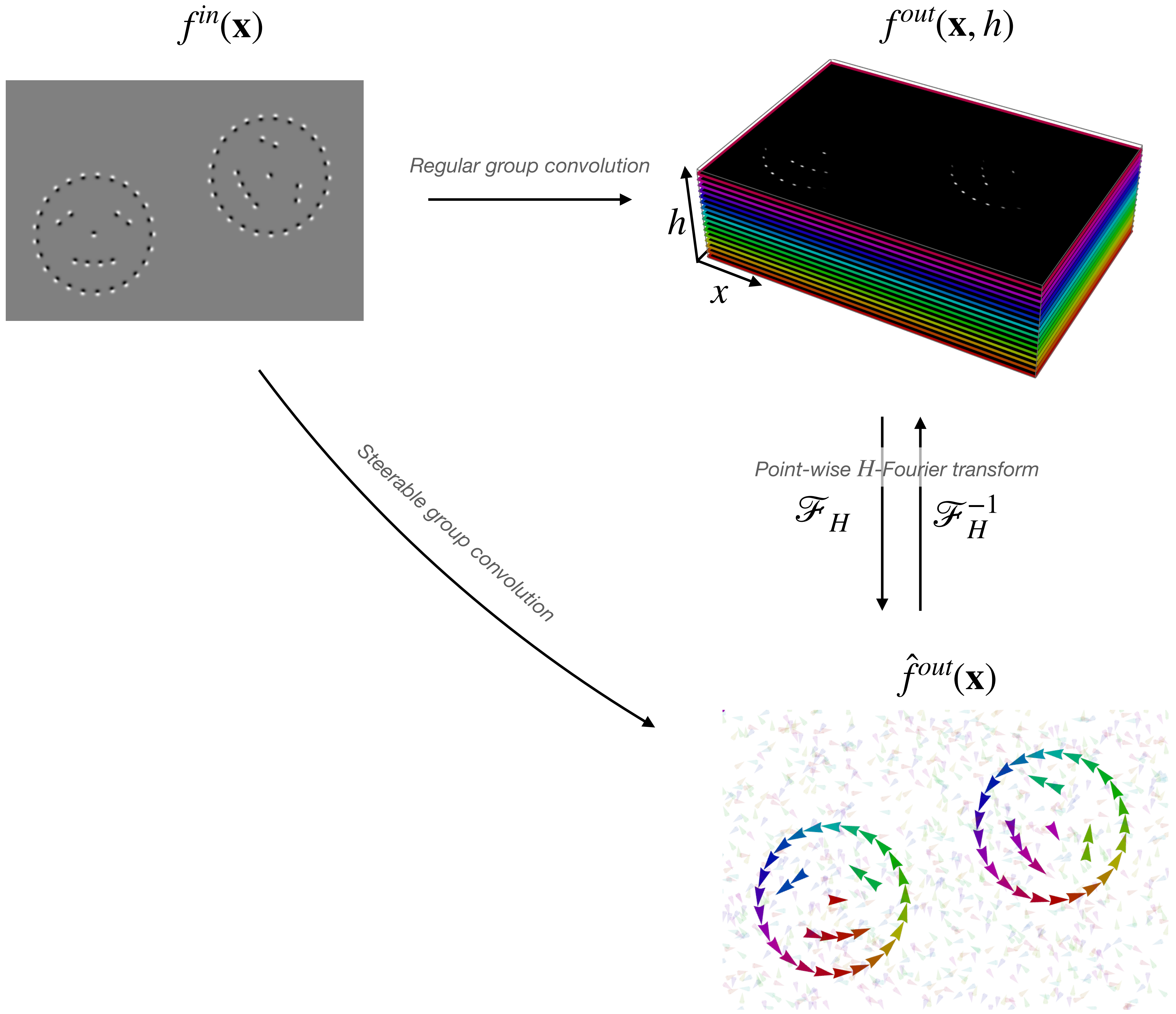
$\hat{f}(\mathbf{x}) = \hat{f}^Y(\mathbf{x})\;\hat{w}^\dagger$

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

Inverse H-Fourier transform!

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

From regular to steerable via a Fourier transform



$f^{in}(\mathbf{x})$

Regular group convolution

$f^{out}(\mathbf{x}, h)$

Point-wise H-Fourier transform

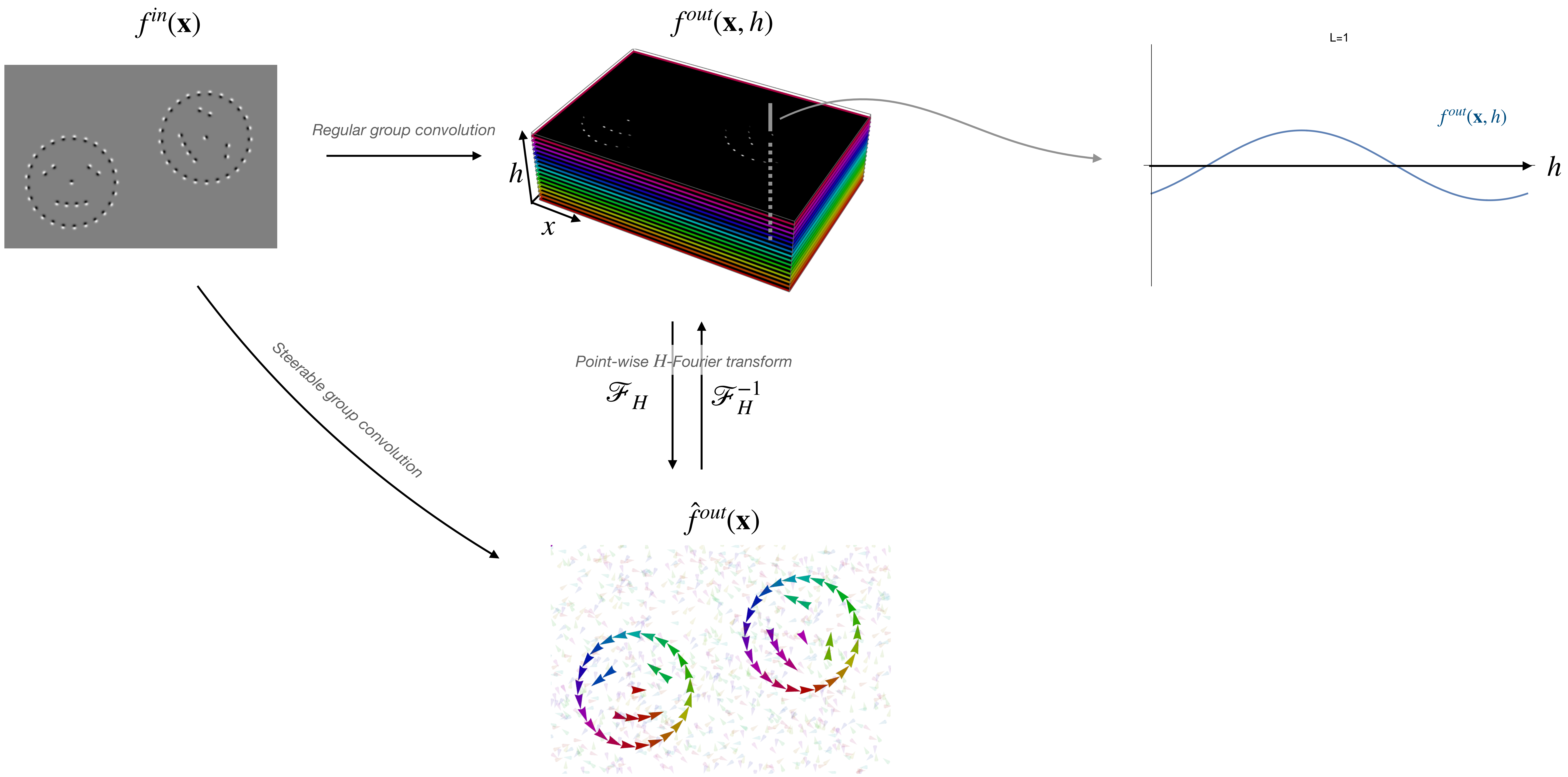
$\mathscr{F}_H$

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

Steerable group convolution

$\hat{f}^{out}(\mathbf{x})$

From regular to steerable via a Fourier transform



$f^{in}(\mathbf{x})$

Regular group convolution

$f^{out}(\mathbf{x},h)$

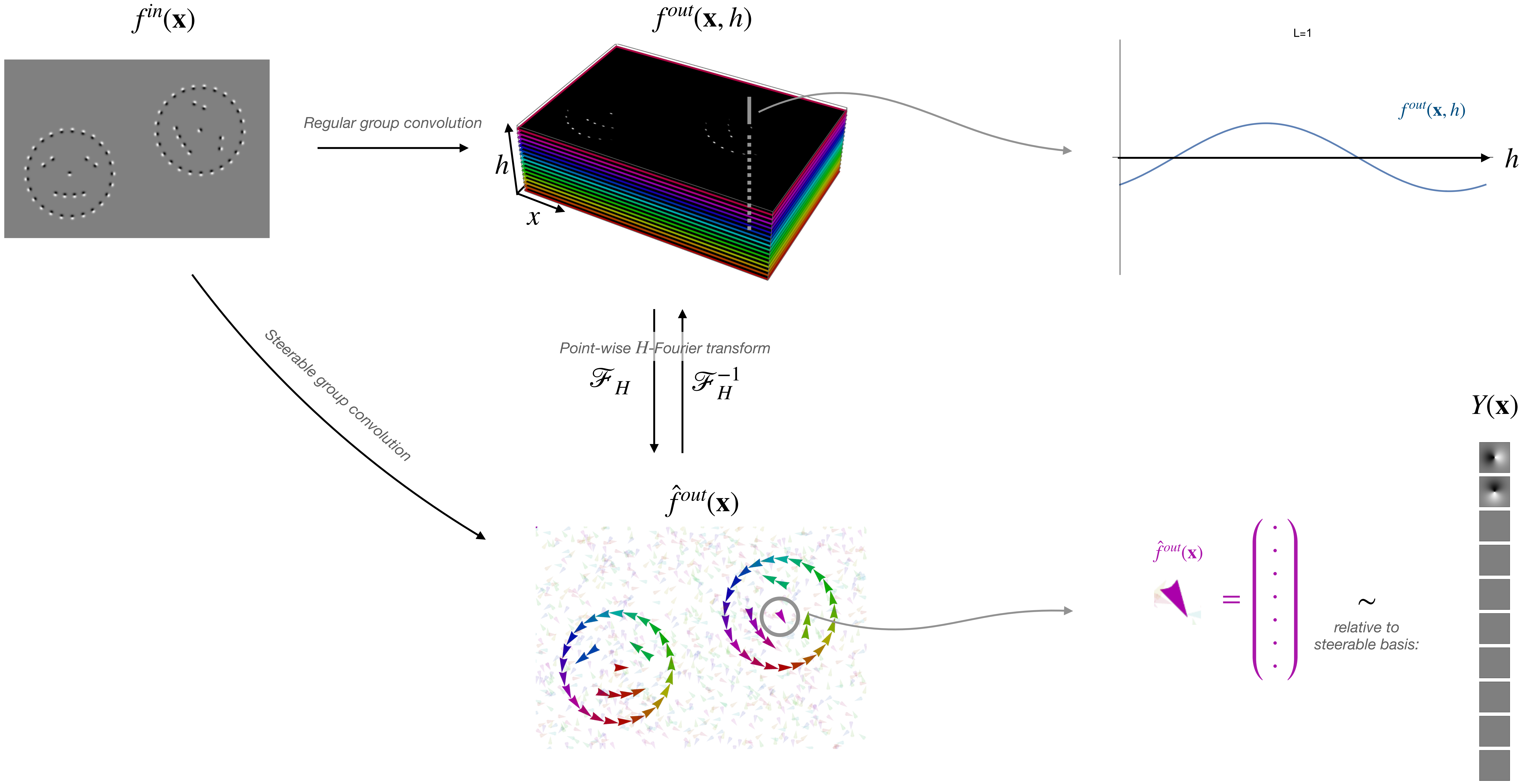
Steerable group convolution

Point-wise H-Fourier transform

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

$\hat{f}^{out}(\mathbf{x})$

From regular to steerable via a Fourier transform



$f^{in}(\mathbf{x})$

Regular group convolution

$f^{out}(\mathbf{x}, h)$

Steerable group convolution

Point-wise H-Fourier transform

$\mathcal{F}_H$

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

$\hat{f}^{out}(\mathbf{x})$

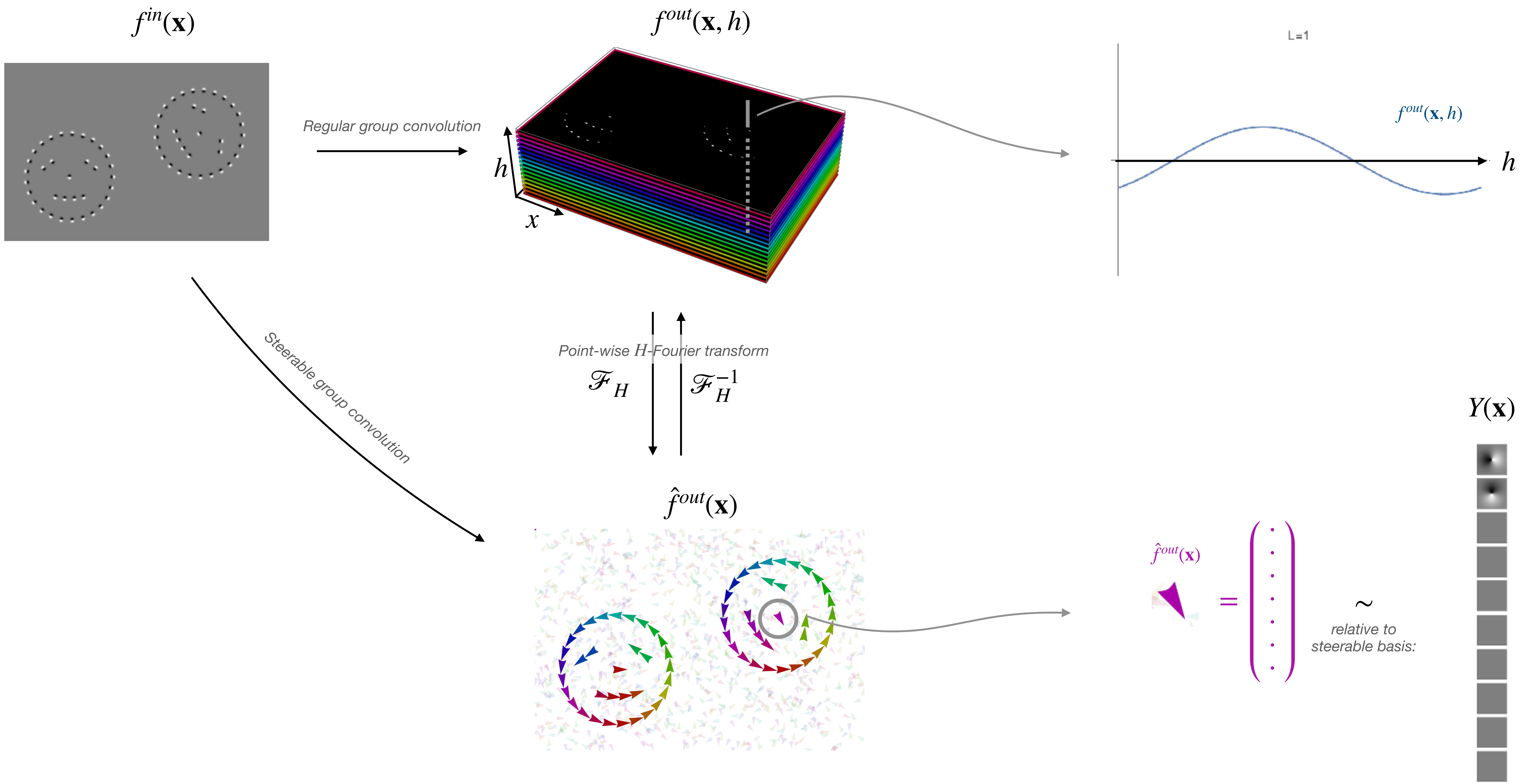
$Y(\mathbf{x})$

$\hat{f}^{out}(\mathbf{x})$ =

~

relative to steerable basis:

From regular to steerable via a Fourier transform



$f^{in}(\mathbf{x})$

Regular group convolution

$f^{out}(\mathbf{x}, h)$

Steerable group convolution

Point-wise H-Fourier transform

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

$\hat{f}^{out}(\mathbf{x})$

$\hat{f}^{out}(\mathbf{x})$ $=$ $\begin{pmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{pmatrix}$ $\sim$ relative to steerable basis:

$Y(\mathbf{x})$

From regular to steerable via a Fourier transform

Regular group convolutions:
Domain expanded feature maps

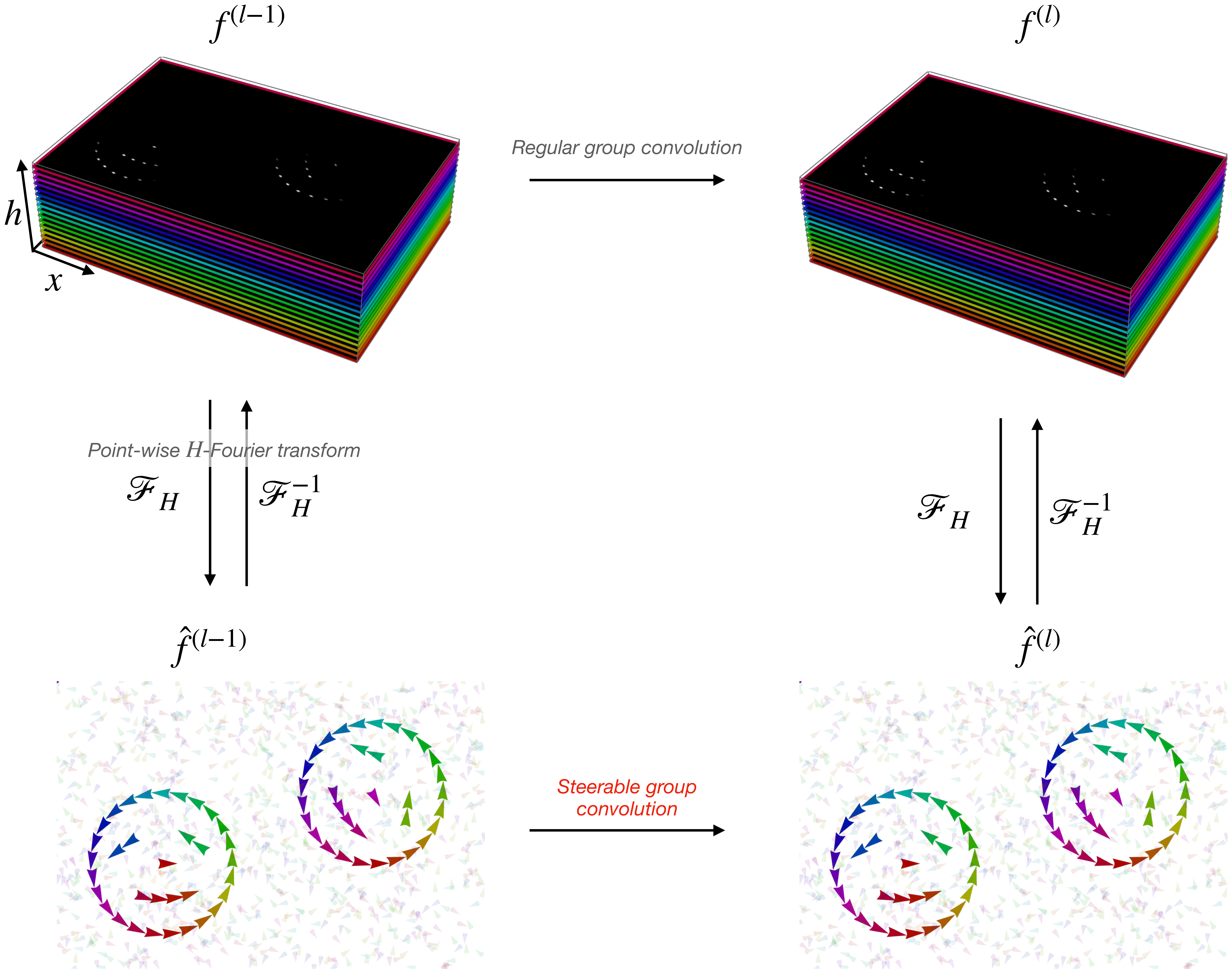
$$f^{(l)} : \mathbb{R}^d \times {\color{red}H} \rightarrow \mathbb{R}$$

added axis

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

$$\hat{f}^{(l)} : \mathbb{R}^d \rightarrow {\color{red}V_H}$$

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



$f^{(l-1)}$

$f^{(l)}$

Regular group convolution

Point-wise H-Fourier transform

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

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

$\hat{f}^{(l-1)}$

$\hat{f}^{(l)}$

Steerable group convolution

1. Motivation	Equivariance → weight-sharing and generalization
2. Pattern matching using group theory	Group theory: symmetries & recognition by components (features have “poses”)
3. Group convolutions	Template matching over groups
4. Example	Effective representation learning and generalization
5. G-convs are all you need!	Any equivariant linear layer is a group convolution
6. Steerable group convolutions	Efficient (band-limited) **grid-free** g-convs
7. Feature fields and escnn library	
8. Equivariant tensor product layers	
9. Equivariant graph NNs	

1. Motivation	Equivariance → weight-sharing and generalization
2. Pattern matching using group theory	Group theory: symmetries & recognition by components (features have "poses")
3. Group convolutions	Template matching over groups
4. Example	Effective representation learning and generalization
5. G-convs are all you need!	Any equivariant linear layer is a group convolution
6. Steerable group convolutions	Efficient (band-limited) **grid-free** g-convs
7. Feature fields and escnn library	
8. Equivariant tensor product layers	
9. Equivariant graph NNs	

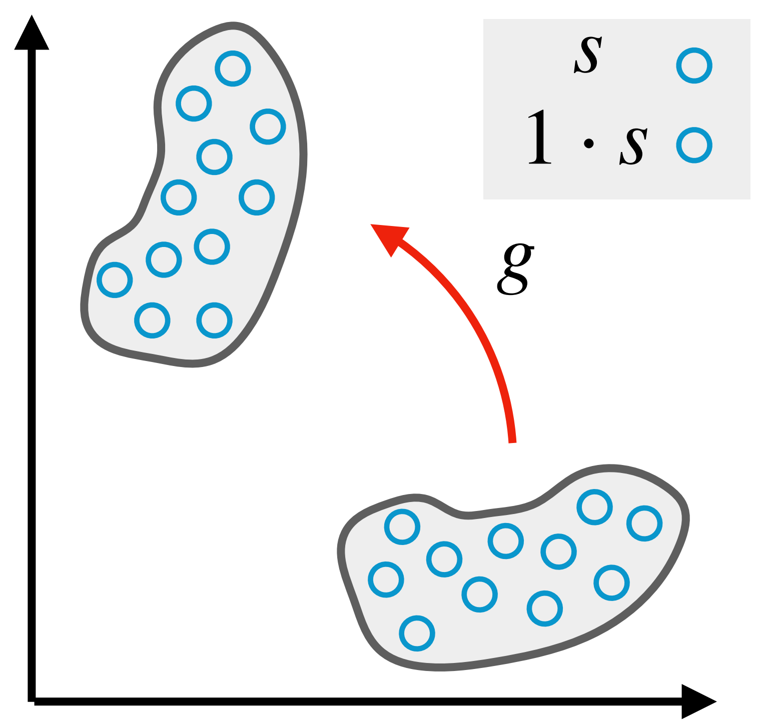
Feature field and induced representation

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

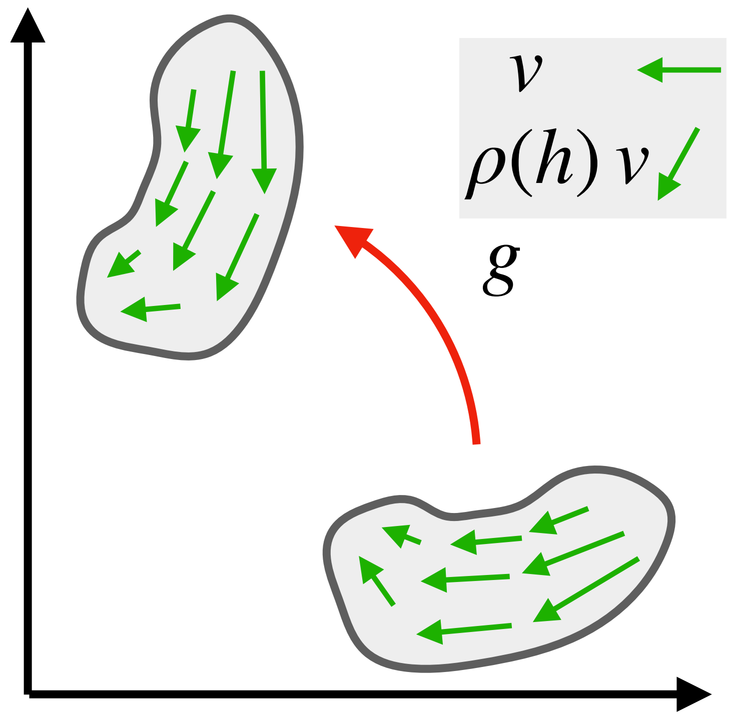
codomain	transforms via a *representation*	$\rho(h)$	of H
domain	transforms via the action	g^{-1}	of $G = (\mathbb{R}^d, +) \rtimes H$

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 ($\rho(h) = 1$)

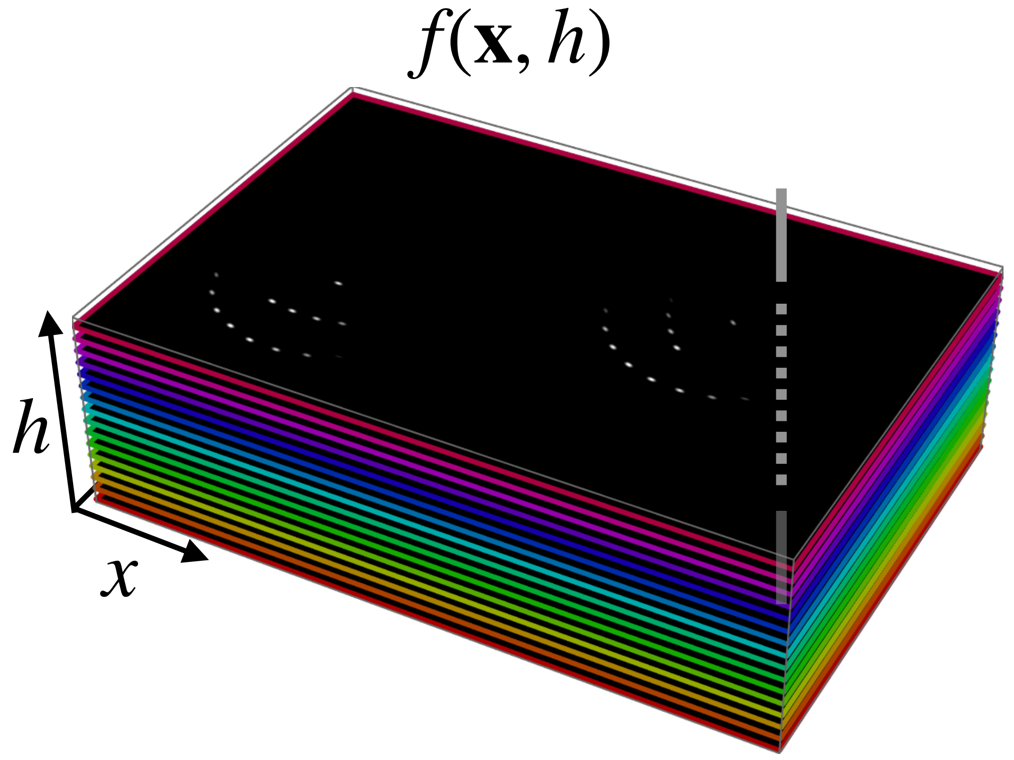


vector field

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.

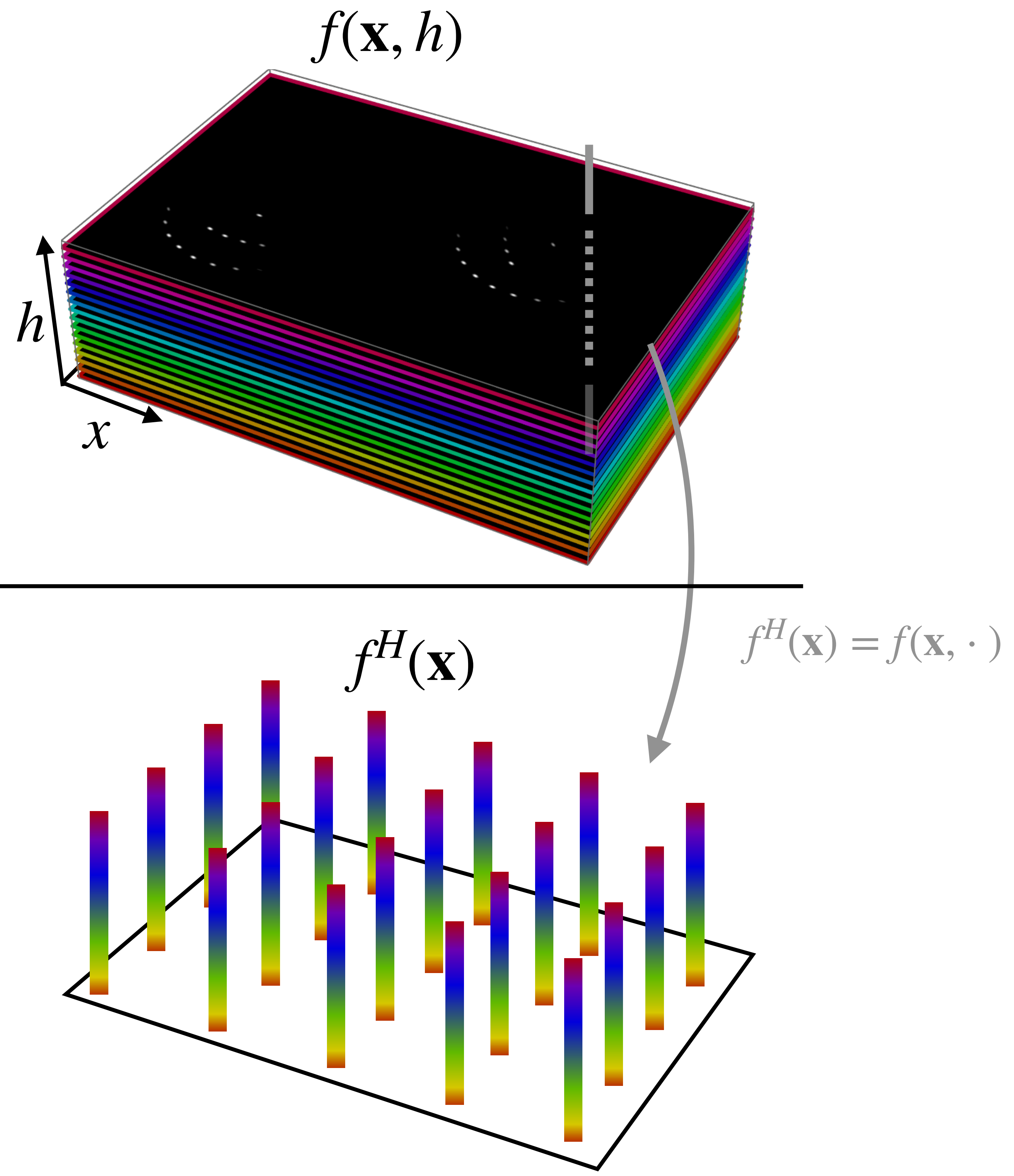


$$(\mathscr{L}_g f)(\mathbf{x}', h') = f(h^{-1}(\mathbf{x}' - \mathbf{x}), h^{-1}h)$$

Feature field and induced representation

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

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

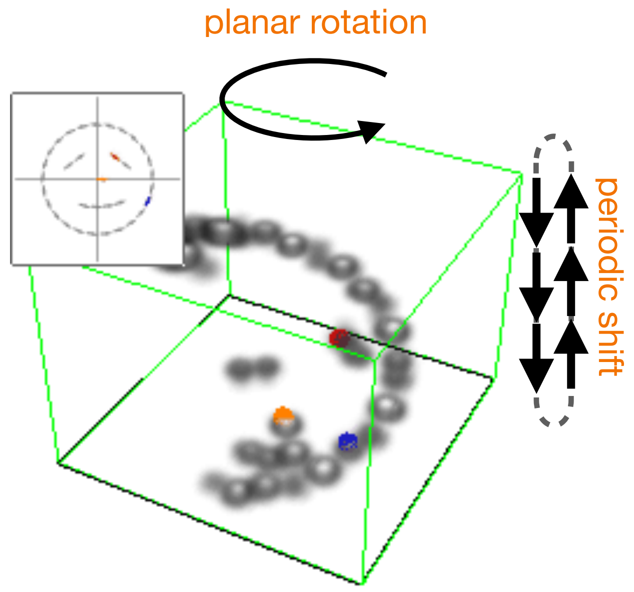


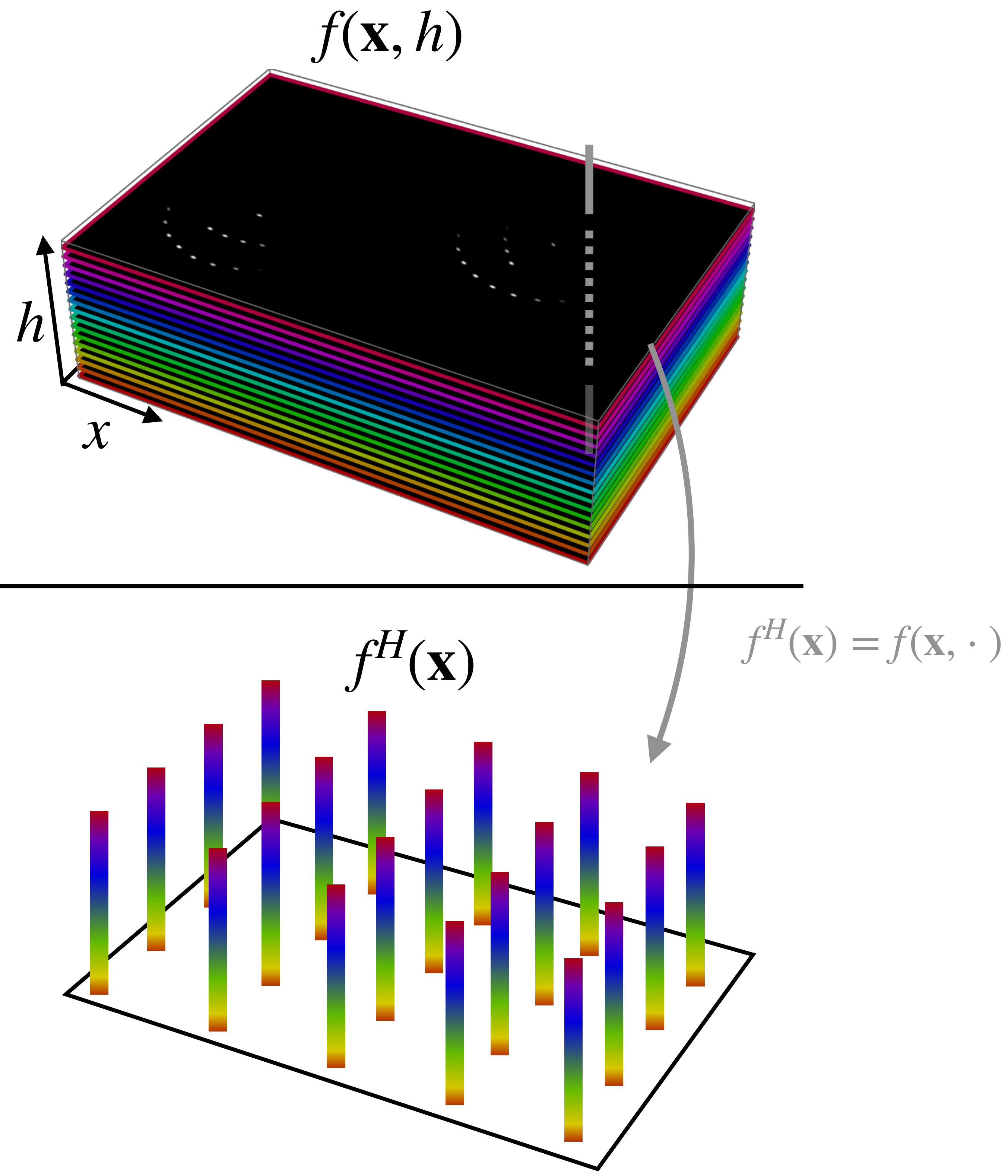
Regular H feature fields: Let $f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$ be the field of functions $f^H(\mathbf{x}) : H \to \mathbb{R}$ on the subgroup H, then the functions (**fibers**) transform via the regular representation $\mathscr{L}_h^H$ (recall. $\mathscr{L}_h^H f(h') = f(h^{-1}h')$)

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

Feature field and induced representation

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





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

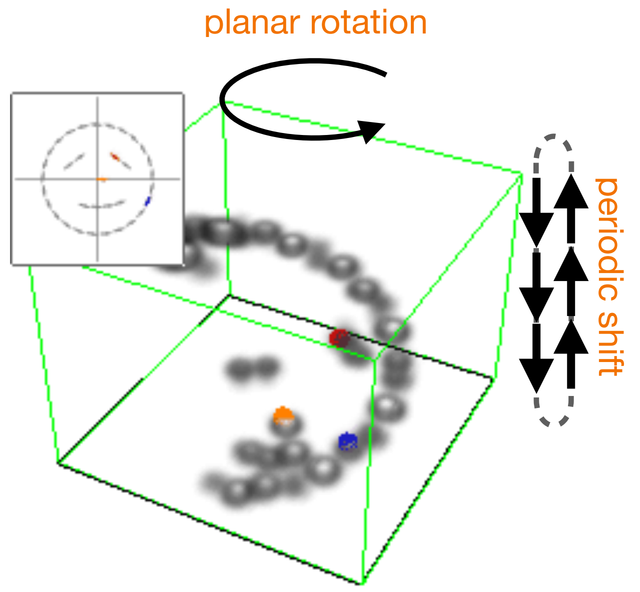
Regular H feature fields: Let $f^H(\mathbf{x}) = f(\mathbf{x}, \cdot\,)$ be the field of functions $f^H(\mathbf{x}) : H \to \mathbb{R}$ on the subgroup H, then the functions (**fibers**) transform via the regular representation $\mathscr{L}^H_h$ (recall. $\mathscr{L}^H_h f(h') = f(h^{-1}h')$)

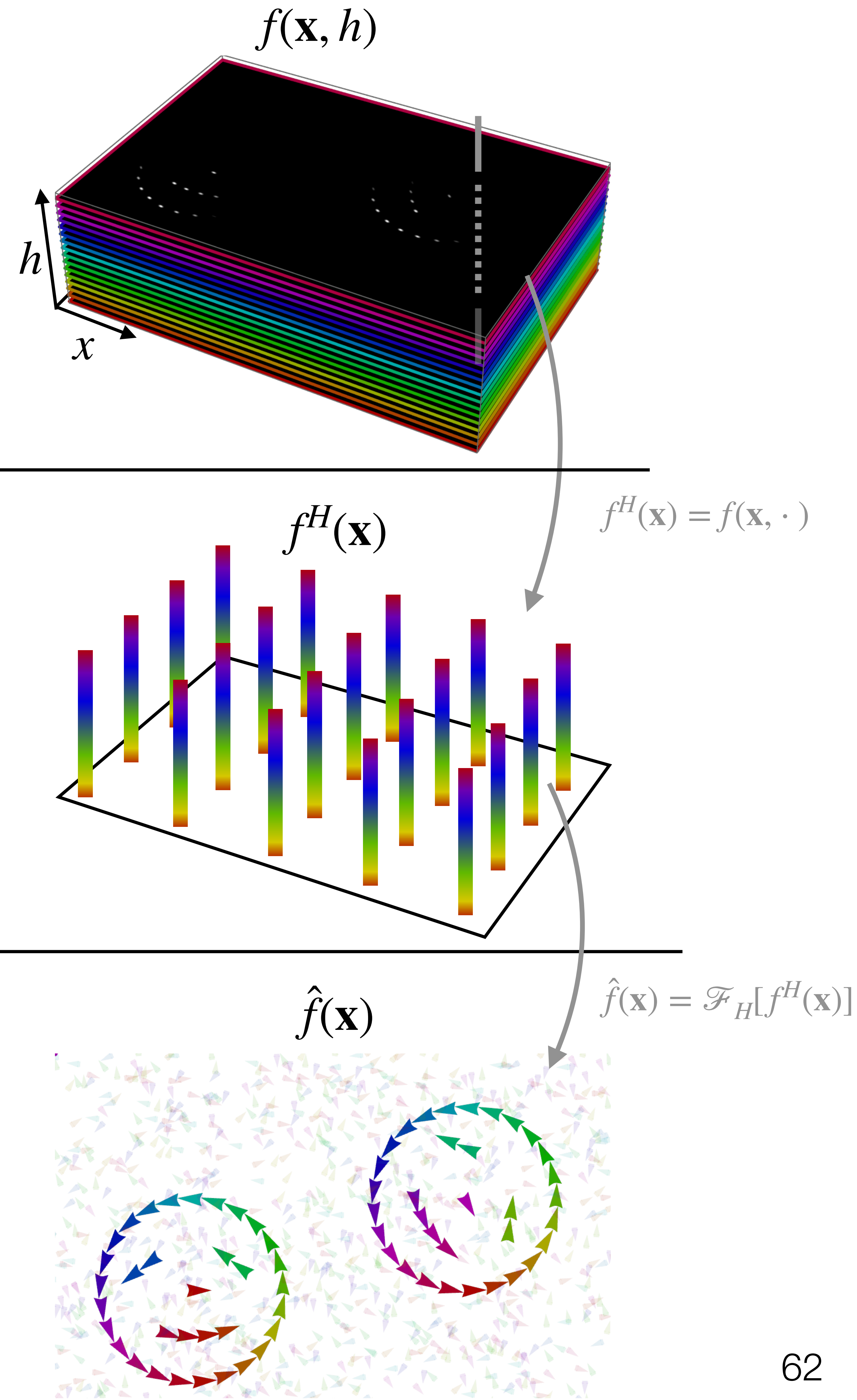
$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}^G_H[\ \mathscr{L}^H_h\ ](\mathbf{x}, h) f^H\right)(\mathbf{x}')$$

