

# Geometric Deep Learning

---

Erik Schultheis, Kate Haitsiukevich, Çağlar Hızlı, Alison Pouplin, Vikas Garg

8.2.2024

## Course overview

- Introductory lectures: Group Theory, Graph Neural Networks, GDL Blueprint, Manifolds
- Exercises (first set: next week; deadline 6.11.24)
- Presentations by You

## Tentative Timeline

Week	Date	Session 1 (45')	Session 2 (45')
<b>October - Period II</b>			
Week 43	24.10	Introduction	Group Theory
Week 44	31.10	GNNs - Course I	GNNs - Course II
<b>November - Period II</b>			
Week 45	07.11	GNNs - Solutions	Sym. - Course
Week 46	14.11	Sym. - Solutions	Man. - Course I
Week 47	21.11	Man. - Course II	Paper
Week 48	28.11	Man. - Solutions	Paper
<b>December- Period II</b>			
Week 49	05.12	Evaluation week	
Week 50	12.12	Evaluation week / Neurips	
Week 51	19.12	Holidays	

## Tentative Timeline

Week	Date	Session 1 (45')	Session 2 (45')
<b>January - Period III</b>			
Week 02	09.01	Paper	Paper
Week 03	16.01	Paper	Paper
Week 04	23.01	Paper	Paper
Week 05	30.01	Paper	Paper
<b>February - Period III</b>			
Week 06	6.02	Paper	Paper
Week 07	13.02	Paper	Paper
Week 08	20.02	<b>Evaluation week</b>	

## Resources

- Geometric Deep Learning book
- Graph Representation Learning book
- Equivariant and coordinate-independent CNNs book



## Evaluation

- Participation in course
- Notebook exercises
- Paper presentation
- Writing assignments

## Part I: Why Geometric Deep Learning

---

# Learning on generic vector spaces

## classification with real-valued data

Given training dataset  $x_1, \dots, x_n \in \mathbb{R}^d$  and labels  $y_1, \dots, y_n \in [C]$ , generate a function so that

$$\mathbb{E}_{X,Y}[\ell(f(X), Y)] \rightarrow \min .$$

# Learning on generic vector spaces

## classification with real-valued data

Given training dataset  $x_1, \dots, x_n \in \mathbb{R}^d$  and labels  $y_1, \dots, y_n \in [C]$ , generate a function so that

$$\mathbb{E}_{X,Y}[\ell(f(X), Y)] \rightarrow \min .$$

Requires *huge* amounts of data.



Real data usually contains structure

Handwritten digits, original

0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7
8	8	8	8	8	8	8
9	9	9	9	9	9	9

As a vector space, these contain  
the same information

# Real data usually contains structure

Handwritten digits, permuted



As a vector space, these contain  
the same information

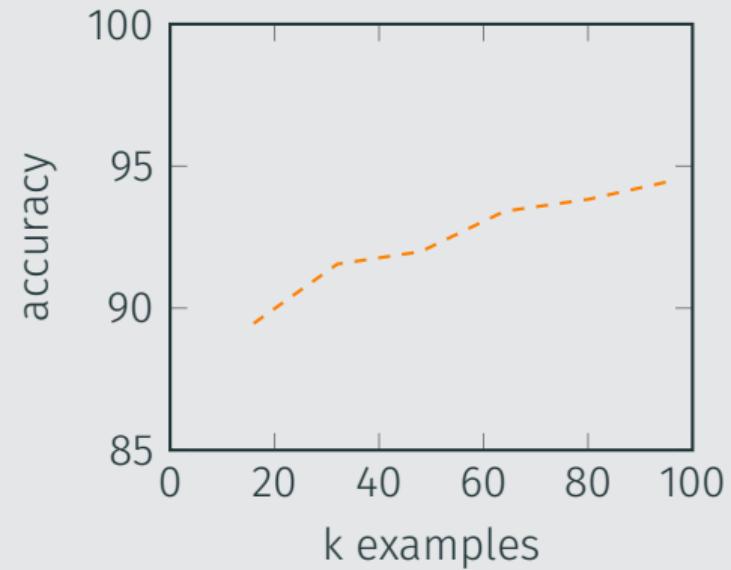
# Real data usually contains structure

Handwritten digits, permuted



As a vector space, these contain  
the same information

Structure enables sample efficiency



MLP (permutation invariant)

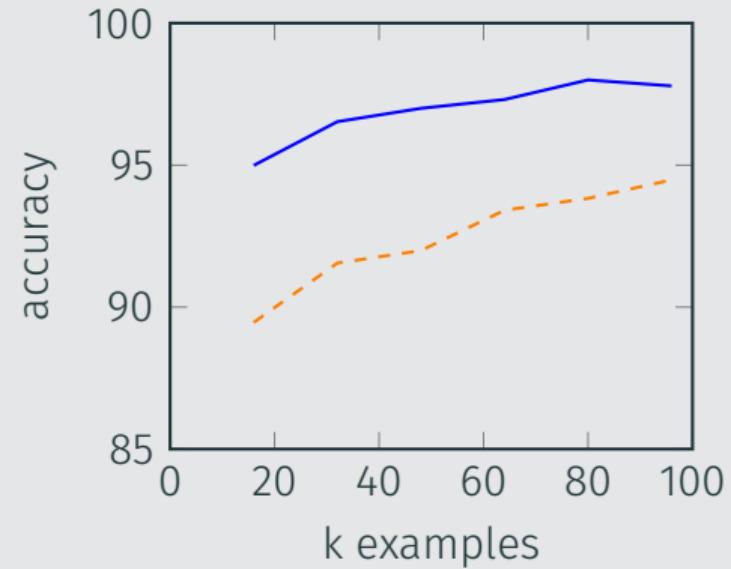
# Real data usually contains structure

Handwritten digits, permuted



As a vector space, these contain  
the same information

Structure enables sample efficiency



CNN (exploits structure)

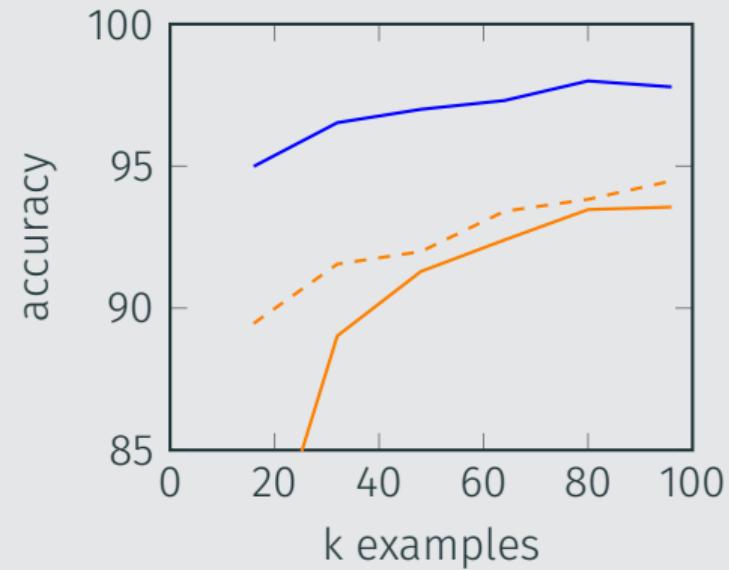
# Real data usually contains structure

Handwritten digits, permuted



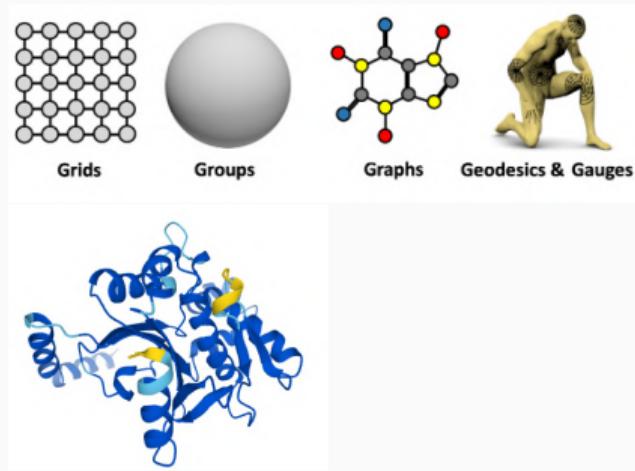
As a vector space, these contain  
the same information

Structure enables sample efficiency



CNN (permuted images)

# Examples of structured data



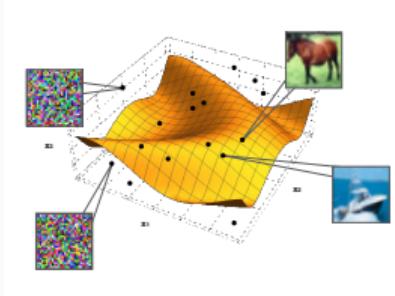
Architecture	Domain	Symmetry
CNN	Grid	Translation
Spherical CNN	Sphere / $SO(3)$	Rotation $SO(3)$
Mesh CNN	Manifold	Gauge Symmetry
GNN	Graph	Permutation
Deep Sets	Set	Permutation
Transformer	Complete graph	Permutation
LSTM	1D Grid	Time translation

## Part II: Statistical Learning Theory

---

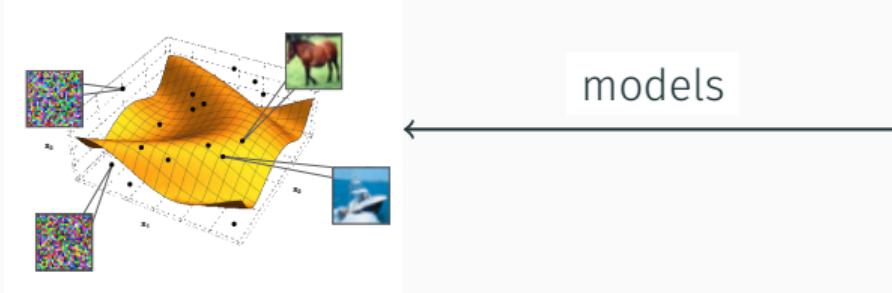
# Key Components of Statistical Learning

## Data distribution



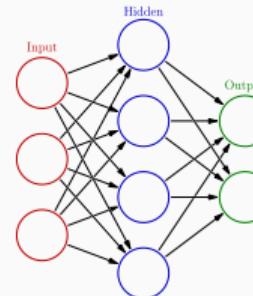
# Key Components of Statistical Learning

Data distribution



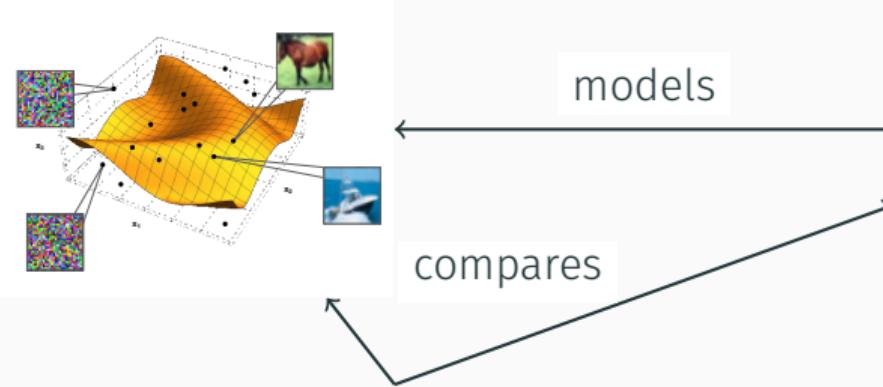
models

Approximation model

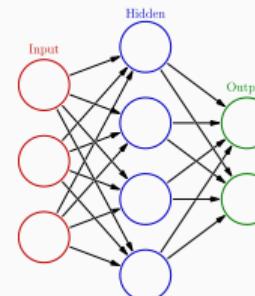


# Key Components of Statistical Learning

Data distribution



Approximation model



Error Metric

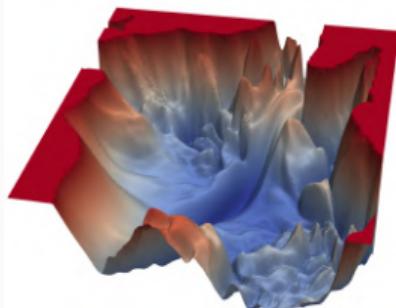


Image: Li et al. (2018)

# Key Components of Statistical Learning

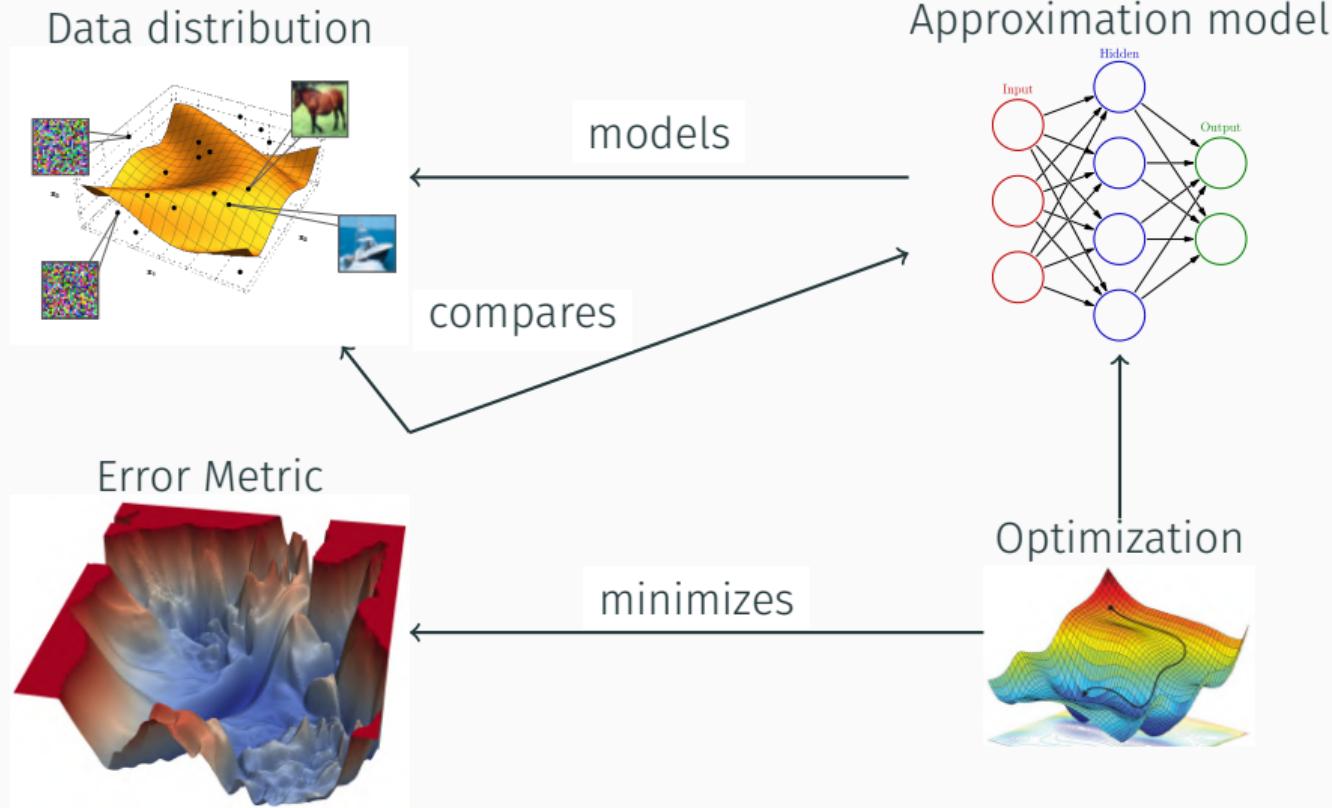


Image: Amini et al. (2019)