Feature field and induced representation

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

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




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

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

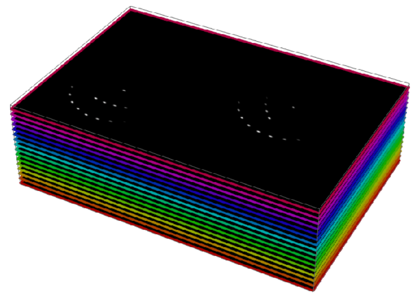
$$(\mathscr{L}_g f)(\mathbf{x}', h') \iff \left(\mathrm{Ind}_H^G[\ \mathscr{L}_h^H\ ](\mathbf{x}, h) f^H\right)(\mathbf{x}')$$

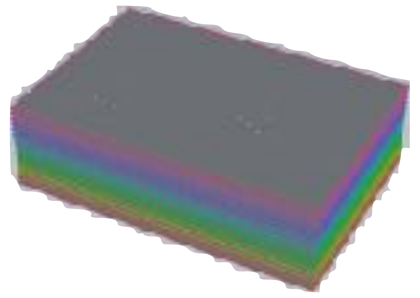
Steerable H feature fields: Since the fibers $f^H(\mathbf{x})$ are functions on H we can represent them via their Fourier coefficients $\hat{f}(\mathbf{x}) = \mathscr{F}_H[f^H(\mathbf{x})]$. These vectors of coefficients transform via irreps $\rho(h) = \bigoplus_l \rho_l(h)$

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

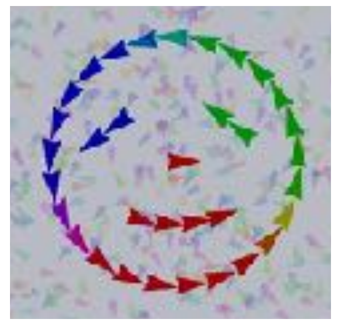
Steerable group convolutions

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





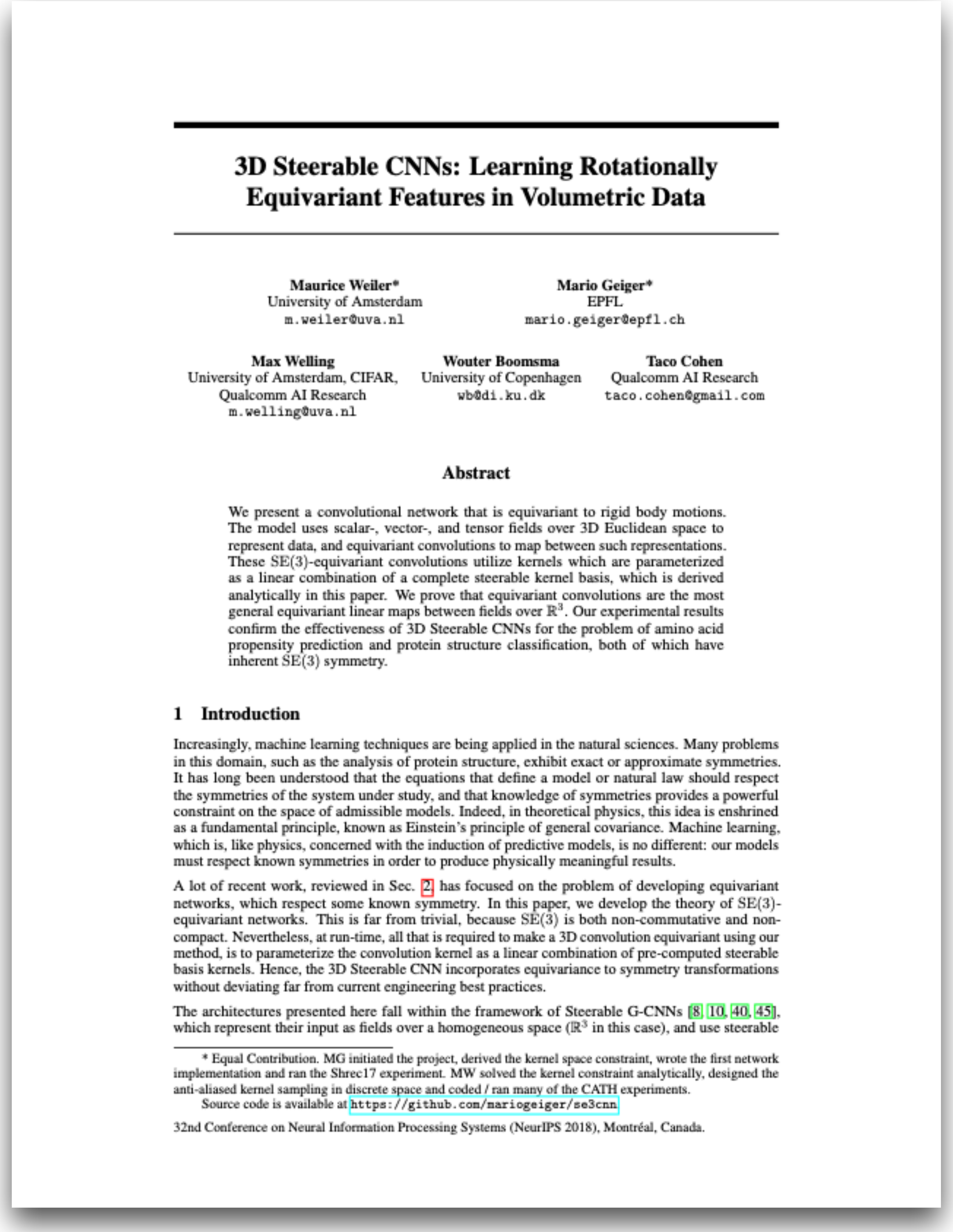
$$\begin{tikzcd}
f^{in} \arrow[r, "\mathcal{K}"] \arrow[d, "\mathcal{F}_H"'] & f^{out} \arrow[d, "\mathcal{F}_H"] \\
\hat{f}^{in} \arrow[r, "\hat{\mathcal{K}}"'] & \hat{f}^{out}
\end{tikzcd}$$



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

but with kernel $k : \mathbb{R}^d \rightarrow \mathbb{R}^{d_Y \times d_X}$ 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.

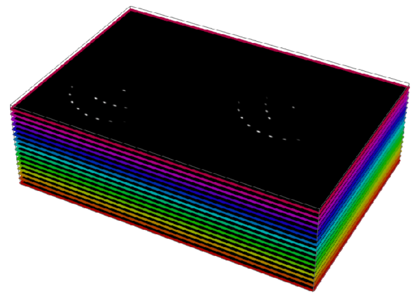
The architectures presented here fall within the framework of Steerable G-CNNs [8, 10, 40, 45], which represent their input as fields over a homogeneous space ($\mathbb{R}^3$ in this case), and use steerable

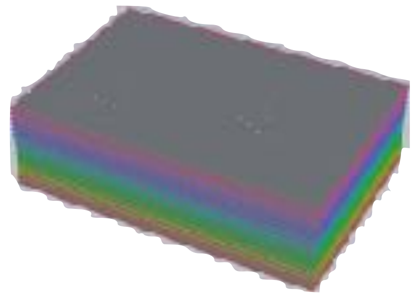
* Equal Contribution. MG initiated the project, derived the kernel space constraint, wrote the first network implementation and ran the Shrec17 experiment. MW solved the kernel constraint analytically, designed the anti-aliased kernel sampling in discrete space and coded / ran many of the CATH experiments.
Source code is available at https://github.com/mariogeiger/se3cnn

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

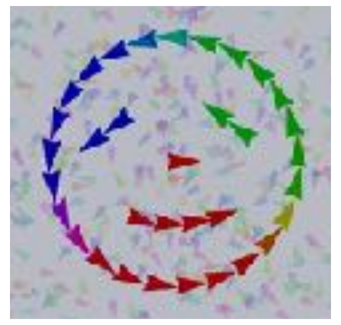
Steerable group convolutions

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





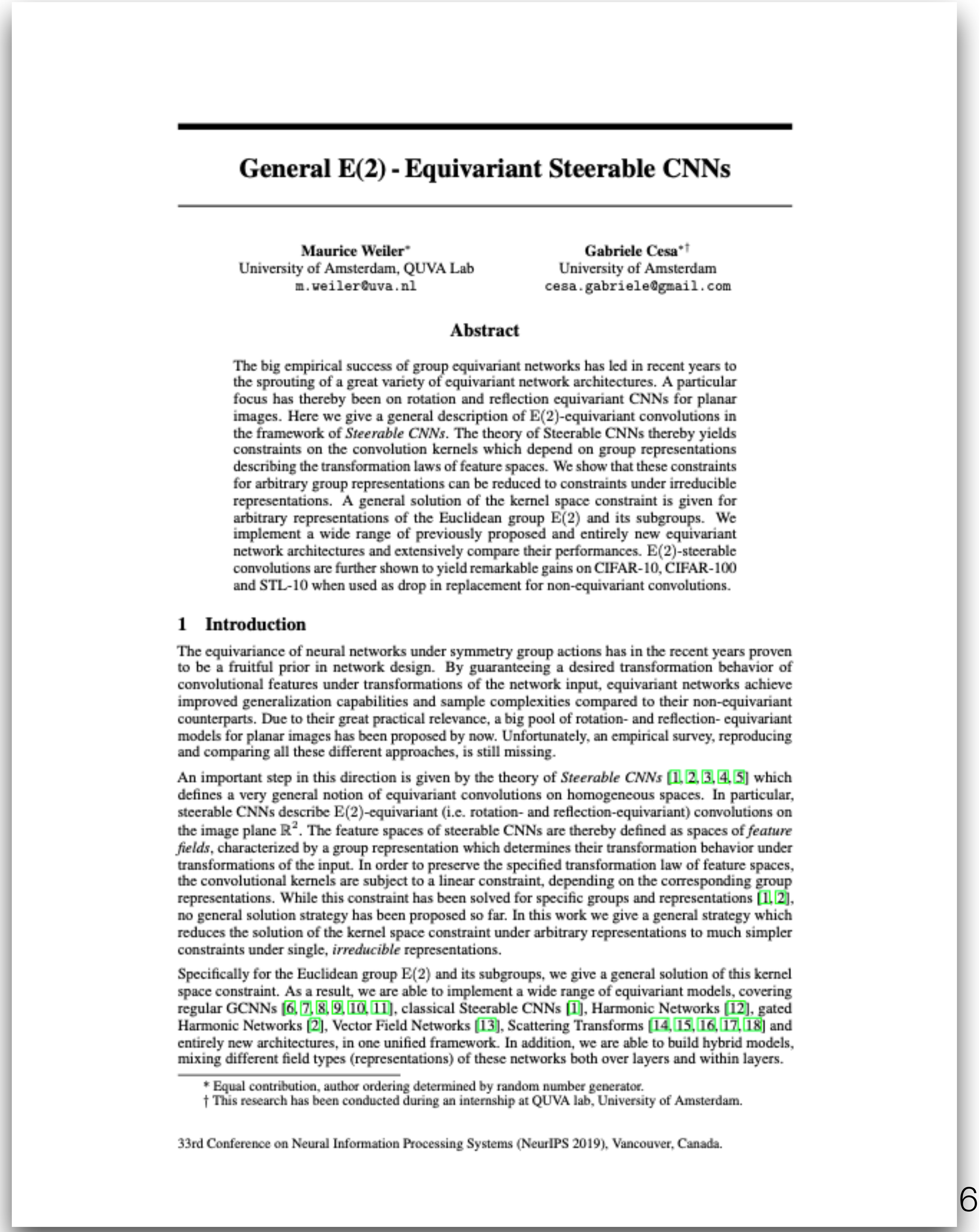
$$\begin{tikzcd}
f^{in} \arrow[r, "\mathcal{K}"] \arrow[d, "\mathcal{F}_H"'] & f^{out} \arrow[d, "\mathcal{F}_H"] \\
\hat{f}^{in} \arrow[r, "\hat{\mathcal{K}}"'] & \hat{f}^{out}
\end{tikzcd}$$



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

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

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



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.

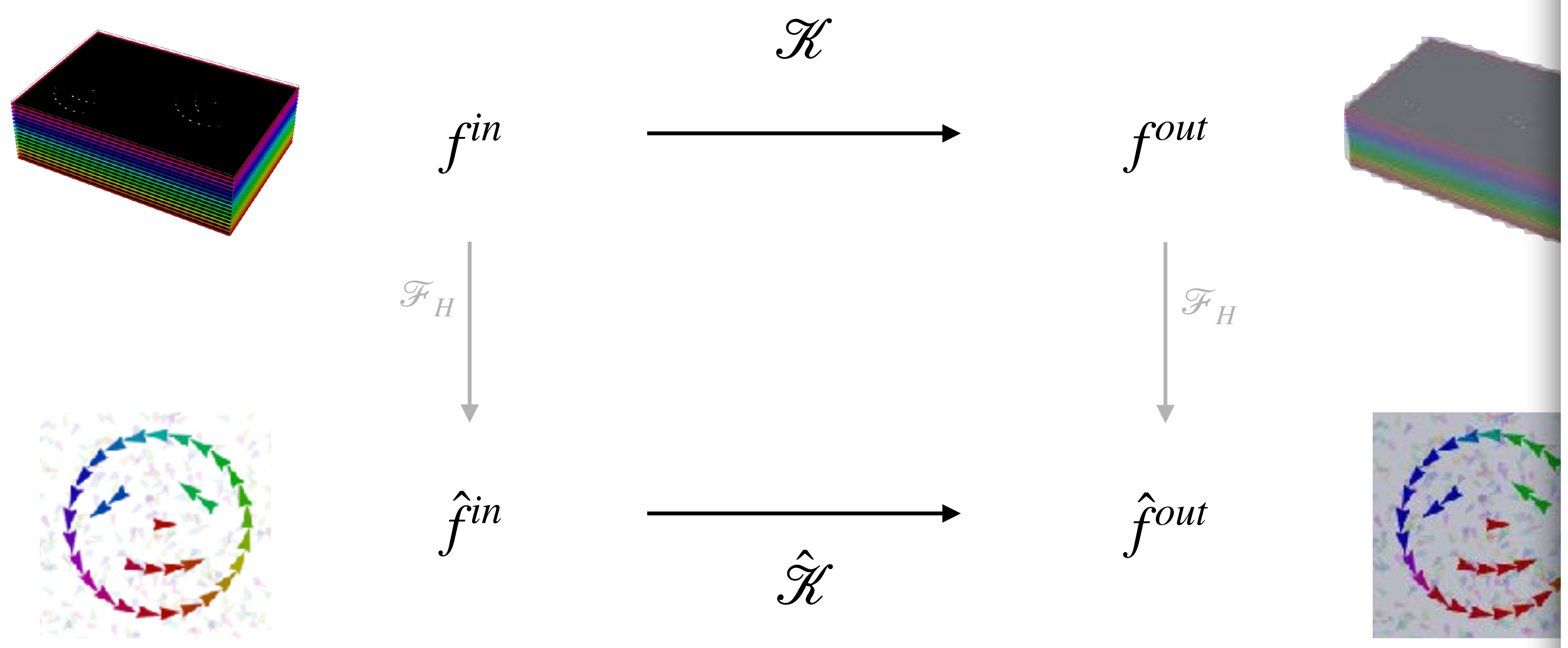
Specifically for the Euclidean group E(2) and its subgroups, we give a general solution of this kernel space constraint. As a result, we are able to implement a wide range of equivariant models, covering regular GCNNs [6, 7, 8, 9, 10, 11], classical Steerable CNNs [1], Harmonic Networks [12], gated Harmonic 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 $\mathcal{K}[f](g) = \int_G k(g^{-1}g')f(g)\mathrm{d}\mathbf{x}'\mathrm{d}g$

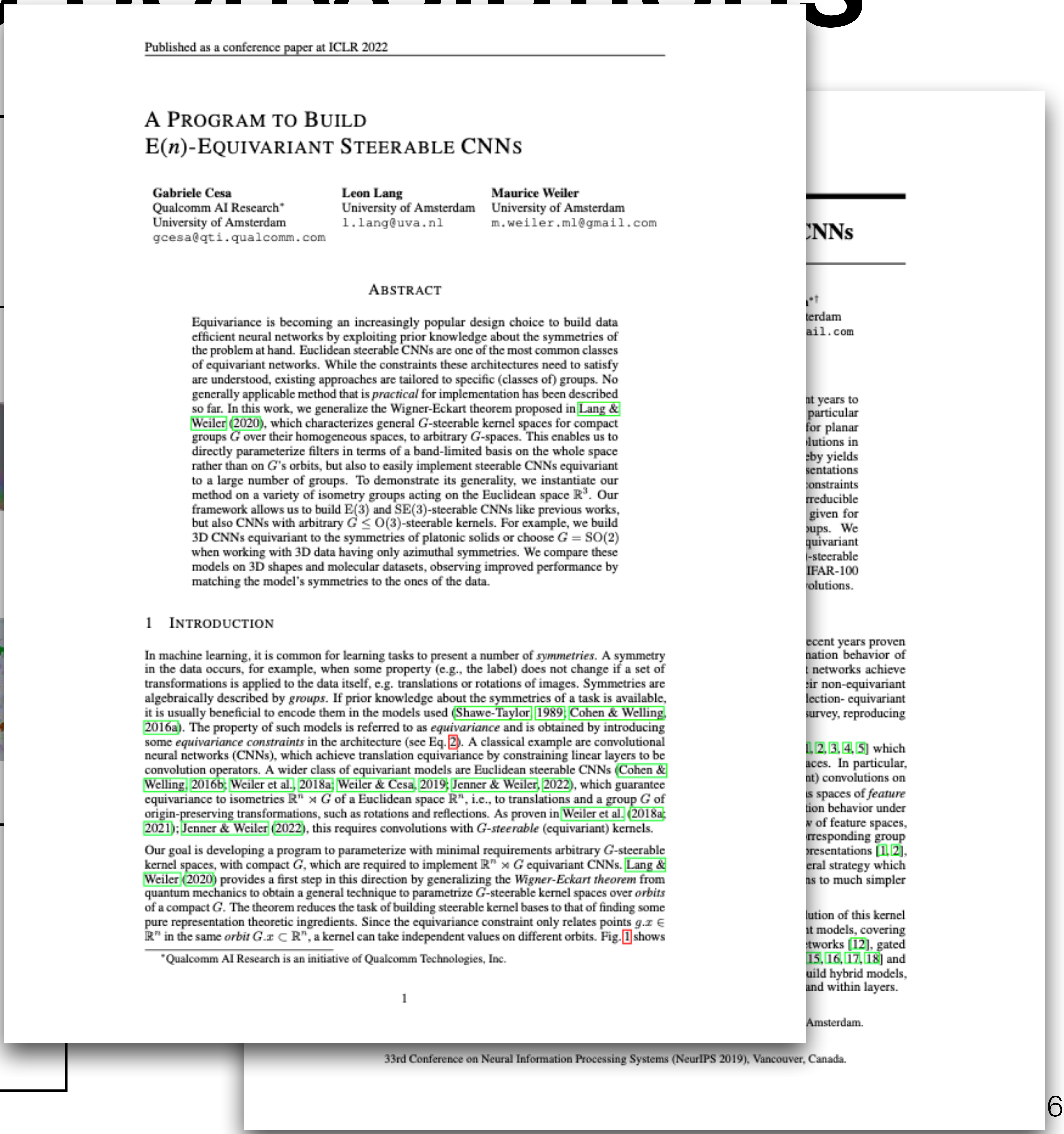


$$\begin{tikzcd}
f^{in} \arrow[r, "\mathcal{K}"] \arrow[d, "\mathcal{F}_H"'] & f^{out} \arrow[d, "\mathcal{F}_H"'] \\
\hat{f}^{in} \arrow[r, "\hat{\mathcal{K}}"'] & \hat{f}^{out}
\end{tikzcd}$$

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

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

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



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.

CNNs

a*†
terdam
ail.com

nt years to
particular
for planar
olutions in
eby yields
sentations
constraints
rreducible
given for
oups. We
quivariant
-steerable
IFAR-100
volutions.

ecent years proven
nation behavior of
t networks achieve
eir non-equivariant
lection- equivariant
survey, reproducing

1, 2, 3, 4, 5] which
aces. In particular,
nt) convolutions on
s spaces of *feature*
tion behavior under
v of feature spaces,
rresponding group
presentations [1, 2],
eral strategy which
ns to much simpler

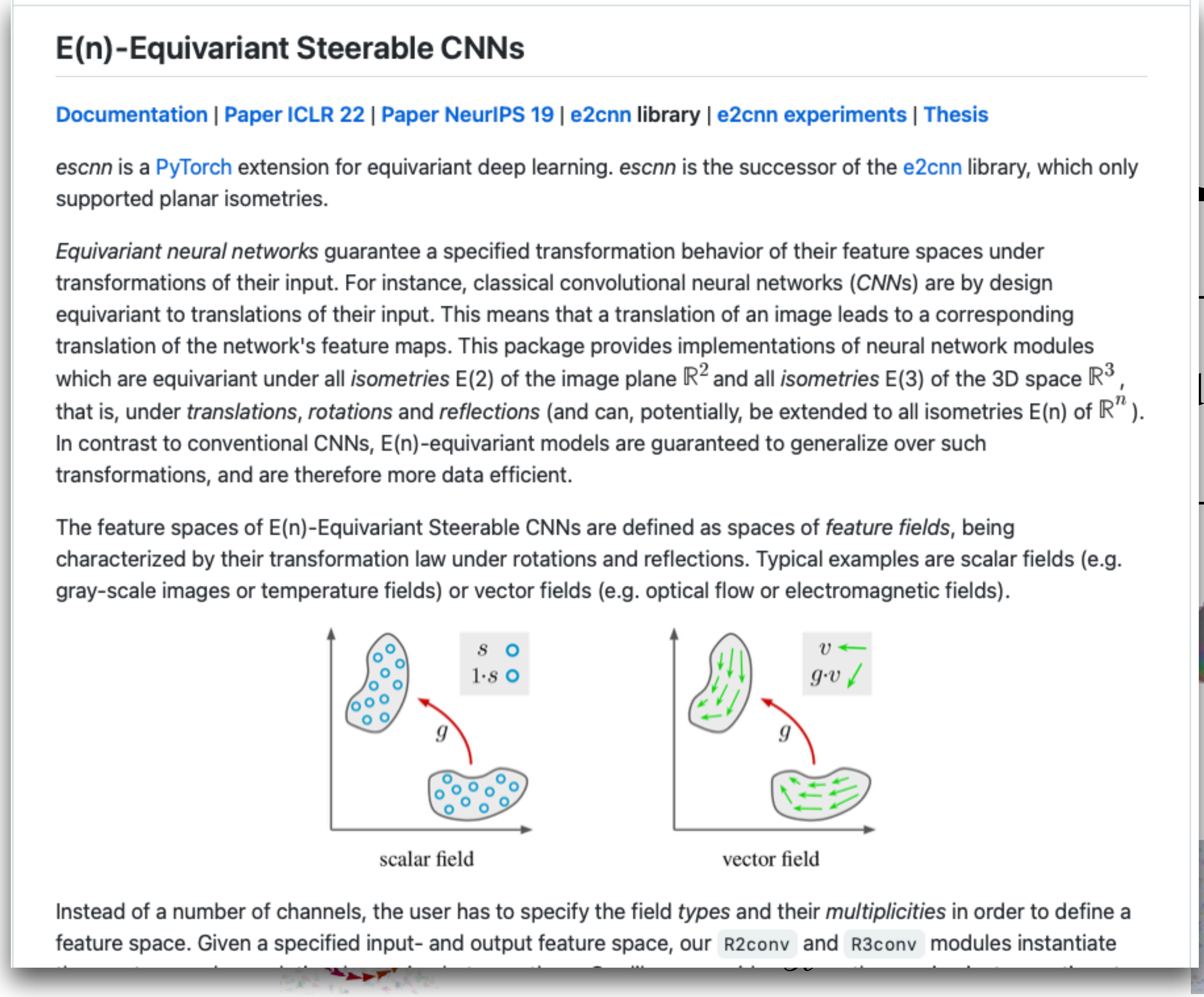
lution of this kernel
nt models, covering
etworks [12], gated
15, 16, 17, 18] and
uild hybrid models,
and within layers.

Amsterdam.

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

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

up convolutions



E(n)-Equivariant Steerable CNNs

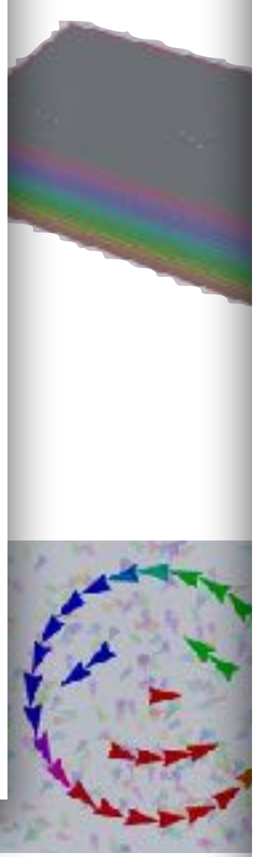
Documentation | Paper ICLR 22 | Paper NeurIPS 19 | e2cnn library | e2cnn experiments | Thesis

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 (*CNN*s) 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).

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.

1

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

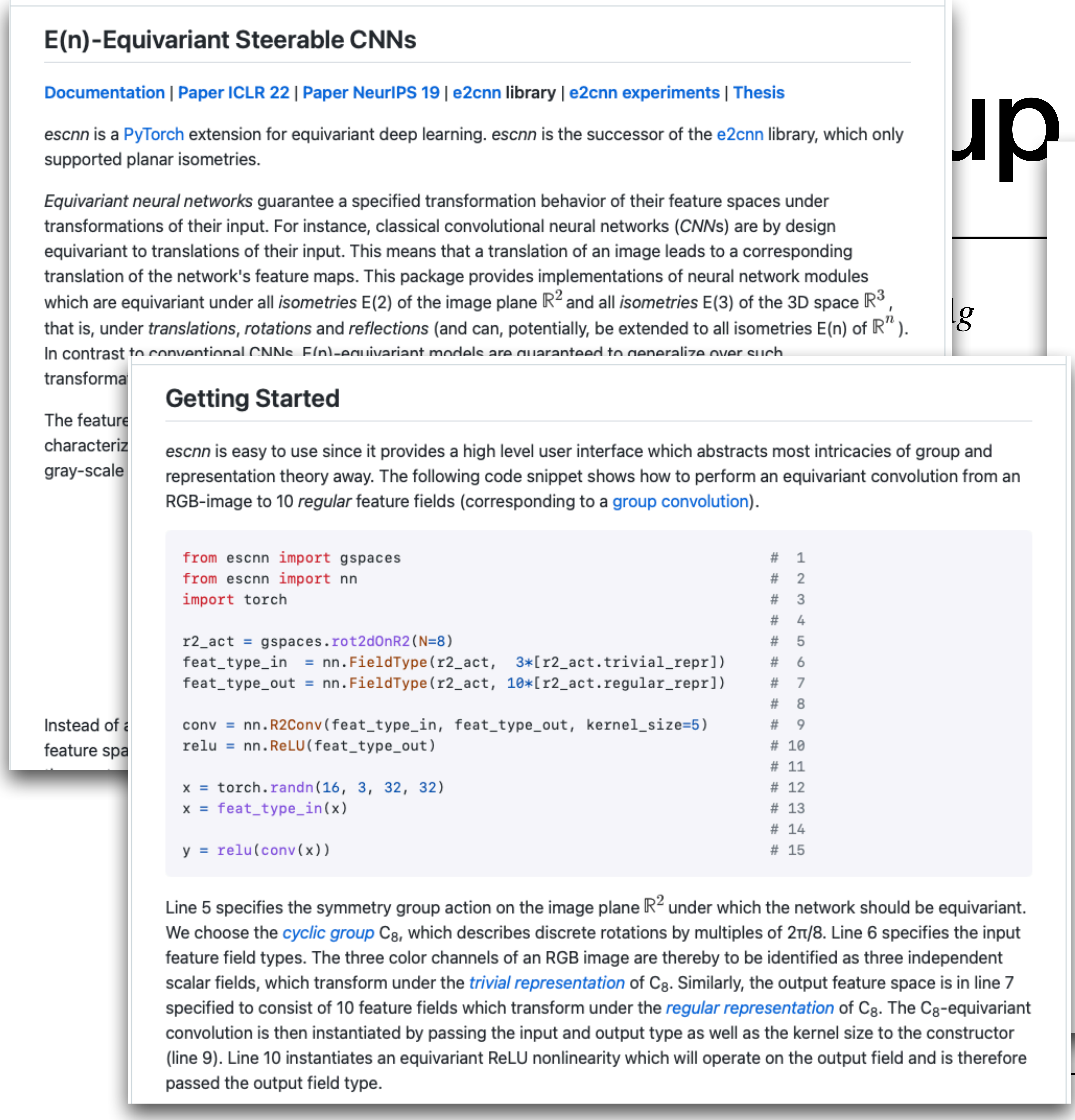
Normal convolution $\mathscr{K}[\hat{f}](\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \mathrm{d}\mathbf{x}'$

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

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

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 (*CNN*s) 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.

lg

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 $G \leq O(3)$-steerable kernels. For example, we build 3D CNNs equivariant to the symmetries of platonic solids or choose $G = SO(2)$ 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.

1

NNs

r*†
terdam
ail.com

nt years to
particular
for planar
lutions in
eby yields
sentations
constraints
rreducible
given for
ups. We
quivariant
-steerable
IFAR-100
olutions.

ecent years proven
nation behavior of
 networks achieve
ir non-equivariant
lection- equivariant
survey, reproducing

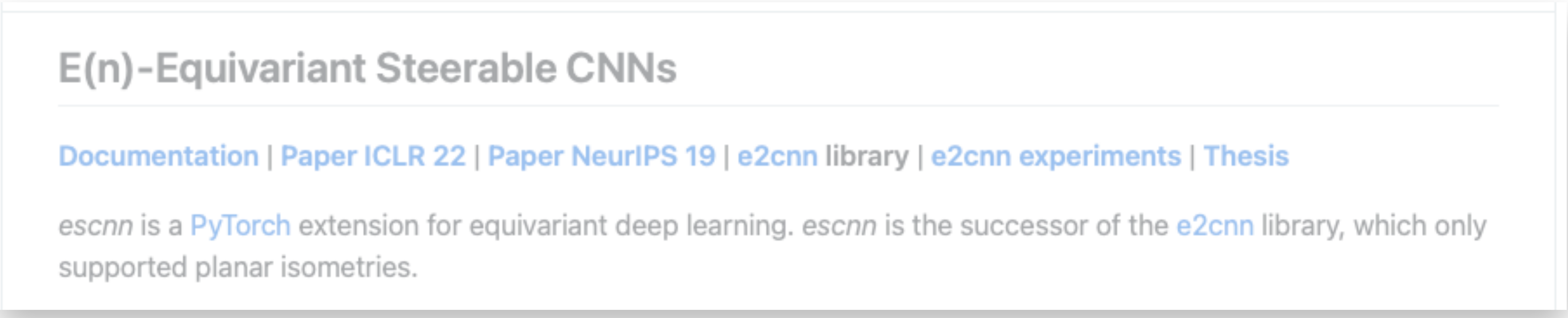
1, 2, 3, 4, 5] which
aces. In particular,
nt) convolutions on
s spaces of *feature*
tion behavior under
v of feature spaces,
rresponding group
presentations [1, 2],
eral strategy which
ns to much simpler

ution 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



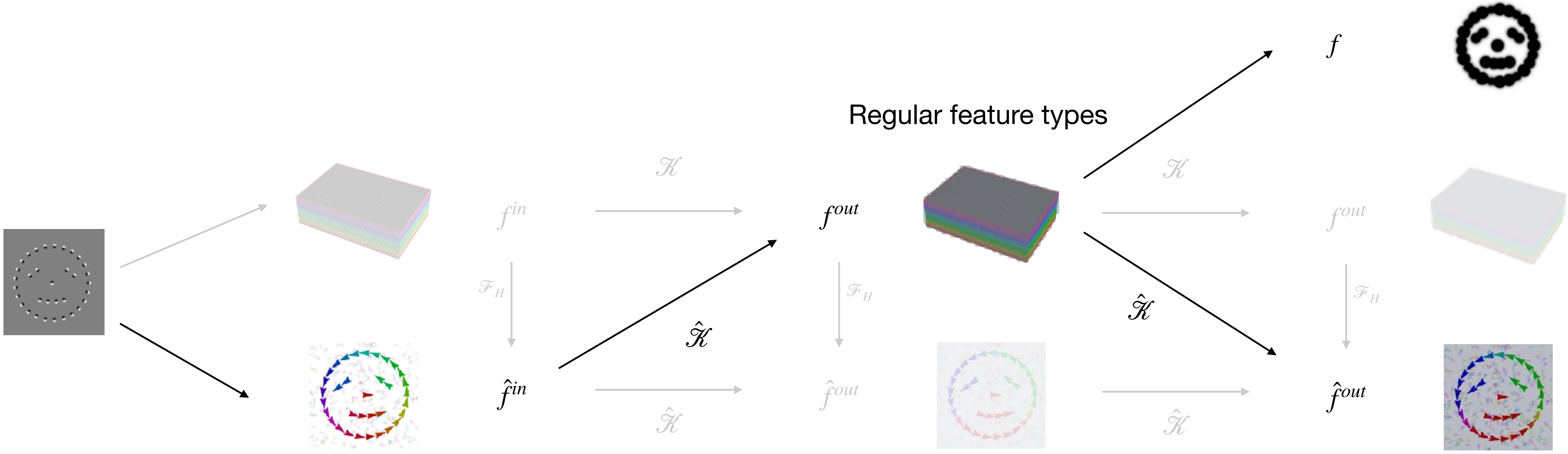
E(n)-Equivariant Steerable CNNs

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)



Regular feature types

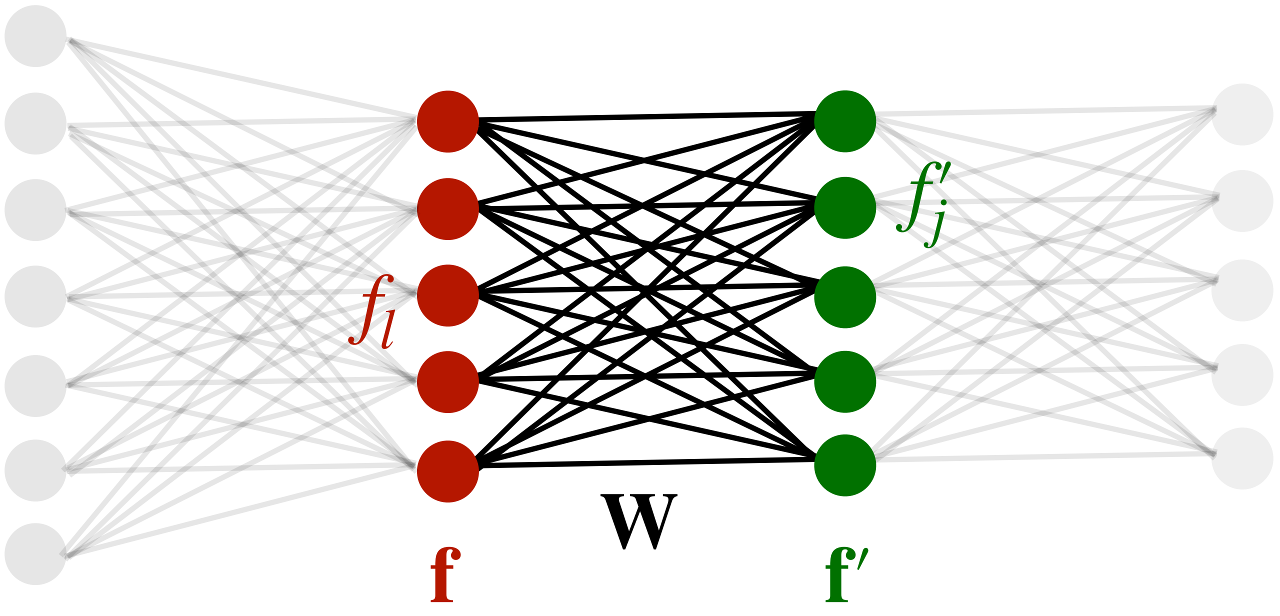
Steerable (irrep) feature types

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

1. Motivation	Equivariance → weight-sharing and generalization
2. Pattern matching using group theory	Group theory: symmetries & recognition by components (features have "poses")
3. Group convolutions	Template matching over groups
4. Example	Effective representation learning and generalization
5. G-convs are all you need!	Any equivariant linear layer is a group convolution
6. Steerable group convolutions	Efficient (band-limited) **grid-free** g-convs
7. Feature fields and escnn library	Flexible framework for equivariant layers
8. Equivariant tensor product layers	
9. Equivariant graph NNs	

1. Motivation	Equivariance → weight-sharing and generalization
2. Pattern matching using group theory	Group theory: symmetries & recognition by components (features have “poses”)
3. Group convolutions	Template matching over groups
4. Example	Effective representation learning and generalization
5. G-convs are all you need!	Any equivariant linear layer is a group convolution
6. Steerable group convolutions	Efficient (band-limited) **grid-free** g-convs
7. Feature fields and escnn library	Flexible framework for equivariant layers
8. Equivariant tensor product layers	
9. Equivariant graph NNs	

Equivariant MLP



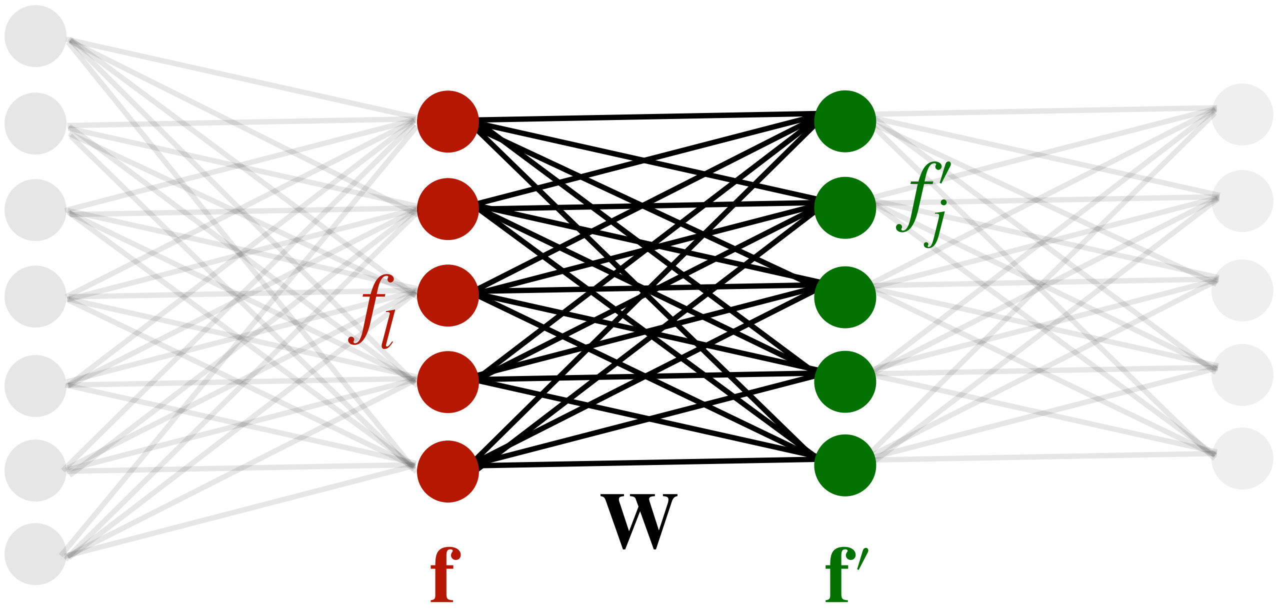
$$\mathbf{f} \quad \mapsto \quad \underbrace{\mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f}}_{\text{linear layer}} \quad \mapsto \quad \underbrace{\mathbf{f}'' = \sigma(\,\mathbf{f}'\,)}_{\text{activation}}$$

(Repeat L times)

Linear layer (matrix-vector multiplication)

$$\mathbf{f}' = \mathbf{W}\ \mathbf{f} \qquad\qquad f'_j = \sum_l w_l^j\, f_l$$

Equivariant MLP



$$\mathbf{f} \underset{\text{linear layer}}{\mapsto} \mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \underset{\text{activation}}{\mapsto} \mathbf{f}'' = \sigma(\,\mathbf{f}'\,)$$

(Repeat L times)

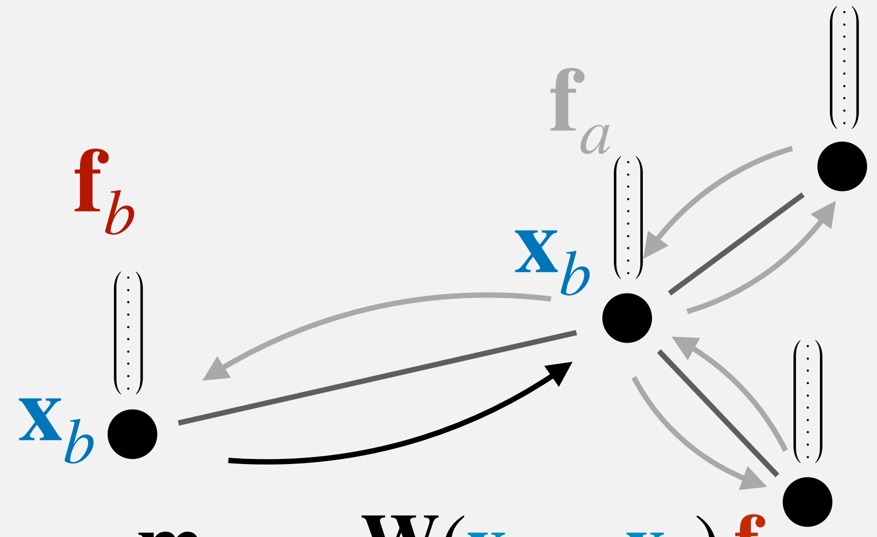
Linear layer (matrix-vector multiplication)

$$\mathbf{f}' = \mathbf{W}\ \mathbf{f} \qquad f'_j = \sum_l w_l^j\, f_l$$

Conditional linear layer (weight matrix depends on $\mathbf{x}_b - \mathbf{x}_a$)

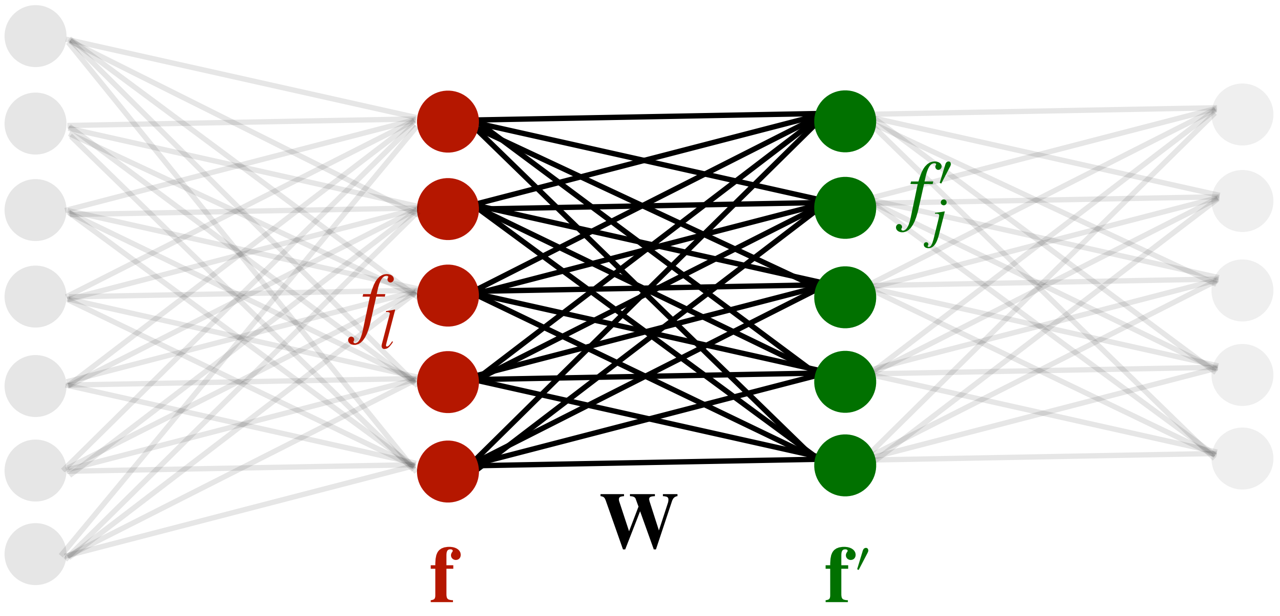
$$\mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\ \mathbf{f} \qquad f'_j = \sum_l w_l^j(\mathbf{x}_b - \mathbf{x}_a)\, f_l$$

Convolutional message passing



$$\mathbf{m}_{ab} = \mathbf{W}\underbrace{(\mathbf{x}_b - \mathbf{x}_a)}_{\text{geometric attribute}}\,\mathbf{f}_b$$

Equivariant MLP



$$\mathbf{f} \underset{\text{linear layer}}{\mapsto} \mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \underset{\text{activation}}{\mapsto} \mathbf{f}'' = \sigma(\,\mathbf{f}'\,)$$

(Repeat L times)

Linear layer (matrix-vector multiplication)

$$\mathbf{f}' = \mathbf{W}\;\mathbf{f} \qquad\qquad f'_j = \sum_l w^j_l\, f_l$$

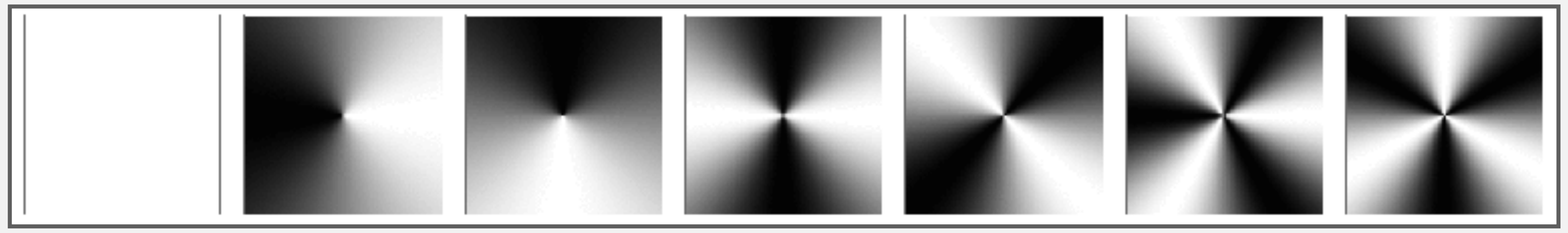
Conditional linear layer (weight matrix depends on $\mathbf{x}_b - \mathbf{x}_a$)

$$\mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \qquad\qquad f'_j = \sum_l w^j_l(\mathbf{x}_b - \mathbf{x}_a)\, f_l$$

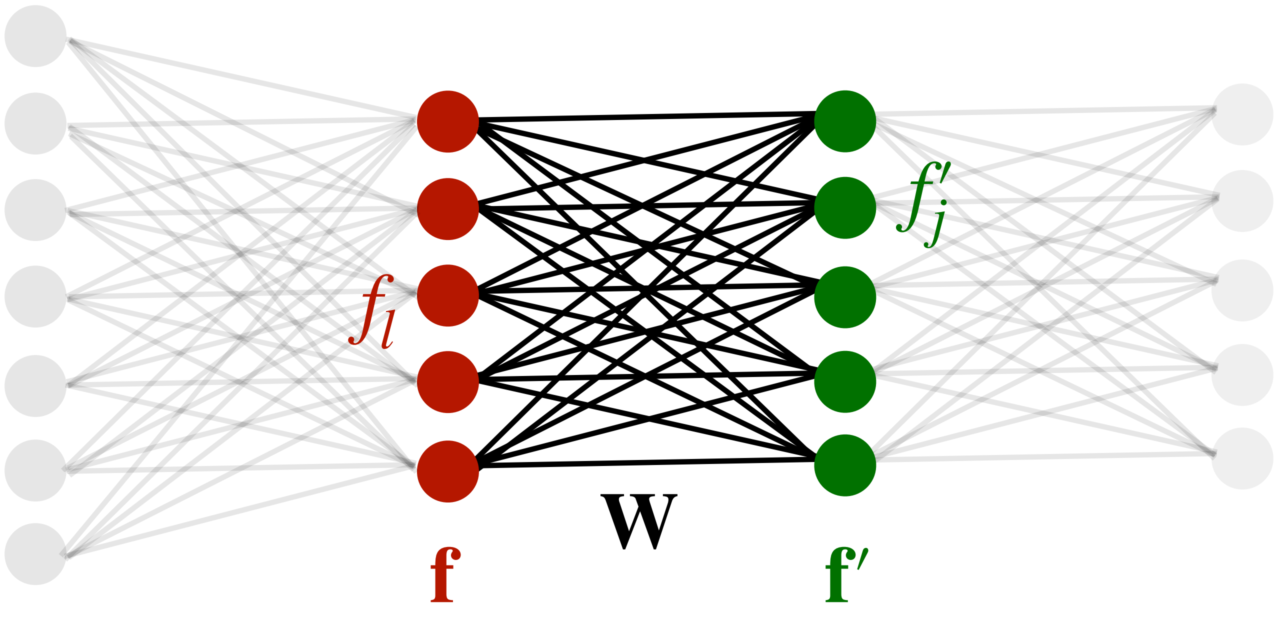
Let $\mathbf{W} : \mathbb{R}^3 \to \mathbb{R}^{C' \times C}$ a matrix valued function (conv-kernel)

- Expanded in a basis $Y(\mathbf{x}) = \begin{pmatrix} \vdots \\ Y_J(\mathbf{x}) \\ \vdots \end{pmatrix}$
- Basis (coordinate embedding) functions $Y_J : \mathbb{R}^3 \to \mathbb{R}$
- Matrix-valued weights $\mathbf{W}_J$ with elements w^j_{Jl}

$$\mathbf{W}(\mathbf{x}_b - \mathbf{x}_a) = \sum_J \mathbf{W}_J\, Y_J(\mathbf{x}_b - \mathbf{x}_a)$$



Equivariant MLP



$$\mathbf{f} \quad \mapsto \quad \underbrace{\mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f}}_{\text{linear layer}} \quad \mapsto \quad \underbrace{\mathbf{f}'' = \sigma(\,\mathbf{f}'\,)}_{\text{activation}}$$

(Repeat L times)

Linear layer (matrix-vector multiplication)

$$\mathbf{f}' = \mathbf{W}\;\mathbf{f} \qquad\qquad f'_j = \sum_l w^j_l\, f_l$$

Conditional linear layer (weight matrix depends on $\mathbf{x}_b - \mathbf{x}_a$)

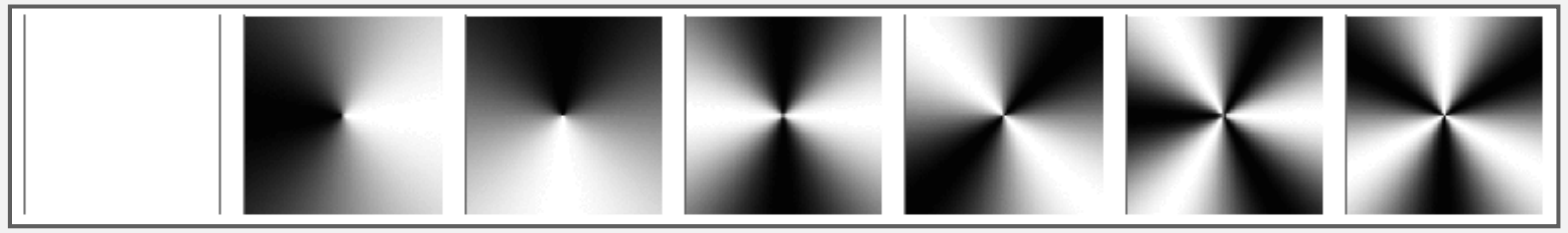
$$\mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \qquad\qquad f'_j = \sum_l w^j_l(\mathbf{x}_b - \mathbf{x}_a)\, f_l$$

$$\mathbf{f}' = \mathbf{f} \overset{bilinear}{W} Y_J(\mathbf{x}_b - \mathbf{x}_a) \qquad\qquad f'_j = \sum_l \sum_J w^j_{Jl}\, Y_J(\mathbf{x}_b - \mathbf{x}_a)\, f_l$$

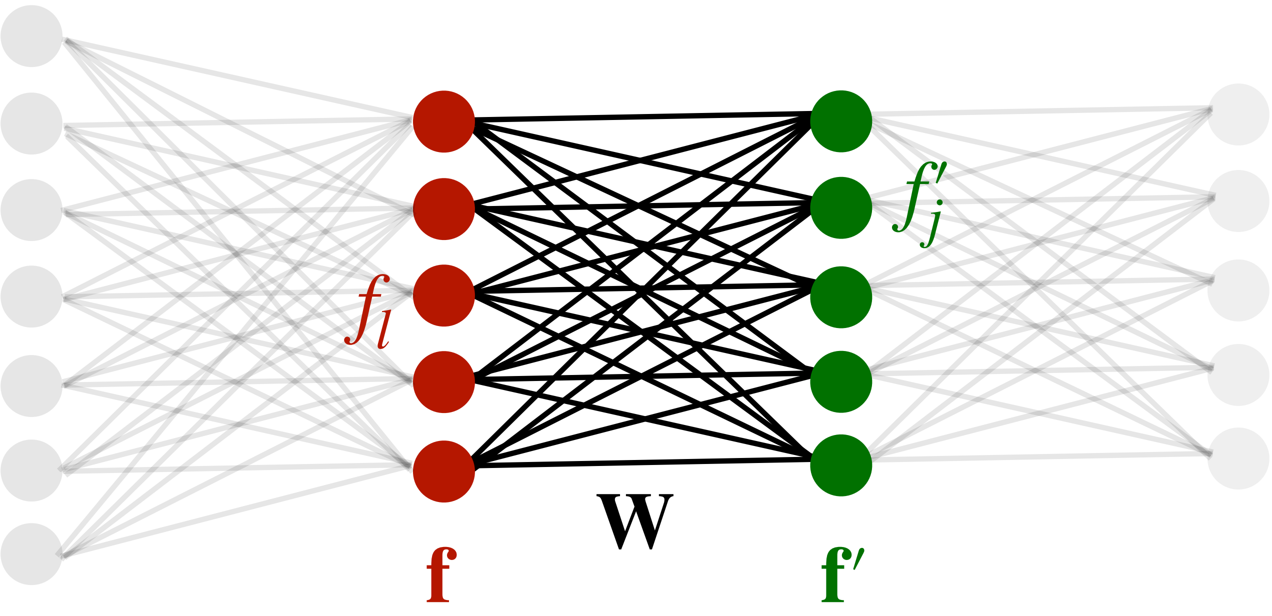
Let $\mathbf{W} : \mathbb{R}^3 \rightarrow \mathbb{R}^{C' \times C}$ a matrix valued function (conv-kernel)

- Expanded in a basis $Y(\mathbf{x}) = \begin{pmatrix} \vdots \\ Y_J(\mathbf{x}) \\ \vdots \end{pmatrix}$
- Basis (coordinate embedding) functions $Y_J : \mathbb{R}^3 \rightarrow \mathbb{R}$
- Matrix-valued weights $\mathbf{W}_J$ with elements w^j_{Jl}

$$\mathbf{W}(\mathbf{x}_b - \mathbf{x}_a) = \sum_J \mathbf{W}_J\, Y_J(\mathbf{x}_b - \mathbf{x}_a)$$



Equivariant MLP



$$\mathbf{f} \quad \underset{\text{linear layer}}{\mapsto} \quad \mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \quad \underset{\text{activation}}{\mapsto} \quad \mathbf{f}'' = \sigma(\,\mathbf{f}'\,)$$

(Repeat L times)

Linear layer (matrix-vector multiplicati

$$\mathbf{f}' = \mathbf{W}\ \mathbf{f}$$

Conditional linear layer

$$\mathbf{x}_b - \mathbf{x}_a)$$

$$\mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \qquad f_j' = \sum_l w^j_l(\mathbf{x}_b - \mathbf{x}_a)\, f_l$$

$$\mathbf{f}' = \mathbf{f} \overset{bilinear}{W} Y_J(\mathbf{x}_b - \mathbf{x}_a) \qquad f_j' = \sum_l \sum_J w^j_{Jl}\, Y_J(\mathbf{x}_b - \mathbf{x}_a)\, f_l$$

Conditional linear layers are tensor products!!!

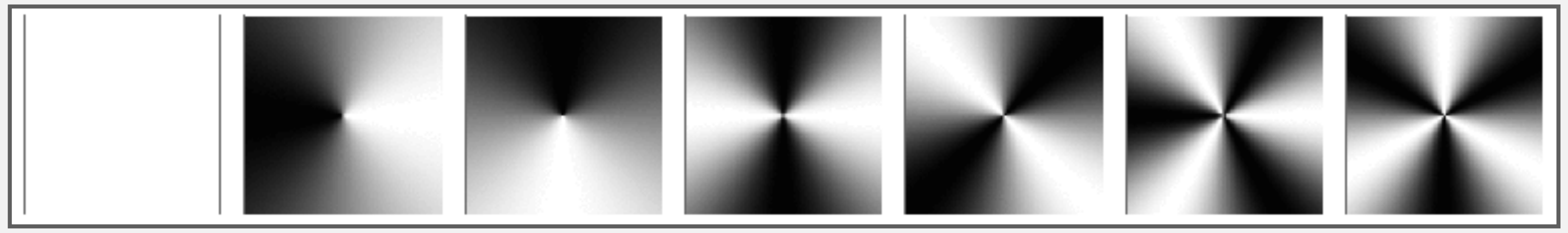
$$\mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \quad \Leftrightarrow \quad \mathbf{f}' = \mathbf{f} \otimes^{\mathbf{w}} Y_J(\mathbf{x}_b - \mathbf{x}_a)$$

atrix valued function (conv-kernel)

xpanded in a basis $Y(\mathbf{x}) = \begin{pmatrix} \vdots \\ Y_J(\mathbf{x}) \\ \vdots \end{pmatrix}$

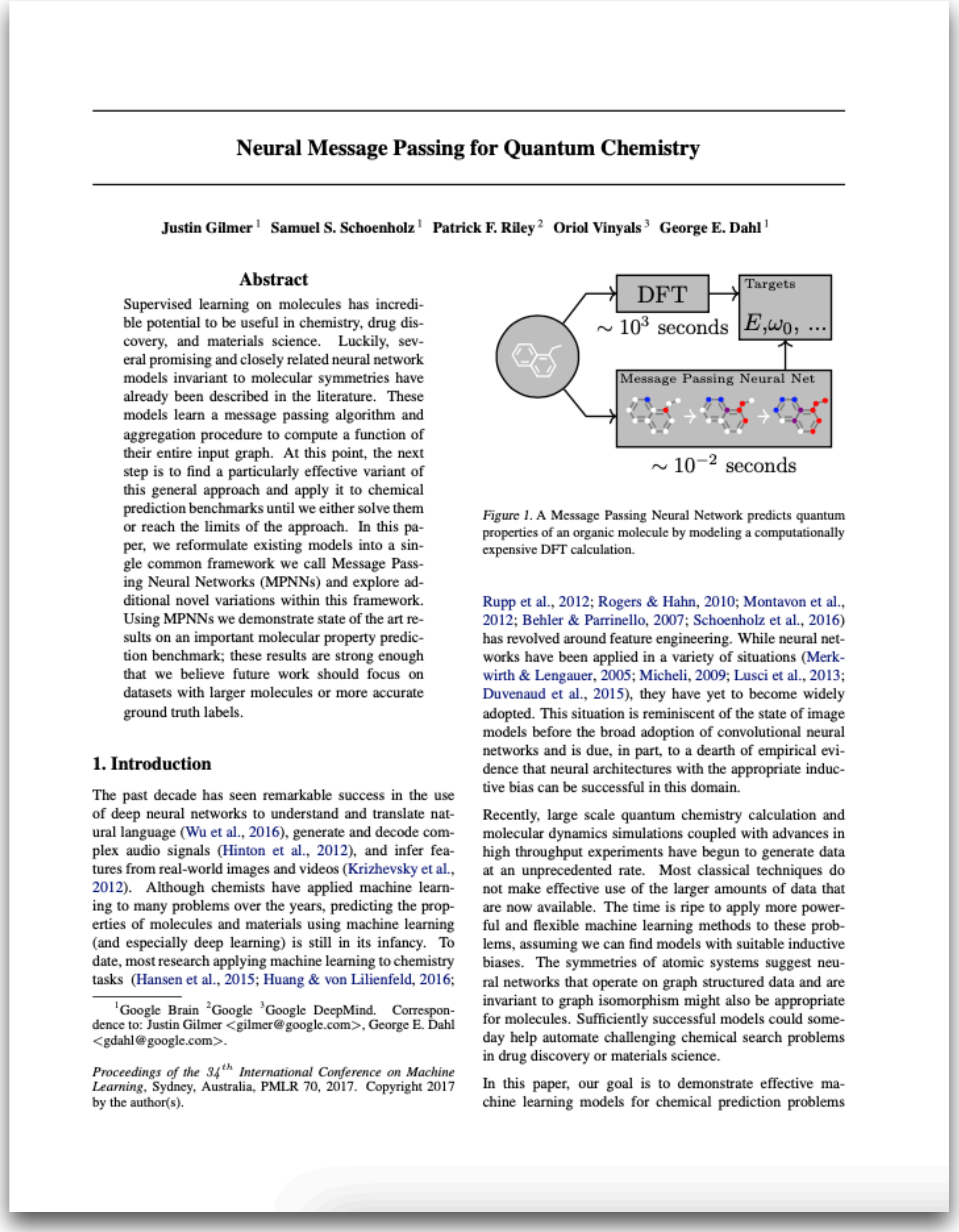
- Basis (coordinate embedding) functions $Y_J : \mathbb{R}^3 \rightarrow \mathbb{R}$
- Matrix-valued weights $\mathbf{W}_J$ with elements w^j_{Jl}

$$\mathbf{W}(\mathbf{x}_b - \mathbf{x}_a) = \sum_J \mathbf{W}_J\, Y_J(\mathbf{x}_b - \mathbf{x}_a)$$



1. Motivation	Equivariance → weight-sharing and generalization
2. Pattern matching using group theory	Group theory: symmetries & recognition by components (features have “poses”)
3. Group convolutions	Template matching over groups
4. Example	Effective representation learning and generalization
5. G-convs are all you need!	Any equivariant linear layer is a group convolution
6. Steerable group convolutions	Efficient (band-limited) **grid-free** g-convs
7. Feature fields and escnn library	Flexible framework for equivariant layers
8. Equivariant tensor product layers	Conv layers ↔ TPs with coordinate embeddings (Clebsch-Gordan: equivariant TP)
9. Equivariant graph NNs	

1. Motivation	Equivariance → weight-sharing and generalization
2. Pattern matching using group theory	Group theory: symmetries & recognition by components (features have “poses”)
3. Group convolutions	Template matching over groups
4. Example	Effective representation learning and generalization
5. G-convs are all you need!	Any equivariant linear layer is a group convolution
6. Steerable group convolutions	Efficient (band-limited) **grid-free** g-convs
7. Feature fields and escnn library	Flexible framework for equivariant layers
8. Equivariant tensor product layers	Conv layers ↔ TPs with coordinate embeddings (Clebsch-Gordan: equivariant TP)
9. Equivariant graph NNs	



Neural Message Passing for Quantum Chemistry

Justin Gilmer[1] **Samuel S. Schoenholz**[1] **Patrick F. Riley**[2] **Oriol Vinyals**[3] **George E. Dahl**[1]

Abstract

Supervised learning on molecules has incredible potential to be useful in chemistry, drug discovery, and materials science. Luckily, several promising and closely related neural network models invariant to molecular symmetries have already been described in the literature. These models learn a message passing algorithm and aggregation procedure to compute a function of their entire input graph. At this point, the next step is to find a particularly effective variant of this general approach and apply it to chemical prediction benchmarks until we either solve them or reach the limits of the approach. In this paper, we reformulate existing models into a single common framework we call Message Passing Neural Networks (MPNNs) and explore additional novel variations within this framework. Using MPNNs we demonstrate state of the art results on an important molecular property prediction benchmark; these results are strong enough that we believe future work should focus on datasets with larger molecules or more accurate ground truth labels.

1. Introduction

The past decade has seen remarkable success in the use of deep neural networks to understand and translate natural language (Wu et al., 2016), generate and decode complex audio signals (Hinton et al., 2012), and infer features from real-world images and videos (Krizhevsky et al., 2012). Although chemists have applied machine learning to many problems over the years, predicting the properties of molecules and materials using machine learning (and especially deep learning) is still in its infancy. To date, most research applying machine learning to chemistry tasks (Hansen et al., 2015; Huang & von Lilienfeld, 2016; Rupp et al., 2012; Rogers & Hahn, 2010; Montavon et al., 2012; Behler & Parrinello, 2007; Schoenholz et al., 2016) has revolved around feature engineering. While neural networks have been applied in a variety of situations (Merkwirth & Lengauer, 2005; Micheli, 2009; Lusci et al., 2013; Duvenaud et al., 2015), they have yet to become widely adopted. This situation is reminiscent of the state of image models before the broad adoption of convolutional neural networks and is due, in part, to a dearth of empirical evidence that neural architectures with the appropriate inductive bias can be successful in this domain.

Figure 1. A Message Passing Neural Network predicts quantum properties of an organic molecule by modeling a computationally expensive DFT calculation.

Recently, large scale quantum chemistry calculation and molecular dynamics simulations coupled with advances in high throughput experiments have begun to generate data at an unprecedented rate. Most classical techniques do not make effective use of the larger amounts of data that are now available. The time is ripe to apply more powerful and flexible machine learning methods to these problems, assuming we can find models with suitable inductive biases. The symmetries of atomic systems suggest neural networks that operate on graph structured data and are invariant to graph isomorphism might also be appropriate for molecules. Sufficiently successful models could someday help automate challenging chemical search problems in drug discovery or materials science.

In this paper, our goal is to demonstrate effective machine learning models for chemical prediction problems

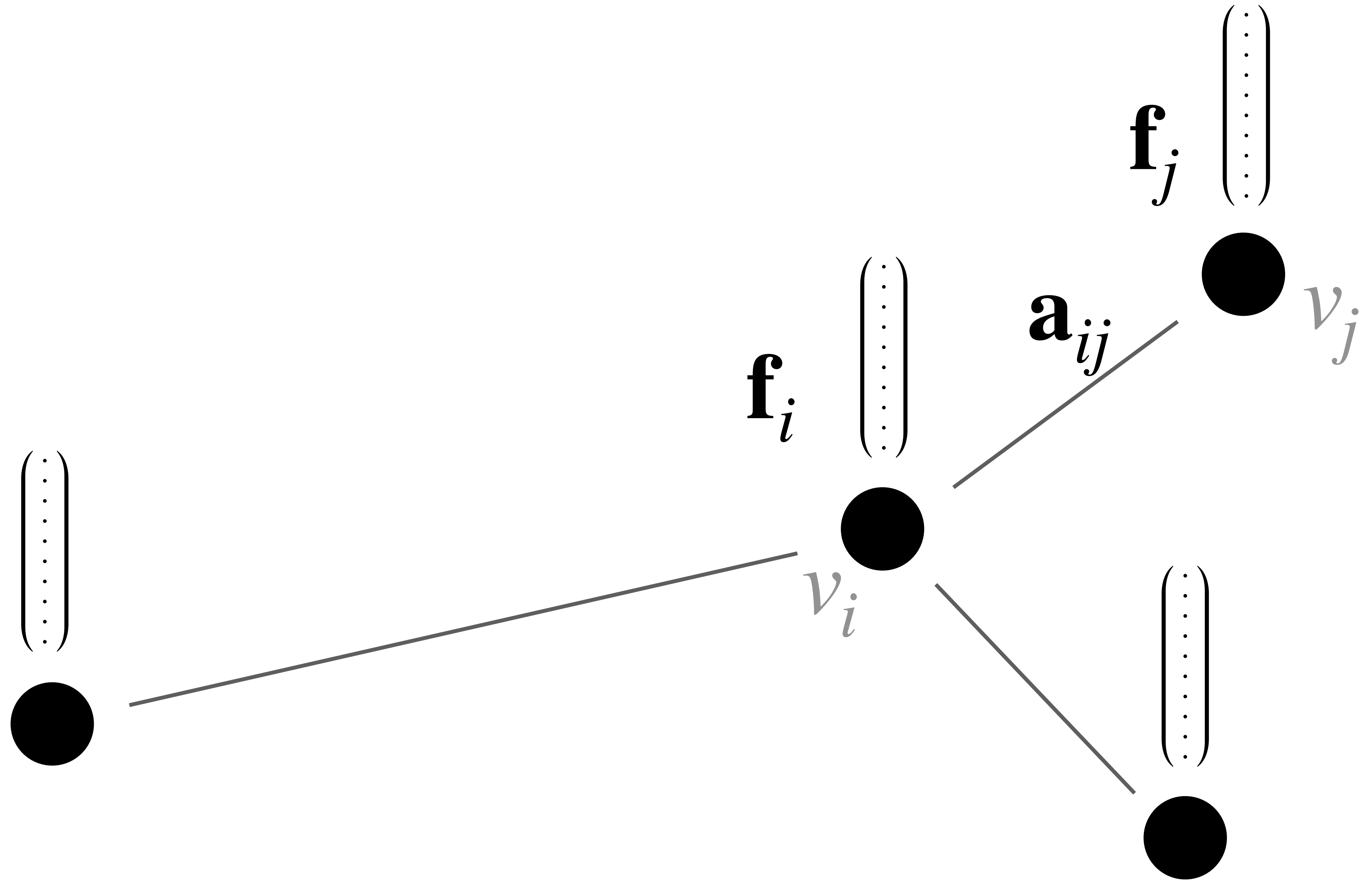
[1] Google Brain [2] Google [3] Google DeepMind. Correspondence to: Justin Gilmer <gilmer@google.com>, George E. Dahl <gdahl@google.com>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

The Message Passing Framework

Graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$

- nodes $v_i \in \mathscr{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

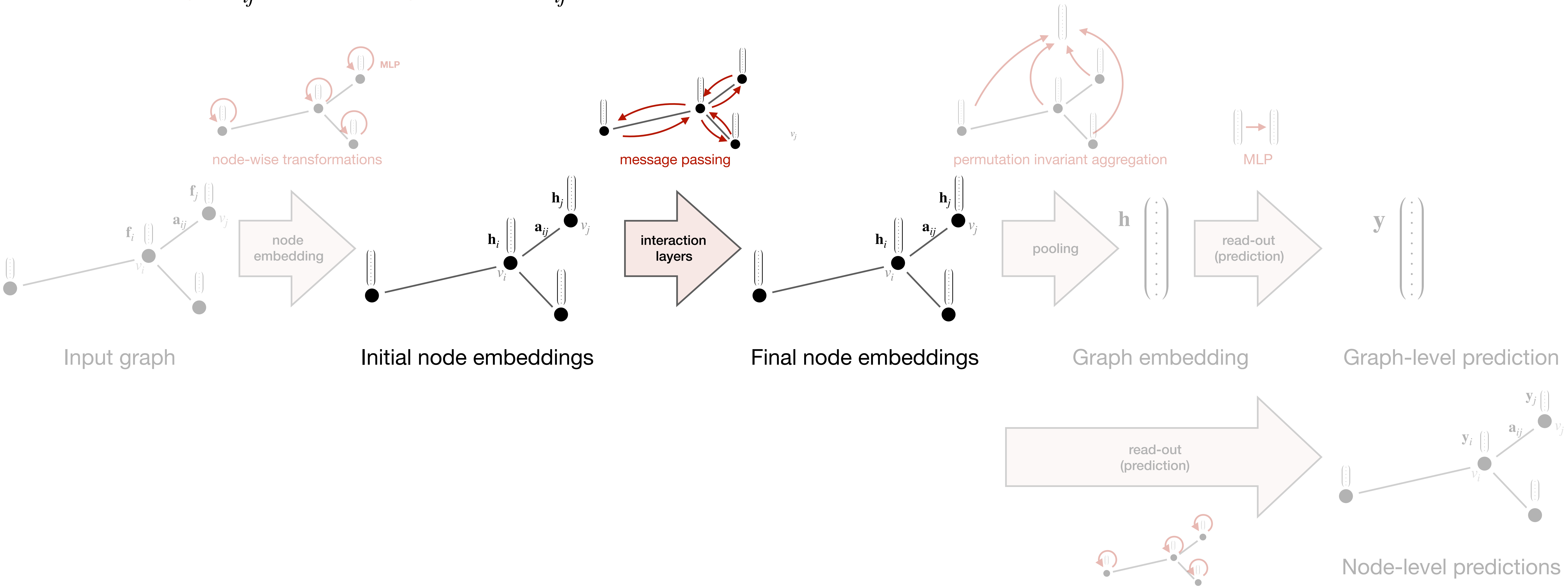


The Message Passing Framework

Graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$

- nodes $v_i \in \mathscr{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$



node-wise transformations

message passing

permutation invariant aggregation

MLP

node embedding

interaction layers

pooling

read-out (prediction)

Input graph

Initial node embeddings

Final node embeddings

Graph embedding

Graph-level prediction

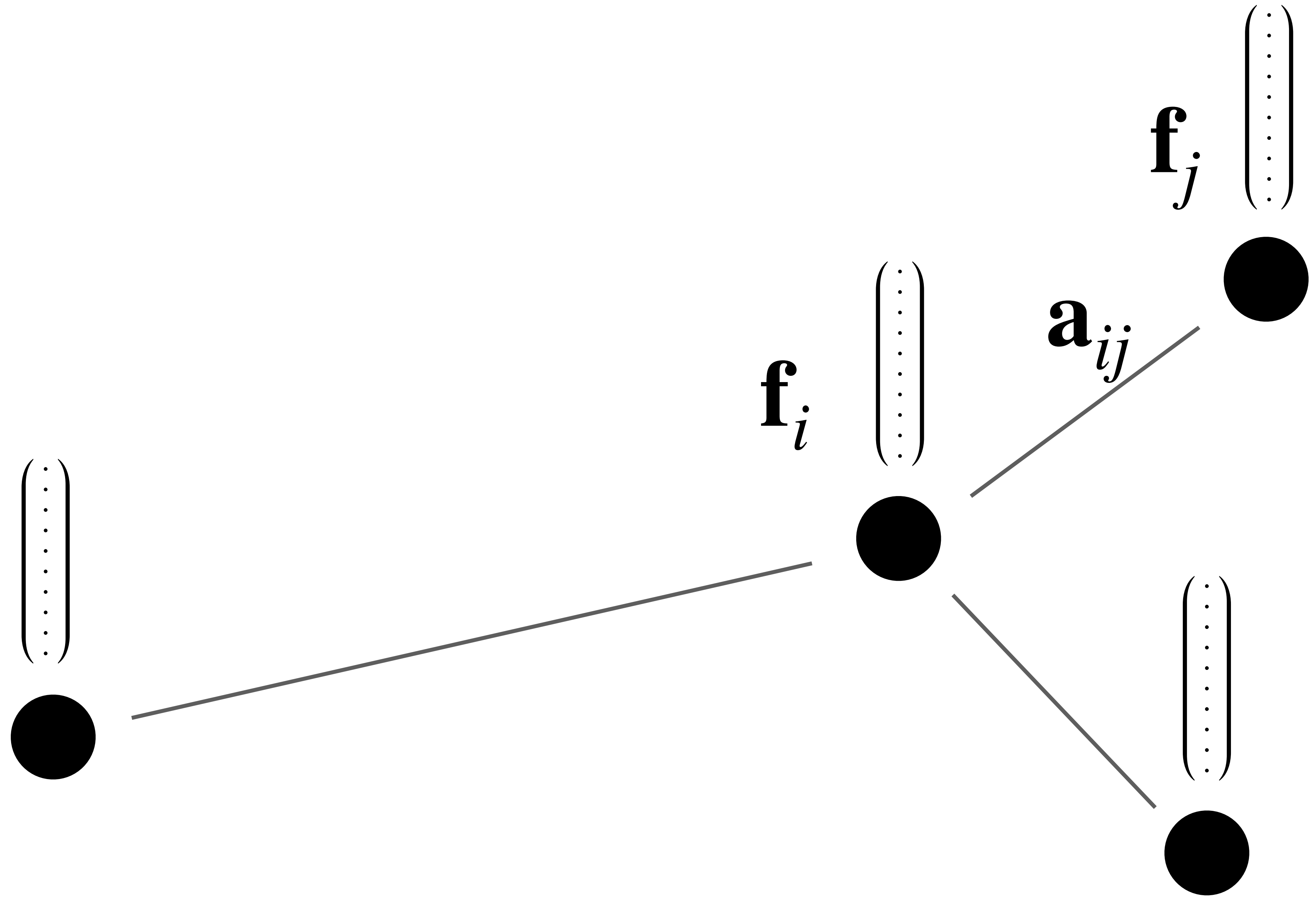
read-out (prediction)

Node-level predictions

The Message Passing Framework

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

- Messages

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

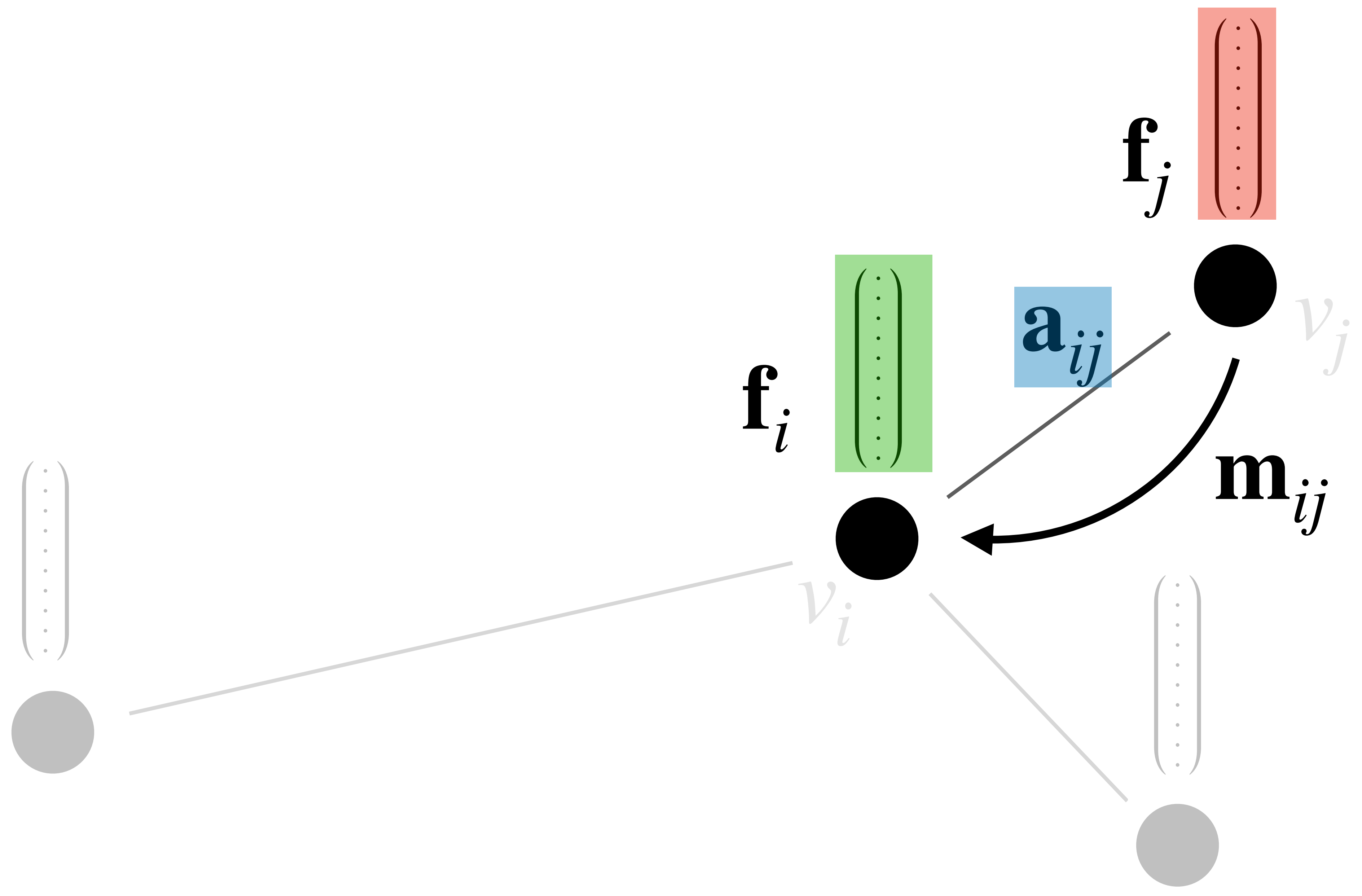
- Aggregate + node updates

$$\mathbf{f}_i' = \phi_f(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij})$$