# Data distribution

## Definitions

- data point/instance  $x_i \in \mathcal{X} = \mathbb{R}^d$
- label  $y_i \in \mathbb{R}$  (regression),  $y_i \in [C]$  (classification)
- data set:  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$
- data distribution:  $x_i \sim_{\text{i.i.d.}} \mathbb{P}$
- ground truth:  $y_i = f^*(x_i)$

## Example

- $x_i = \boxed{2}, y_i = 2$
- $\mathcal{D}$ : 50000 image-label pairs
- $\mathbb{P}$  distribution of hand-drawn images;  $\mathbb{P}(\boxed{2}) > \mathbb{P}(\boxed{3})$
- $f^*(\boxed{2}) = 2$

# Error Measure

## Definitions

- loss  $\ell : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}_{\geq 0}$
- risk  $\mathcal{R}[f] = \mathbb{E}[\ell(f(X), f^*(X))]$
- empirical risk

$$\hat{\mathcal{R}}[f] = n^{-1} \sum_i [\ell(f(x_i), y_i)]$$

## Examples

- Squared error, absolute error (regression)
- cross-entropy (classification)

# Error Measure

## Definitions

- loss  $\ell : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}_{\geq 0}$
- risk  $\mathcal{R}[f] = \mathbb{E}[\ell(f(X), f^*(X))]$
- empirical risk  
$$\hat{\mathcal{R}}[f] = n^{-1} \sum_i [\ell(f(x_i), y_i)]$$

## Examples

- Squared error, absolute error (regression)
- cross-entropy (classification)

Task: minimize  $\mathcal{R}[f]$  with access only to  $\hat{\mathcal{R}}[f]$

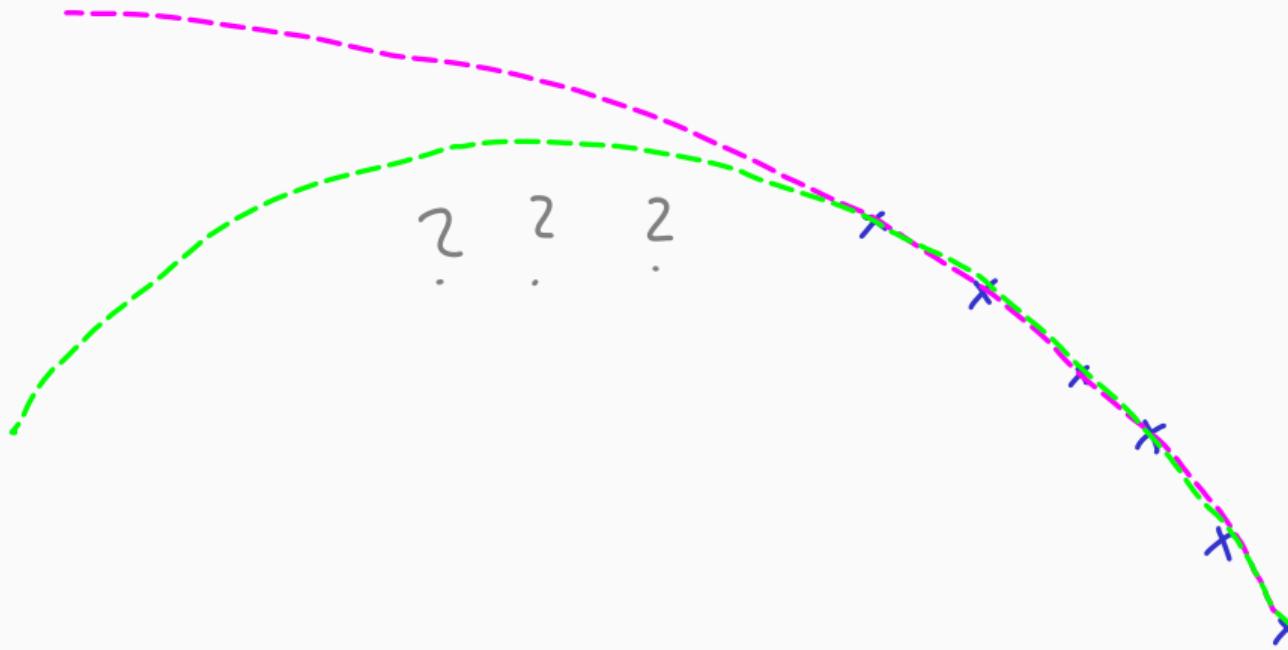
We need to make some concessions

- \* training instances



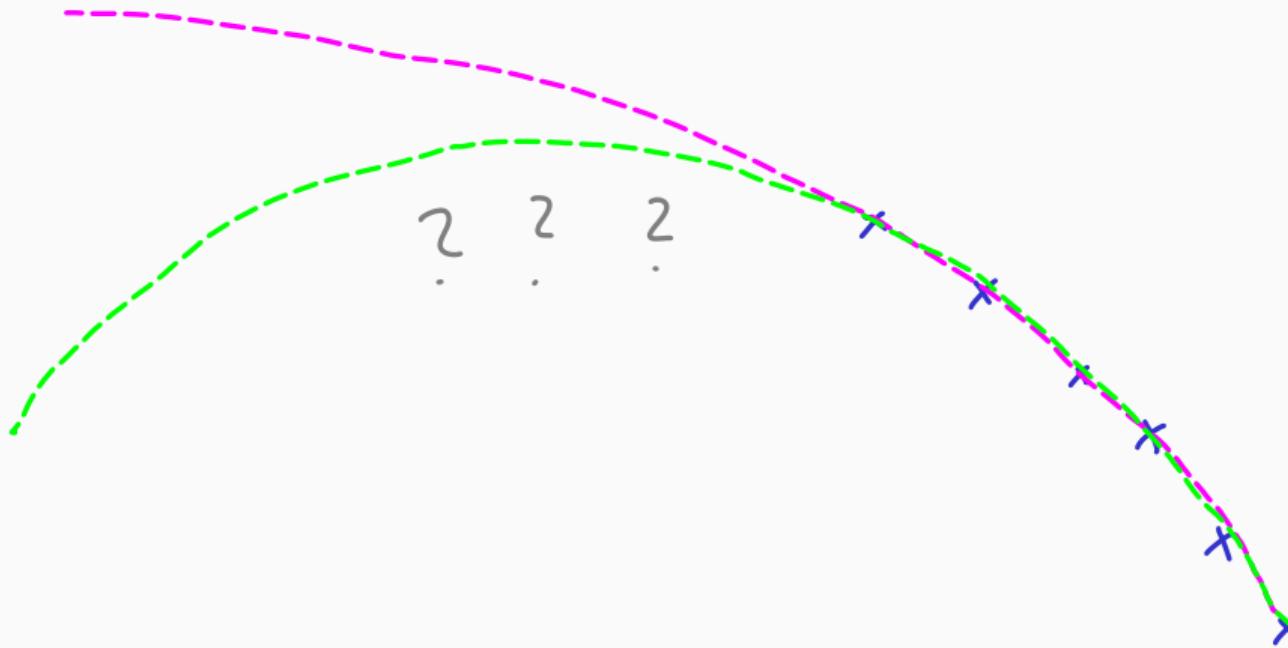
We need to make some concessions

\* training instances



We need to make some concessions

\* training instances



We might be unlucky with the i.i.d. sample we draw

# Most of the time, we should be mostly correct

## Probably Approximately Correct (PAC) Learning

An algorithm  $A$  is a PCA-learner if there exists a function  $m(\epsilon, \delta)$ , such that for every  $\epsilon, \delta \in (0, 1)$  and every distribution  $\mathbb{P}$ , when running the learning algorithm on a i.i.d. sample  $S$  of size  $m(\epsilon, \delta)$ , with probability at least  $\delta$  we have

$$\mathcal{R}[A(S)] \leq \epsilon.$$

# Most of the time, we should be mostly correct

## Probably Approximately Correct (PAC) Learning

An algorithm  $A$  is a PCA-learner if there exists a function  $m(\epsilon, \delta)$ , such that for every  $\epsilon, \delta \in (0, 1)$  and every distribution  $\mathbb{P}$ , when running the learning algorithm on a i.i.d. sample  $S$  of size  $m(\epsilon, \delta)$ , with probability at least  $\delta$  we have

$$\mathcal{R}[A(S)] \leq \epsilon.$$

*Probably* (with probability  $\delta$ ) we are *approximately* (with tolerance  $\epsilon$ ) correct.

**Can we achieve that, at least?**

Minimizing empirical risk might not minimize population risk

- \* training instances



Minimizing empirical risk might not minimize population risk

\* training instances

\*:  $\hat{R} = 6/7$



x  
x

x  
x

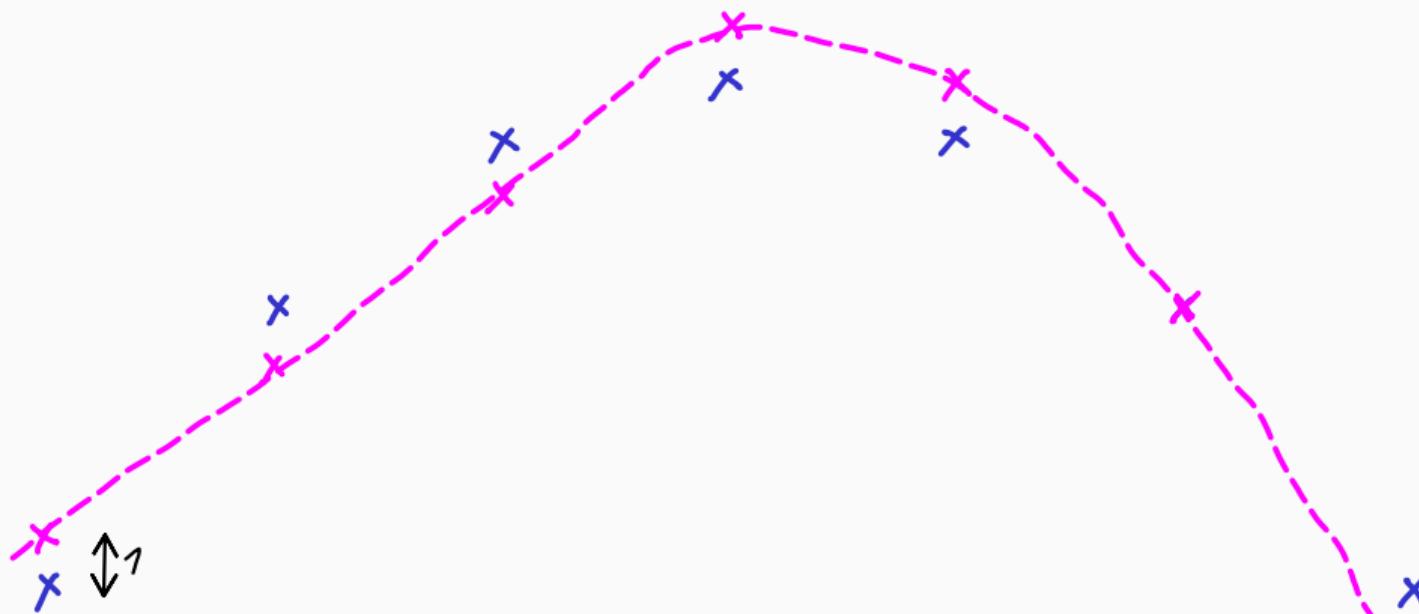
x

x

Minimizing empirical risk might not minimize population risk

\* training instances

\*:  $\hat{R} = 6/7 \quad R \approx 1$

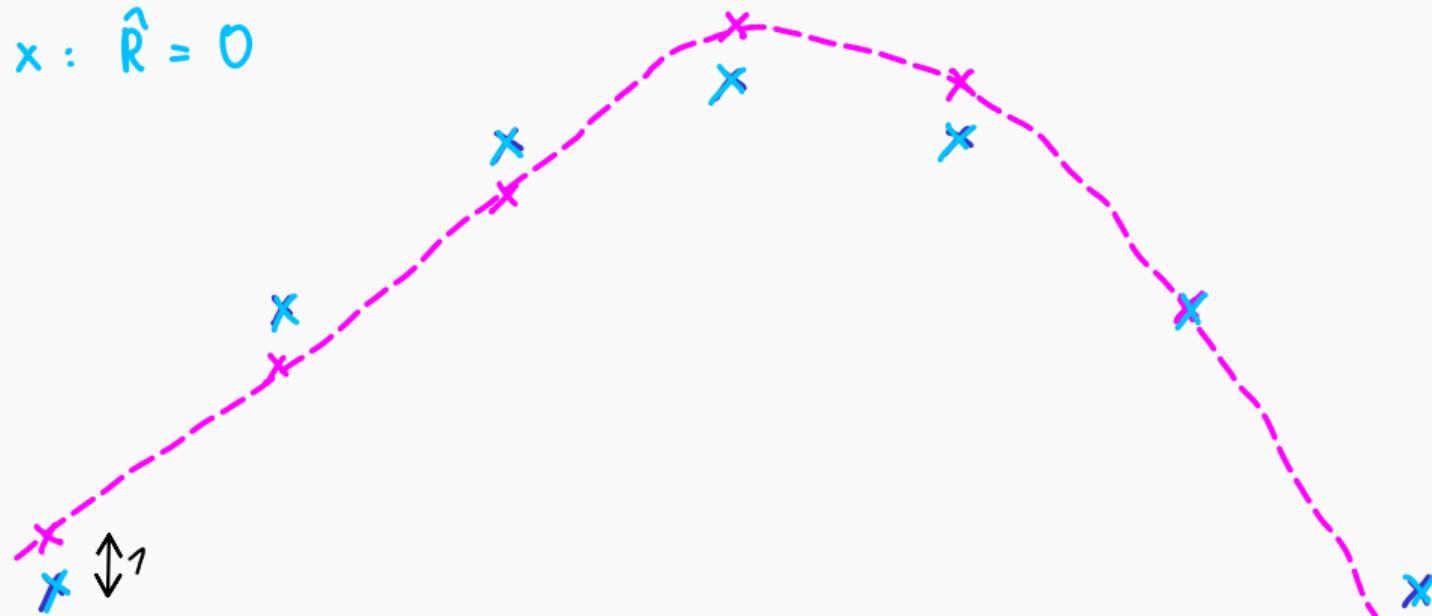


## Minimizing empirical risk might not minimize population risk

\* training instances

\*:  $\hat{R} = 6/7 \quad R \approx 1$

\*:  $\hat{R} = 0$



## Minimizing empirical risk might not minimize population risk

\* training instances

\*:  $\hat{R} = 6/7 \quad R \approx 1$

\*:  $\hat{R} = 0 \quad R \gg 1$



## We need to make some concessions

### No free lunch theorem

Let  $A$  be any learning algorithm for the task of binary classification with 0-1 loss over a domain  $\mathcal{X}$ . Let  $m$  be any number smaller than  $|\mathcal{X}|/2$ , representing a training set size. Then, there exists a distribution  $\mathbb{P}$  over  $\mathcal{X} \times \{0, 1\}$  such that