The Message Passing Framework

Graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$

- nodes $v_i \in \mathscr{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

- Messages

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

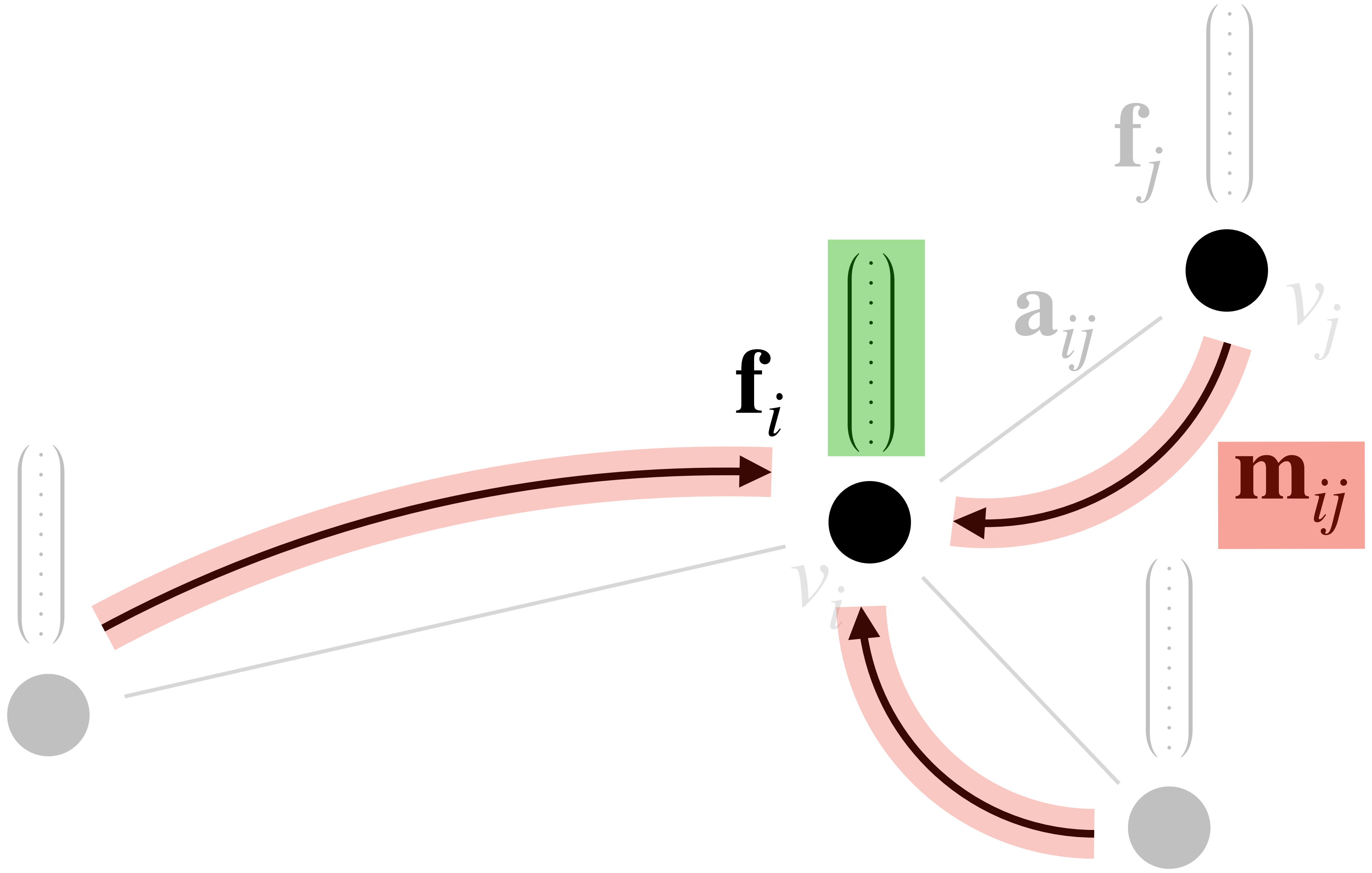
- Aggregate + node updates

$$\mathbf{f}_i' = \phi_f(\mathbf{f}_i, \sum_{j \in \mathscr{N}(i)} \mathbf{m}_{ij})$$

The Message Passing Framework

Graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$

- nodes $v_i \in \mathscr{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

- Messages

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

- Aggregate + node updates

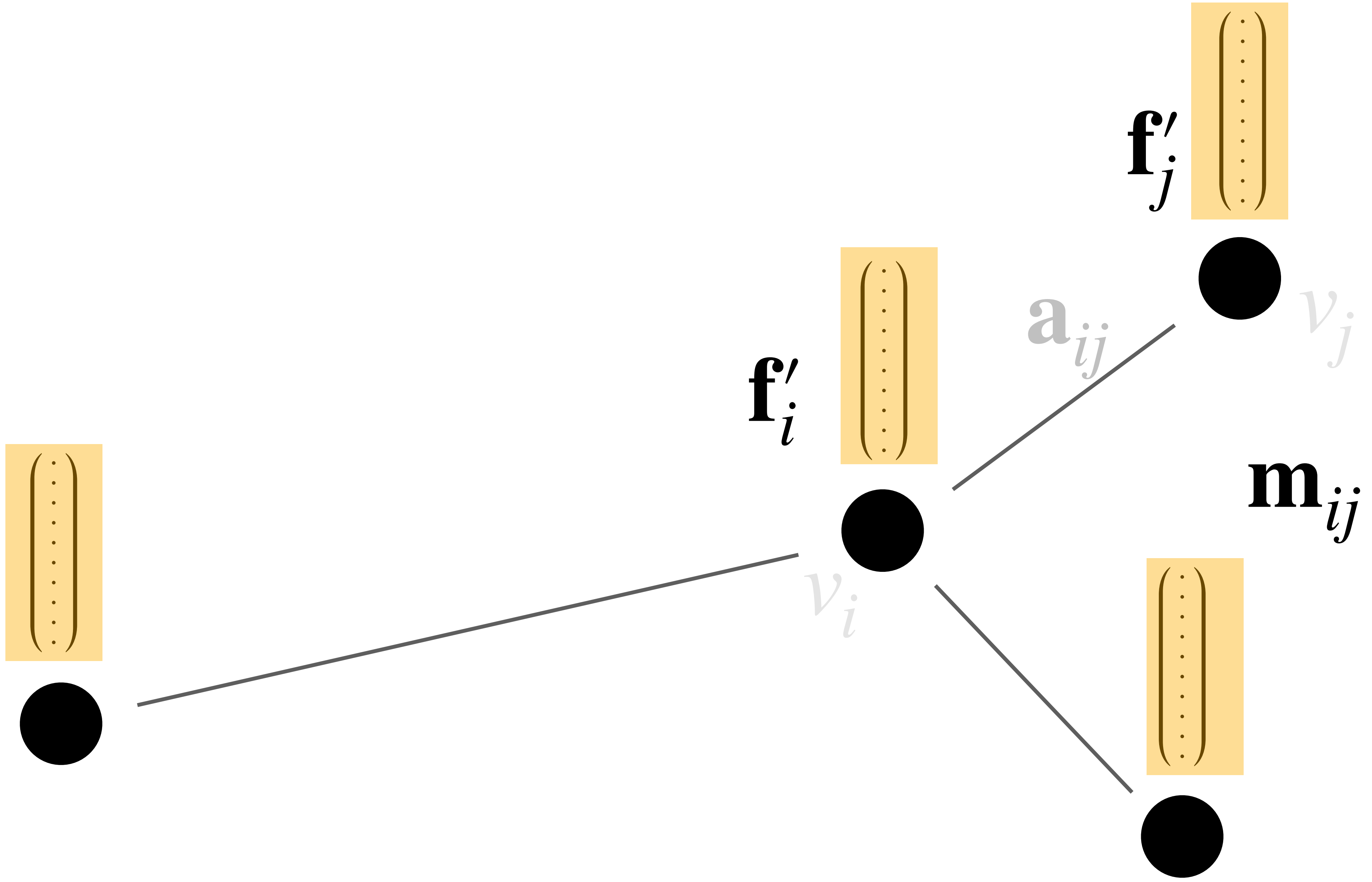
$$\mathbf{f}'_i = \phi_f(\mathbf{f}_i, \sum_{j \in \mathscr{N}(i)} \mathbf{m}_{ij})$$

The Message Passing Framework

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$



Message passing layer:

- Messages

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

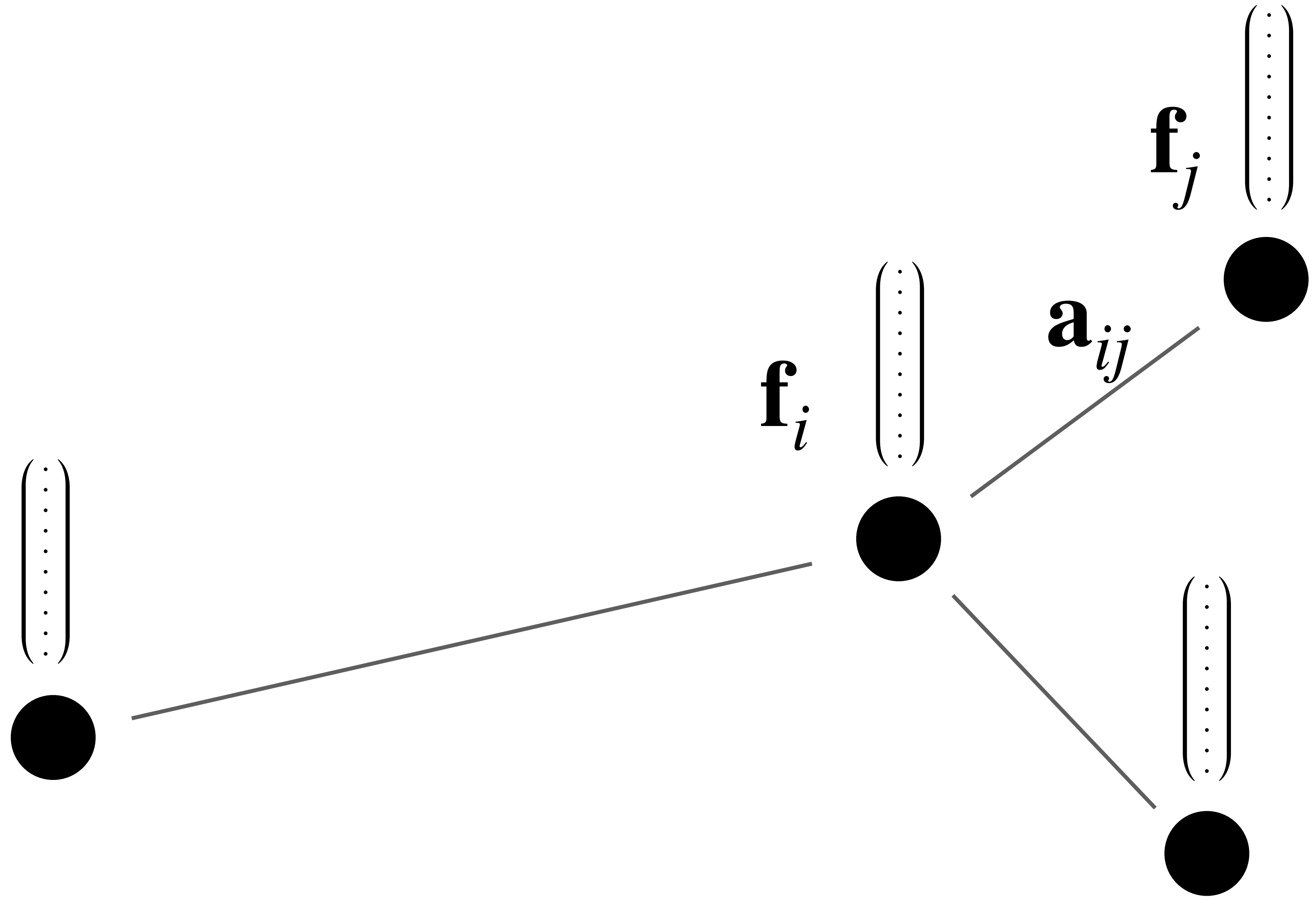
- Aggregate + node updates

$$\mathbf{f}'_i = \phi_f(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij})$$

The Message Passing Framework

Graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$

- nodes $v_i \in \mathscr{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

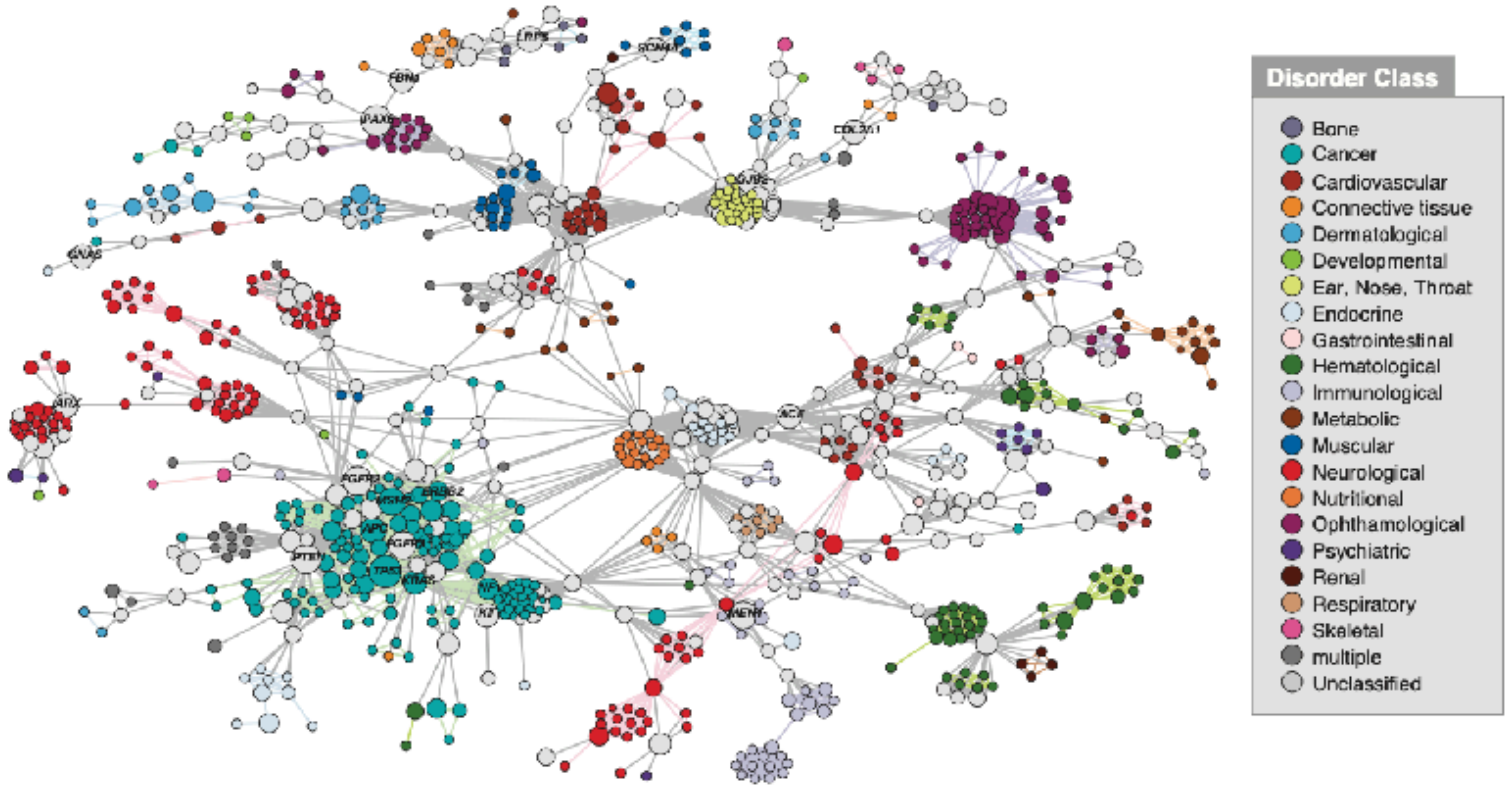
Message passing layer:

- Messages

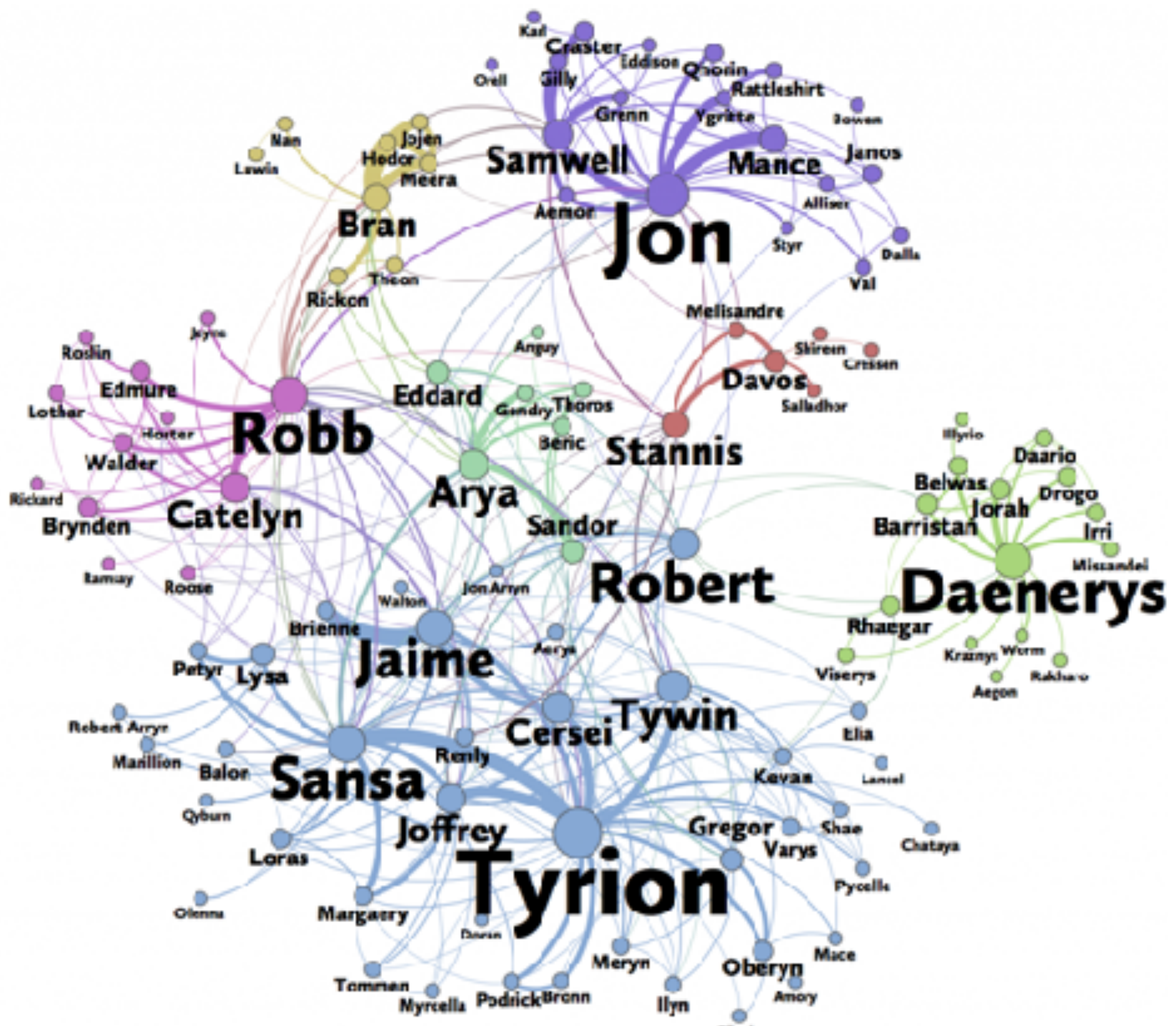
$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

- Aggregate + node updates

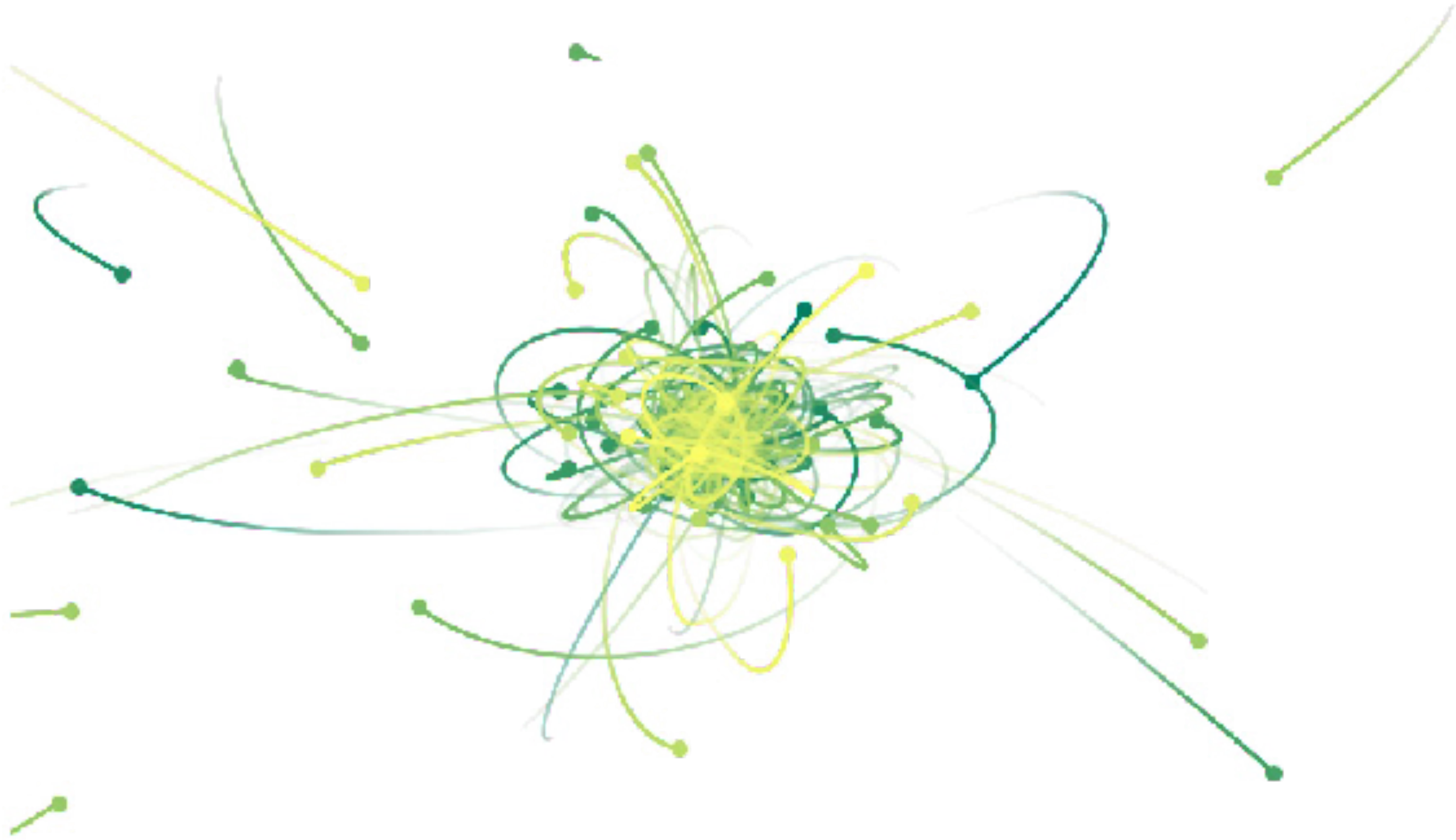
$$\mathbf{f}'_i = \phi_f(\mathbf{f}_i, \sum_{j \in \mathscr{N}(i)} \mathbf{m}_{ij})$$



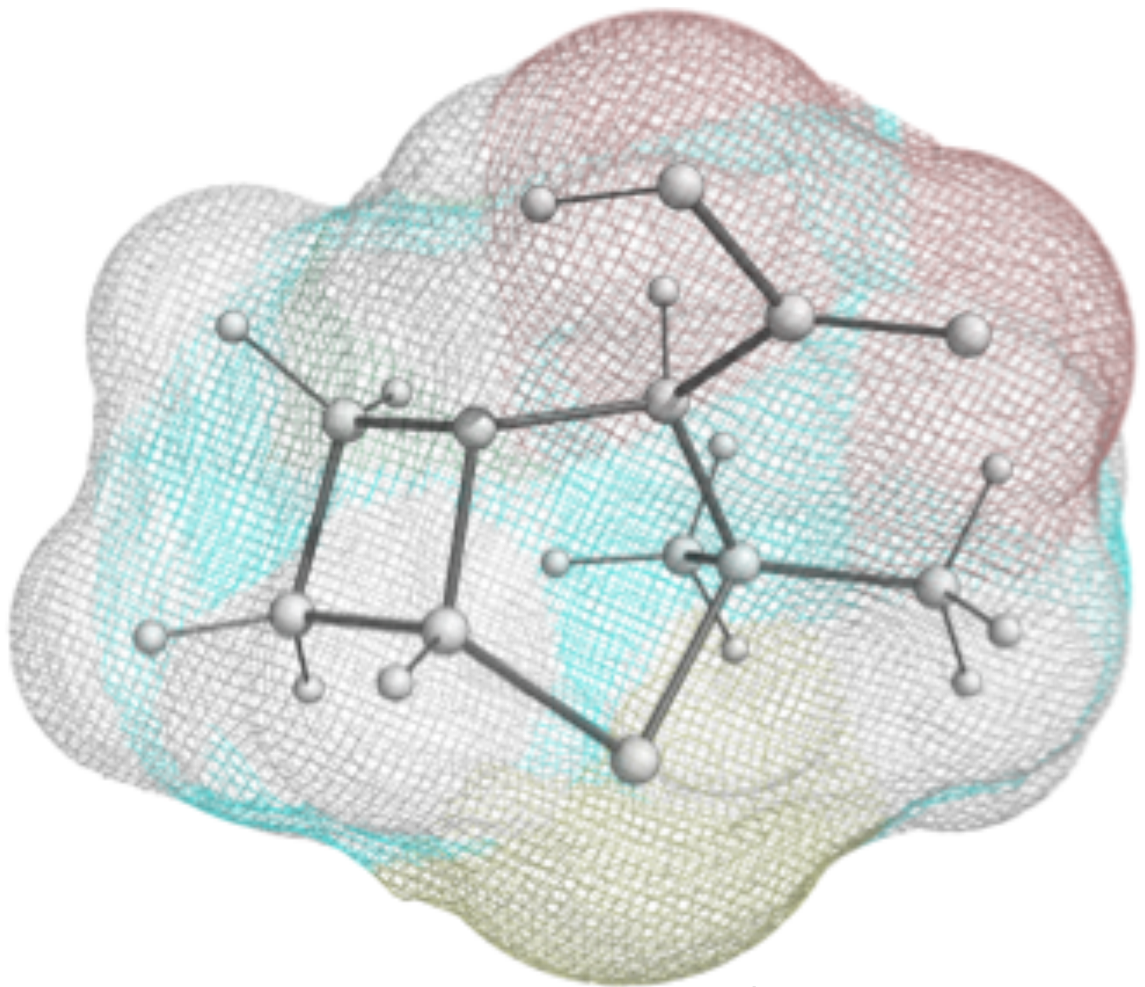
Gene/Protein interaction graphs[1]



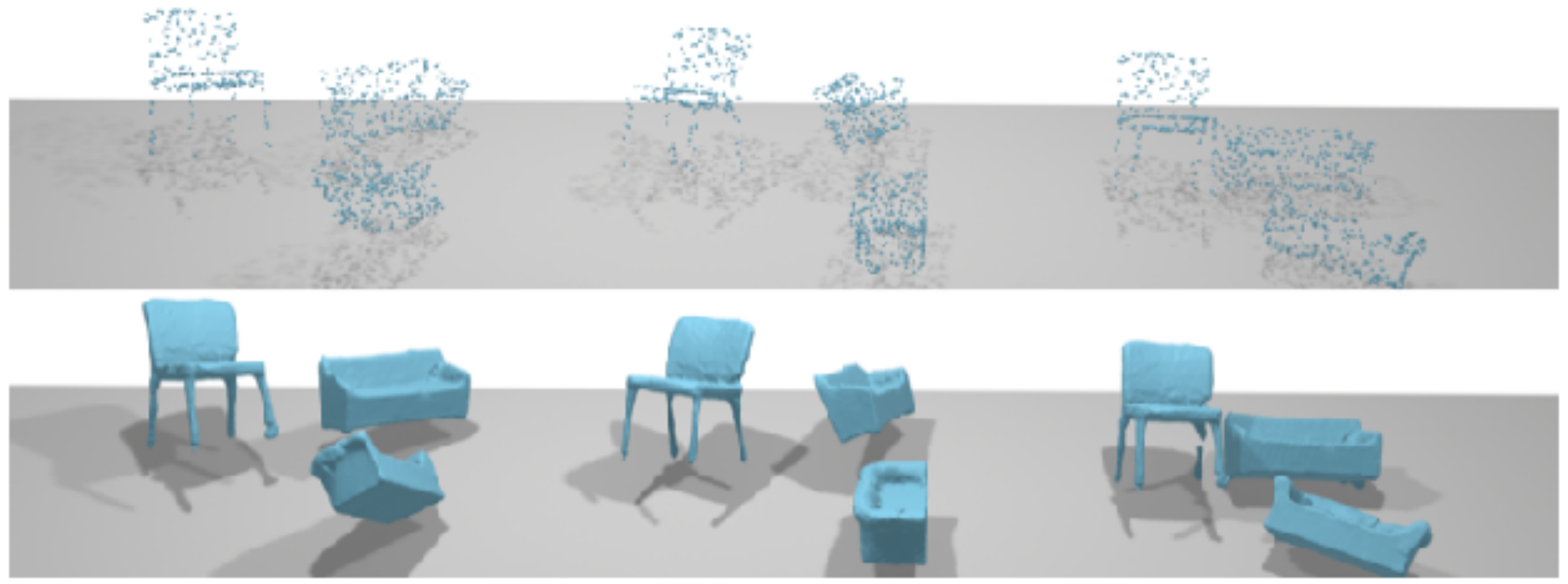
Social network graph[2]



Physical system[3]



Molecule[4]



Point cloud/shapes[5]

Figures from: [1] Goh, K. I., Cusick, M. E., Valle, D., Childs, B., Vidal, M., & Barabási, A. L. (2007). The human disease network. Proceedings of the National Academy of Sciences, 104(21), 8685-8690.

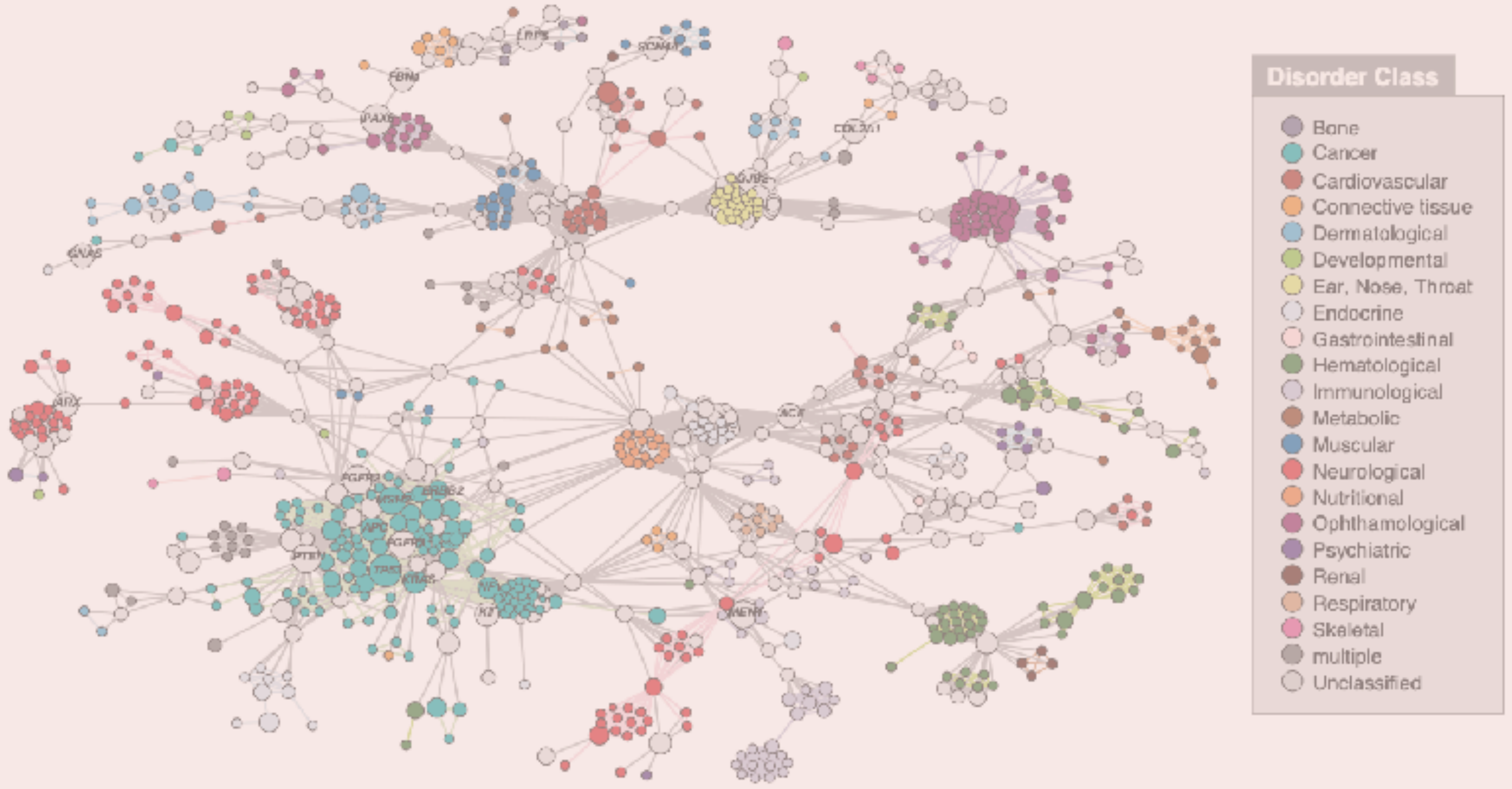
[2] https://predictivehacks.com/social-network-analysis-of-game-of-thrones/

[3] Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E., & Welling, M. (2021). Geometric and Physical Quantities improve E (3) Equivariant Message Passing. In ICLR 2022

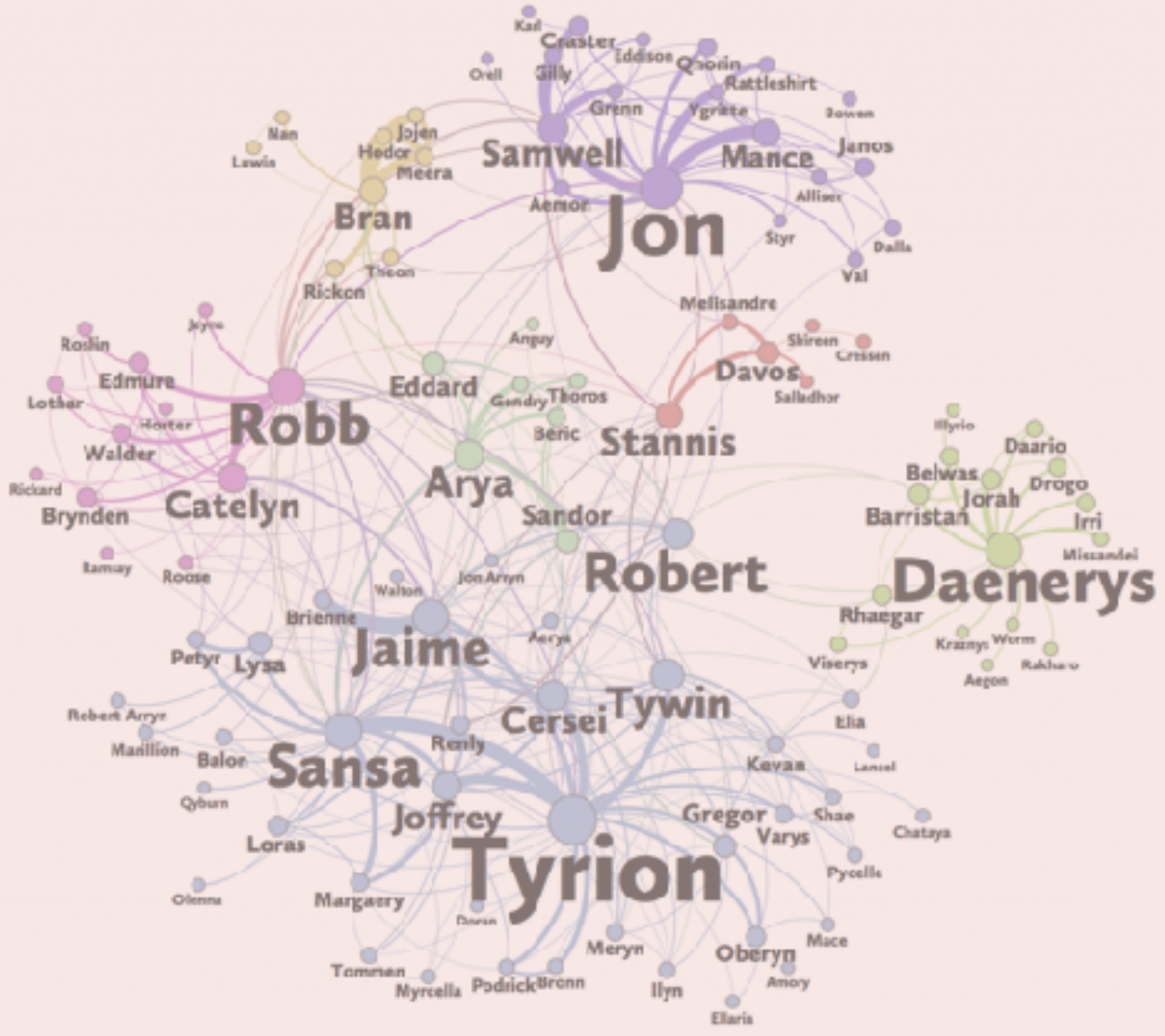
[4] Atz, K., Grisoni, F., & Schneider, G. (2021). Geometric deep learning on molecular representations. Nature Machine Intelligence, 1-10.

[5] Chatzipantazis, E., Pertigkiozoglou, S., Dobriban, E., & Daniilidis, K. (2022). SE (3)-Equivariant Attention Networks for Shape Reconstruction in Function Space. arXiv preprint arXiv:2204.02394.

General graphs



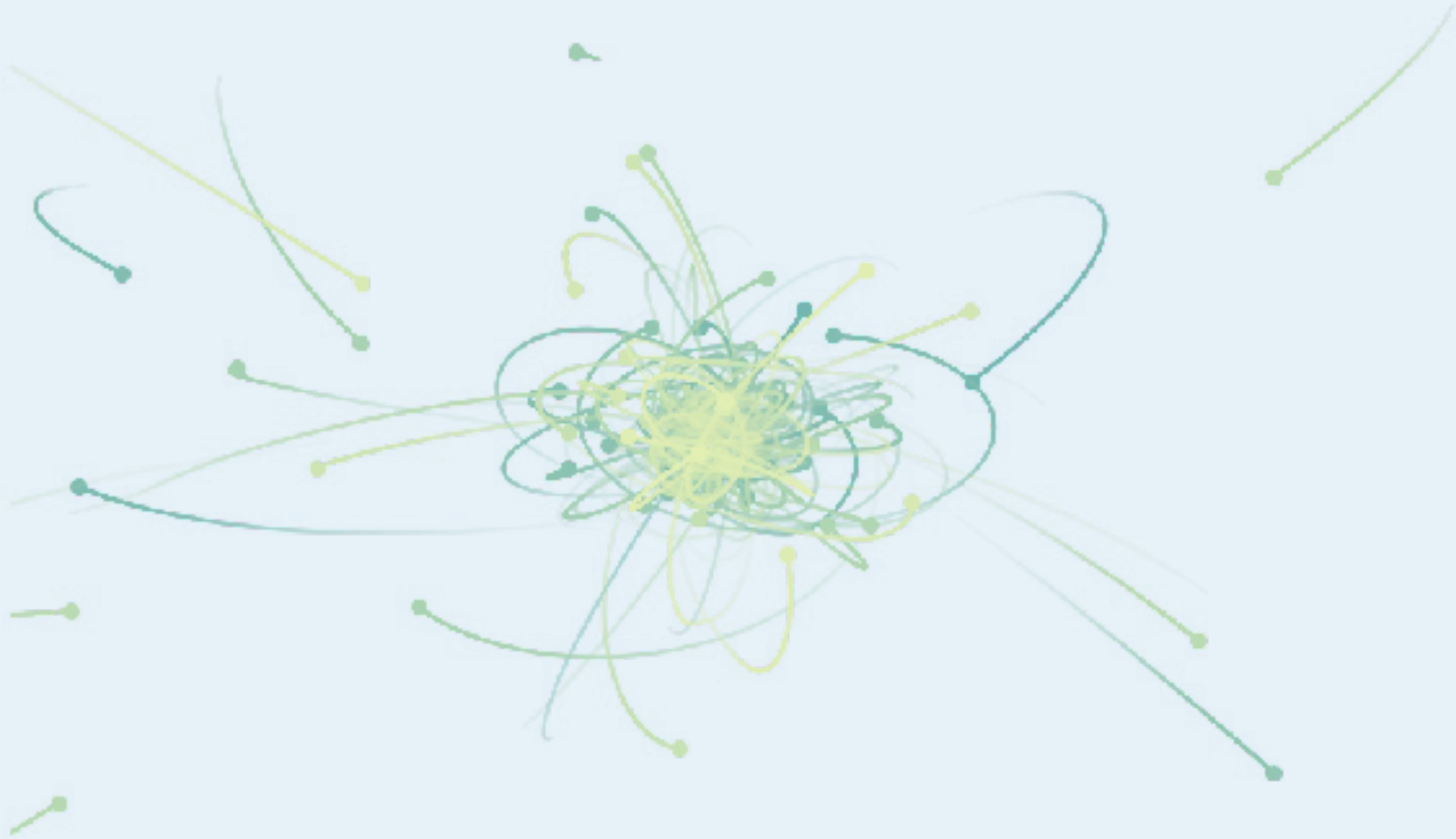
Gene/Protein interaction graphs[1]



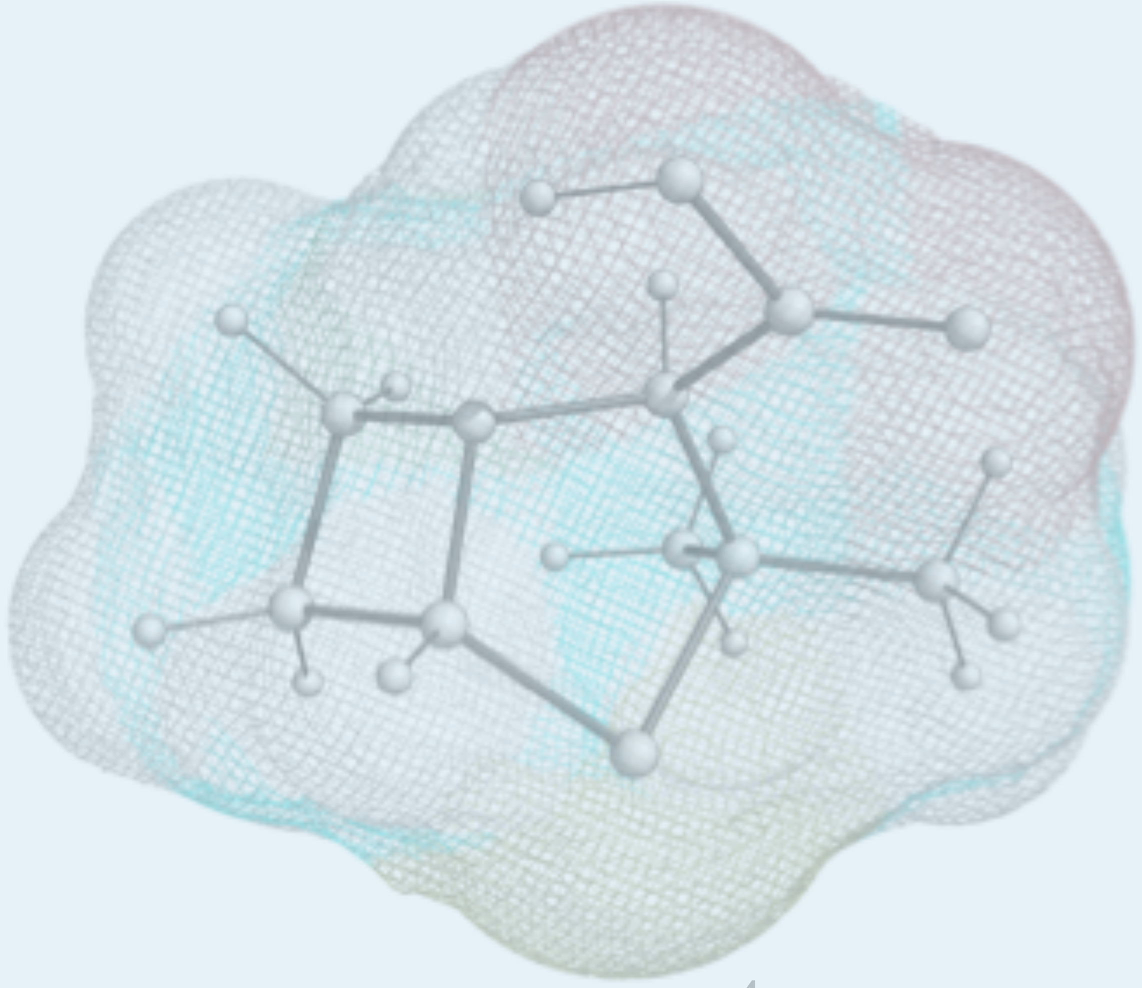
Social network graph[2]

Geometric graphs

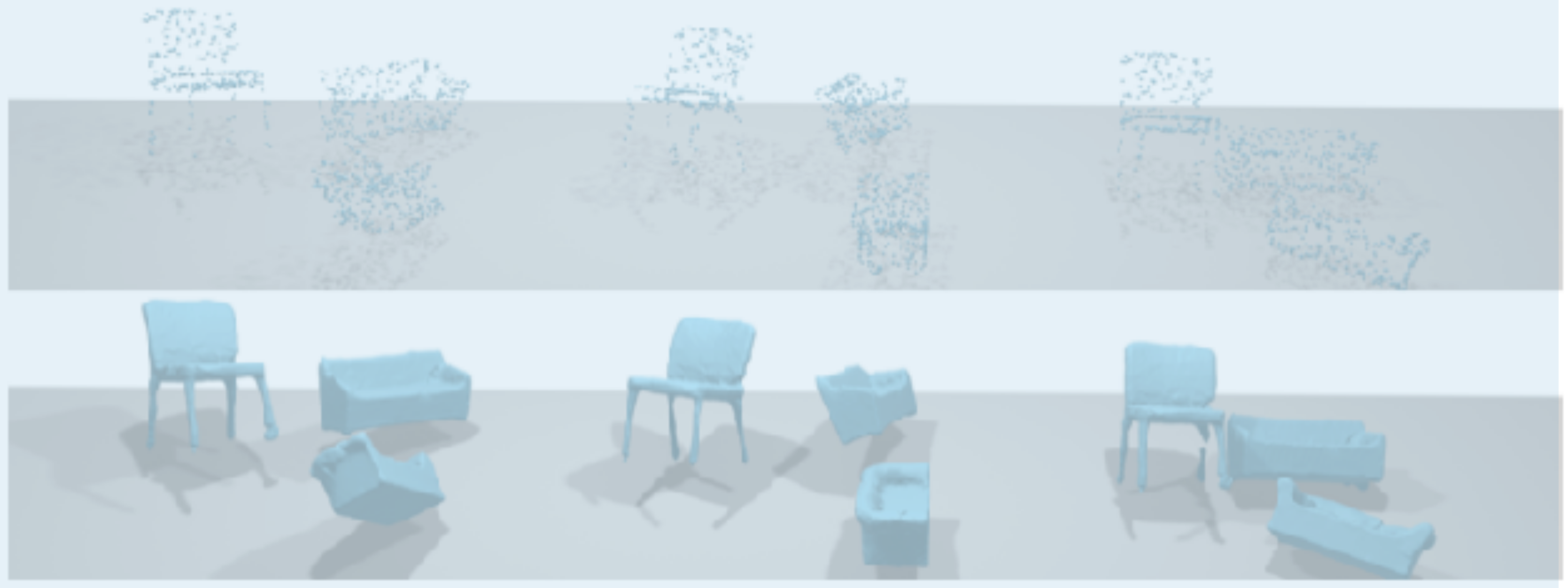
(nodes correspond to points in a manifold)



Physical system[3]



Molecule[4]



Point cloud/shapes[5]

Figures from: [1]Goh, K. I., Cusick, M. E., Valle, D., Childs, B., Vidal, M., & Barabási, A. L. (2007). The human disease network. Proceedings of the National Academy of Sciences, 104(21), 8685-8690.

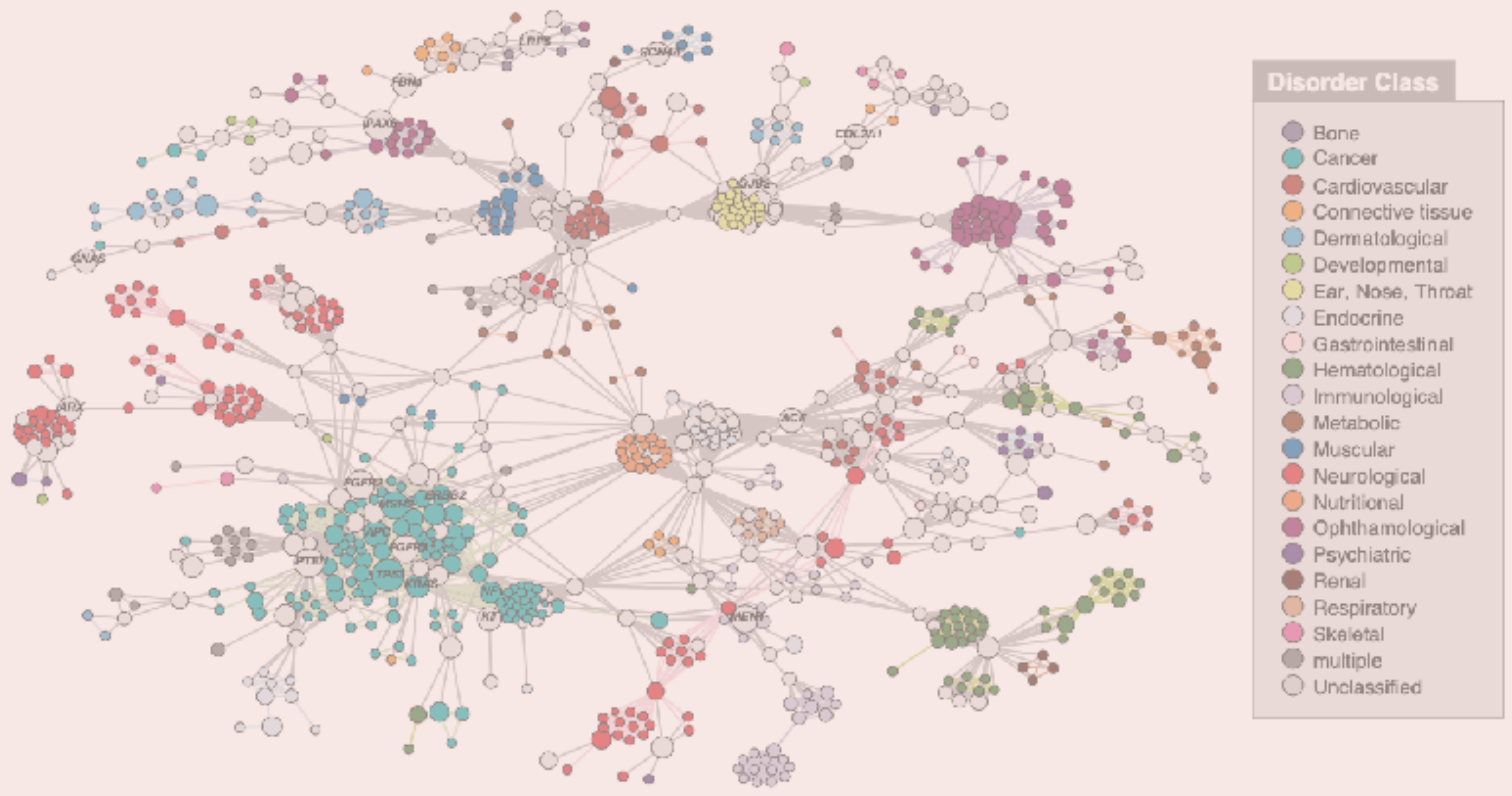
[2]https://predictivehacks.com/social-network-analysis-of-game-of-thrones/

[3]Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E., & Welling, M. (2021). Geometric and Physical Quantities improve E (3) Equivariant Message Passing. In ICLR 2022

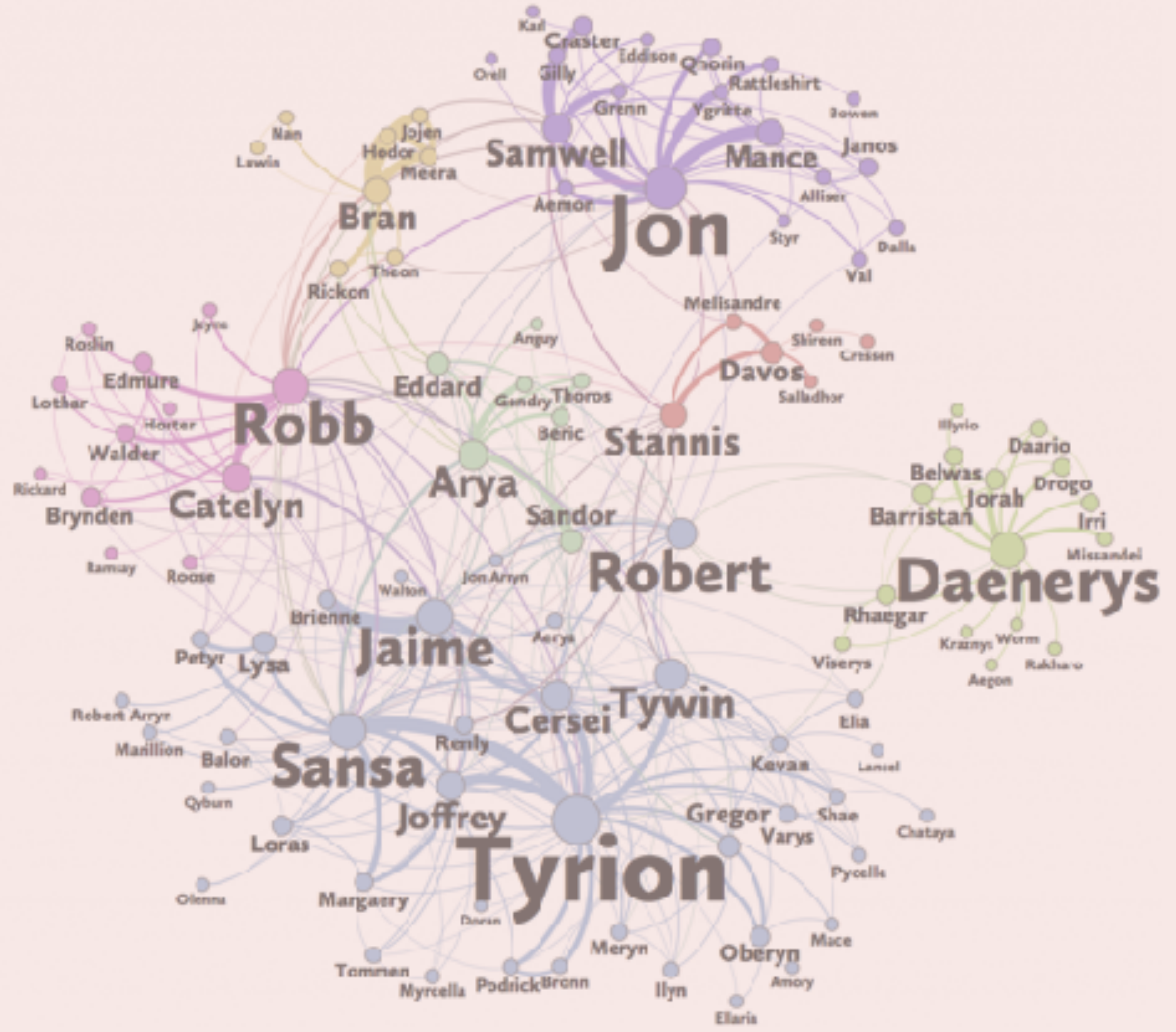
[4]Atz, K., Grisoni, F., & Schneider, G. (2021). Geometric deep learning on molecular representations. Nature Machine Intelligence, 1-10.

[5]Chatzipantazis, E., Pertigkiozoglou, S., Dobriban, E., & Daniilidis, K. (2022). SE (3)-Equivariant Attention Networks for Shape Reconstruction in Function Space. arXiv preprint arXiv:2204.02394.

General graphs

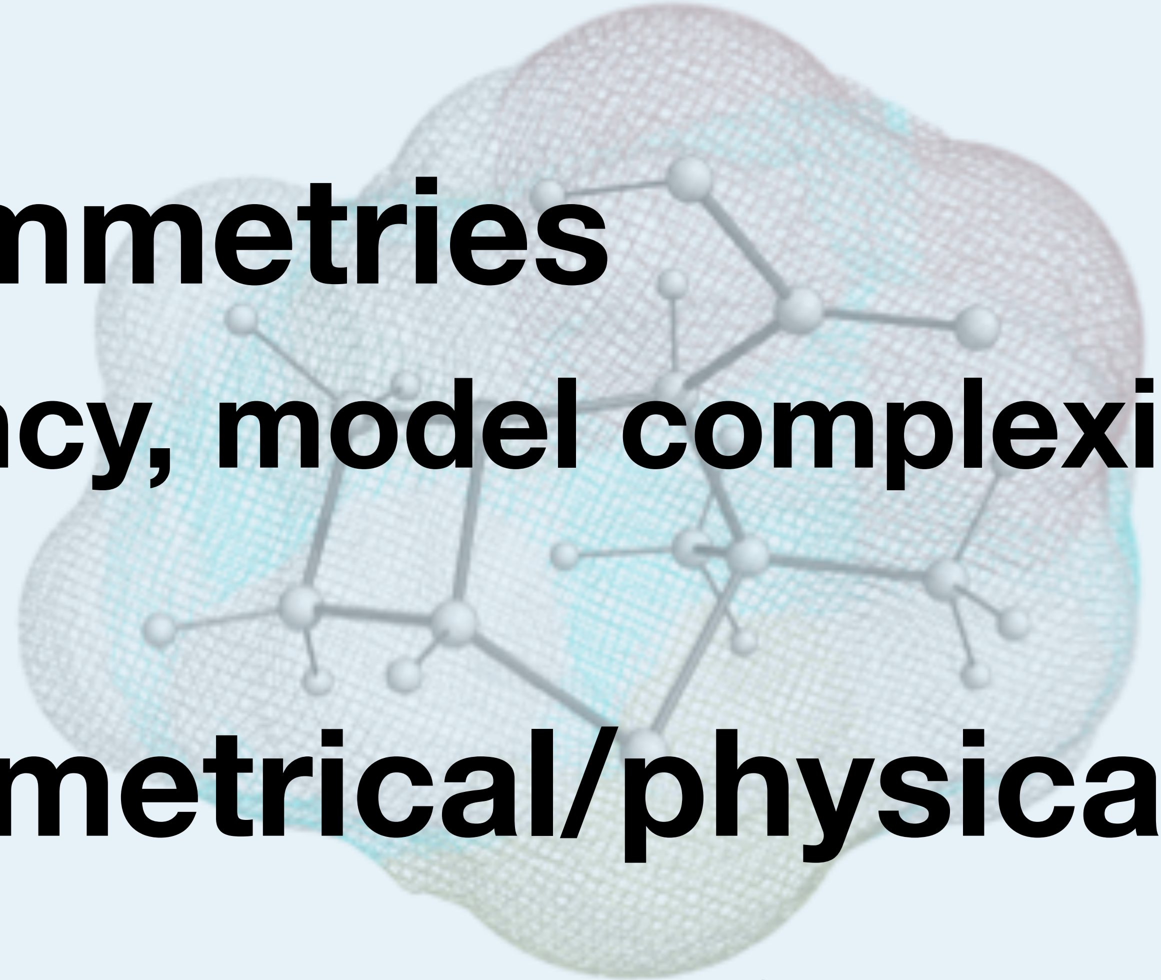


Gene/Protein interaction graphs[1]



Social network graph[2]

Geometric graphs
(nodes correspond to points in a manifold)



1. Leverage symmetries

(sample efficiency, model complexity, generalizability)

2. Respect geometrical/physical constraints

Physical system[3]

Molecule[4]

Point cloud/shapes[5]

Figures from: [1] Goh, K. I., Cusick, M. E., Valle, D., Childs, B., Vidal, M., & Barabási, A. L. (2007). The human disease network. Proceedings of the National Academy of Sciences, 104(21), 8685-8690.

[2] https://predictivehacks.com/social-network-analysis-of-game-of-thrones/

[3] Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E., & Welling, M. (2021). Geometric and Physical Quantities improve E (3) Equivariant Message Passing. In ICLR 2022

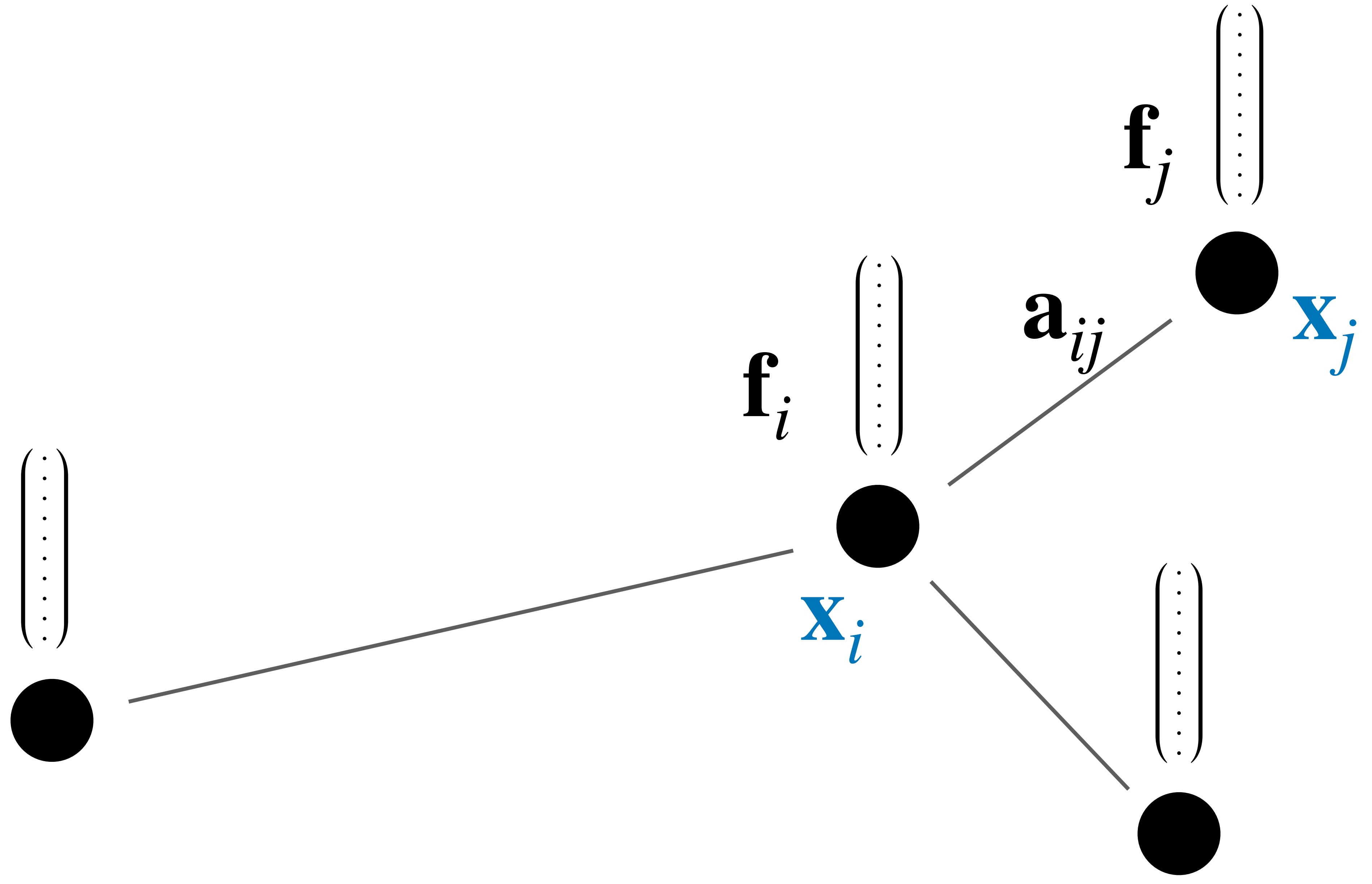
[4] Atz, K., Grisoni, F., & Schneider, G. (2021). Geometric deep learning on molecular representations. Nature Machine Intelligence, 1-10.

[5] Chatzipantazis, E., Pertigkiozoglou, S., Dobriban, E., & Daniilidis, K. (2022). SE (3)-Equivariant Attention Networks for Shape Reconstruction in Function Space. arXiv preprint arXiv:2204.02394.

The *Geometric* Message Passing Framework

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

- Messages

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

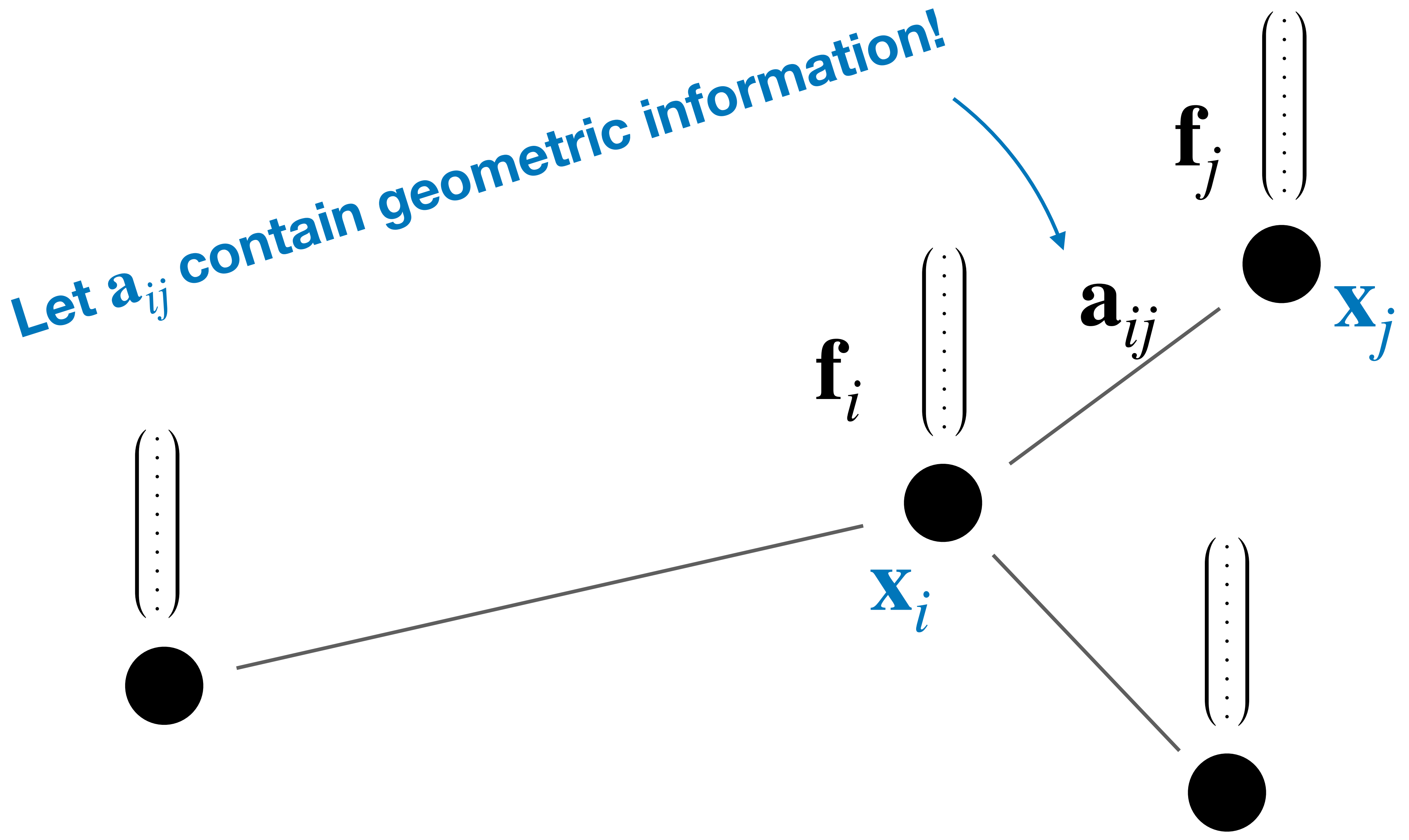
- Aggregate + node updates

$$\mathbf{f}_i' = \phi_f(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij})$$

The *Geometric* Message Passing Framework

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Let $\mathbf{a}_{ij}$ contain geometric information!

Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

- Messages

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

"Condition" messages on geometry

- Aggregate + node updates

$$\mathbf{f}_i' = \phi_f(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij})$$

The *Geometric* Message Passing Framework

Graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$

- nodes $v_i \in \mathscr{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

$\mathbf{f}_j$
$\mathbf{a}_{ij}$
$\mathbf{x}_j$
$\mathbf{f}_i$
$\mathbf{x}_i$

Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

- Messages

$$(X = \mathbb{R}^d) \quad \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$$

The *Geometric* Message Passing Framework

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

$\mathbf{f}_j$
$\mathbf{a}_{ij}$
$\mathbf{x}_j$
$\mathbf{f}_i$
$\mathbf{x}_i$

Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

- Messages

$$(X = \mathbb{R}^d) \quad \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$$

Only equivariant to translations…

The *Geometric* Message Passing Framework

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

$\mathbf{f}_j$
$\mathbf{a}_{ij}$
$\mathbf{x}_j$
$\mathbf{f}_i$
$\mathbf{x}_i$

Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

- Messages

$$(X = \mathbb{R}^d) \quad \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$$

Only equivariant to translations…

$$(X = \mathbb{R}^d) \quad \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \|\mathbf{x}_j - \mathbf{x}_i\|)$$

The *Geometric* Message Passing Framework

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

$\mathbf{f}_j$
$\mathbf{a}_{ij}$
$\mathbf{x}_j$
$\mathbf{f}_i$
$\mathbf{x}_i$

Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

- Messages

$(X = \mathbb{R}^d)$ $$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$$

Only equivariant to translations…

$(X = \mathbb{R}^d)$ $$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \|\mathbf{x}_j - \mathbf{x}_i\|)$$

Full $E(3)$ equivariance, but a bit restrictive…

The *Geometric* Message Passing Framework

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

$\mathbf{f}_j$
$\mathbf{a}_{ij}$
$\mathbf{x}_j$
$\mathbf{f}_i$
$\mathbf{x}_i$

Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

- Messages

$(X = \mathbb{R}^d)$ $$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$$

Only equivariant to translations...

$(X = \mathbb{R}^d)$ $$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \|\mathbf{x}_j - \mathbf{x}_i\|)$$

Full $E(3)$ equivariance, but a bit restrictive...

$(X = G)$ $$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, g_j^{-1} g_i)$$

Solution 1: Lift to the group!

The *Geometric* Message Passing Framework

Graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$

- nodes $v_i \in \mathscr{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

$\mathbf{f}_j$
$\mathbf{a}_{ij}$
$\mathbf{x}_j$
$\mathbf{f}_i$
$\mathbf{x}_i$

Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

- Messages

$(X = \mathbb{R}^d)$ $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$

Only equivariant to translations…

$(X = \mathbb{R}^d)$ $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \|\mathbf{x}_j - \mathbf{x}_i\|)$

Full $E(3)$ equivariance, but a bit restrictive…

$(X = G)$ $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, g_j^{-1} g_i)$

Solution 1: Lift to the group!

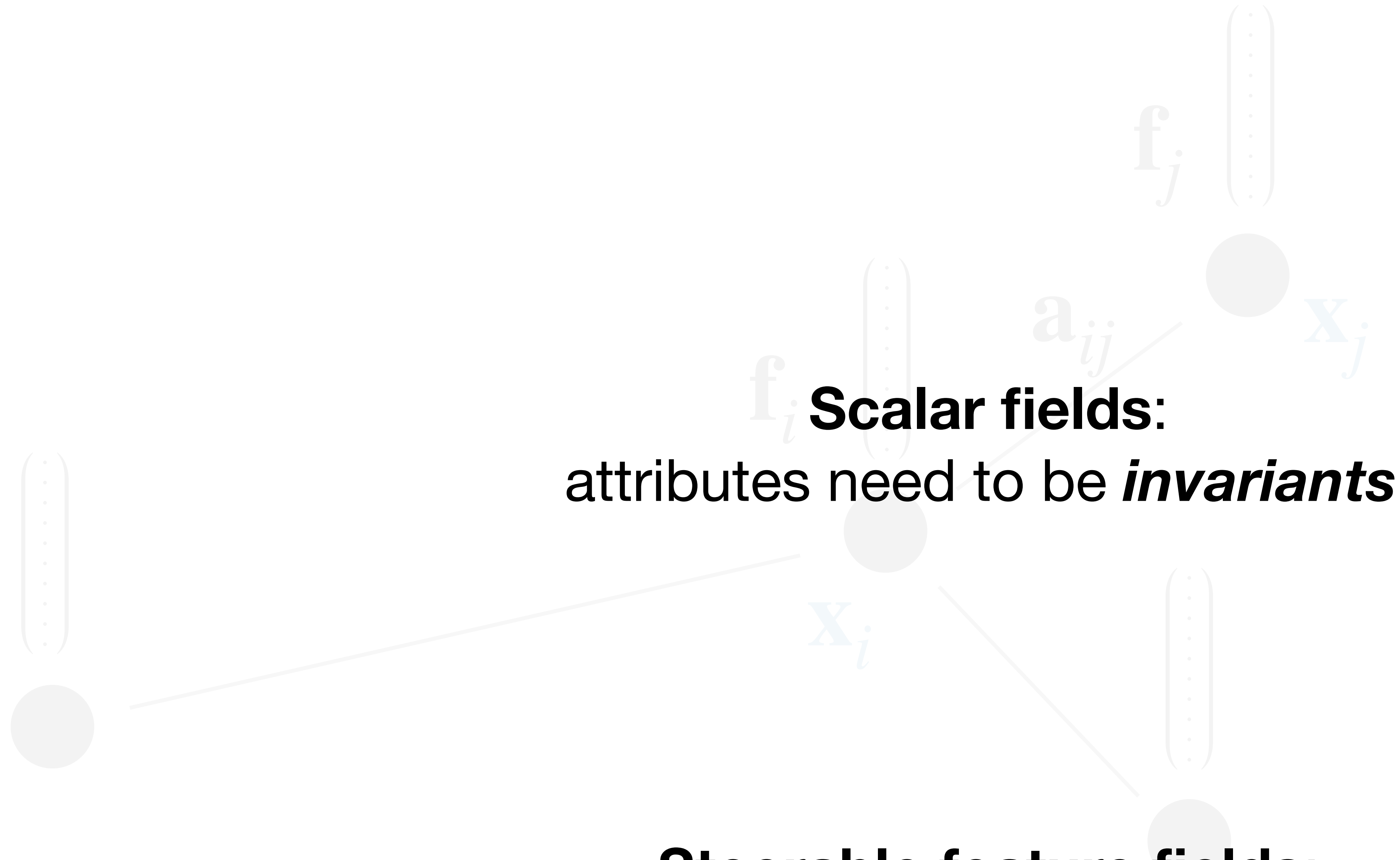
$(X = \mathbb{R}^d)$ $\hat{\mathbf{m}}_{ij} = \hat{\phi}_m(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, Y(\mathbf{x}_j - \mathbf{x}_i))$

Solution 2: work with steerable feature fields!

The *Geometric* Message Passing Framework

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Scalar fields:
attributes need to be ***invariants*!**

Steerable feature fields:
attributes can be ***covariants*!**

Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

- Messages

$$(X = \mathbb{R}^d) \quad \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$$

Only equivariant to translations…

$$(X = \mathbb{R}^d) \quad \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \|\mathbf{x}_j - \mathbf{x}_i\|)$$

Full $E(3)$ equivariance, but a bit restrictive…

$$(X = G) \quad \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, g_j^{-1} g_i)$$

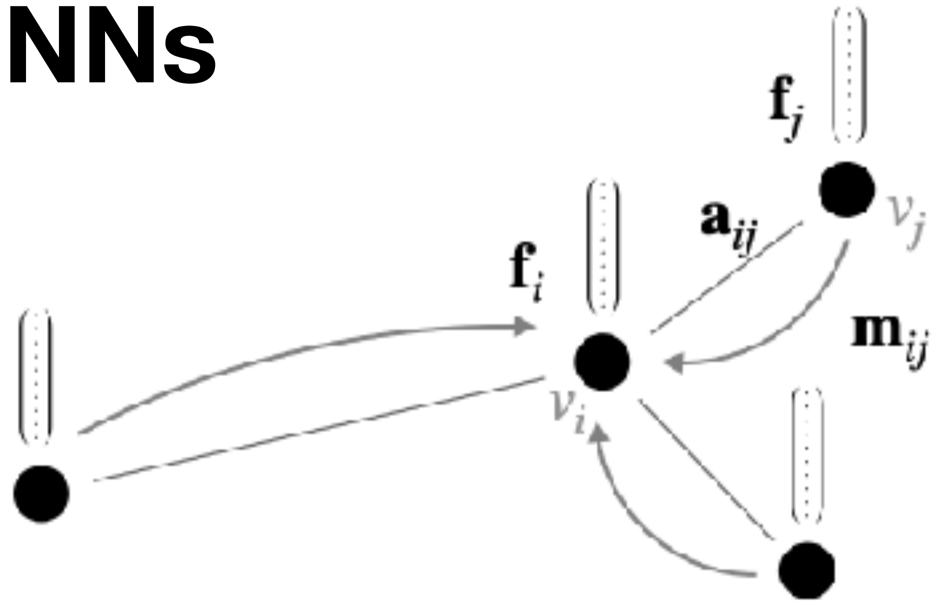
Solution 1: Lift to the group!

$$(X = \mathbb{R}^d) \quad \hat{\mathbf{m}}_{ij} = \hat{\phi}_m(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, Y(\mathbf{x}_j - \mathbf{x}_i))$$

Solution 2: work with steerable feature fields!