- There exists a function  $f : \mathcal{X} \rightarrow \{0, 1\}$  with  $\mathcal{R}[f] = 0$
- With probability of at least  $1/7$  over the choice of training set  $\mathcal{S} \sim \mathbb{P}^m$  we have  $\mathcal{R}[A(\mathcal{S})] \geq 1/8$

## We need to make some concessions

### No free lunch theorem

Let  $A$  be any learning algorithm for the task of binary classification with 0-1 loss over a domain  $\mathcal{X}$ . Let  $m$  be any number smaller than  $|\mathcal{X}|/2$ , representing a training set size. Then, there exists a distribution  $\mathbb{P}$  over  $\mathcal{X} \times \{0, 1\}$  such that

- There exists a function  $f : \mathcal{X} \rightarrow \{0, 1\}$  with  $\mathcal{R}[f] = 0$
- With probability of at least  $1/7$  over the choice of training set  $\mathcal{S} \sim \mathbb{P}^m$  we have  $\mathcal{R}[A(\mathcal{S})] \geq 1/8$

⇒ We need to restrict the set of admissible functions

## We need to make some concessions

### No free lunch theorem

Let  $A$  be any learning algorithm for the task of binary classification with 0-1 loss over a domain  $\mathcal{X}$ . Let  $m$  be any number smaller than  $|\mathcal{X}|/2$ , representing a training set size. Then, there exists a distribution  $\mathbb{P}$  over  $\mathcal{X} \times \{0, 1\}$  such that

- There exists a function  $f : \mathcal{X} \rightarrow \{0, 1\}$  with  $\mathcal{R}[f] = 0$
- With probability of at least  $1/7$  over the choice of training set  $\mathcal{S} \sim \mathbb{P}^m$  we have  $\mathcal{R}[A(\mathcal{S})] \geq 1/8$

⇒ We need to restrict the set of admissible functions  
⇒ approximation model

# Approximation model and complexity measure

## Hypothesis class

The model (or hypothesis) class is a subset  $\mathcal{F} \subset \{f : \mathcal{X} \rightarrow \mathbb{R}\}$

## Examples

- Polynomials up to degree  $k$
- Neural networks of a given architecture

# Approximation model and complexity measure

## Hypothesis class

The model (or hypothesis) class is a subset  $\mathcal{F} \subset \{f : \mathcal{X} \rightarrow \mathbb{R}\}$

## Examples

- Polynomials up to degree  $k$
- Neural networks of a given architecture

## Complexity measure

A non-negative mapping  $\gamma : \mathcal{F} \rightarrow \mathbb{R}_{\geq 0}$  that captures how “complex” a hypothesis is.

## Examples

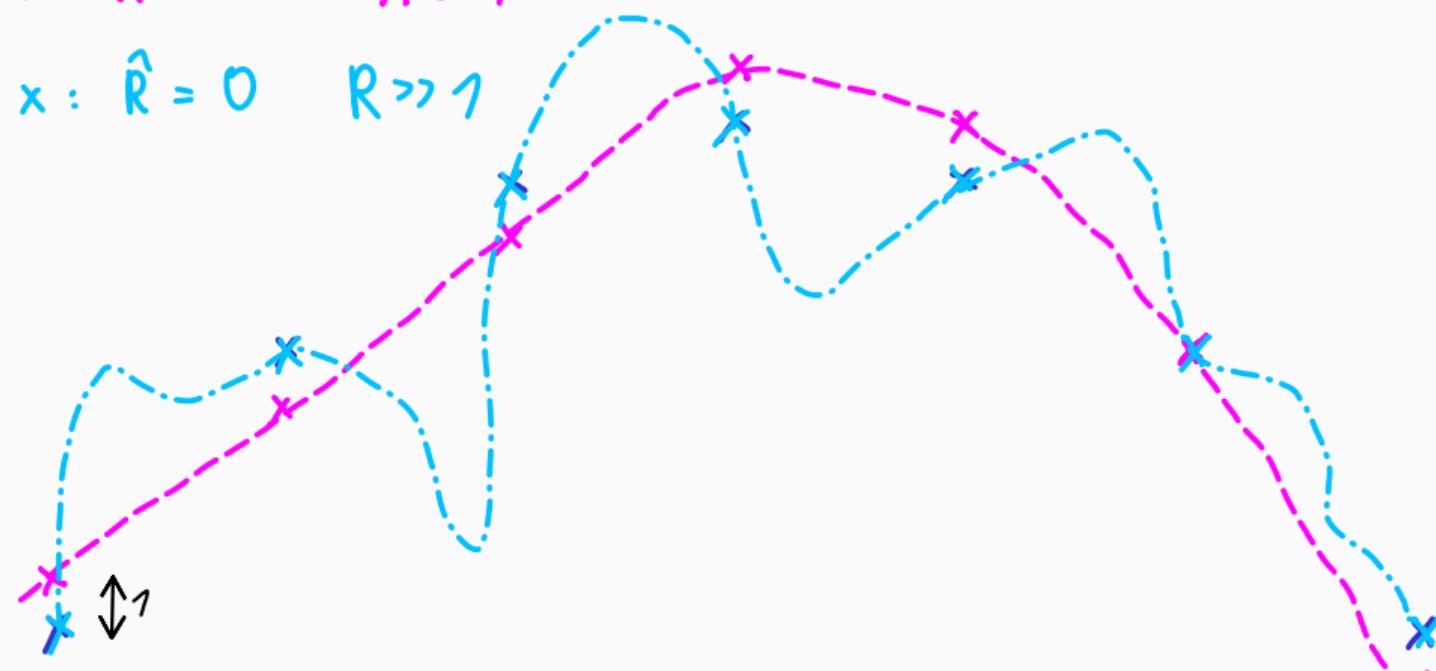
- Degree of polynomial
- Number of neurons in network

Less complex functions less likely to overfit

\* training instances

\*:  $\hat{R} = 6/7 \quad R \approx 1$

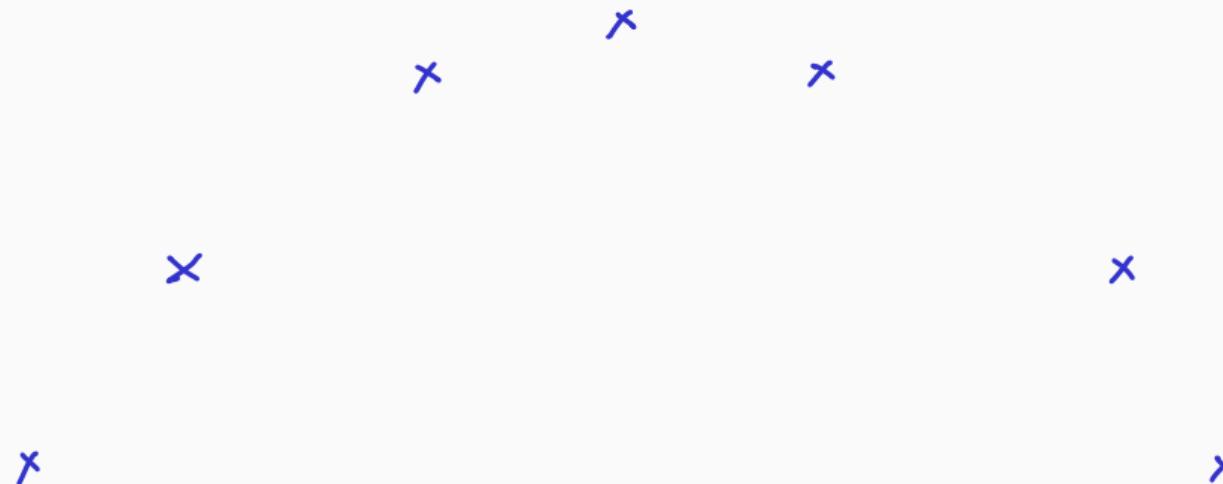
\*:  $\hat{R} = 0 \quad R \gg 1$



Now we get to make a trade-off

Restrict to simple hypothesis  $\mathcal{F}_\beta := \{f \in \mathcal{F} : \gamma(f) \leq \beta\}$ :

- \* training instances

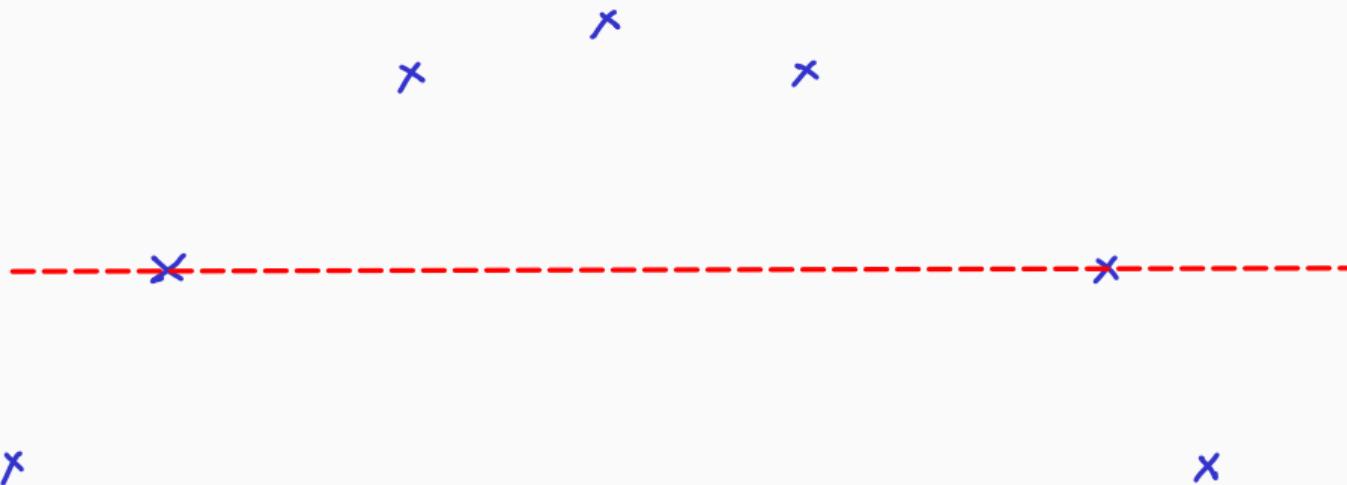


Now we get to make a trade-off

Restrict to simple hypothesis  $\mathcal{F}_\beta := \{f \in \mathcal{F} : \gamma(f) \leq \beta\}$ :

- \* training instances

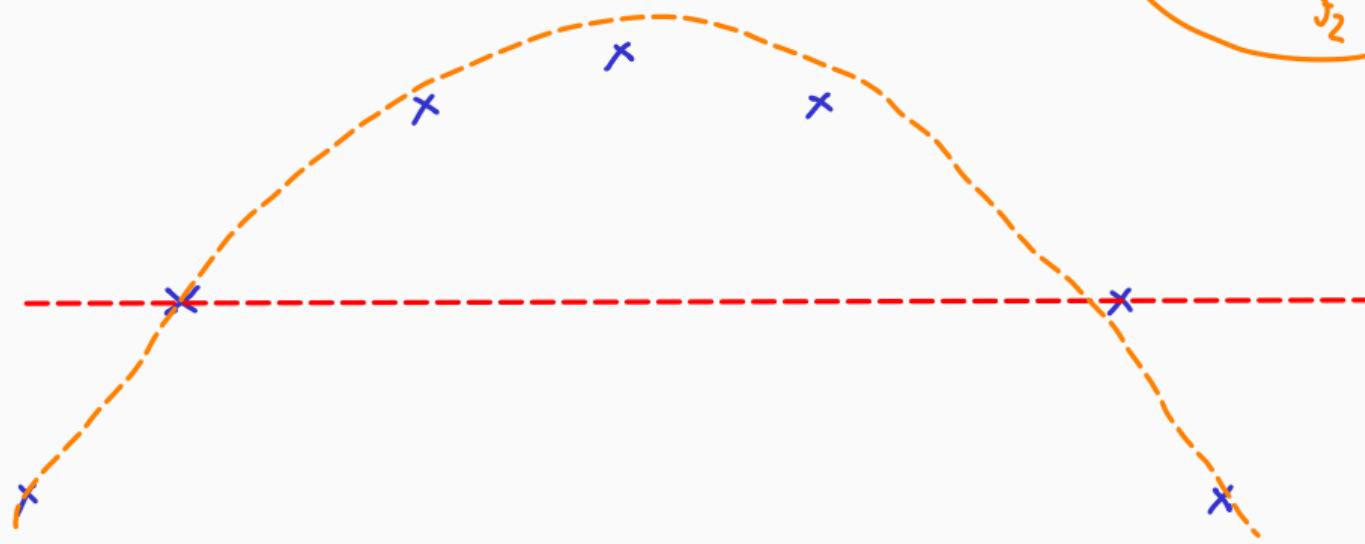
$$\mathcal{F}_1 = \text{--}$$



Now we get to make a trade-off

Restrict to simple hypothesis  $\mathcal{F}_\beta := \{f \in \mathcal{F} : \gamma(f) \leq \beta\}$ :

- \* training instances



We can generalize this example to a generic error decomposition

$$\begin{aligned}\mathcal{R}[\hat{f}] - \inf_{f \in \mathcal{F}} \mathcal{R}[f] &= \left( \mathcal{R}[\hat{f}] - \inf_{f \in \mathcal{F}_\delta} \mathcal{R}[f] \right) + \left( \inf_{f \in \mathcal{F}_\delta} \mathcal{R}[f] - \inf_{f \in \mathcal{F}} \mathcal{R}[f] \right) \\ &= \left( \hat{\mathcal{R}}[\hat{f}] - \inf_{f \in \mathcal{F}_\delta} \hat{\mathcal{R}}[f] \right) + \left( \inf_{f \in \mathcal{F}_\delta} \hat{\mathcal{R}}[f] - \inf_{f \in \mathcal{F}_\delta} \mathcal{R}[f] \right) + \left( \mathcal{R}[\hat{f}] - \hat{\mathcal{R}}[\hat{f}] \right) + \epsilon_{\text{approx}} \\ &\leq \epsilon_{\text{opt}} + 2 \sup_{f \in \mathcal{F}_\delta} |\hat{\mathcal{R}}[f] - \mathcal{R}[f]| + \epsilon_{\text{approx}} \\ &= \epsilon_{\text{opt}} + \epsilon_{\text{stat}} + \epsilon_{\text{approx}}\end{aligned}$$

## Controlling sources of error simultaneously is challenging

- In practice, SGD seems to be successfull at minimizing  $\epsilon_{\text{opt}}$
- Small hypothesis class  $\mathcal{F}_\delta$ :  $\downarrow \epsilon_{\text{stat}}, \uparrow \epsilon_{\text{approx}}$
- Large hypothesis class  $\mathcal{F}_\delta$ :  $\uparrow \epsilon_{\text{stat}}, \downarrow \epsilon_{\text{approx}}$

## Lipschitzness is too weak – Staticical curse of dimensionality

### Lipschitz

A function  $f$  is  $L$ -Lipschitz if  $\forall x_1, x_2 \in \mathcal{X}$ :

$$\|f(x_1) - f(x_2)\| \leq L \cdot \|x_1 - x_2\|.$$

⇒ limits rate of variation of the function

### Curse of dimensionality

How many training points do we need to learn a 1-Lipschitz function on a  $d$ -dimensional hypercube?

## Barron functions too strong – Approximation curse of dimensionality

A function  $f$  with Fourier-transform  $\hat{f}$  is in the Barron-class, if

$$\int \hat{f}(\omega) \|\omega\|_2^2 d\omega < \infty.$$

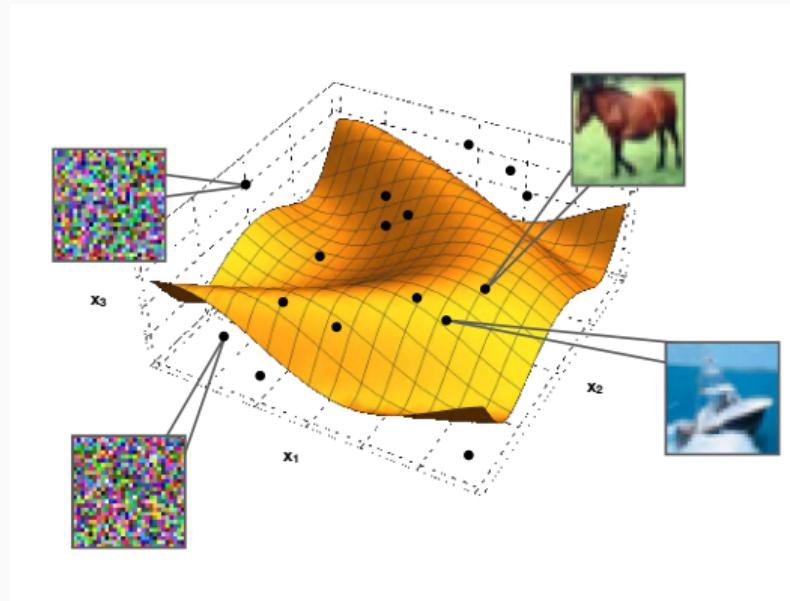
## Barron functions too strong – Approximation curse of dimensionality

A function  $f$  with Fourier-transform  $\hat{f}$  is in the Barron-class, if

$$\int \hat{f}(\omega) \|\omega\|_2^2 d\omega < \infty.$$

⇒ High frequency components need to decay faster than  $\|\omega\|^{-3}$ ; functions need to be very smooth.

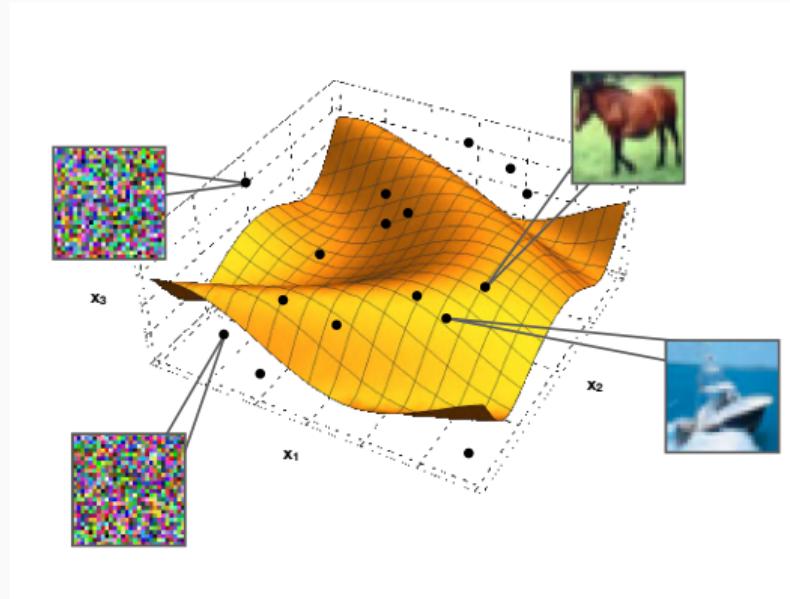
The input data distribution is assumed to have low-dimensional structure *embedded* in a high-dimensional space



---

Goldt et al. (2020). "Modeling the Influence of Data Structure on Learning in Neural Networks: The Hidden Manifold Model"

The input data distribution is assumed to have low-dimensional structure *embedded* in a high-dimensional space



Can we exploit that?

---

Goldt et al. (2020). "Modeling the Influence of Data Structure on Learning in Neural Networks: The Hidden Manifold Model"

## Part III: Signals over geometric domains

---

## Geometric Domains

MNIST image:  $x \in \mathbb{R}^{28 \times 28}$  as a vector. Defined on  
a grid of  $28 \times 28$  pixels.

⇒ Mapping from pixel to intensity.

## Geometric Domains

MNIST image:  $x \in \mathbb{R}^{28 \times 28}$  as a vector. Defined on a grid of  $28 \times 28$  pixels.

⇒ Mapping from pixel to intensity.

### Signal

Given some domain  $\Omega$ , a signal is a mapping

$$x: \Omega \longrightarrow \mathcal{C}$$

from domain locations to  $c$ -dimensional vectors (*channels*) in vector space  $\mathcal{C}$ .

The space of all signals is  $\mathcal{X}(\Omega, \mathcal{C})$ .

# Geometric Domains

MNIST image:  $x \in \mathbb{R}^{28 \times 28}$  as a vector. Defined on a grid of  $28 \times 28$  pixels.  
⇒ Mapping from pixel to intensity.

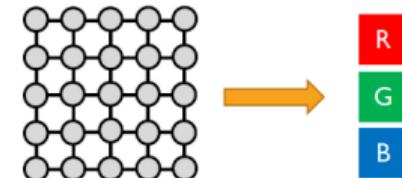
## Signal

Given some domain  $\Omega$ , a signal is a mapping

$$x: \Omega \longrightarrow \mathcal{C}$$

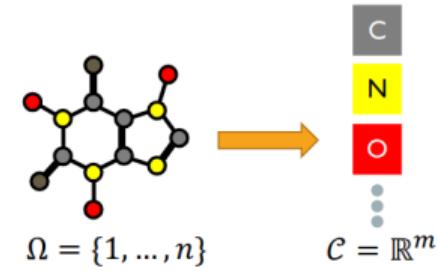
from domain locations to  $c$ -dimensional vectors (*channels*) in vector space  $\mathcal{C}$ .

The space of all signals is  $\mathcal{X}(\Omega, \mathcal{C})$ .



$$\Omega = \mathbb{Z}_n \times \mathbb{Z}_n \quad \mathcal{C} = \mathbb{R}^3$$

Example:  $n \times n$  RGB image



$$\Omega = \{1, \dots, n\} \quad \mathcal{C} = \mathbb{R}^m$$

Example: molecular graph

We can imbue signals with a Hilbert-space structure

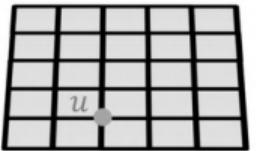
### Addition and scalar multiplication

For two signals  $x$  and  $y$  on  $\Omega$ ,  $\alpha, \beta \in \mathbb{R}$ , define  $\alpha x + \beta y$  through

$$(\alpha x + \beta y)(\omega) := \alpha x(\omega) + \beta y(\omega) \quad \forall \omega \in \Omega.$$

⇒ vector space

signals  $\mathcal{X}(\Omega)$



domain  $\Omega$

# We can imbue signals with a Hilbert-space structure

## Addition and scalar multiplication

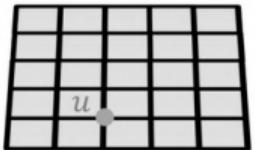
For two signals  $x$  and  $y$  on  $\Omega$ ,  $\alpha, \beta \in \mathbb{R}$ , define  $\alpha x + \beta y$  through

$$(\alpha x + \beta y)(\omega) := \alpha x(\omega) + \beta y(\omega) \quad \forall \omega \in \Omega.$$

$\implies$  vector space

$$\begin{matrix} \text{ant} \\ \text{bee} \end{matrix} + \begin{matrix} \text{bee} \end{matrix} = \begin{matrix} \text{ant} \\ \text{bee} \end{matrix}$$

signals  $\mathcal{X}(\Omega)$



domain  $\Omega$

# We can imbue signals with a Hilbert-space structure

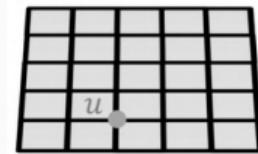
## Inner product

With an inner product  $\langle \cdot, \cdot \rangle_{\mathcal{C}}$  on  $\mathcal{C}$ , and a measure  $\mu$  on  $\Omega$ , define inner product on  $\mathcal{X}(\Omega, \mathcal{C})$ :

$$\langle x, y \rangle := \int_{\Omega} \langle x(\omega), y(\omega) \rangle_{\mathcal{C}} d\mu(\omega).$$

⇒ Hilbert space

signals  $\mathcal{X}(\Omega)$



domain  $\Omega$

# We can imbue signals with a Hilbert-space structure

## Inner product

With an inner product  $\langle \cdot, \cdot \rangle_{\mathcal{C}}$  on  $\mathcal{C}$ , and a measure  $\mu$  on  $\Omega$ , define inner product on  $\mathcal{X}(\Omega, \mathcal{C})$ :

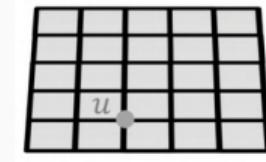
$$\langle x, y \rangle := \int_{\Omega} \langle x(\omega), y(\omega) \rangle_{\mathcal{C}} d\mu(\omega).$$

⇒ Hilbert space

Example: MNIST - single channel, counting measure

$$\langle x, y \rangle = \sum_{i=1}^{28} \sum_{j=1}^{28} x[i, j] \cdot y[i, j]$$

signals  $\mathcal{X}(\Omega)$



domain  $\Omega$

# Symmetries

A transformation of an object that leaves the object unchanged.

# Symmetries

A transformation of an object that leaves the object unchanged.

## Symmetry of the label function

Recall:  $f^* : \mathcal{X} \rightarrow \mathcal{Y}$  ground-truth label function.

A transformation  $g : \mathcal{X} \rightarrow \mathcal{X}$  is a symmetry of the label function, if  $f^* \equiv f^* \circ g$ .

# Symmetries

A transformation of an object that leaves the object unchanged.

## Symmetry of the label function

Recall:  $f^* : \mathcal{X} \rightarrow \mathcal{Y}$  ground-truth label function.

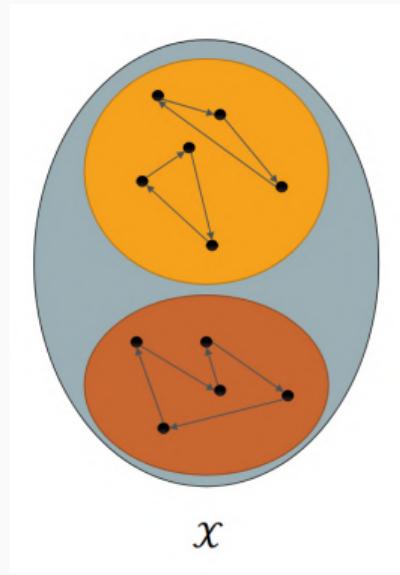
A transformation  $g : \mathcal{X} \rightarrow \mathcal{X}$  is a symmetry of the label function, if  $f^* \equiv f^* \circ g$ .