Linear vs non-linear (group) convolutions

Message passing NNs

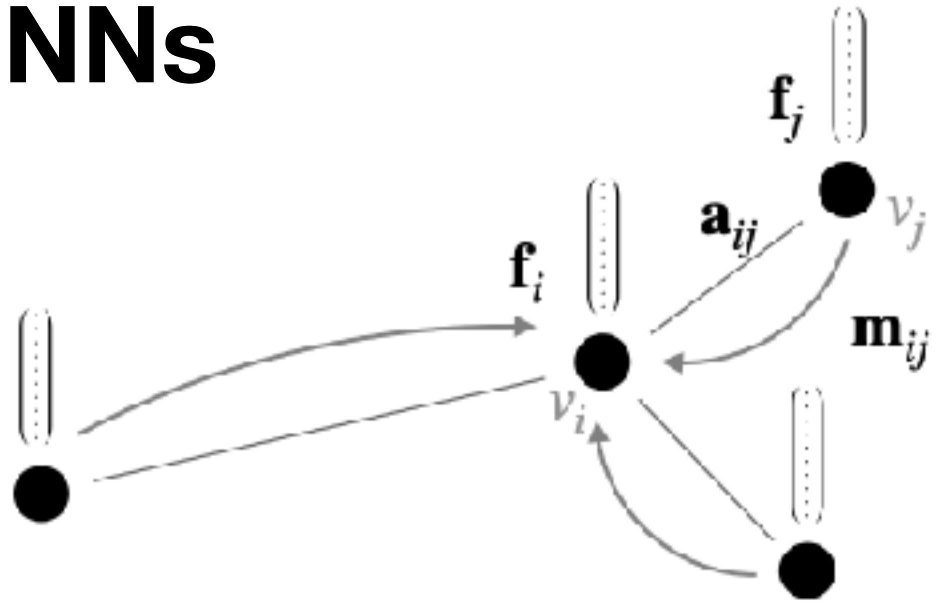


Compute messages: $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$

Aggregate and update: $\mathbf{f}_i' = \phi_f\left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}\right)$

Linear vs non-linear (group) convolutions

Message passing NNs



Compute messages: $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$

Aggregate and update: $\mathbf{f}_i' = \phi_f\left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}\right)$

Classic point convolutions

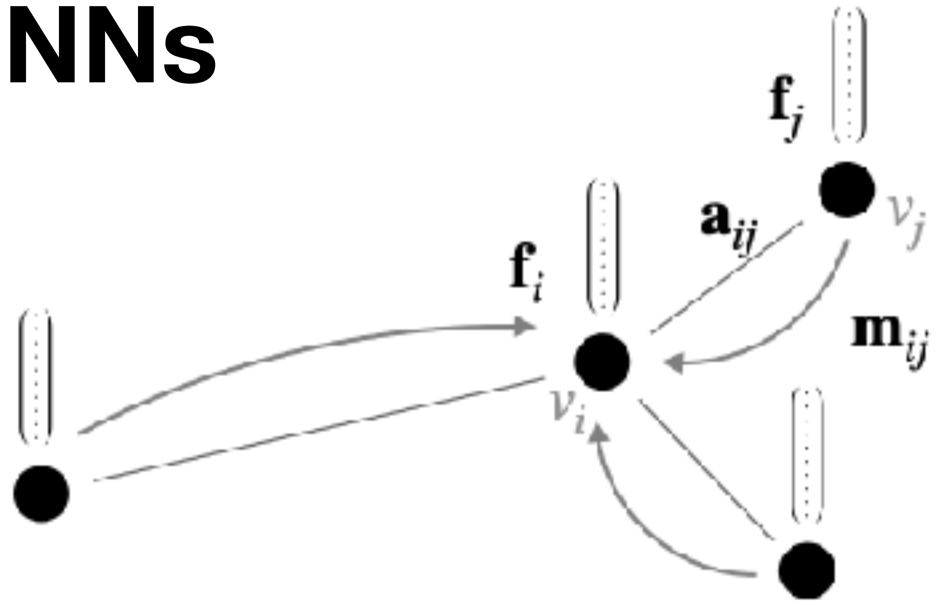
(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$

$$\mathbf{m}_{ij} = \mathbf{W}(g_i^{-1} g_j)\mathbf{f}_j$$

Linear vs non-linear (group) convolutions

Message passing NNs



Compute messages: $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$

Aggregate and update: $\mathbf{f}_i' = \phi_f\left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}\right)$

Classic point convolutions

(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$

$$\mathbf{m}_{ij} = \mathbf{W}(g_i^{-1} g_j)\mathbf{f}_j$$

Steerable G-CNNs

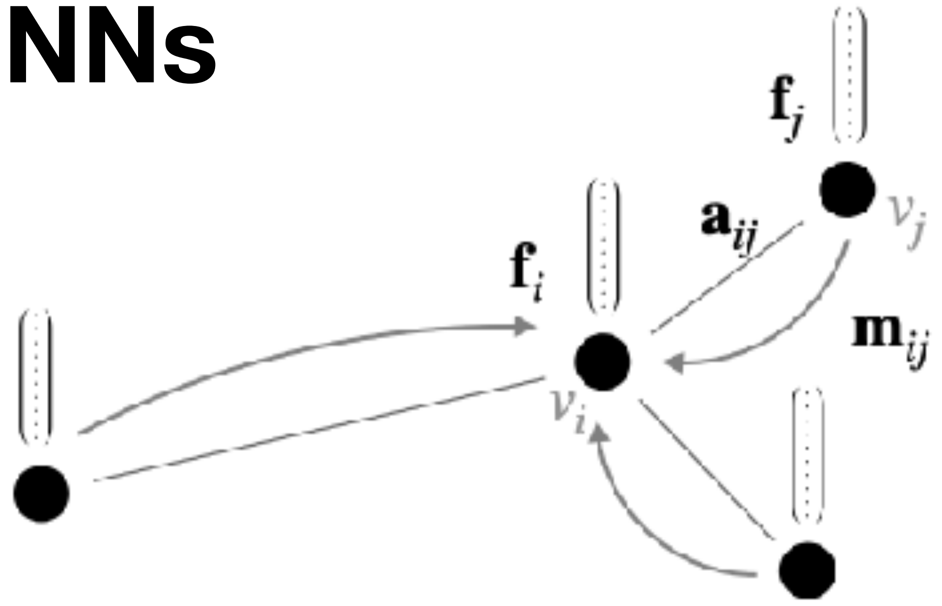
(Lecture 2: steerable g-convs)

$$\mathbf{m}_{ij} = \mathbf{W}_{\hat{\mathbf{a}}_{ij}}(\|\mathbf{x}_j - \mathbf{x}_i\|)\hat{\mathbf{f}}_j$$

$$:= \hat{\mathbf{f}}_j \otimes_{cg}^{\mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)} \hat{\mathbf{a}}_{ij}$$

Linear vs non-linear (group) convolutions

Message passing NNs



Compute messages: $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$

Aggregate and update: $\mathbf{f}_i' = \phi_f\left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}\right)$

Classic point convolutions

(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$

$$\mathbf{m}_{ij} = \mathbf{W}(g_i^{-1} g_j)\mathbf{f}_j$$

$\mathscr{F}_H$

Steerable G-CNNs

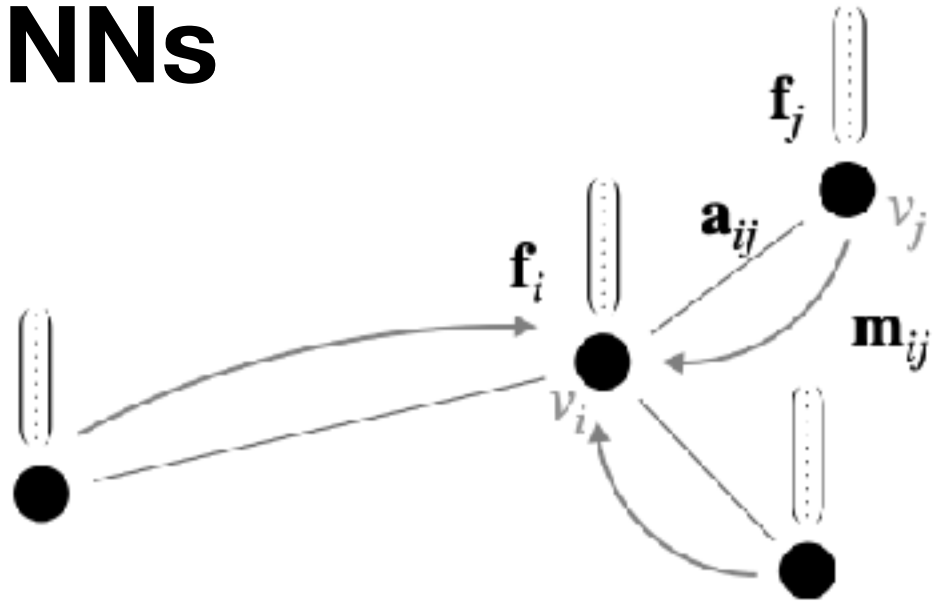
(Lecture 2: steerable g-convs)

$$\mathbf{m}_{ij} = \mathbf{W}_{\hat{\mathbf{a}}_{ij}}(\|\mathbf{x}_j - \mathbf{x}_i\|)\hat{\mathbf{f}}_j$$

$$:= \hat{\mathbf{f}}_j \otimes_{cg}^{\mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)} \hat{\mathbf{a}}_{ij}$$

Linear vs non-linear (group) convolutions

Message passing NNs



Compute messages: $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$

Aggregate and update: $\mathbf{f}'_i = \phi_f\left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}\right)$

Classic point convolutions

(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$

$$\mathbf{m}_{ij} = \mathbf{W}(g_i^{-1} g_j)\mathbf{f}_j$$

$\mathscr{F}_H$

Steerable G-CNNs

(Lecture 2: steerable g-convs)

$$\mathbf{m}_{ij} = \mathbf{W}_{\hat{\mathbf{a}}_{ij}}(\|\mathbf{x}_j - \mathbf{x}_i\|)\hat{\mathbf{f}}_j$$

$$:= \hat{\mathbf{f}}_j \otimes_{cg}^{\mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)} \hat{\mathbf{a}}_{ij}$$

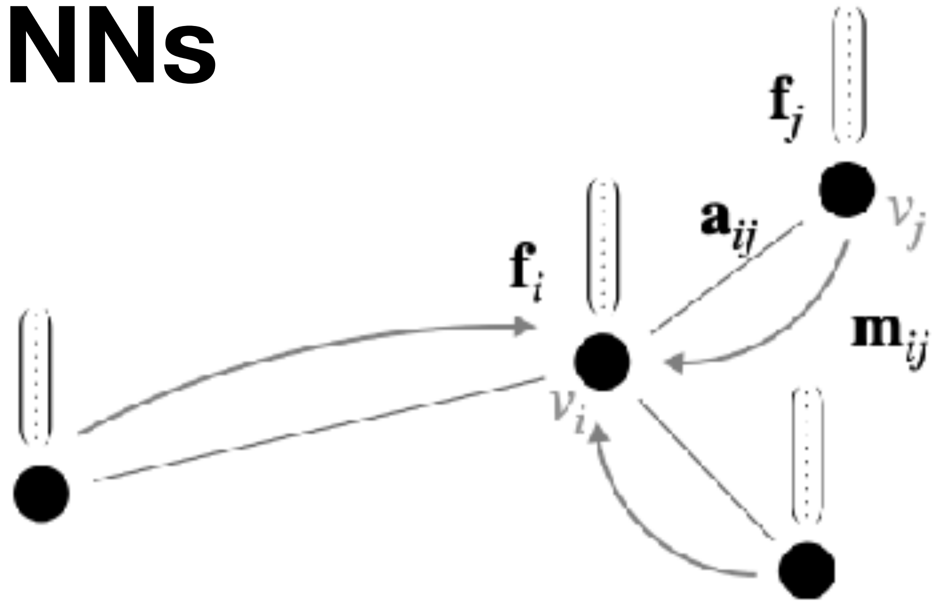
Invariant Message Passing NNs

(Lecture 3)

$$\mathbf{m}_{ij} = \mathrm{MLP}(\mathbf{f}_i, \mathbf{f}_j, \|\mathbf{x}_j - \mathbf{x}_i\|)$$

Linear vs non-linear (group) convolutions

Message passing NNs



Compute messages: $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$

Aggregate and update: $\mathbf{f}'_i = \phi_f\left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}\right)$

Classic point convolutions

(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$

$$\mathbf{m}_{ij} = \mathbf{W}(g_i^{-1} g_j)\mathbf{f}_j$$

$\mathcal{F}_H$

Steerable G-CNNs

(Lecture 2: steerable g-convs)

$$\mathbf{m}_{ij} = \mathbf{W}_{\hat{\mathbf{a}}_{ij}}(\|\mathbf{x}_j - \mathbf{x}_i\|)\hat{\mathbf{f}}_j$$

$$:= \hat{\mathbf{f}}_j \otimes_{cg}^{\mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)} \hat{\mathbf{a}}_{ij}$$

Invariant Message Passing NNs

(Lecture 3)

$$\mathbf{m}_{ij} = \text{MLP}(\mathbf{f}_i, \mathbf{f}_j, \|\mathbf{x}_j - \mathbf{x}_i\|)$$

Equivariant (Steerable) Message Passing NNs

(Lecture 3)

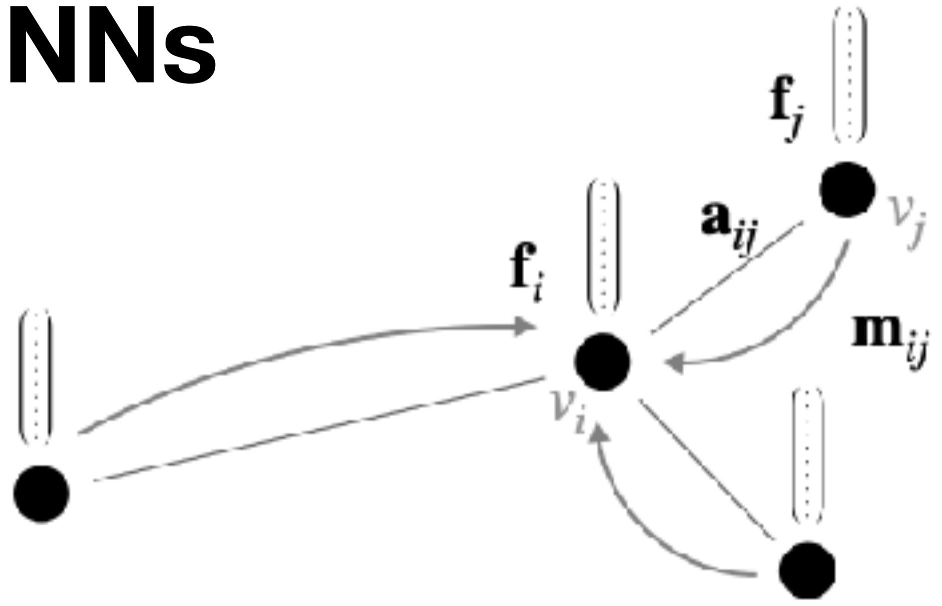
$$\hat{\mathbf{m}}_{ij} = \widehat{\text{MLP}}(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \mathbf{x}_j - \mathbf{x}_i)$$

With steerable MLP:

$$\widehat{\text{MLP}}_{\hat{\mathbf{a}}_{ij}}(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \|\mathbf{x}_j - \mathbf{x}_i\|) := \sigma(\mathbf{W}^{(n)}_{\hat{\mathbf{a}}_{ij}}(\ldots(\sigma(\mathbf{W}^{(1)}_{\hat{\mathbf{a}}_{ij}}\hat{\mathbf{h}}_i))))$$

Linear vs non-linear (group) convolutions

Message passing NNs



Compute messages: $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$

Aggregate and update: $\mathbf{f}'_i = \phi_f\left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}\right)$

Classic point convolutions

(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$

$$\mathbf{W}(g_i^{-1} g_j)\mathbf{f}_j$$

Linear convolution

$\mathscr{F}_H$

Steerable G-CNNs

(Lecture 2: steerable g-convs)

$$\mathbf{m}_{ij} = \mathbf{W}_{\hat{\mathbf{a}}_{ij}}(\|\mathbf{x}_j - \mathbf{x}_i\|)\hat{\mathbf{f}}_j$$

$$:= \hat{\mathbf{f}}_j \otimes_{cg}^{\mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)} \hat{\mathbf{a}}_{ij}$$

Invariant Message Passing NNs

(Lecture 3)

$$\mathbf{m}_{ij} = \mathrm{MLP}(\mathbf{f}_i, \mathbf{f}_j, \|\mathbf{x}_j - \mathbf{x}_i\|)$$

Equivariant (Steerable) Message Passing NNs

(Lecture 3)

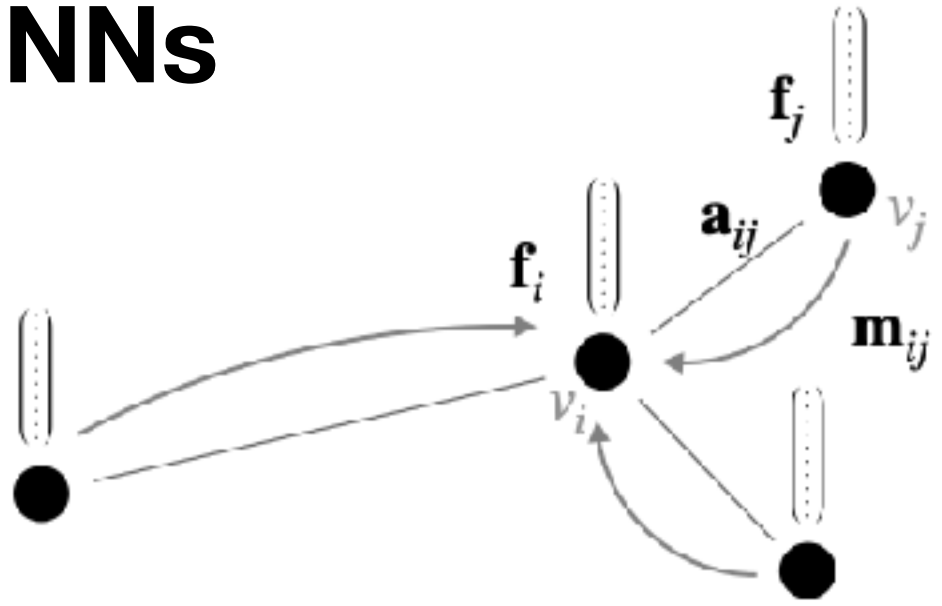
$$\hat{\mathbf{m}}_{ij} = \widehat{\mathrm{MLP}}(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \mathbf{x}_j - \mathbf{x}_i)$$

With steerable MLP:

$$\widehat{\mathrm{MLP}}_{\hat{\mathbf{a}}_{ij}}(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \|\mathbf{x}_j - \mathbf{x}_i\|) := \sigma(\mathbf{W}^{(n)}_{\hat{\mathbf{a}}_{ij}}(\ldots(\sigma(\mathbf{W}^{(1)}_{\hat{\mathbf{a}}_{ij}}\hat{\mathbf{h}}_i))))$$

Linear vs non-linear (group) convolutions

Message passing NNs



Compute messages: $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$

Aggregate and update: $\mathbf{f}_i' = \phi_f\left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}\right)$

Classic point convolutions

(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$

$$\mathbf{W}(g_i^{-1} g_j)\mathbf{f}_j$$

Linear convolution

$\mathcal{F}_H$

Steerable G-CNNs

(Lecture 2: steerable g-convs)

$$\mathbf{m}_{ij} = \mathbf{W}_{\hat{\mathbf{a}}_{ij}}(\|\mathbf{x}_j - \mathbf{x}_i\|)\hat{\mathbf{f}}_j$$

$$:= \hat{\mathbf{f}}_j \otimes_{cg}^{\mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)} \hat{\mathbf{a}}_{ij}$$

Invariant Message Passing NNs

(Lecture 3)

$$\mathbf{m}_{ij} = \text{MLP}(\mathbf{f}_i, \mathbf{f}_j, \|\mathbf{x}_j - \mathbf{x}_i\|)$$

Non-linear “convolution”

Equivariant (Steerable) Message Passing NNs

(Lecture 3)

$$\hat{\mathbf{m}}_{ij} = \widehat{\text{MLP}}(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \mathbf{x}_j - \mathbf{x}_i)$$

With steerable MLP:

$$\widehat{\text{MLP}}_{\hat{\mathbf{a}}_{ij}}(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \|\mathbf{x}_j - \mathbf{x}_i\|) := \sigma(\mathbf{W}^{(n)}_{\hat{\mathbf{a}}_{ij}}(\ldots(\sigma(\mathbf{W}^{(1)}_{\hat{\mathbf{a}}_{ij}}\hat{\mathbf{h}}_i))))$$

Geometric and Physical Quantities improve E(3) Equivariant Message Passing

Anonymous authors
Paper under double-blind review

Abstract

Including covariant information, such as position, force, velocity or spin is important in many tasks in computational physics and chemistry. We introduce Steerable E(3) Equivariant Graph Neural Networks (SEGNNs) that generalise equivariant graph networks, such that node and edge attributes are not restricted to invariant scalars, but can contain covariant information, such as vectors or tensors. This model, composed of steerable MLPs, is able to incorporate geometric and physical information in both the message and update functions. Through the definition of steerable node attributes, the MLPs provide a new class of activation functions for general use with steerable feature fields. We discuss ours and related work through the lens of *equivariant non-linear convolutions*, which further allows us to pin-point the successful components of SEGNNs: *non-linear* message aggregation improves upon classic *linear* (steerable) point convolutions; *steerable messages* improve upon recent equivariant graph networks that send invariant messages. We demonstrate the effectiveness of our method on several tasks in computational physics and chemistry and provide extensive ablation studies.

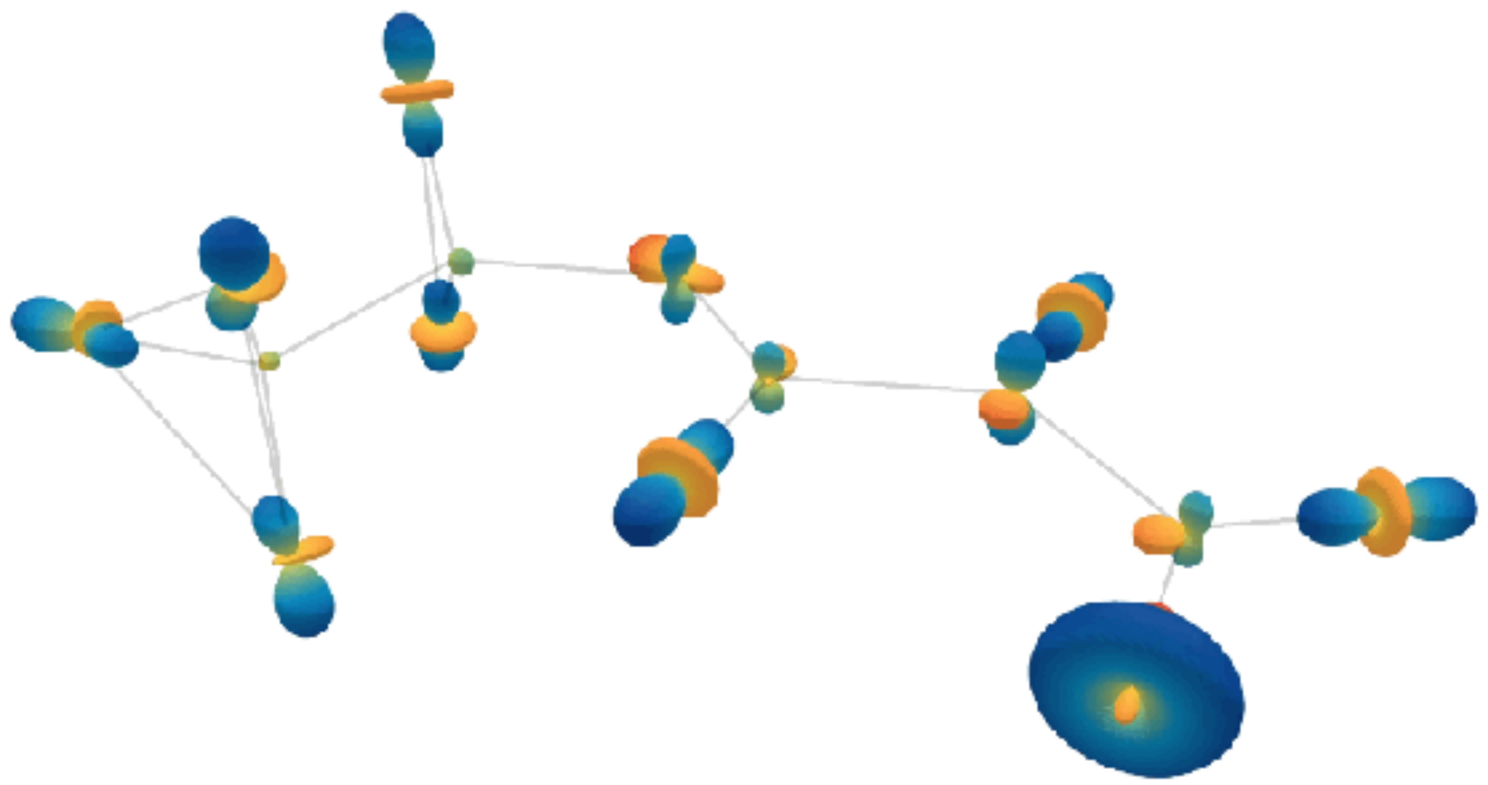
1 Introduction

The success of Convolutional Neural Networks (CNNs) (LeCun et al., 1998; 2015; Schmidhuber, 2015; Krizhevsky et al., 2012) is a key factor for the rise of deep learning, attributed to their capability of exploiting translation symmetries, hereby introducing a strong inductive bias. Recent work has shown that designing CNNs to exploit additional symmetries via group convolutions has even further increased their performance (Cohen & Welling, 2016; 2017; Worrall et al., 2017; Cohen et al., 2018; Kondor & Trivedi, 2018; Weiler et al., 2018; Bekkers et al., 2018; Bekkers, 2019; Weiler & Cesa, 2019). Graph neural networks (GNNs) and CNNs are closely related to each other via their aggregation of local information. More precisely, CNNs can be formulated as message passing layers (Gilmer et al., 2017) based on a sum aggregation of messages that are obtained by relative position-dependent *linear* transformations of neighbouring node features. The power of message passing layers is, however, that node features are transformed and propagated in a highly *non-linear* manner. Equivariant GNNs have been proposed before as either PointConv-type (Wu et al., 2019; Kristof et al., 2017) implementations of steerable (Thomas et al., 2018; Anderson et al., 2019; Fuchs et al., 2020) or regular group convolutions (Finzi et al., 2020). The most important component in these methods are the convolution layers. Although powerful, such layers only (pseudo[1]) linearly transform the graphs and non-linearity is only obtained via point-wise activations.

In this paper, we propose non-linear E(3) *equivariant message passing* layers using the same principles that underlie steerable group convolutions, and view them as *non-linear group convolutions*. Central to our method is the use of steerable vectors and their equivariant transformations to represent and process node features; we present the underlying mathematics of both in Sec. 2 and illustrate it in Fig. 1 on a molecular graph. As a consequence, information at nodes and edges can now be rotationally invariant (scalar) or covariant (vector, tensor). In steerable message passing frameworks, the Clebsch-Gordan (CG) tensor product is used to steer the update and message functions by geometric information such as relative orientation (pose). Through a notion of steerable node attributes we provide a new class of equivariant activation functions for general use with steerable

[1] Methods such as SE(3)-transformers (Fuchs et al., 2020) and Cormorant (Anderson et al., 2019) include an input-dependent attention component that augments the convolutions.

Task: Molecular property prediction



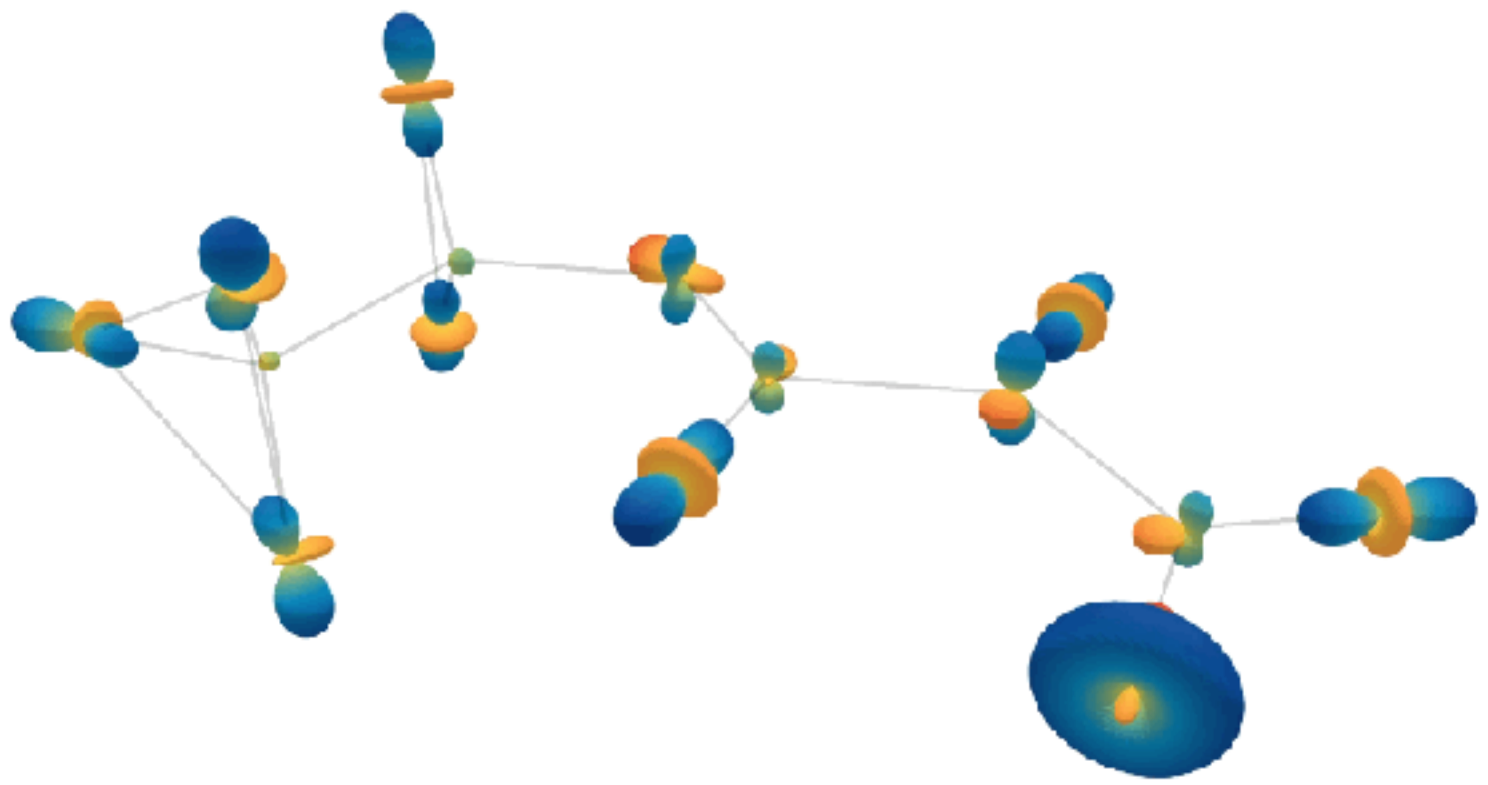
→ Property

	Task		α	$\Delta\varepsilon$	$\varepsilon_{\mathrm{HOMO}}$	$\varepsilon_{\mathrm{LUMO}}$	μ	C_ν	
	Units	Cutoff radius	bohr^3	meV	meV	meV	D	cal/mol K	Time [s]
Isotropic (fully connected graph)	(S)EGNN ($l_f = 0, l_a = 0$)	-	.091	53	34	28	.042	.043	0.016
Isotropic (local)	(S)EGNN ($l_f = 0, l_a = 0$)	2Å	.24	98	60	60	.34	.077	0.014
Anisotropic (local)	SEGNN ($l_f = 1, l_a = 2$)	2Å	.074	48	27	25	.031	.035	0.048
	SEGNN ($l_f = 2, l_a = 3$)	2Å	.060	42	24	21	.023	.031	0.097

(Steerable) G-CNNs allow for local connectivity (Scales to large proteins!!!)

isotropic convs require full connectivity in order to infer the geometry

Task: Molecular property prediction



Property

	Task		α	$\Delta\varepsilon$	$\varepsilon_{\text{HOMO}}$	$\varepsilon_{\text{LUMO}}$	μ	C_ν	
	Units	Cutoff radius	bohr^3	meV	meV	meV	D	cal/mol K	Time [s]
Isotropic (fully connected graph)	(S)EGNN ($l_f = 0, l_a = 0$)	-	.091	53	34	28	.042	.043	0.016
Isotropic (local)	(S)EGNN ($l_f = 0, l_a = 0$)	2Å	.24	98	60	60	.34	.077	0.014
Anisotropic (local)	SEGNN ($l_f = 1, l_a = 2$)	2Å	.074	48	27	25	.031	.035	0.048
	SEGNN ($l_f = 2, l_a = 3$)	2Å	.060	42	24	21	.023	.031	0.097

(Steerable) G-CNNs allow for local connectivity (Scales to large proteins!!!)

isotropic convs require full connectivity in order to infer the geometry

Task: Trajectory prediction N-body problem

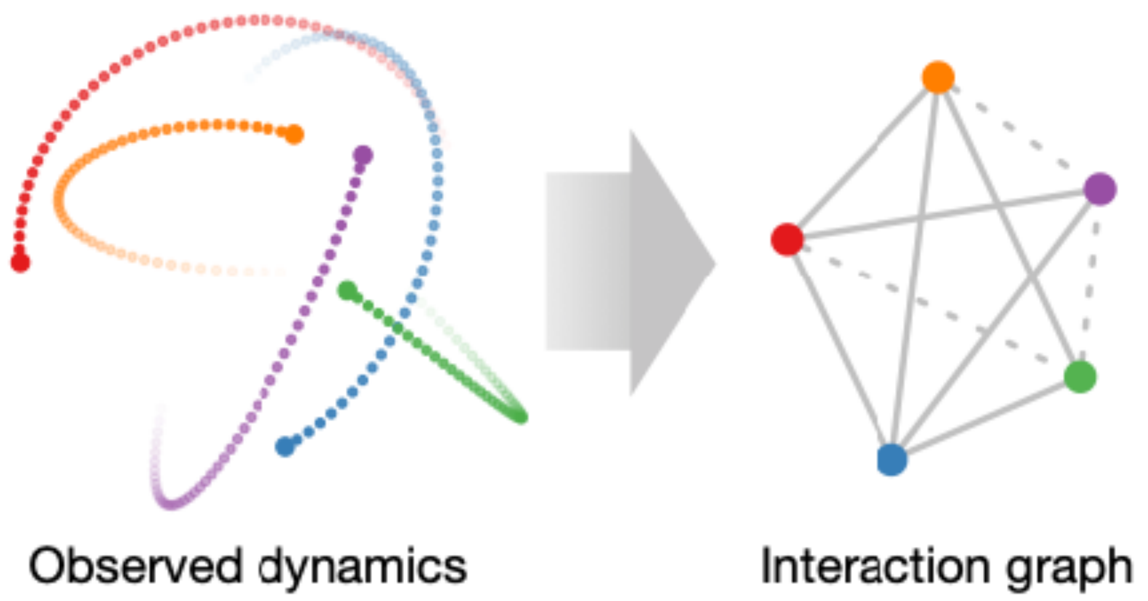
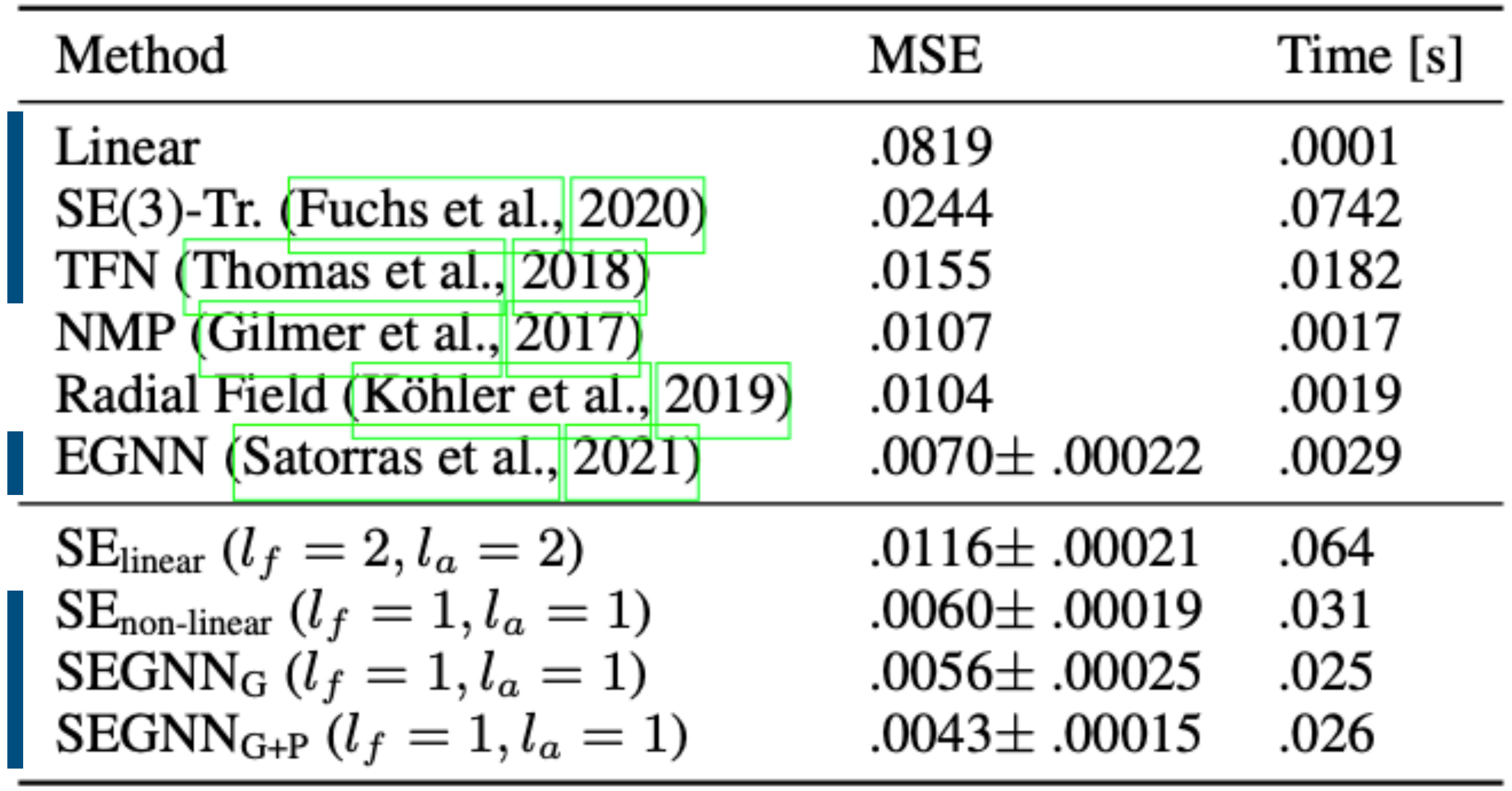


*Figure from Kipf et al 2018



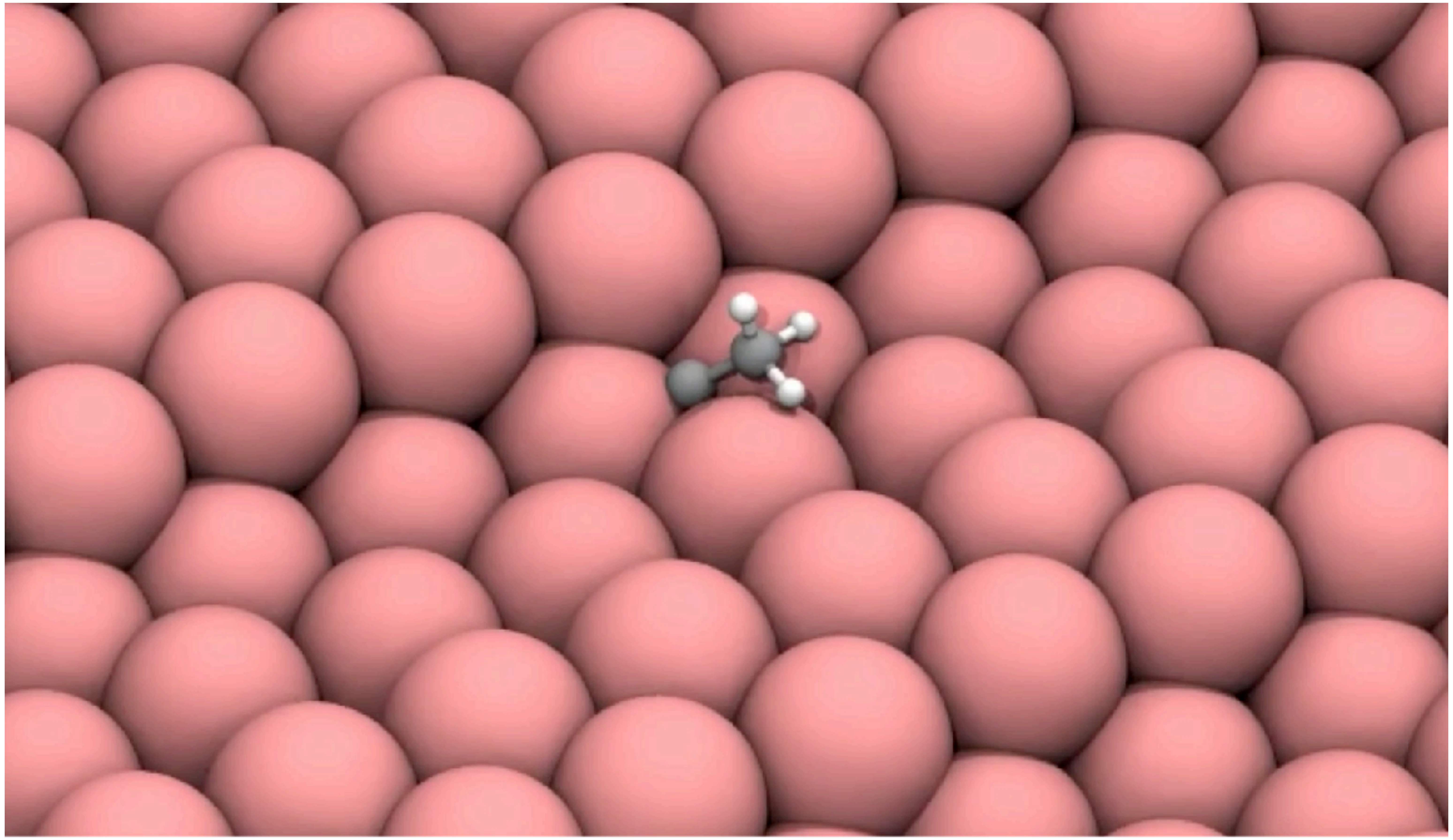
	Method	MSE	Time [s]
G-CNNs	Linear	.0819	.0001
	SE(3)-Tr. (Fuchs et al., 2020)	.0244	.0742
	TFN (Thomas et al., 2018)	.0155	.0182
	NMP (Gilmer et al., 2017)	.0107	.0017
	Radial Field (Köhler et al., 2019)	.0104	.0019
Isotropic “non-linear CNNs”	EGNN (Satorras et al., 2021)	.0070± .00022	.0029
	$\text{SE}_{\text{linear}}$ $(l_f = 2, l_a = 2)$	.0116± .00021	.064
“non-linear G-CNNs”	$\text{SE}_{\text{non-linear}}$ $(l_f = 1, l_a = 1)$	.0060± .00019	.031
	SEGNN_{G} $(l_f = 1, l_a = 1)$	.0056± .00025	.025
	$\text{SEGNN}_{\text{G+P}}$ $(l_f = 1, l_a = 1)$	.0043± .00015	.026

G-CNNs outperform CNNs with isotropic kernels

“Non-linear convolutions” outperform linear convolutions

Steerable methods for computational chemistry

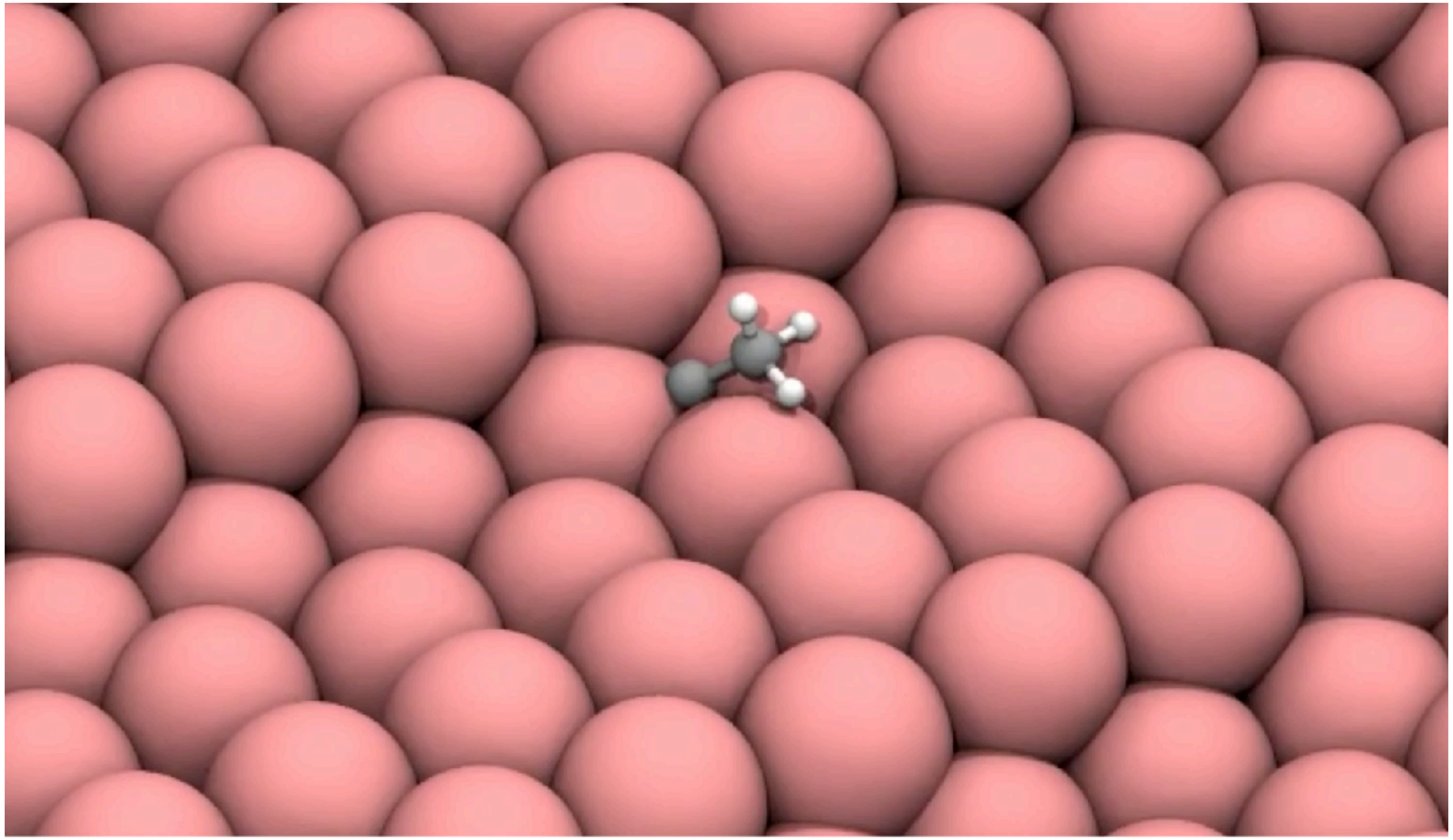
Brandstetter, Hesselink, van der Pol, Bekkers, Welling **Geometric and Physical Quantities Improve E(3) Equivariant Message Passing** - arXiv:2110.02905



Video: Open Catalyst Project

Steerable methods for computational chemistry

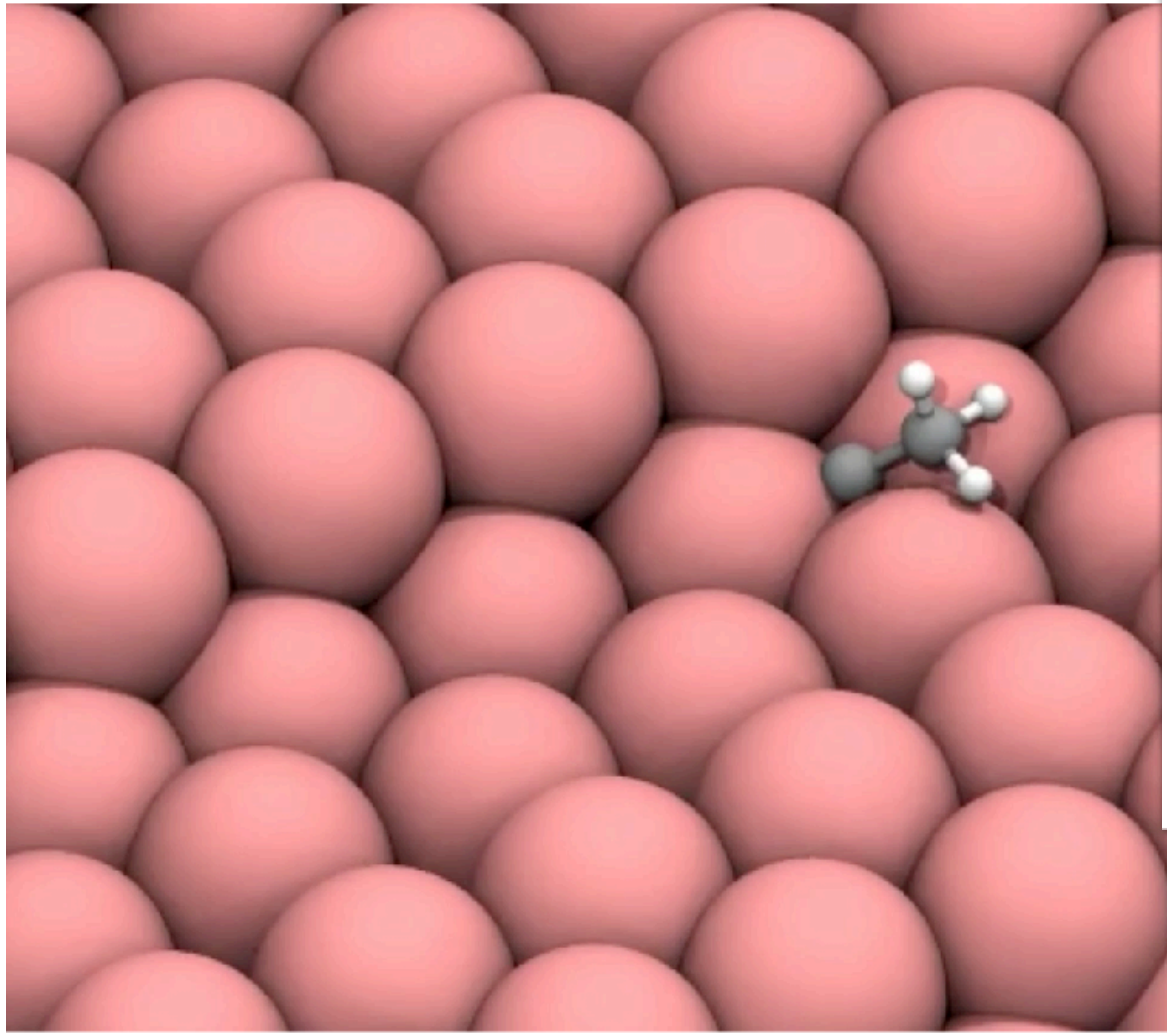
Brandstetter, Hesselink, van der Pol, Bekkers, Welling **Geometric and Physical Quantities Improve E(3) Equivariant Message Passing** - arXiv:2110.02905

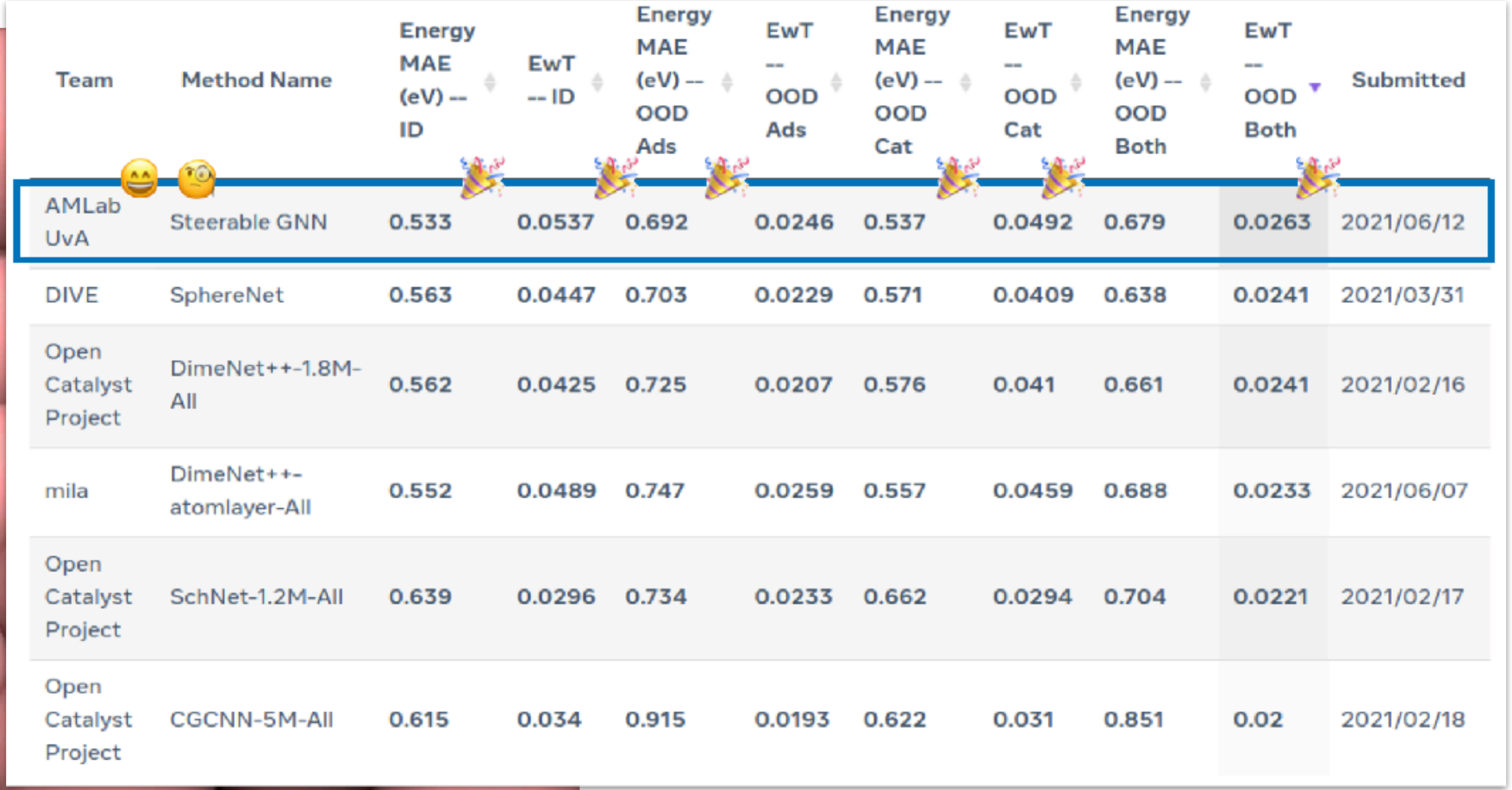


Video: Open Catalyst Project

Steerable methods for computational chemistry

Brandstetter, Hesselink, van der Pol, Bekkers, Welling **Geometric and Physical Quantities Improve E(3) Equivariant Message Passing** - arXiv:2110.02905





Team	Method Name	Energy MAE (eV) -- ID	EwT -- ID	Energy MAE (eV) -- OOD Ads	EwT -- OOD Ads	Energy MAE (eV) -- OOD Cat	EwT -- OOD Cat	Energy MAE (eV) -- OOD Both	EwT -- OOD Both	Submitted
AMLab UvA	Steerable GNN	0.533	0.0537	0.692	0.0246	0.537	0.0492	0.679	0.0263	2021/06/12
DIVE	SphereNet	0.563	0.0447	0.703	0.0229	0.571	0.0409	0.638	0.0241	2021/03/31
Open Catalyst Project	DimeNet++-1.8M-All	0.562	0.0425	0.725	0.0207	0.576	0.041	0.661	0.0241	2021/02/16
mila	DimeNet++-atomlayer-All	0.552	0.0489	0.747	0.0259	0.557	0.0459	0.688	0.0233	2021/06/07
Open Catalyst Project	SchNet-1.2M-All	0.639	0.0296	0.734	0.0233	0.662	0.0294	0.704	0.0221	2021/02/17
Open Catalyst Project	CGCNN-5M-All	0.615	0.034	0.915	0.0193	0.622	0.031	0.851	0.02	2021/02/18

Video: Open Catalyst Project

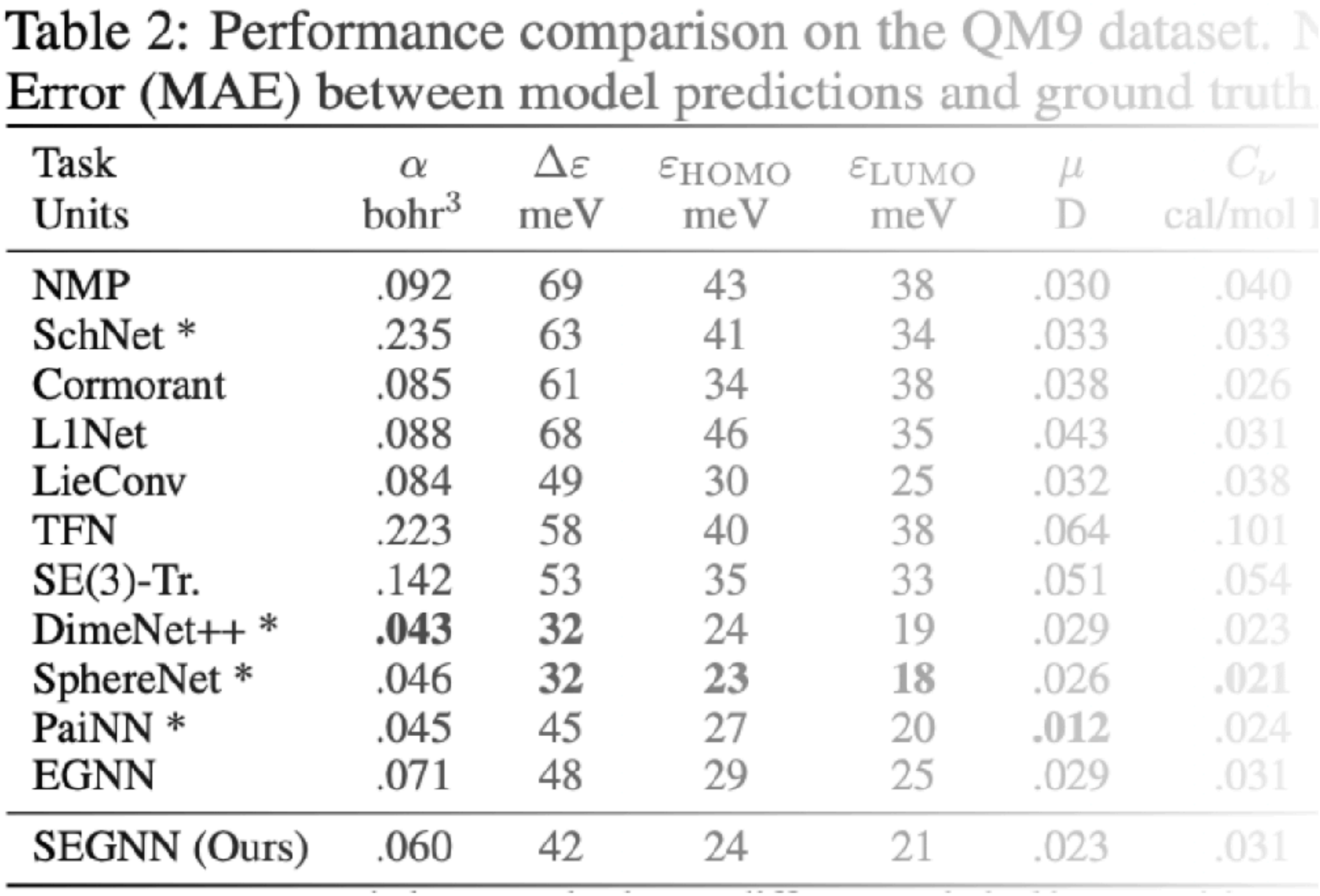


Table 2: Performance comparison on the QM9 dataset. N
Error (MAE) between model predictions and ground truth

			Task Units	α bohr^3	$\Delta\varepsilon$ meV	$\varepsilon_{\text{HOMO}}$ meV	$\varepsilon_{\text{LUMO}}$ meV	μ D	C_ν cal/mol
non-linear		no geometry	NMP	.092	69	43	38	.030	.040
	regular	$\mathbb{R}^3$	SchNet *	.235	63	41	34	.033	.033
pseudo-linear	steerable	$\mathbb{R}^3$	Cormorant	.085	61	34	38	.038	.026
	steerable	$SE(3)$	L1Net	.088	68	46	35	.043	.031
	regular	G	LieConv	.084	49	30	25	.032	.038
	steerable	$SE(3)$	TFN	.223	58	40	38	.064	.101
pseudo-linear	steerable	$SE(3)$	SE(3)-Tr.	.142	53	35	33	.051	.054
non-linear	regular	$\mathbb{R}^3 \times S^2 \times \mathbb{R}^+$	DimeNet++ *	**.043**	**32**	24	19	.029	.023
non-linear	regular	$\mathbb{R}^3 \times S^2 \times \mathbb{R}^+$	SphereNet *	.046	**32**	**23**	**18**	.026	**.021**
non-linear	reguleerable?	$SE(3)$	PaiNN *	.045	45	27	20	**.012**	.024
non-linear	regular	$\mathbb{R}^3$	EGNN	.071	48	29	25	.029	.031
non-linear	steerable	$SE(3)$	SEGNN (Ours)	.060	42	24	21	.023	.031

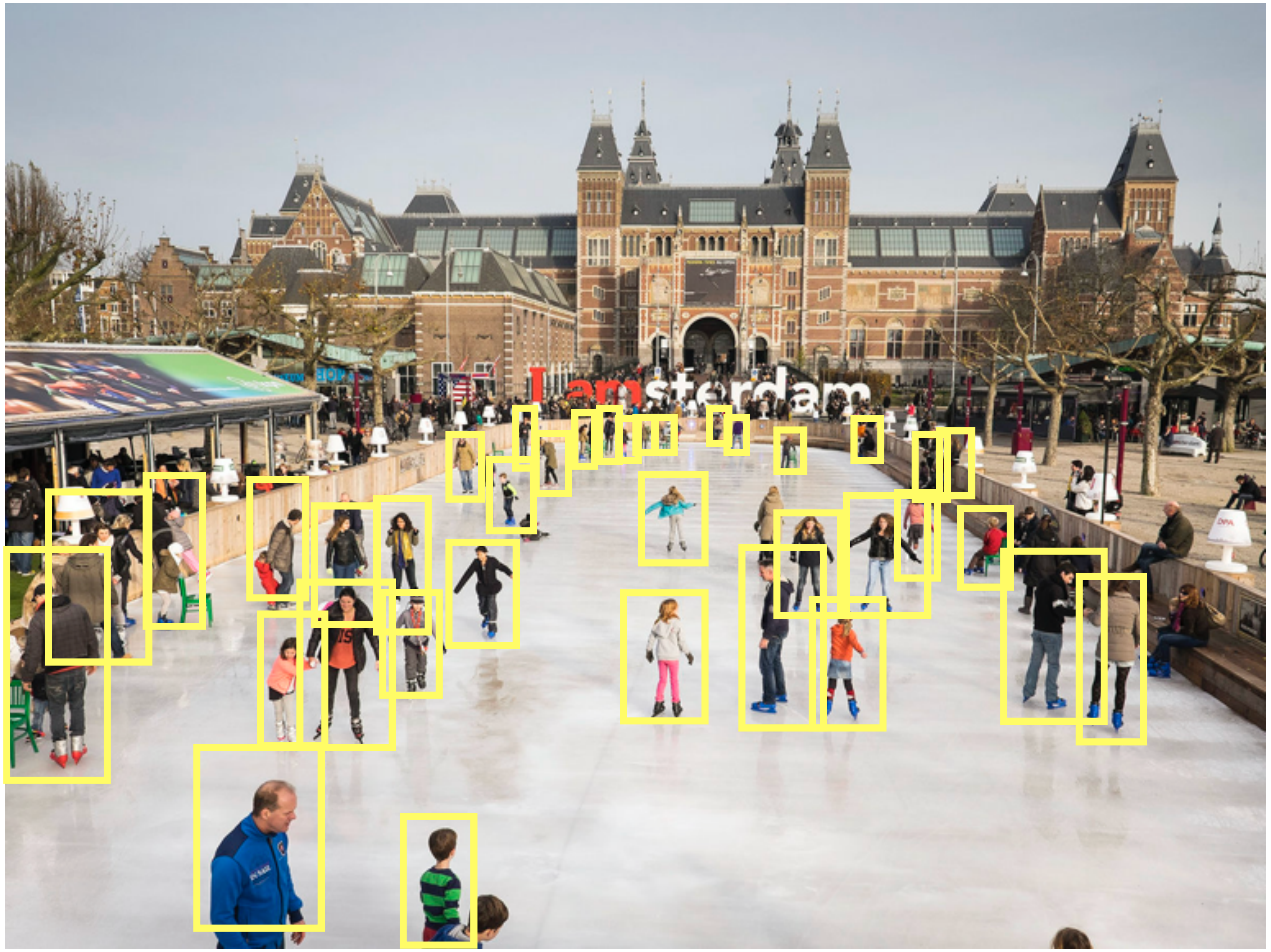
Table 2: Comparison on QM9.

1. Motivation	Equivariance → weight-sharing and generalization
2. Pattern matching using group theory	Group theory: symmetries & recognition by components (features have “poses”)
3. Group convolutions	Template matching over groups
4. Example	Effective representation learning and generalization
5. G-convs are all you need!	Any equivariant linear layer is a group convolution
6. Steerable group convolutions	Efficient (band-limited) **grid-free** g-convs
7. Feature fields and escnn library	Flexible framework for equivariant layers
8. Equivariant tensor product layers	Conv layers ↔ TPs with coordinate embeddings (Clebsch-Gordan: equivariant TP)
9. Equivariant graph NNs	Equivariant MP via geometry conditioned layers

1. Motivation	Equivariance → weight-sharing and generalization
2. Pattern matching using group theory	Group theory: symmetries & recognition by components (features have “poses”)
3. Group convolutions	Template matching over groups
4. Example	Effective representation learning and generalization
5. G-convs are all you need!	Any equivariant linear layer is a group convolution
6. Steerable group convolutions	Efficient (band-limited) **grid-free** g-convs
7. Feature fields and escnn library	Flexible framework for equivariant layers
8. Equivariant tensor product layers	Conv layers ↔ TPs with coordinate embeddings (Clebsch-Gordan: equivariant TP)
9. Equivariant graph NNs	Equivariant MP via geometry conditioned layers

Conclusion

Geometric guarantees (equivariance)



Importance of equivariance:

- No information is lost when the input is transformed
- Guaranteed stability to (local + global) transformations

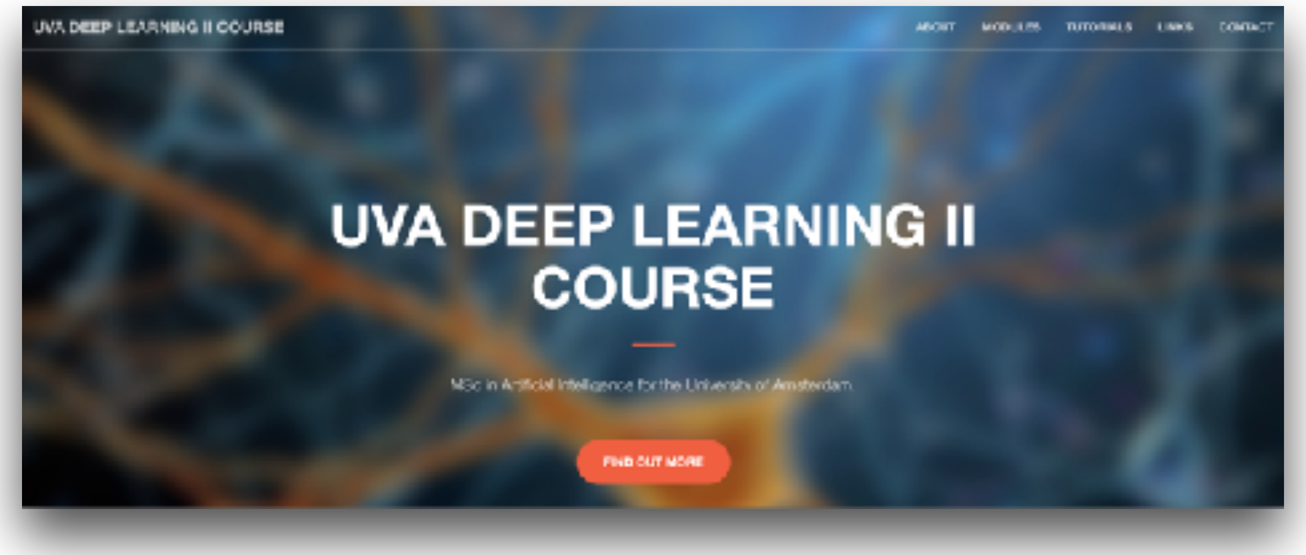
Group convolutions:

- Equivariance beyond translations
- Geometric guarantees
- Increased weight sharing

$\Big\}$ + **performance**
+ **generalization**

G-CNNs are not only relevant for invariant problems but for any type of structured data!

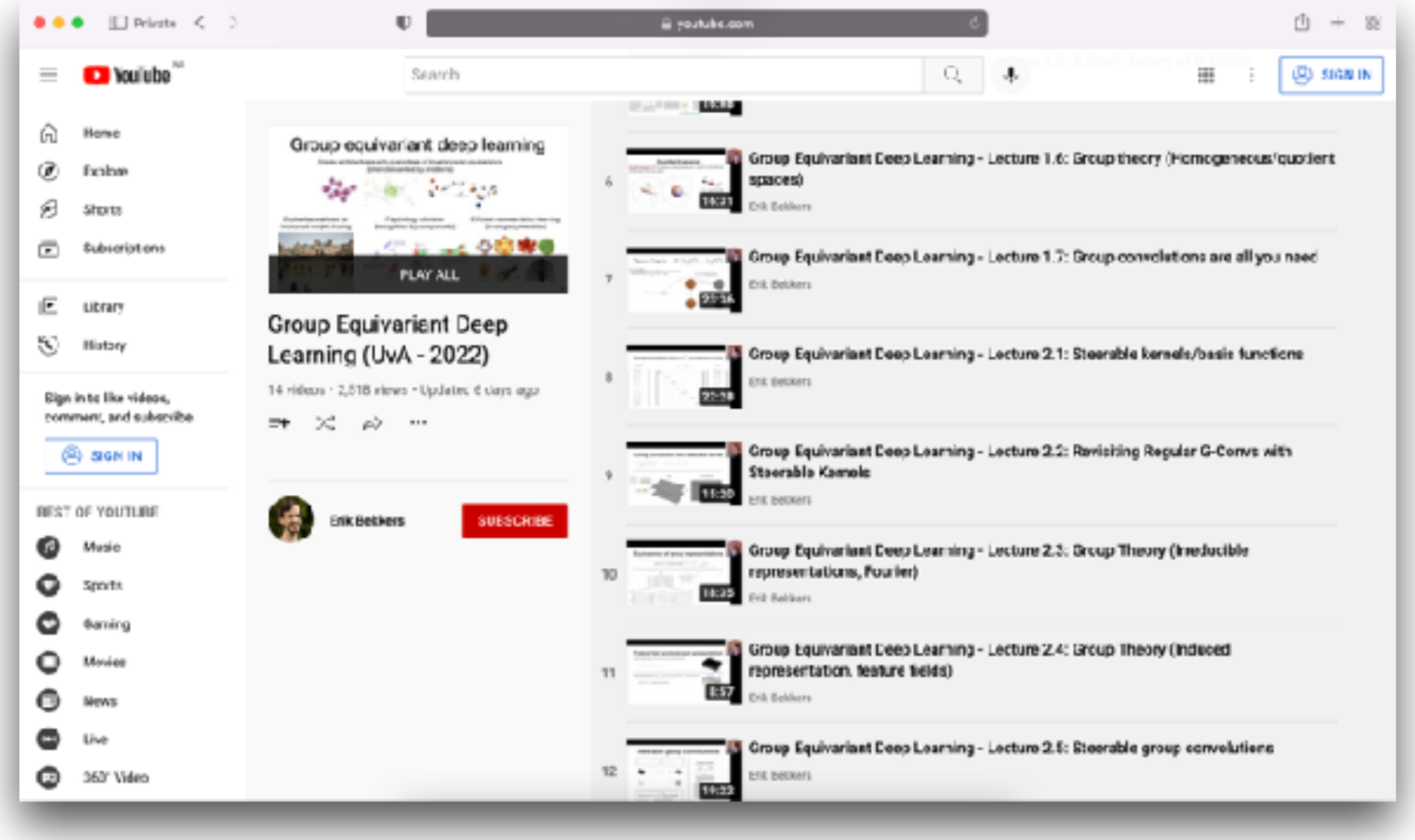
UvA course on group equivariant deep learning (https://uvagedl.github.io)



UVA DEEP LEARNING II COURSE

MSc in Artificial Intelligence for the University of Amsterdam

Youtube playlist



Group Equivariant Deep Learning (UvA - 2022)

14 videos · 2,018 views · Updated 6 days ago

Erik Bekkers

6 Group Equivariant Deep Learning - Lecture 1.6: Group theory (Homogeneous/quotient spaces) Erik Bekkers

7 Group Equivariant Deep Learning - Lecture 1.7: Group convolutions are all you need Erik Bekkers

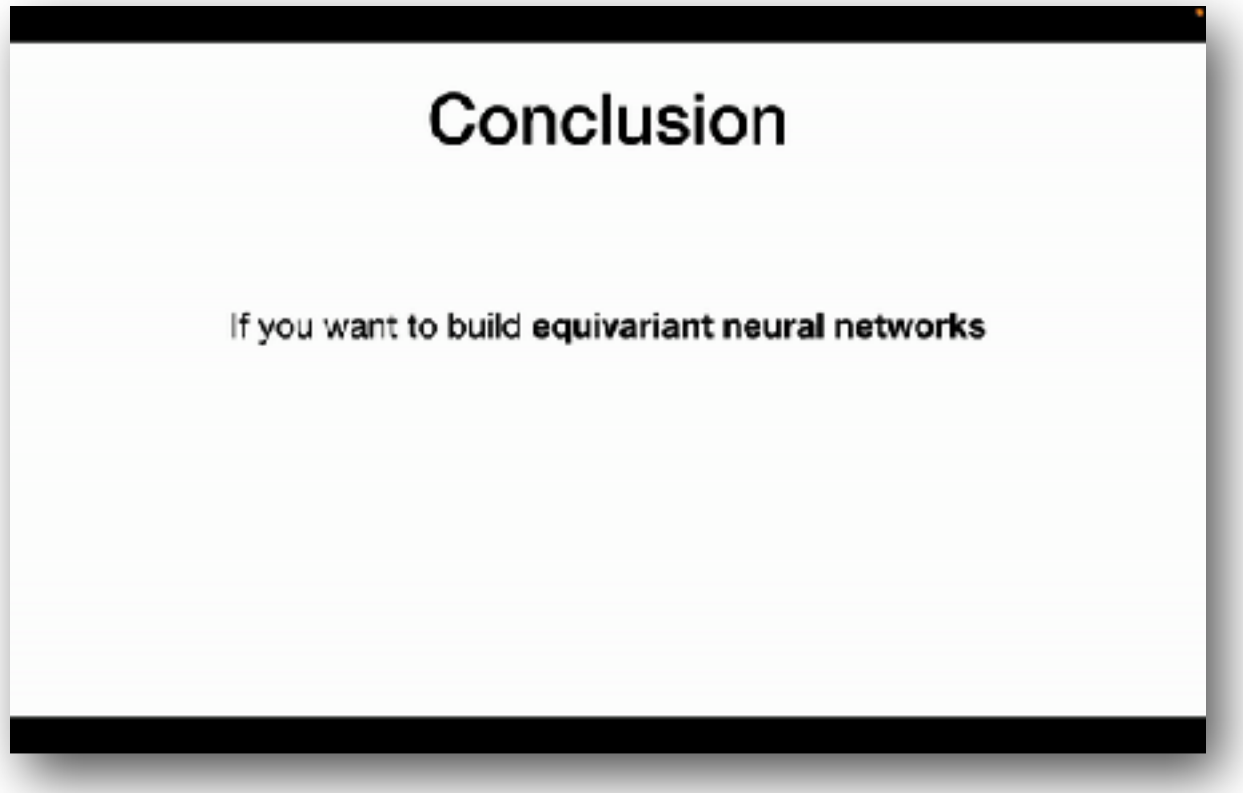
8 Group Equivariant Deep Learning - Lecture 2.1: Steerable kernels/basis functions Erik Bekkers

9 Group Equivariant Deep Learning - Lecture 2.2: Revisiting Regular G-Convs with Steerable Kernels Erik Bekkers

10 Group Equivariant Deep Learning - Lecture 2.3: Group Theory (Irreducible representations, Fourier) Erik Bekkers

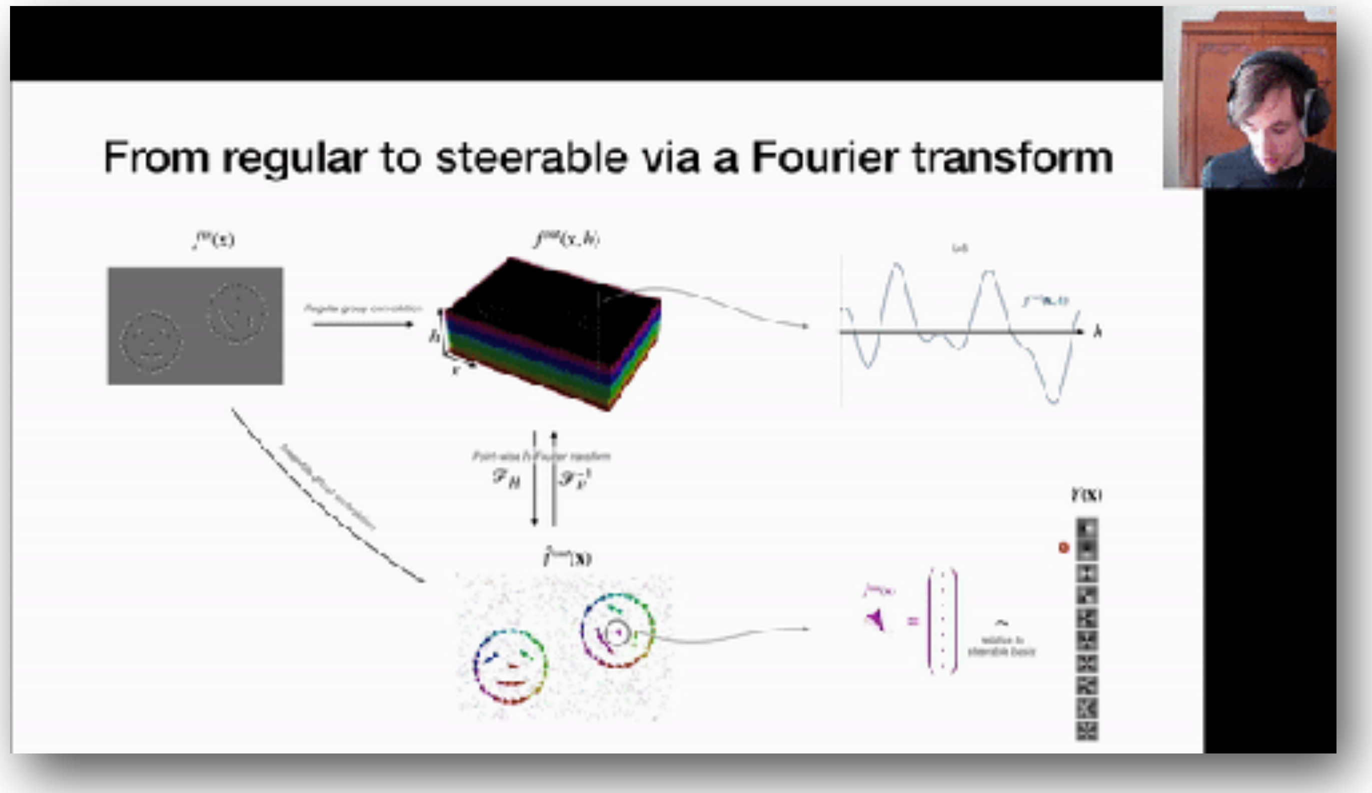
11 Group Equivariant Deep Learning - Lecture 2.4: Group Theory (Induced representation, feature fields) Erik Bekkers

12 Group Equivariant Deep Learning - Lecture 2.5: Steerable group convolutions Erik Bekkers



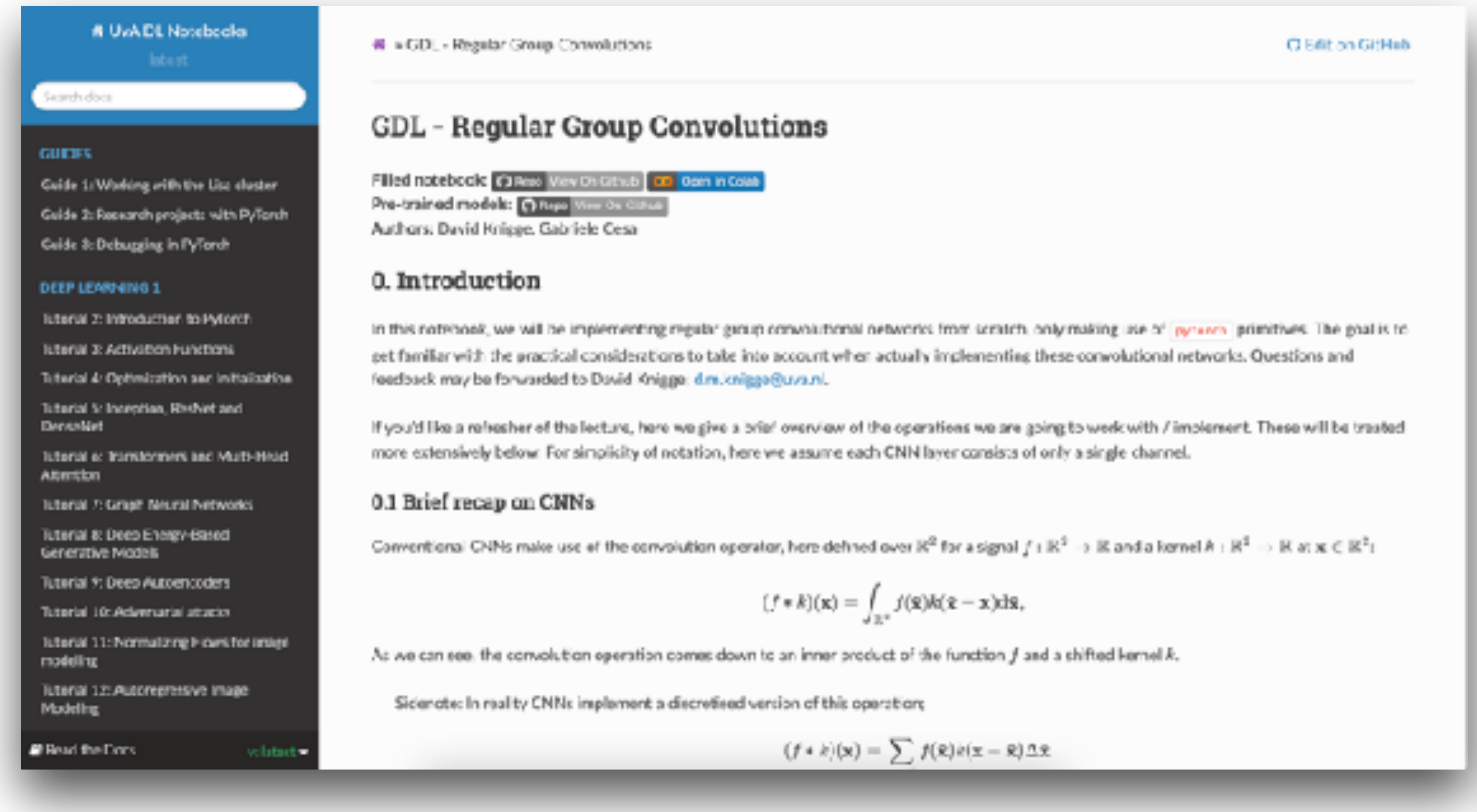
Conclusion

If you want to build **equivariant neural networks**



From regular to steerable via a Fourier transform

Tutorial notebooks



GDL - Regular Group Convolutions

Filled notebook: Repo View On Github | Open in Colab
Pre-trained models: Repo View On Github
Authors: David Knigge, Gabriele Cesa

0. Introduction

In this notebook, we will be implementing regular group convolutional networks from scratch, only making use of pytorch primitives. The goal is to get familiar with the practical considerations to take into account when actually implementing these convolutional networks. Questions and feedback may be forwarded to David Knigge: d.m.knigge@uva.nl.

If you'd like a refresher of the lecture, here we give a brief overview of the operations we are going to work with / implement. These will be treated more extensively below. For simplicity of notation, here we assume each CNN layer consists of only a single channel.