Example: Horizontal flip

$$f^*\left(\begin{array}{c} \text{Image of a dog standing on grass} \end{array}\right) = f^*\left(\begin{array}{c} \text{Image of a dog standing on grass, horizontally flipped} \end{array}\right) = \text{"dog".}$$

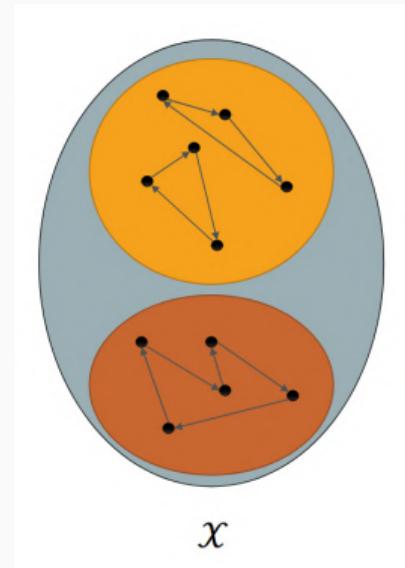
## Symmetries of the label function

- Any bijective function that respects class boundaries is a symmetry of the label function
- $\Rightarrow$  if we knew all symmetries, a *single* instance per class would be enough to learn.
- If we know some symmetries, less training data is needed.



## Symmetries of the label function

- Any bijective function that respects class boundaries is a symmetry of the label function
- $\Rightarrow$  if we knew all symmetries, a *single* instance per class would be enough to learn.
- If we know some symmetries, less training data is needed.
- Can exploit symmetries of the underlying domain.

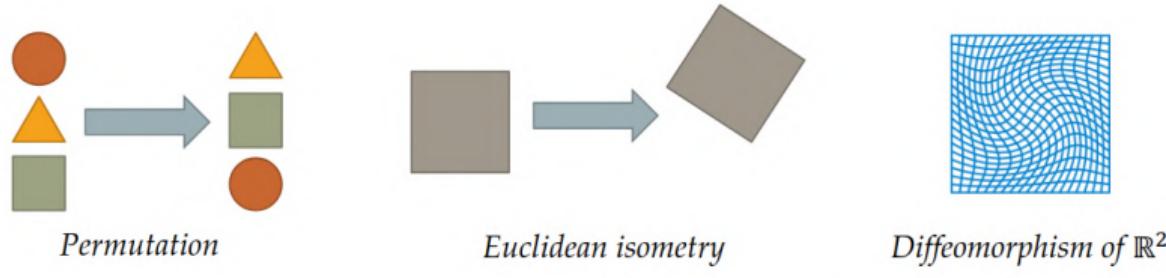


## Symmetries of geometric domains

- A transformation  $g : \Omega \longrightarrow \Omega$  is a symmetry of the domain  $\Omega$  if it preserves its structure.

# Symmetries of geometric domains

- A transformation  $g : \Omega \longrightarrow \Omega$  is a symmetry of the domain  $\Omega$  if it preserves its structure.
- Permutation of elements in a set preserves set membership
- Euclidian isometries (rotation, translation, reflection) preserve angles and distances in Euclidian spaces ( $\mathbb{R}^d$ )
- Diffeomorphism preserves manifold structure



We can lift symmetries on the domain to symmetries on signals

Given a symmetry  $g : \Omega \longrightarrow \Omega$ , we can define a symmetry  $\tilde{g}$  on  $\mathcal{X}(\Omega, \mathcal{C})$  through:

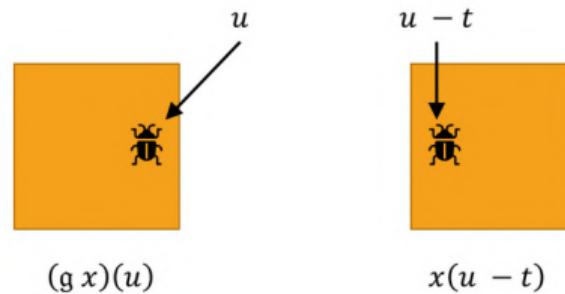
$$\tilde{g}(x)(\omega) = x(g^{-1}(\omega)).$$

We can lift symmetries on the domain to symmetries on signals

Given a symmetry  $g : \Omega \longrightarrow \Omega$ , we can define a symmetry  $\tilde{g}$  on  $\mathcal{X}(\Omega, \mathcal{C})$  through:

$$\tilde{g}(x)(\omega) = x(g^{-1}(\omega)) .$$

$g = (t_x, t_y)$ , a translation



The mathematical theory of symmetries is **group theory**

## Part IV: Group Theory

---

# What is a group?

## Definition

A collection of *abstract* transformations

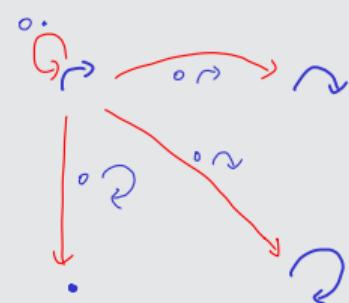
$G = (\{u, v, w, \dots\}, \circ)$  that can be combined with  
a “multiplication”  $\circ$ :

**Closedness:**  $u \circ v \in G$

**Associativity:**  $u \circ (v \circ w) = (u \circ v) \circ w$

**Neutral Element:**  $\exists e \in G : eu = ue = u.$

**Inverse:**  $\forall u \in G : \exists u^{-1} : u^{-1}u = e$



# What is a group?

## Definition

A collection of *abstract* transformations

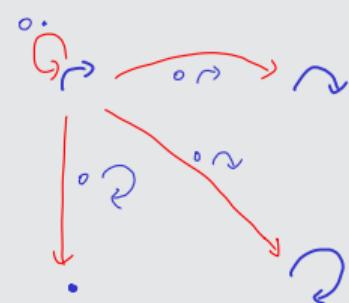
$G = (\{u, v, w, \dots\}, \circ)$  that can be combined with  
a “multiplication”  $\circ$ :

**Closedness:**  $u \circ v \in G$

**Associativity:**  $u \circ (v \circ w) = (u \circ v) \circ w$

**Neutral Element:**  $\exists e \in G : eu = ue = u.$

**Inverse:**  $\forall u \in G : \exists u^{-1} : u^{-1}u = e$



## Examples

Natural numbers  $\mathbb{N} = (\{1, 2, \dots\}, +)$

# What is a group?

## Definition

A collection of *abstract* transformations

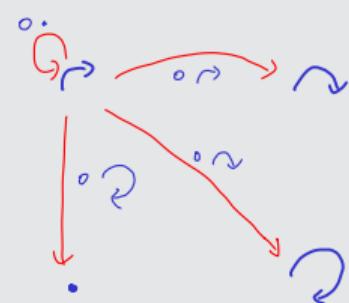
$G = (\{u, v, w, \dots\}, \circ)$  that can be combined with  
a “multiplication”  $\circ$ :

**Closedness:**  $u \circ v \in G$

**Associativity:**  $u \circ (v \circ w) = (u \circ v) \circ w$

**Neutral Element:**  $\exists e \in G : eu = ue = u.$

**Inverse:**  $\forall u \in G : \exists u^{-1} : u^{-1}u = e$



## Examples

Natural numbers  $\mathbb{N} = (\{1, 2, \dots\}, +)$  **No**

# What is a group?

## Definition

A collection of *abstract* transformations

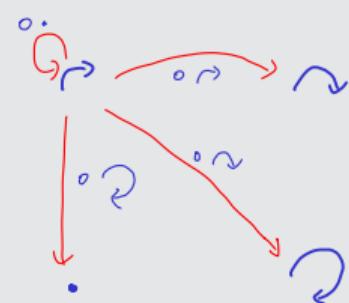
$G = (\{u, v, w, \dots\}, \circ)$  that can be combined with  
a “multiplication”  $\circ$ :

**Closedness:**  $u \circ v \in G$

**Associativity:**  $u \circ (v \circ w) = (u \circ v) \circ w$

**Neutral Element:**  $\exists e \in G : eu = ue = u.$

**Inverse:**  $\forall u \in G : \exists u^{-1} : u^{-1}u = e$



## Examples

Natural numbers  $\mathbb{N} = (\{1, 2, \dots\}, +)$  **No**

Integers  $\mathbb{Z} = (\{\dots, -1, 0, 1, \dots\}, +)$

# What is a group?

## Definition

A collection of *abstract* transformations

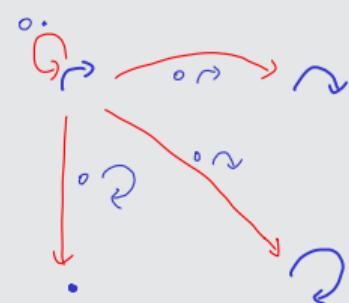
$G = (\{u, v, w, \dots\}, \circ)$  that can be combined with  
a “multiplication”  $\circ$ :

**Closedness:**  $u \circ v \in G$

**Associativity:**  $u \circ (v \circ w) = (u \circ v) \circ w$

**Neutral Element:**  $\exists e \in G : eu = ue = u.$

**Inverse:**  $\forall u \in G : \exists u^{-1} : u^{-1}u = e$



## Examples

Natural numbers  $\mathbb{N} = (\{1, 2, \dots\}, +)$  **No**

Integers  $\mathbb{Z} = (\{\dots, -1, 0, 1, \dots\}, +)$  **Yes**

# What is a group?

## Definition

A collection of *abstract* transformations

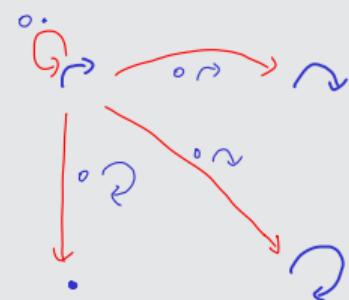
$G = (\{u, v, w, \dots\}, \circ)$  that can be combined with  
a “multiplication”  $\circ$ :

**Closedness:**  $u \circ v \in G$

**Associativity:**  $u \circ (v \circ w) = (u \circ v) \circ w$

**Neutral Element:**  $\exists e \in G : eu = ue = u.$

**Inverse:**  $\forall u \in G : \exists u^{-1} : u^{-1}u = e$



## Examples

Natural numbers  $\mathbb{N} = (\{1, 2, \dots\}, +)$  **No** Nonzero reals  $\mathbb{R}^* = (\mathbb{R} \setminus \{0\}, \cdot)$

Integers  $\mathbb{Z} = (\{\dots, -1, 0, 1, \dots\}, +)$  **Yes**

# What is a group?

## Definition

A collection of *abstract* transformations

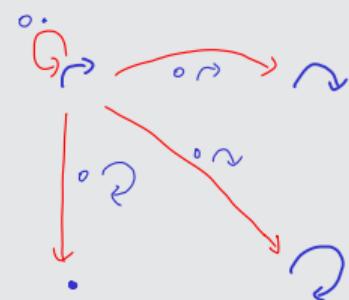
$G = (\{u, v, w, \dots\}, \circ)$  that can be combined with  
a “multiplication”  $\circ$ :

**Closedness:**  $u \circ v \in G$

**Associativity:**  $u \circ (v \circ w) = (u \circ v) \circ w$

**Neutral Element:**  $\exists e \in G : eu = ue = u.$

**Inverse:**  $\forall u \in G : \exists u^{-1} : u^{-1}u = e$



## Examples

Natural numbers  $\mathbb{N} = (\{1, 2, \dots\}, +)$  **No** Nonzero reals  $\mathbb{R}^* = (\mathbb{R} \setminus \{0\}, \cdot)$  **Yes**

Integers  $\mathbb{Z} = (\{\dots, -1, 0, 1, \dots\}, +)$  **Yes**

# What is a group?

## Definition

A collection of *abstract* transformations

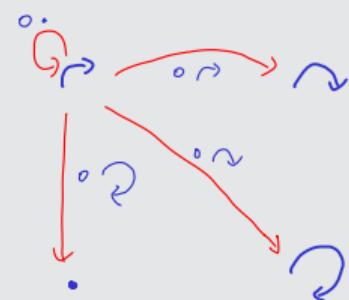
$G = (\{u, v, w, \dots\}, \circ)$  that can be combined with  
a “multiplication”  $\circ$ :

**Closedness:**  $u \circ v \in G$

**Associativity:**  $u \circ (v \circ w) = (u \circ v) \circ w$

**Neutral Element:**  $\exists e \in G : eu = ue = u.$

**Inverse:**  $\forall u \in G : \exists u^{-1} : u^{-1}u = e$



## Examples

Natural numbers  $\mathbb{N} = (\{1, 2, \dots\}, +)$  **No**   Nonzero reals  $\mathbb{R}^* = (\mathbb{R} \setminus \{0\}, \cdot)$  **Yes**

Integers  $\mathbb{Z} = (\{\dots, -1, 0, 1, \dots\}, +)$  **Yes**   Nonzero reals  $\mathbb{R}^* = (\mathbb{R} \setminus \{0\}, +)$

# What is a group?

## Definition

A collection of *abstract* transformations

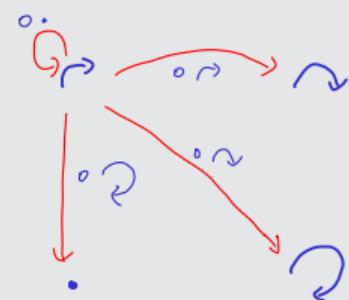
$G = (\{u, v, w, \dots\}, \circ)$  that can be combined with  
a “multiplication”  $\circ$ :

**Closedness:**  $u \circ v \in G$

**Associativity:**  $u \circ (v \circ w) = (u \circ v) \circ w$

**Neutral Element:**  $\exists e \in G : eu = ue = u.$

**Inverse:**  $\forall u \in G : \exists u^{-1} : u^{-1}u = e$



## Examples

Natural numbers  $\mathbb{N} = (\{1, 2, \dots\}, +)$  **No**   Nonzero reals  $\mathbb{R}^* = (\mathbb{R} \setminus \{0\}, \cdot)$  **Yes**

Integers  $\mathbb{Z} = (\{\dots, -1, 0, 1, \dots\}, +)$  **Yes**   Nonzero reals  $\mathbb{R}^* = (\mathbb{R} \setminus \{0\}, +)$  **No**