0.1 Brief recap on CNNs

Conventional CNNs make use of the convolution operator, here defined over $\mathbb{R}^2$ for a signal $f: \mathbb{R}^2 \rightarrow \mathbb{R}$ and a kernel $k: \mathbb{R}^2 \rightarrow \mathbb{R}$ at $\mathbf{x} \in \mathbb{R}^2$:

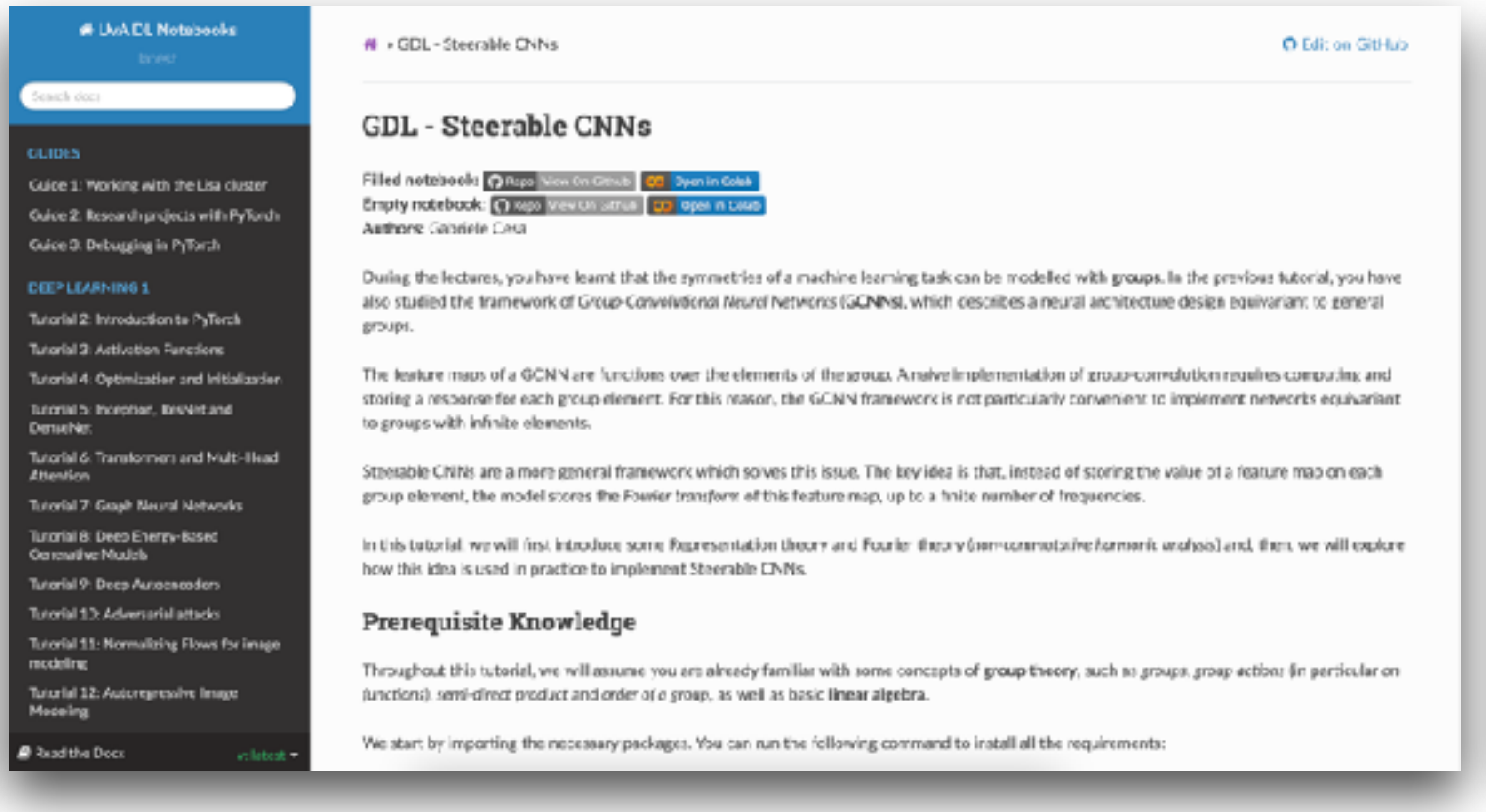
$$(f * k)(\mathbf{x}) = \int_{\mathbb{R}^2} f(\tilde{\mathbf{x}})k(\tilde{\mathbf{x}} - \mathbf{x})d\tilde{\mathbf{x}},$$

As we can see, the convolution operation comes down to an inner product of the function f and a shifted kernel k.

Sidenote: In reality CNNs implement a discretised version of this operation:

$$(f * k)(\mathbf{x}) = \sum f(\tilde{\mathbf{x}})k(\tilde{\mathbf{x}} - \mathbf{x})\Delta\tilde{\mathbf{x}}$$

UvA DL Notebooks
Edit on GitHub
GUIDES
Guide 1: Working with the Lisa cluster
Guide 2: Research projects with PyTorch
Guide 3: Debugging in PyTorch
DEEP LEARNING 1
Tutorial 2: Introduction to PyTorch
Tutorial 3: Activation Functions
Tutorial 4: Optimization and Initialization
Tutorial 5: Inception, ResNet and DenseNet
Tutorial 6: Transformers and Multi-Head Attention
Tutorial 7: Graph Neural Networks
Tutorial 8: Deep Energy-Based Generative Models
Tutorial 9: Deep Autoencoders
Tutorial 10: Adversarial attacks
Tutorial 11: Normalizing Flows for image modeling
Tutorial 12: Autoregressive Image Modeling
Read the Docs
v: latest



GDL - Steerable CNNs

Filled notebook: Repo View On Github | Open in Colab
Empty notebook: Repo View On Github | Open in Colab
Authors: Gabriele Cesa

During the lectures, you have learnt that the symmetries of a machine learning task can be modelled with **groups**. In the previous tutorial, you have also studied the framework of *Group-Convolutional Neural Networks* (**GCNNs**), which describes a neural architecture design equivariant to general groups.

The feature maps of a GCNN are functions over the elements of the group. A naive implementation of group-convolution requires computing and storing a response for each group element. For this reason, the GCNN framework is not particularly convenient to implement networks equivariant to groups with infinite elements.

Steerable CNNs are a more general framework which solves this issue. The key idea is that, instead of storing the value of a feature map on each group element, the model stores the Fourier transform of this feature map, up to a finite number of frequencies.

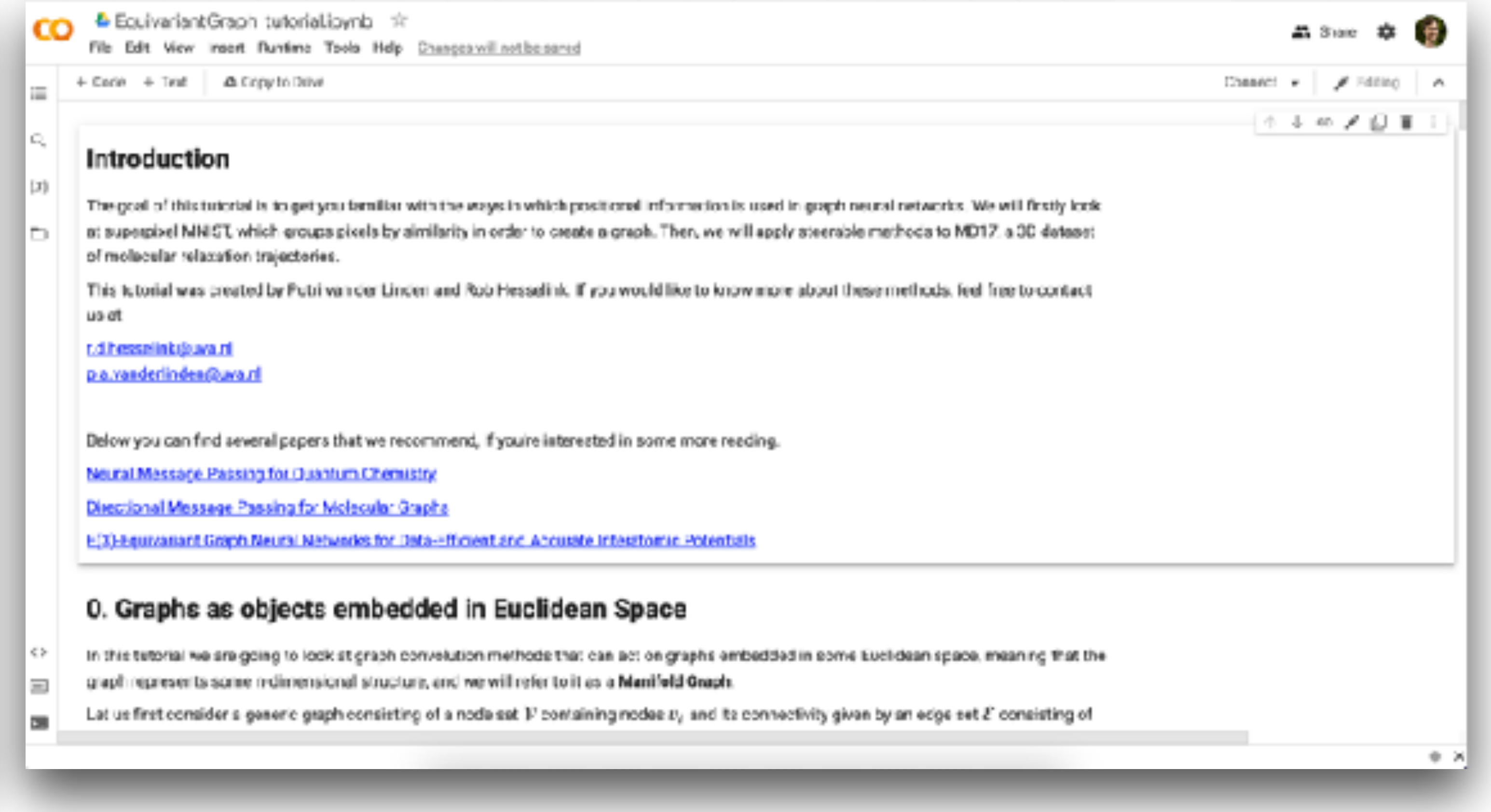
In this tutorial, we will first introduce some Representation theory and Fourier theory (*non-commutative harmonic analysis*) and, then, we will explore how this idea is used in practice to implement Steerable CNNs.

Prerequisite Knowledge

Throughout this tutorial, we will assume you are already familiar with some concepts of **group theory**, such as *groups*, *group actions* (in particular on functions), *semi-direct product* and *order of a group*, as well as basic **linear algebra**.

We start by importing the necessary packages. You can run the following command to install all the requirements:

UvA DL Notebooks
Edit on GitHub
GUIDES
Guide 1: Working with the Lisa cluster
Guide 2: Research projects with PyTorch
Guide 3: Debugging in PyTorch
DEEP LEARNING 1
Tutorial 2: Introduction to PyTorch
Tutorial 3: Activation Functions
Tutorial 4: Optimization and Initialization
Tutorial 5: Inception, ResNet and DenseNet
Tutorial 6: Transformers and Multi-Head Attention
Tutorial 7: Graph Neural Networks
Tutorial 8: Deep Energy-Based Generative Models
Tutorial 9: Deep Autoencoders
Tutorial 10: Adversarial attacks
Tutorial 11: Normalizing Flows for image modeling
Tutorial 12: Autoregressive Image Modeling
Read the Docs
v: latest



EquivariantGraph_tutorial.ipynb

Introduction

The goal of this tutorial is to get you familiar with the ways in which positional information is used in graph neural networks. We will firstly look at superpixel MNIST, which groups pixels by similarity in order to create a graph. Then, we will apply steerable methods to MD17, a 3D dataset of molecular relaxation trajectories.

This tutorial was created by Putri van der Linden and Rob Hesselink. If you would like to know more about these methods, feel free to contact us at

r.d.hesselink@uva.nl
p.a.vanderlinden@uva.nl

Below you can find several papers that we recommend, if you're interested in some more reading.

Neural Message Passing for Quantum Chemistry

Directional Message Passing for Molecular Graphs

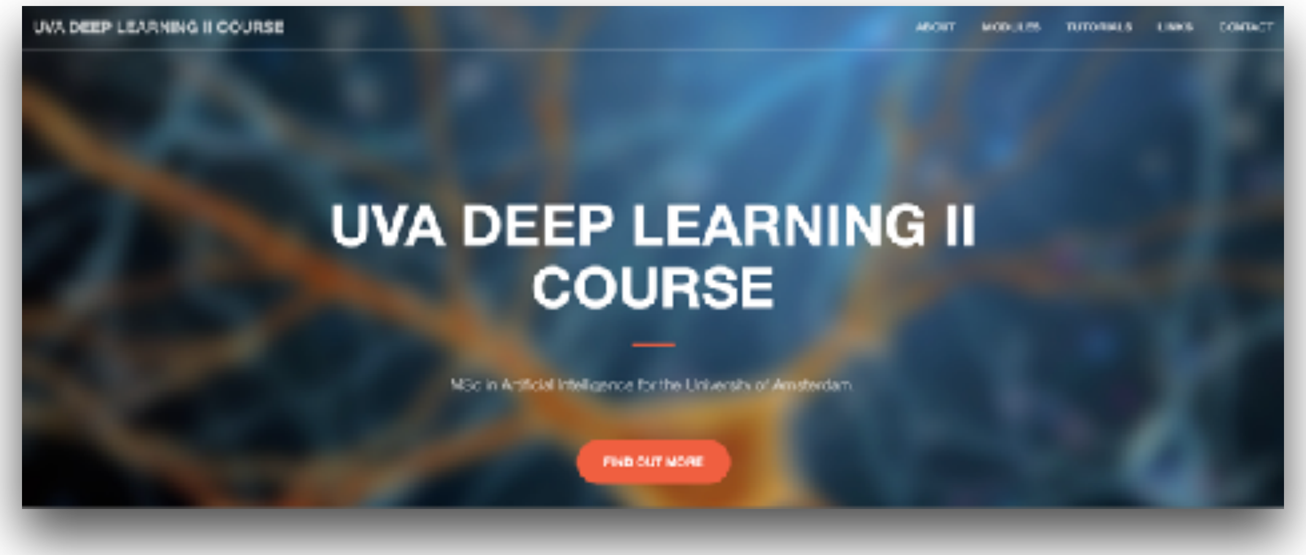
E(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials

0. Graphs as objects embedded in Euclidean Space

In this tutorial we are going to look at graph convolution methods that can act on graphs embedded in some Euclidean space, meaning that the graph represents some n-dimensional structure, and we will refer to it as a **Manifold Graph**.

Let us first consider a generic graph consisting of a node set V containing nodes v_i and its connectivity given by an edge set E consisting of

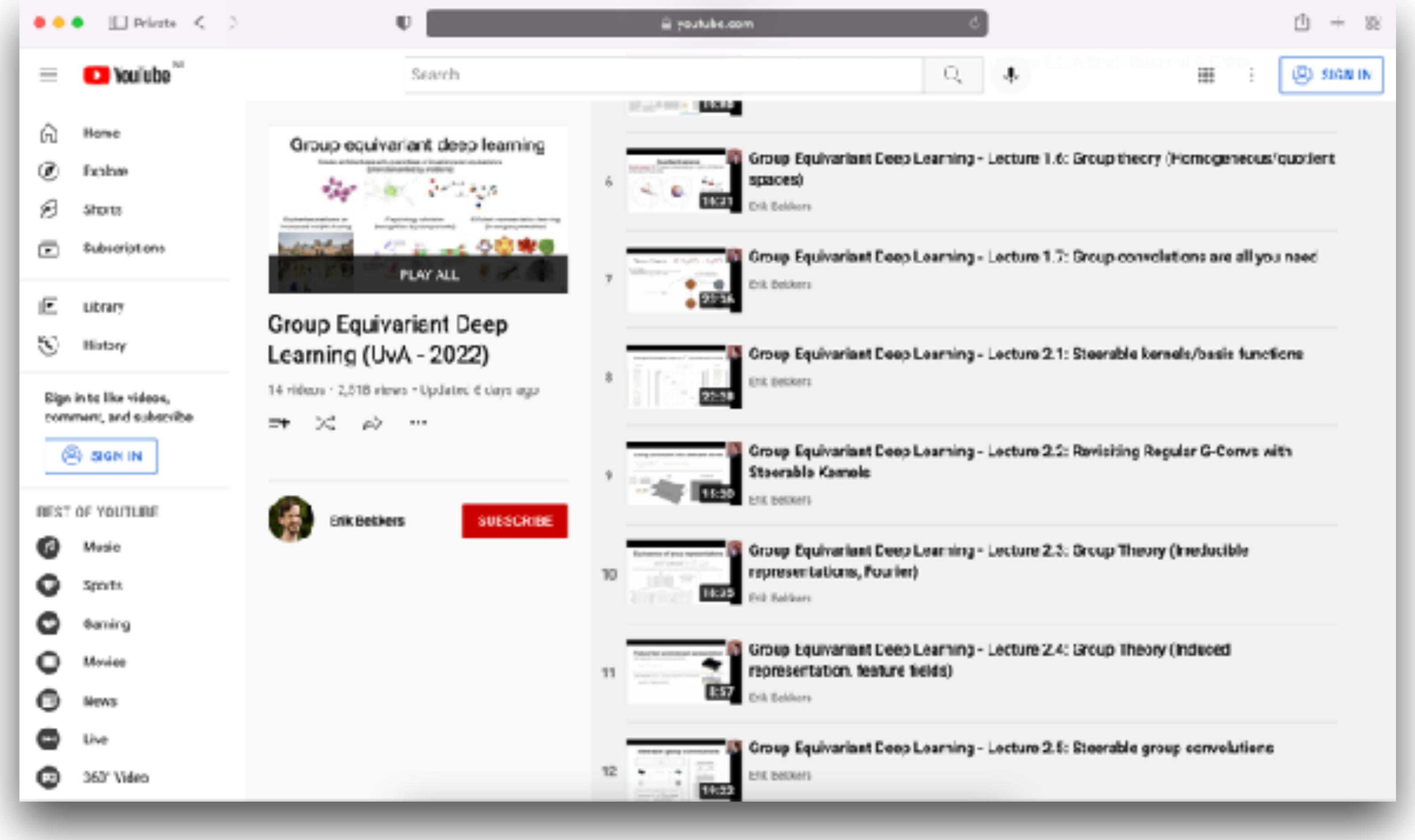
UvA course on group equivariant deep learning (https://uvagedl.github.io)



UVA DEEP LEARNING II COURSE

MSc in Artificial Intelligence for the University of Amsterdam

Youtube playlist



Group equivariant deep learning

Group Equivariant Deep Learning (UvA - 2022)

14 videos · 2,518 views · Updated 4 days ago

Sign in to like videos, comment, and subscribe.

Erik Bekkers

6 Group Equivariant Deep Learning - Lecture 1.6: Group theory (Homogeneous/quotient spaces) Erik Bekkers 10:21

7 Group Equivariant Deep Learning - Lecture 1.7: Group convolutions are all you need Erik Bekkers 22:16

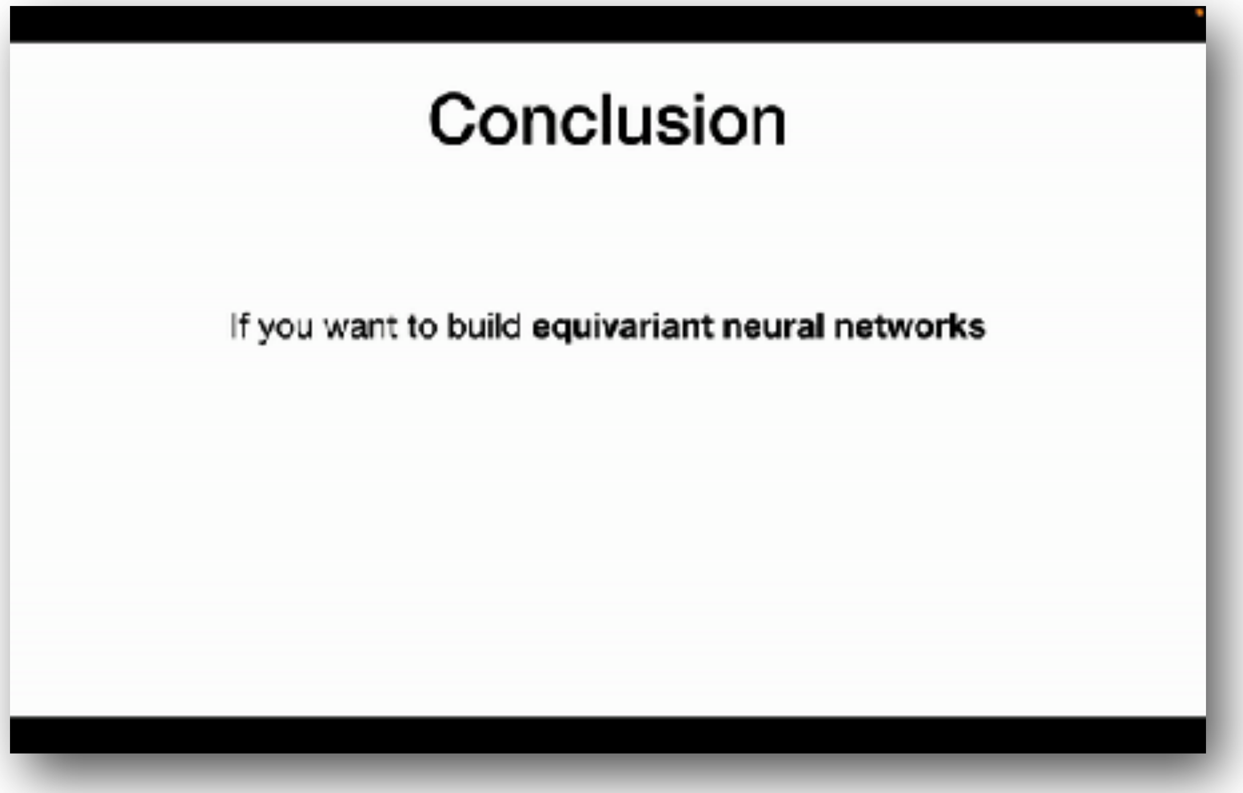
8 Group Equivariant Deep Learning - Lecture 2.1: Steerable kernels/basis functions Erik Bekkers 22:18

9 Group Equivariant Deep Learning - Lecture 2.2: Revisiting Regular G-Convs with Steerable Kernels Erik Bekkers 11:30

10 Group Equivariant Deep Learning - Lecture 2.3: Group Theory (Irreducible representations, Fourier) Erik Bekkers 18:25

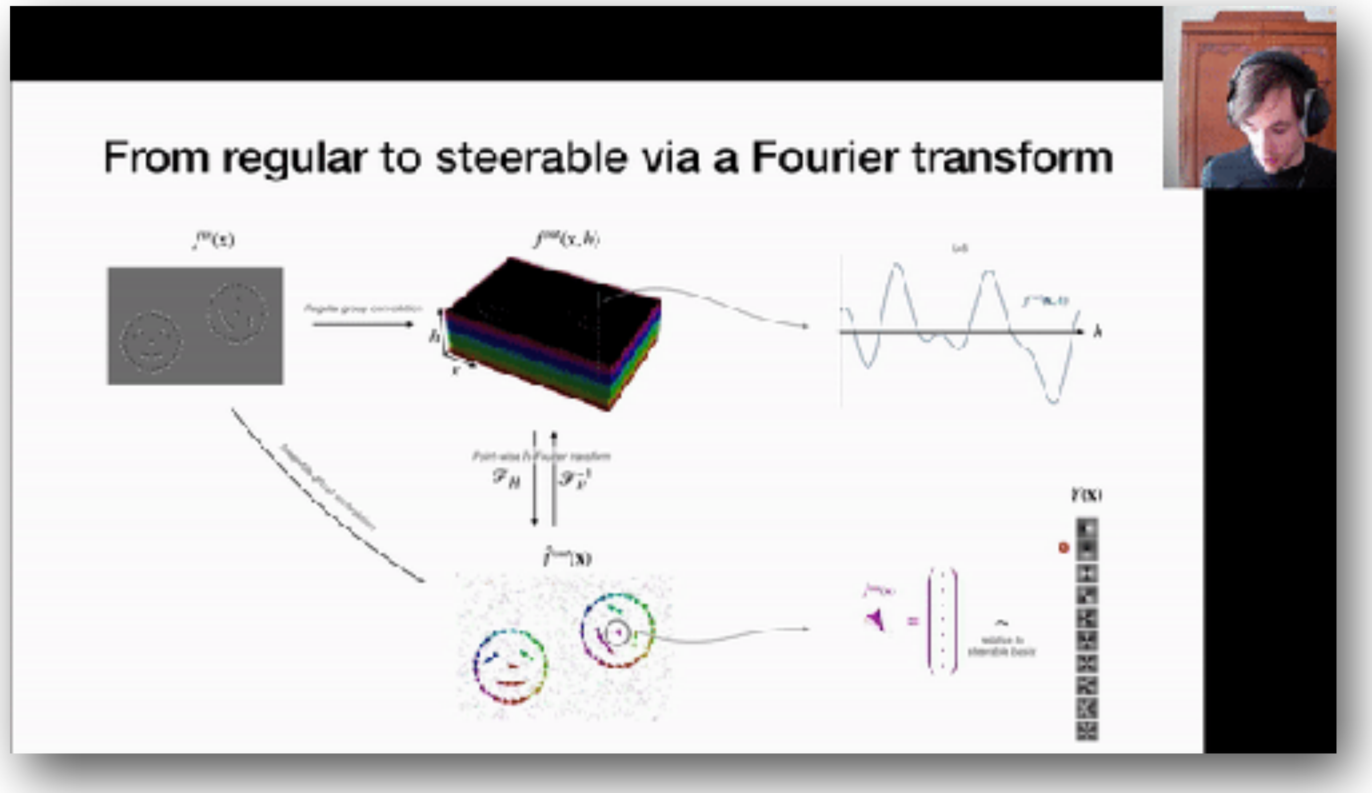
11 Group Equivariant Deep Learning - Lecture 2.4: Group Theory (Induced representation, feature fields) Erik Bekkers 8:57

12 Group Equivariant Deep Learning - Lecture 2.5: Steerable group convolutions Erik Bekkers 19:33



Conclusion

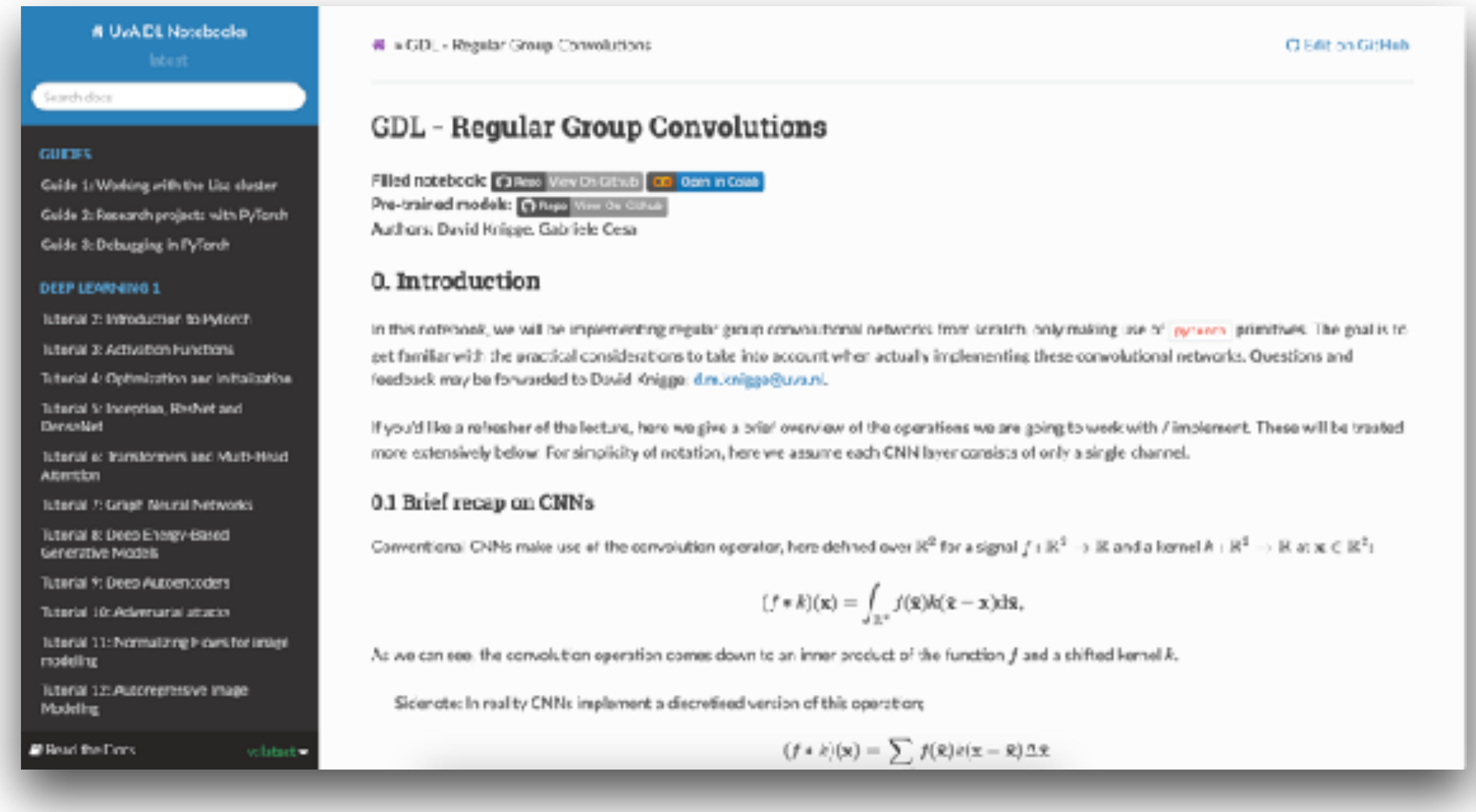
If you want to build **equivariant neural networks**



From regular to steerable via a Fourier transform

$f^{in}(\mathbf{x})$
Regular group convolution
$f^{out}(\mathbf{x},h)$
h
$\mathbf{x}$
$f^{out}(\mathbf{x},\cdot)$
h
Point-wise H-Fourier transform
$\mathscr{F}_H$
$\mathscr{F}_H^{-1}$
Steerable group convolution
$\hat{f}^{out}(\mathbf{x})$
$\hat{f}^{out}(\mathbf{x})$
=
relative to steerable basis
$Y(\mathbf{x})$

Tutorial notebooks



GUIDES

Guide 1: Working with the Lisa cluster

Guide 2: Research projects with PyTorch

Guide 3: Debugging in PyTorch

DEEP LEARNING 1

Tutorial 2: Introduction to PyTorch

Tutorial 3: Activation Functions

Tutorial 4: Optimization and Initialization

Tutorial 5: Inception, ResNet and DenseNet

Tutorial 6: Transformers and Multi-Head Attention

Tutorial 7: Graph Neural Networks

Tutorial 8: Deep Energy-Based Generative Models

Tutorial 9: Deep Autoencoders

Tutorial 10: Adversarial attacks

Tutorial 11: Normalizing Flows for image modeling

Tutorial 12: Autoregressive Image Modeling

» GDL - Regular Group Convolutions

GDL - Regular Group Convolutions

Filled notebook: Repo View On Github Open in Colab
Pre-trained models: Repo View On Github
Authors: David Knigge, Gabriele Cesa

0. Introduction

In this notebook, we will be implementing regular group convolutional networks from scratch, only making use of pytorch primitives. The goal is to get familiar with the practical considerations to take into account when actually implementing these convolutional networks. Questions and feedback may be forwarded to David Knigge: d.m.knigge@uva.nl.

If you'd like a refresher of the lecture, here we give a brief overview of the operations we are going to work with / implement. These will be treated more extensively below. For simplicity of notation, here we assume each CNN layer consists of only a single channel.

0.1 Brief recap on CNNs

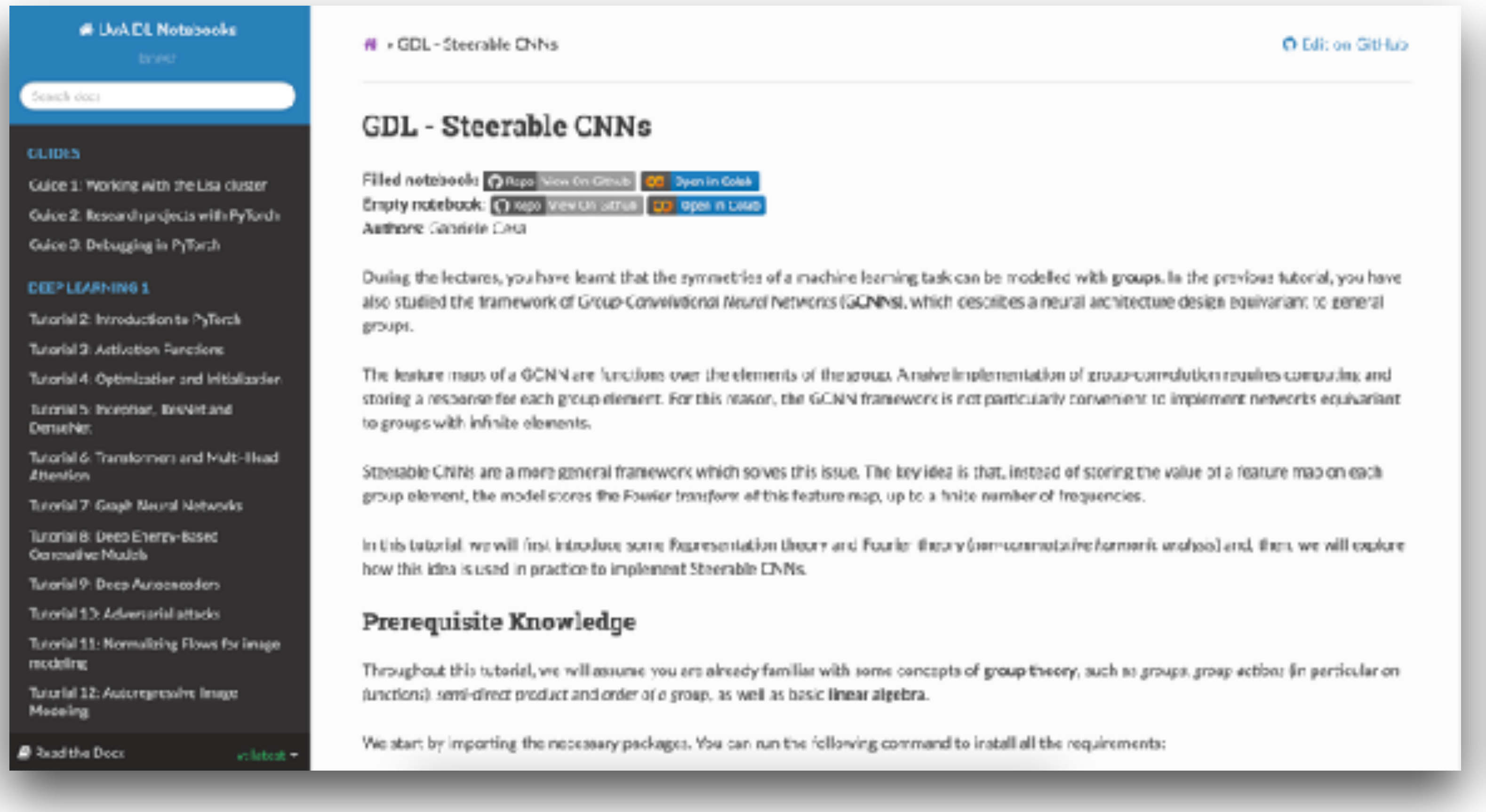
Conventional CNNs make use of the convolution operator, here defined over $\mathbb{R}^2$ for a signal $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ and a kernel $k : \mathbb{R}^2 \rightarrow \mathbb{R}$ at $\mathbf{x} \in \mathbb{R}^2$:

$$(f * k)(\mathbf{x}) = \int_{\mathbb{R}^2} f(\tilde{\mathbf{x}})k(\tilde{\mathbf{x}} - \mathbf{x})d\tilde{\mathbf{x}},$$

As we can see, the convolution operation comes down to an inner product of the function f and a shifted kernel k.

Sidenote: In reality CNNs implement a discretised version of this operation:

$$(f * k)(\mathbf{x}) = \sum f(\tilde{\mathbf{x}})k(\mathbf{x} - \tilde{\mathbf{x}})\Delta\tilde{\mathbf{x}}$$



GUIDES

Guide 1: Working with the Lisa cluster

Guide 2: Research projects with PyTorch

Guide 3: Debugging in PyTorch

DEEP LEARNING 1

Tutorial 2: Introduction to PyTorch

Tutorial 3: Activation Functions

Tutorial 4: Optimization and Initialization

Tutorial 5: Inception, ResNet and DenseNet

Tutorial 6: Transformers and Multi-Head Attention

Tutorial 7: Graph Neural Networks

Tutorial 8: Deep Energy-Based Generative Models

Tutorial 9: Deep Autoencoders

Tutorial 10: Adversarial attacks

Tutorial 11: Normalizing Flows for image modeling

Tutorial 12: Autoregressive Image Modeling

» GDL - Steerable CNNs

GDL - Steerable CNNs

Filled notebook: Repo View On Github Open in Colab
Empty notebook: Repo View On Github Open in Colab
Authors: Gabriele Cesa

During the lectures, you have learnt that the symmetries of a machine learning task can be modelled with **groups**. In the previous tutorial, you have also studied the framework of *Group-Convolutional Neural Networks* (**GCNNs**), which describes a neural architecture design equivariant to general groups.

The feature maps of a GCNN are functions over the elements of the group. A naive implementation of group-convolution requires computing and storing a response for each group element. For this reason, the GCNN framework is not particularly convenient to implement networks equivariant to groups with infinite elements.

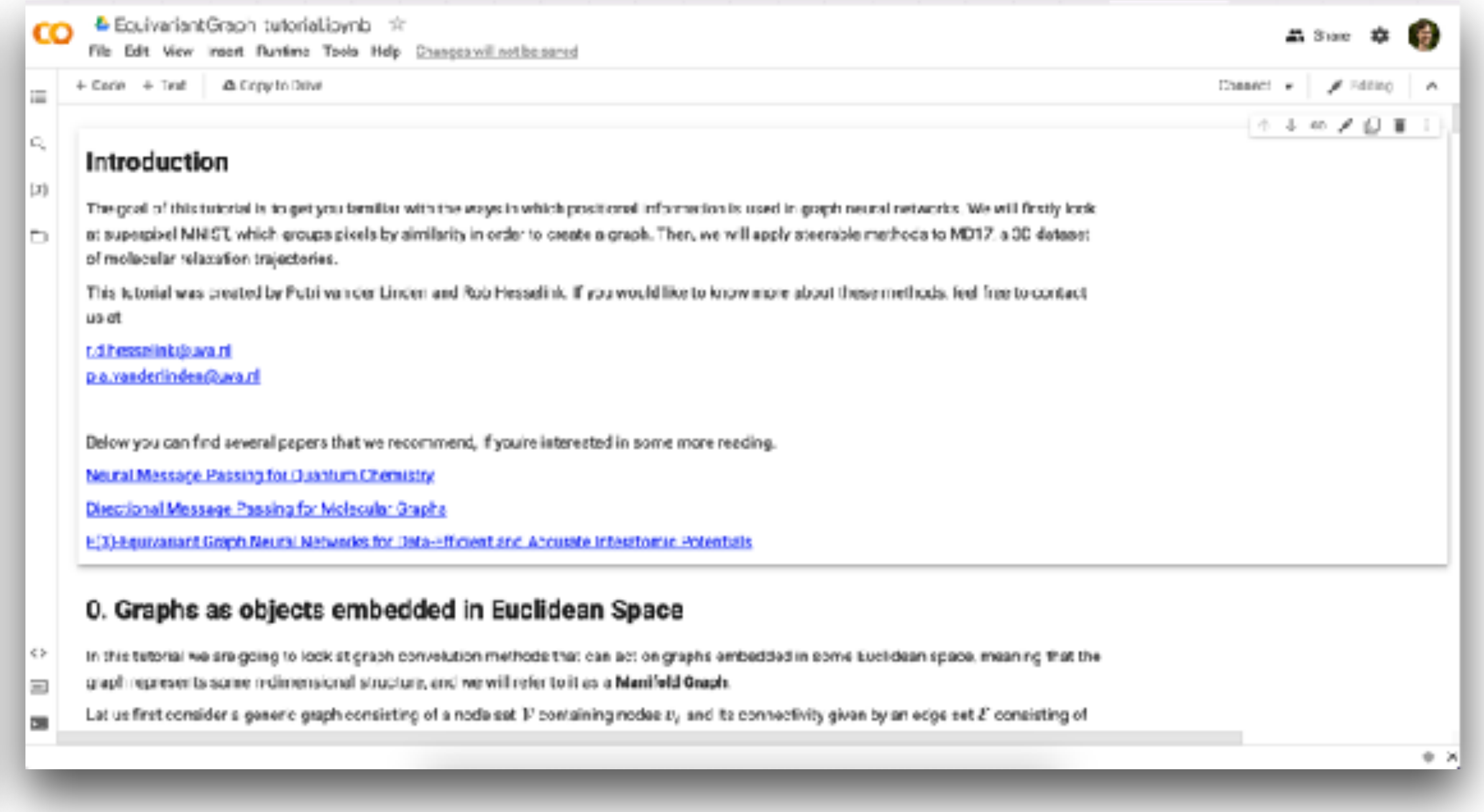
Steerable CNNs are a more general framework which solves this issue. The key idea is that, instead of storing the value of a feature map on each group element, the model stores the Fourier transform of this feature map, up to a finite number of frequencies.

In this tutorial, we will first introduce some Representation theory and Fourier theory (*non-commutative harmonic analysis*) and, then, we will explore how this idea is used in practice to implement Steerable CNNs.

Prerequisite Knowledge

Throughout this tutorial, we will assume you are already familiar with some concepts of **group theory**, such as *groups*, *group actions* (in particular on functions), *semi-direct product* and *order of a group*, as well as basic **linear algebra**.

We start by importing the necessary packages. You can run the following command to install all the requirements:



CO
EquivariantGraph tutorial.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text
Copy to Drive
Connect
Editing

Introduction

The goal of this tutorial is to get you familiar with the ways in which positional information is used in graph neural networks. We will firstly look at superpixel MNIST, which groups pixels by similarity in order to create a graph. Then, we will apply steerable methods to MD17, a 3D dataset of molecular relaxation trajectories.

This tutorial was created by Putri van der Linden and Rob Hesselink. If you would like to know more about these methods, feel free to contact us at

r.d.hesselink@uva.nl
p.a.vanderlinden@uva.nl

Below you can find several papers that we recommend, if you're interested in some more reading.

Neural Message Passing for Quantum Chemistry

Directional Message Passing for Molecular Graphs

E(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials

0. Graphs as objects embedded in Euclidean Space

In this tutorial we are going to look at graph convolution methods that can act on graphs embedded in some Euclidean space, meaning that the graph represents some n-dimensional structure, and we will refer to it as a **Manifold Graph**.

Let us first consider a generic graph consisting of a node set V containing nodes v_i and its connectivity given by an edge set E consisting of

Thank you for listening!