The group captures the *structure* of how transformations combine

Structurally, these  
are the same group:

$$(\{0\}, +) \equiv (\{1\}, \cdot)$$

The group captures the *structure* of how transformations combine

Structurally, these  
are the same group:  
 $(\{0\}, +) \equiv (\{1\}, \cdot)$

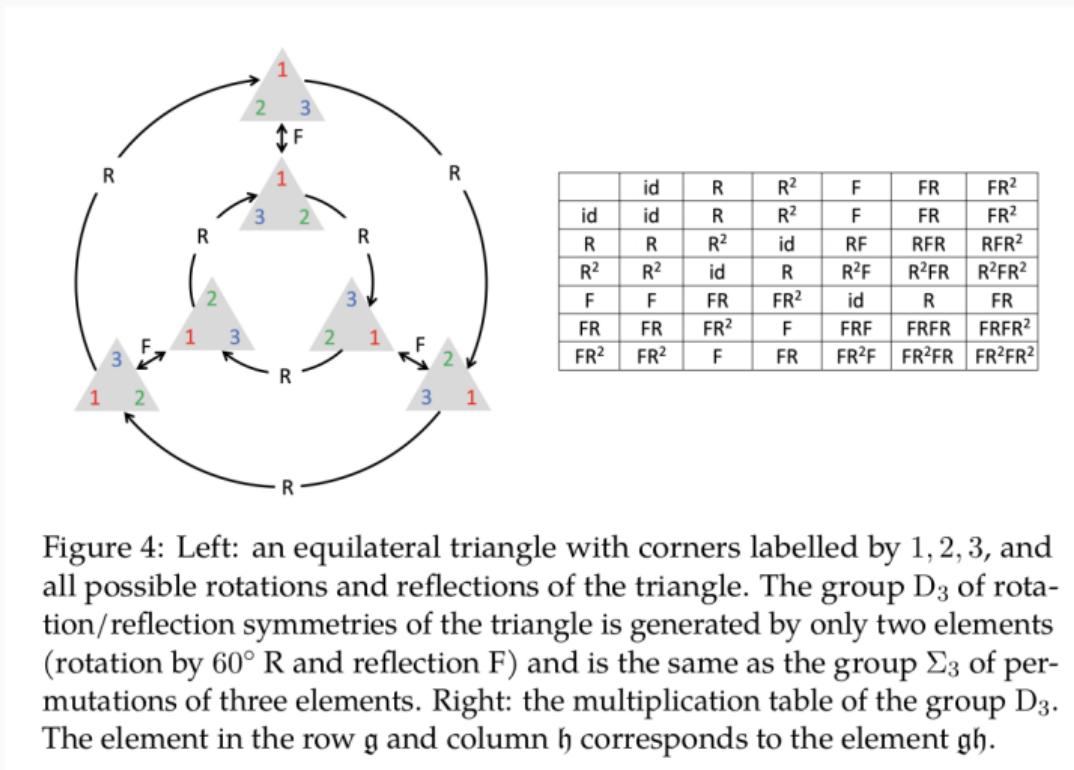
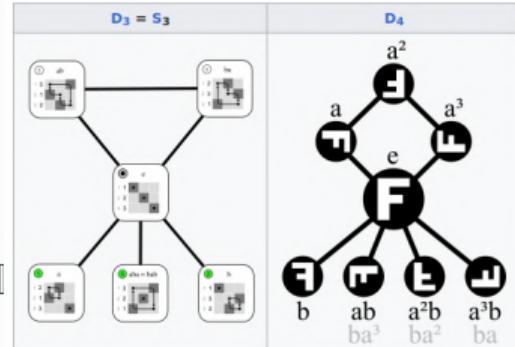


Figure 4: Left: an equilateral triangle with corners labelled by 1, 2, 3, and all possible rotations and reflections of the triangle. The group  $D_3$  of rotation/reflection symmetries of the triangle is generated by only two elements (rotation by  $60^\circ$  R and reflection F) and is the same as the group  $\Sigma_3$  of permutations of three elements. Right: the multiplication table of the group  $D_3$ . The element in the row  $g$  and column  $h$  corresponds to the element  $gh$ .

# Some important groups

$C(n)$	Cyclic group	$x \mapsto x + 1 \bmod n$
$D(n)$	Dihedral	Vertices of a regular $n$ -gon under reflection and rotation
$S(n)$	Permutation	Arbitrary permutation
$SO(n)$	Special ortho.	Rotations in $\mathbb{R}^n$
$O(n)$	Orthogonal	Rotations and reflections in $\mathbb{R}^n$
$GL(n)$	General linear	Invertible linear transforms of $\mathbb{I}$



$$C(n) \subset D(n) \subset S(n)$$

$$SO(n) \subset O(n) \subset GL(n)$$

## Group Actions: Making group elements actually transform something

### Definition

A group action  $\mathcal{A}$  of a group  $G$  on some set  $\Omega$  is a mapping  $G \times \Omega \longrightarrow \Omega$ ,  $(g, \omega) \mapsto g.\omega$ , that is *compatible* with the group structure, that is,

$$(g \circ h).\omega = g.(h.\omega)$$

$$e.\omega = \omega$$

# Group Actions: Making group elements actually transform something

## Definition

A group action  $\mathcal{A}$  of a group  $G$  on some set  $\Omega$  is a mapping  $G \times \Omega \longrightarrow \Omega$ ,  $(g, \omega) \mapsto g.\omega$ , that is *compatible* with the group structure, that is,

$$(g \circ h).\omega = g.(h.\omega)$$

$$e.\omega = \omega$$

## Examples: The same group can act differently

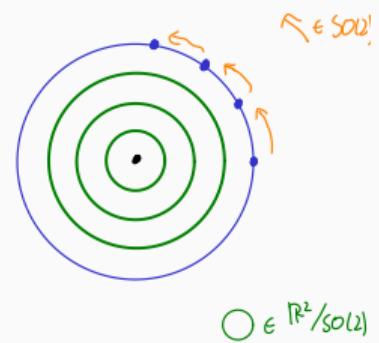
- The group  $(\mathbb{R}, +)$  acting on the set  $\mathbb{R}$ :  $u.v = u + v$
- The group  $(\mathbb{R}, +)$  acting on the vector space  $\mathbb{R}^2$  as a horizontal shift:  
 $u.(x, y) = (x + u, y)$
- The group  $(\mathbb{R}, +)$  acting on the vector space  $\mathbb{R}^2$  as a vertical shift:  
 $u.(x, y) = (x, y + u)$

# Orbits

- The trajectory that can be reached by applying any group element to a fixed starting point
- $G.\omega = \{g.\omega \mid g \in G\}$
- Partitions the space into equivalence classes  
 $[\omega] \in \Omega/G$

$$\Omega = \mathbb{R}^2$$

$$G = SO(2)$$

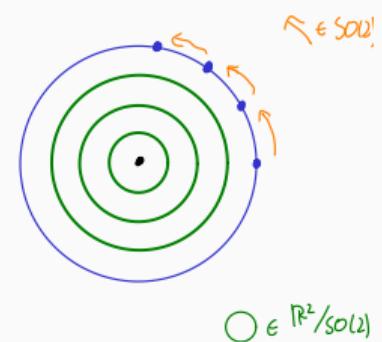


# Orbits

- The trajectory that can be reached by applying any group element to a fixed starting point
- $G.\omega = \{g.\omega \mid g \in G\}$
- Partitions the space into equivalence classes  
 $[\omega] \in \Omega/G$

$$\Omega = \mathbb{R}^2$$

$$G = SO(2)$$



One sample per orbit enough to learn label function

## Summary and Outlook

---

## Today

- Statistical learning theory: Need assumptions to enable successfull learning in high-dimensional spaces
- Inputs as signals on a geometric space: Allows us to exploit symmetries of the underlying structure
- Group theory: Mathematical framework for symmetries

## Next Week

- Sets and Graphs: Permutation group
- Graph neural networks as a first instantiation of the *geometric deep learning blueprint*

## References

---

- [1] Alexander Amini et al. *Spatial Uncertainty Sampling for End-to-End Control*. 2019. arXiv: 1805.04829.
- [2] Sebastian Goldt et al. “Modeling the Influence of Data Structure on Learning in Neural Networks: The Hidden Manifold Model”. In: *Phys. Rev. X* 10 (4 Dec. 2020), p. 041044.
- [3] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *nature* 596.7873 (2021), pp. 583–589.
- [4] Hao Li et al. “Visualizing the Loss Landscape of Neural Nets”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.
- [5] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning -*

# Lecture 2: Graph neural networks

Katsiaryna Haitsiukevich, Alison Pouplin, Çağlar Hızlı, Erik Schultheis,  
Vikas Garg

# Housekeeping

# General

- Course enrollment / confirming participation, [instructions](#)
- GNN exercise is released today.
- **DL for the GNN exercise** (Wed 6.11, 23:00)
- Solution session for the exercise (Thu 7.11)
  - Be prepared to present your solutions!
- Preliminary: the paper list is releases on Friday (1.11)
- **Preliminary**: DL for selecting paper preferences on Tue (11.11)

In the previous episode of GDL...

# Recap of the previous lecture

- Statistical learning theory
  - Assumptions for learning in high-dimensional spaces

# Recap of the previous lecture

- Statistical learning theory
  - Assumptions for learning in high-dimensional spaces
- Perspective on model inputs as signals on geometric domain
  - Exploiting the underlying structure of the data to set assumptions

# Recap of the previous lecture

- Statistical learning theory
  - Assumptions for learning in high-dimensional spaces
- Perspective on model inputs as signals on geometric domain
  - Exploiting the underlying structure of the data to set assumptions
- Group theory
  - Mathematical tool to work with symmetries of the data structure

# Today's lecture

- Sets and Graphs

# Today's lecture

- Sets and Graphs
- Invariance and Equivariance

# Today's lecture

- Sets and Graphs
- Invariance and Equivariance
- Graph Neural Networks (GNNs)

# Today's lecture

- Sets and Graphs
- Invariance and Equivariance
- Graph Neural Networks (GNNs)
- Geometric Deep Learning Blueprint

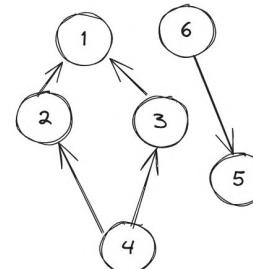
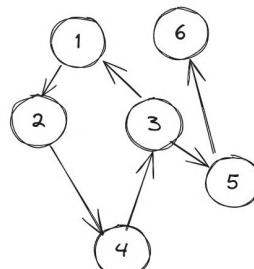
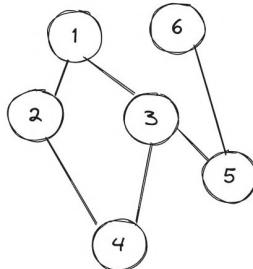
# Graphs: essentials

# What is a graph?

A graph is an unordered tuple of two sets

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

- Set of nodes (vertices)  $V$
- Set of edges 2-tuples (encoding relations)  $E \subseteq (V \times V)$
- And optionally global attributes of a graph can be present



# A special case of a graph: a set

- Graph without edges
- Real world example: points clouds for object representations
- However, not many of the problems can be characterised by sets only
- We need to model connections!

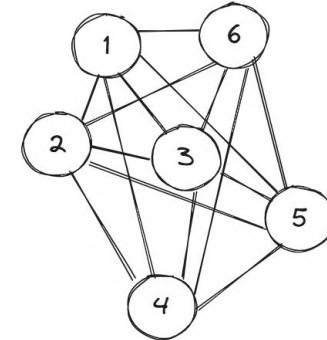
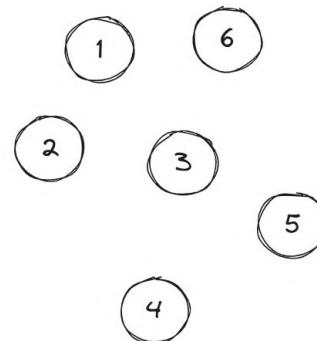
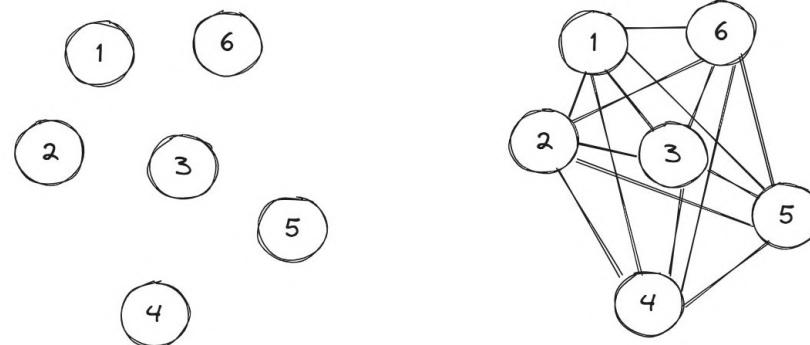


Image source: [Blog post on PointNet++](#)

# A special case of a graph: a fully-connected graph

- Graph where every pair of vertices is connected with an edge
- A synonym: a complete graph
- Real world example: small groups of friends

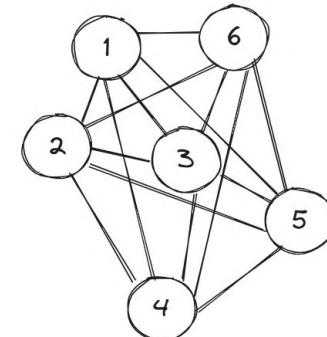
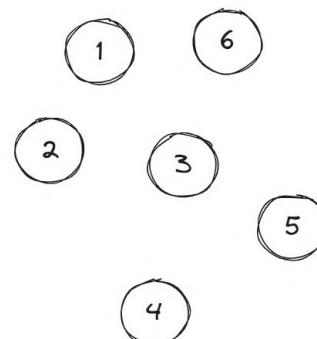


# A special case of a graph: a fully-connected graph

- Graph where every pair of vertices is connected with an edge
- A synonym: a complete graph
- Real world example: small groups of friends

## Question:

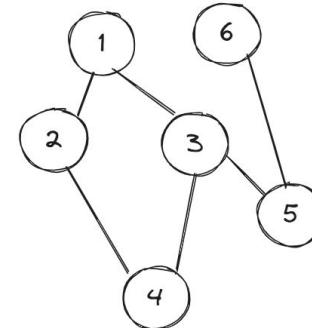
*If the connectivity of a graph  
is unknown, is it the best to  
model it as a set or as a  
fully-connected graph?*



# Adjacency matrix: graph connectivity

- The connectivity information of a graph can be represented as a quadratic matrix called adjacency matrix
- Adjacency matrix consists of 0 and 1
- Value 1 in row  $k$  and column  $m$  corresponds to an edge between vertices  $k$  and  $m$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

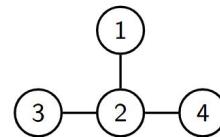


# Graph representations

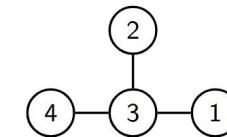
- *How can a graph be represented?*

# Graph representations

- Features of the nodes
- Connectivity information
- Optionally: edges features and global features



$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \\ \mathbf{x}_4^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

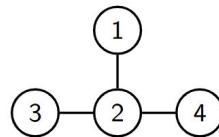


$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_4^\top \\ \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

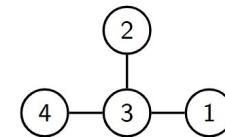
# Graph representations

- Features of the nodes
- Connectivity information
- Optionally: edges features and global features

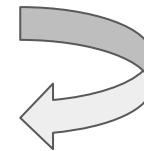
*Two different representations or the same object make modelling tricky*



$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \\ \mathbf{x}_4^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



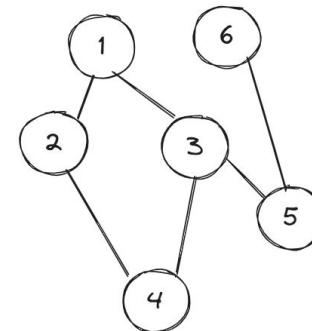
$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_4^\top \\ \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



# Neighborhoods and degree of a node

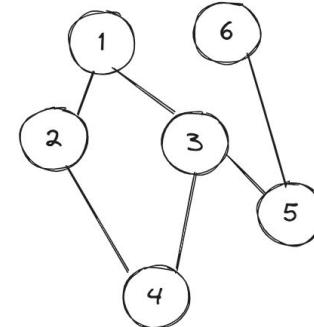
- A neighborhood of a node  
is a set of vertices the node is connected to by an edge

*What is the neighborhood of  
the node number 3?*



# Neighborhoods and degree of a node

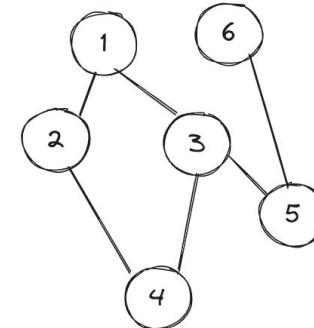
- A neighborhood of a node
  - is a set of vertices the node is connected to by an edge
- A degree of a node
  - is the number of its neighbors



# Neighborhoods and degree of a node

- A neighborhood of a node
  - is a set of vertices the node is connected to by an edge
- A degree of a node
  - is the number of its neighbors

*What is the degree of the  
node number 3?*



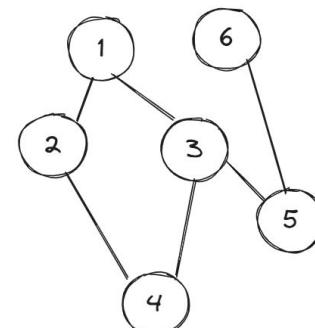
# Laplacian matrix

Laplacian matrix is defined as a difference between

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

- a diagonal degree matrix (*each element is a degree of the corresponding node*) and
- an adjacency matrix

$$\mathbf{D} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$



# Importance of Laplacian matrix

- Laplacian matrix encodes the same information as Adjacency matrix
- Both are used

# Importance of Laplacian matrix

- Laplacian matrix encodes the same information as Adjacency matrix
- Both are used
- Laplacian matrix exhibits desired properties in eigendecomposition
  - The matrix of size  $n \times n$  has  $n$  independent eigenvectors
  - All eigenvalues are real and non-negative
  - The smallest eigenvalue is always zero
  - The eigenvalues indicate connected components in the graph (clusters)

# Importance of Laplacian matrix

- Laplacian matrix encodes the same information as Adjacency matrix
- Both are used
- Laplacian matrix exhibits desired properties in eigendecomposition
  - The matrix of size  $n \times n$  has  $n$  independent eigenvectors
  - All eigenvalues are real and non-negative
  - The smallest eigenvalue is always zero
  - The eigenvalues indicate connected components in the graph (clusters)
- Fourier transform of a graph
  - A linear combination of graph spectral components
  - Utilizes eigenvectors of the Laplacian (similar to sinusoids of a classical Fourier transform)

# Importance of Laplacian matrix

- Laplacian matrix encodes the same information as Adjacency matrix
- Both are used
- Laplacian matrix exhibits desired properties in eigendecomposition
  - The matrix of size  $n \times n$  has  $n$  independent eigenvectors
  - All eigenvalues are real and non-negative
  - The smallest eigenvalue is always zero
  - The eigenvalues indicate connected components in the graph (clusters)
- Fourier transform of a graph
  - A linear combination of graph spectral components
  - Utilizes eigenvectors of the Laplacian (similar to sinusoids of a classical Fourier transform)
- Operation of convolution is equivalent to multiplication in frequency domain

# Motivational examples

# Graphs can represent a lot of data

- Molecules
- Social connections: social networks
- Physical systems: modelling interactions
- Roads: journey planning
- Knowledge graphs: wikipedia
- OCR relations extracted from a PDF document
- Solving PDEs

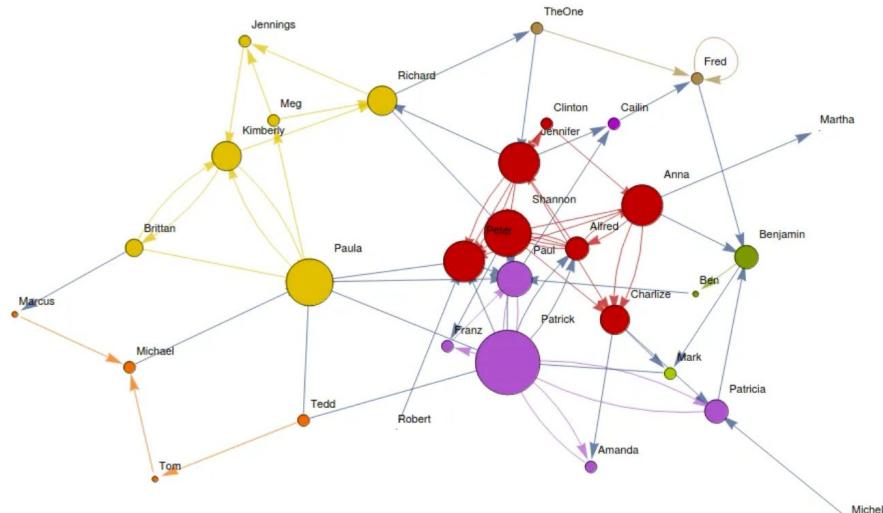


Image source: [Blog post on message-passing](#)

# Types of tasks on Graphs

- Node-level tasks
- Edge-level tasks
- Graph-level tasks

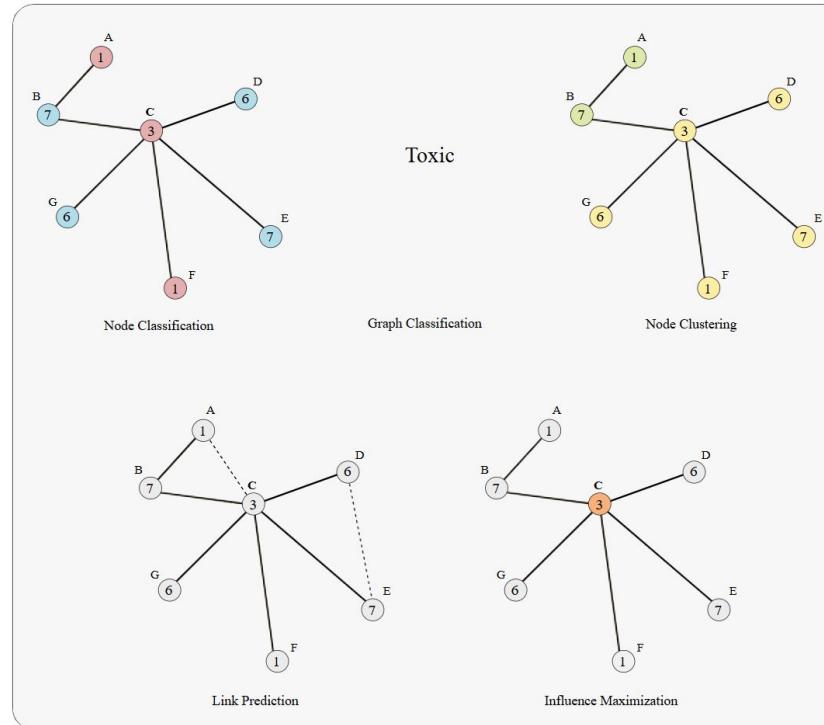


Image source: [Blog post on graph problems](#)

# A note on assumptions

- In the general case, the connectivity information of a graph can be unknown
  - It is known but can be suboptimal for the task at hand

# A note on assumptions

- In the general case, the connectivity information of a graph can be unknown
  - It is known but can be suboptimal for the task at hand
- Learning the structure of a graph is an exciting area of research
  - beyond the scope of this lecture

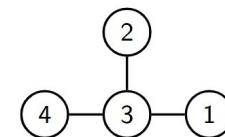
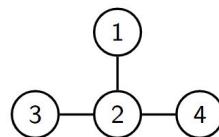
# A note on assumptions

- In the general case, the connectivity information of a graph can be unknown
  - It is known but can be suboptimal for the task at hand
- Learning the structure of a graph is an exciting area of research
  - beyond the scope of this lecture

For now: assume that the structure is known and it is sufficient for the tasks!

# Invariance and Equivariance

Revisiting the problem encountered before:  
Two graphs can have “different representations”

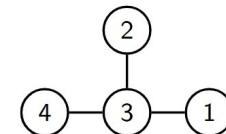
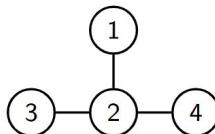


$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \\ \mathbf{x}_4^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_4^\top \\ \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- To learn it directly from data a model needs to see all of the combinations ( $4!$  permutations)  
-> the data requirements grow significantly

# Revisiting the problem encountered before: Two graphs can have “different representations”



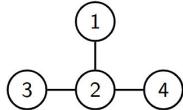
$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \\ \mathbf{x}_4^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_4^\top \\ \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

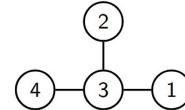
- To learn it directly from data a model needs to see all of the combinations ( $4!$  permutations)
  - > the data requirements grow significantly
    - > we want to exploit the symmetry!

# Linking back to statistical learning

- The assumptions are needed to restrict the class of possible models



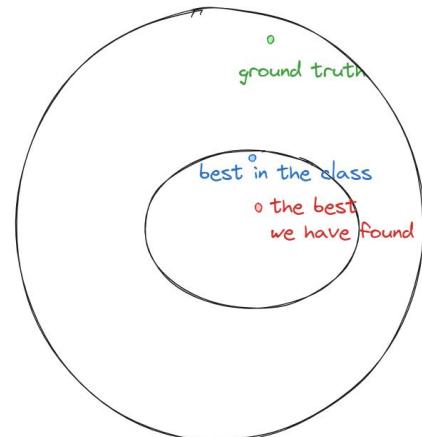
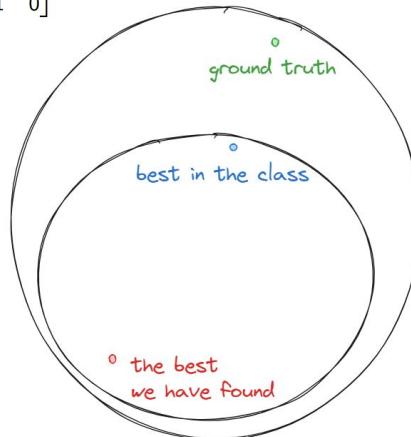
$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \\ \mathbf{x}_4^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



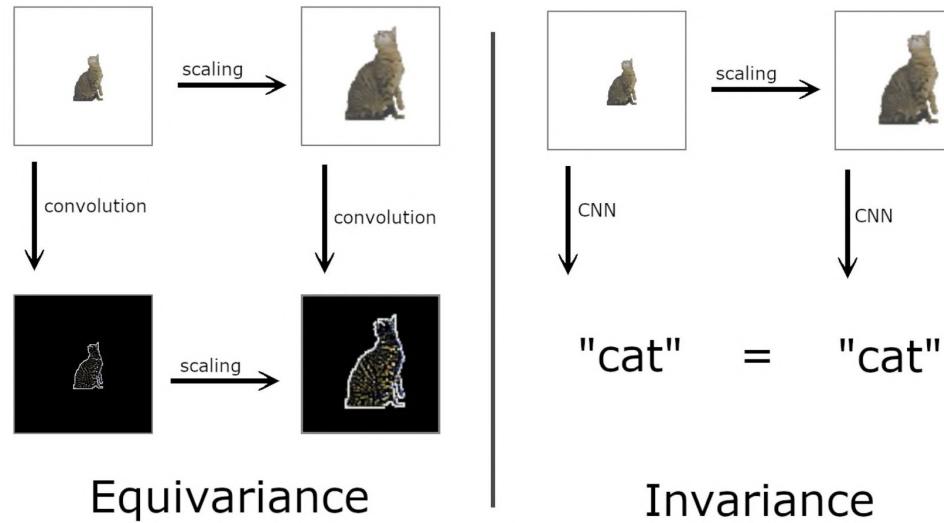
$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_4^\top \\ \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

We want to exploit symmetry:

With a fixed size of a dataset we can find a better approximation if the search space of possible function is smaller



# Invariance and Equivariance



# Permutation invariance for sets

There is no natural ordering in the data

-> the output of the model should not depend on the input order.

$$f \left( \begin{matrix} x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \end{matrix} \right) = \mathbf{y} = f \left( \begin{matrix} x_2 \\ x_5 \\ x_1 \\ x_4 \\ x_3 \end{matrix} \right)$$

Symmetry group  $\mathfrak{G}$ :  $n$ -element Permutation group  $\Sigma_n$   
Group element  $g \in \mathfrak{G}$ : Permutations

# Permutation invariance for graphs

$$f \left( \begin{array}{ccccc} & x_1 & & & \\ x_5 & & & x_2 & \\ & x_4 & & & \\ & & x_3 & & \end{array} \right) = \mathbf{y} = f \left( \begin{array}{ccc} x_2 & & \\ & x_5 & x_4 \\ x_1 & & x_3 \end{array} \right)$$

$$f \left( \begin{array}{ccccc} & x_1 & & & \\ x_5 & & & x_2 & \\ x_4 & & & & \\ & x_3 & & & \end{array} \right) = \mathbf{y} = f \left( \begin{array}{ccc} x_2 & & \\ & x_5 & x_4 \\ x_1 & & x_3 \end{array} \right)$$

# Permutation equivariance

- Ok, permutation invariance is needed to account for ordering  
but why do we need permutation equivariance?

# Permutation equivariance

1. We may actually want the result to change according to the applied transformation, e.g. rotations of a molecule (the coordinates of atoms)

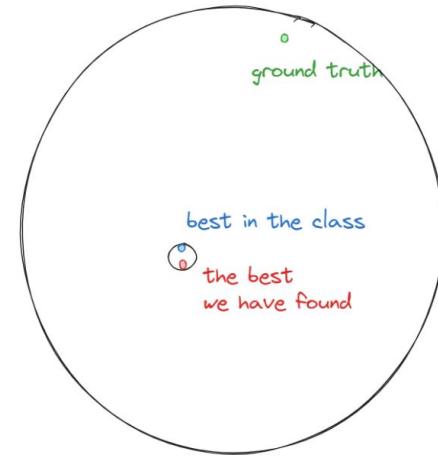
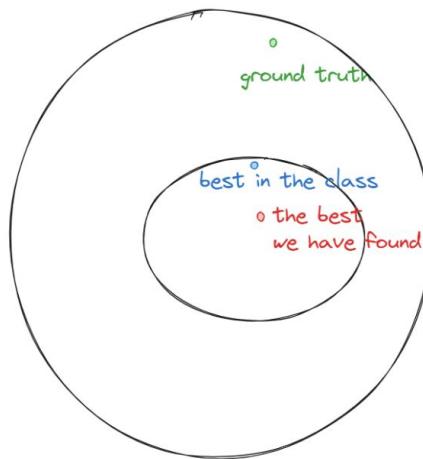
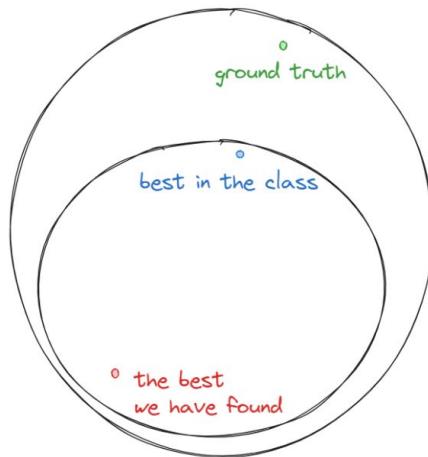
# Permutation equivariance

1. We may actually want the result to change correspondingly to the applied transformation
2. We can express a much larger class of transformations by alternating between invariant and equivariant layers

# Permutation equivariance

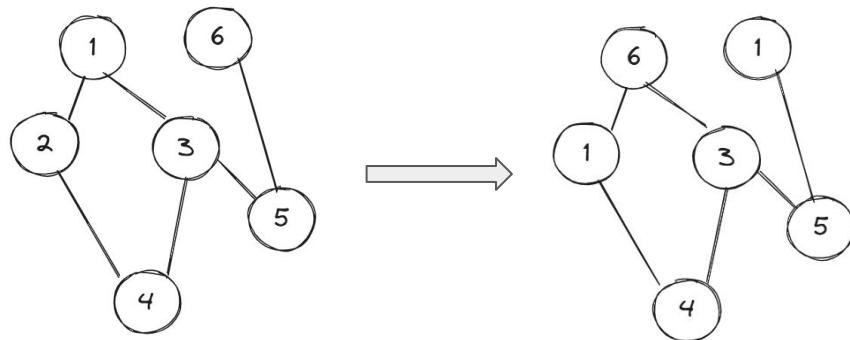
1. We may actually want the result to change correspondingly to the applied transformation
2. We can express a much larger class of transformations by alternating between invariant and equivariant layers
  - a. A class of invariant transformations is quite limited
  - b. Equivariant transformations are more expressive
  - c. By using equivariant + invariant transformation -> the result is invariant, but more expressive
  - d. Can be seen in the GDL blueprint in the end of the lecture (and demonstrated in the exercise)

# Linking back to statistical learning



# Permutations: Permutation matrix in practice

- Permutation matrix should have 1 in every row and every column



$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

# Invariance and equivariance

- Sets ( $P$  is a permutation matrix)

$$f(\mathbf{P}\mathbf{X}) = f(\mathbf{X}) \quad \mathbf{F}(\mathbf{P}\mathbf{X}) = \mathbf{P}\mathbf{F}(\mathbf{X})$$

# Invariance and equivariance

- Sets (P is a permutation matrix)

$$f(\mathbf{PX}) = f(\mathbf{X}) \quad \mathbf{F}(\mathbf{PX}) = \mathbf{PF}(\mathbf{X})$$

- Graphs

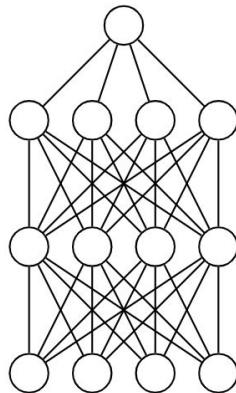
$$f(\mathbf{PX}, \mathbf{PAP}^T) = f(\mathbf{X}, \mathbf{A}) \quad \mathbf{F}(\mathbf{PX}, \mathbf{PAP}^T) = \mathbf{PF}(\mathbf{X}, \mathbf{A})$$

Are simple MLPs permutation invariant?

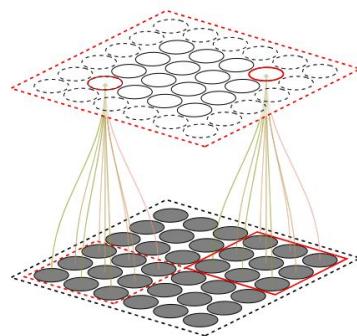
# Are simple MLPs permutation invariant?

- no.

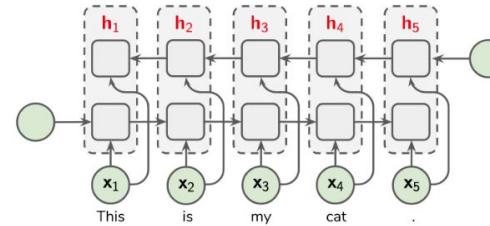
# Are any of these architecture permutation invariant / equivariant?



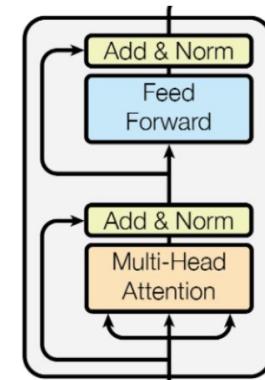
multi-layer  
perceptron



convolutional  
layer

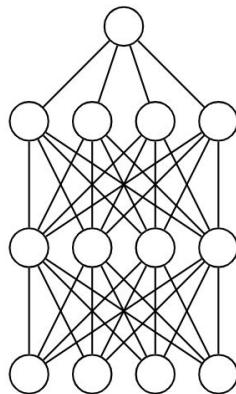


recurrent neural  
network

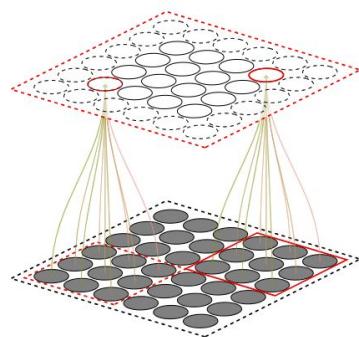


transformer encoder

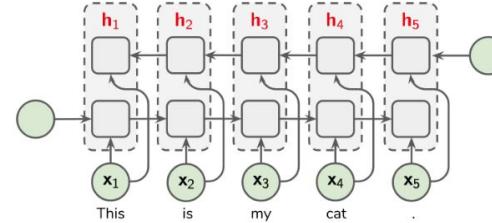
# Are any of these architecture permutation invariant / equivariant?



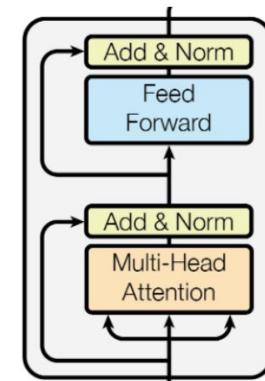
multi-layer  
perceptron



convolutional  
layer



recurrent neural  
network



transformer encoder

**Answer:** transformer encoder with no positional encoding

# Other requirements for GNNs

- Permutation invariance
- +
  - Ability to work with graphs of varying number of nodes
  - Ability to take into account the connectivity information

# Locality: act on neighborhoods

Caution! Do not overdo it with symmetries! Real world is not perfect.

# Locality: act on neighborhoods

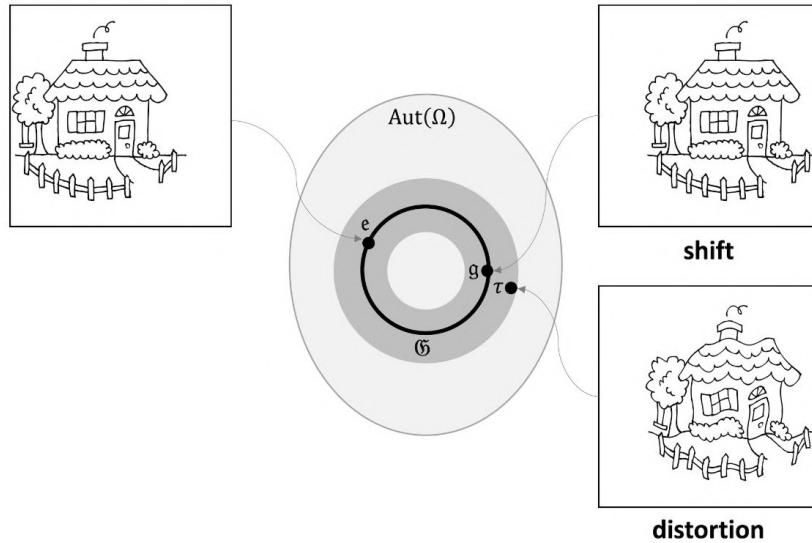
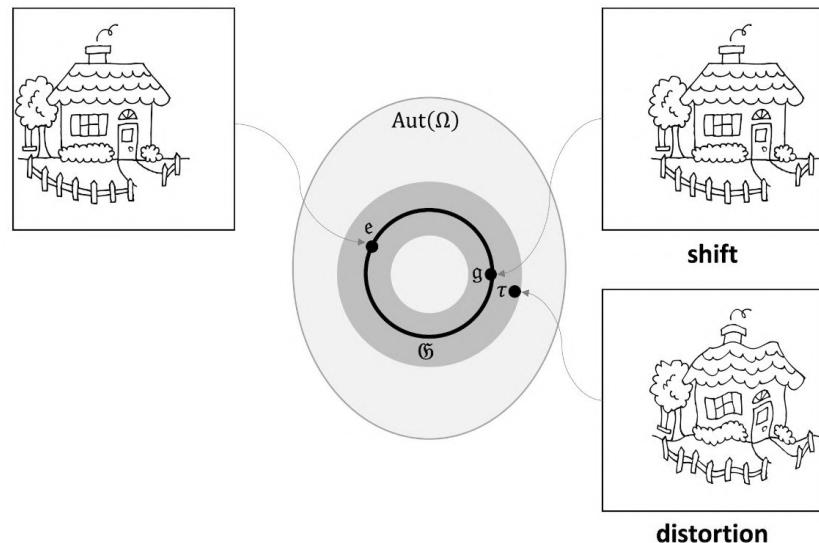


Image source: [GDL book, p. 25](#)

# Locality: act on neighborhoods

## Stability under slight deformations

- CNN layers act locally as well
- GNN layers should be local too
- Link back to the statistical learning -> should not make too strong of the assumptions



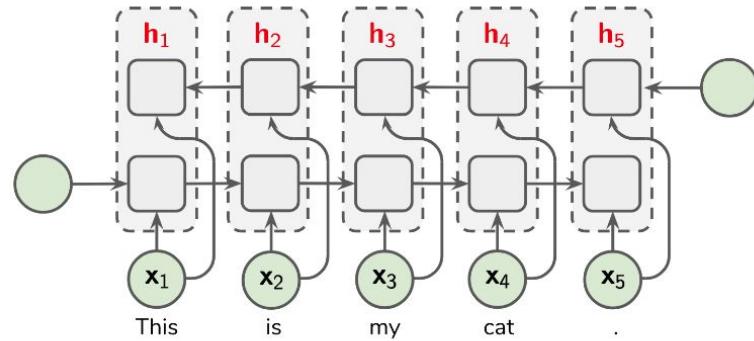
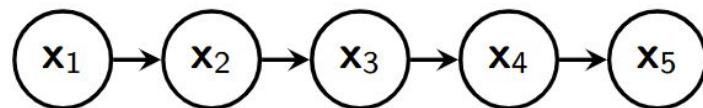
# Special graphs

# DL architectures: CNN, RNN, Transformer

- Even though there is no permutation equivariance in many existing architectures, these models are still **exhibiting some desirable symmetries** and can be classified as “**special cases of GNNs**”.

# Processing special graphs: chains

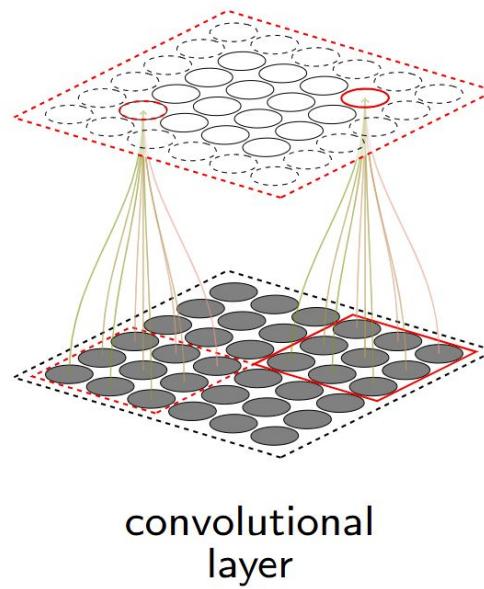
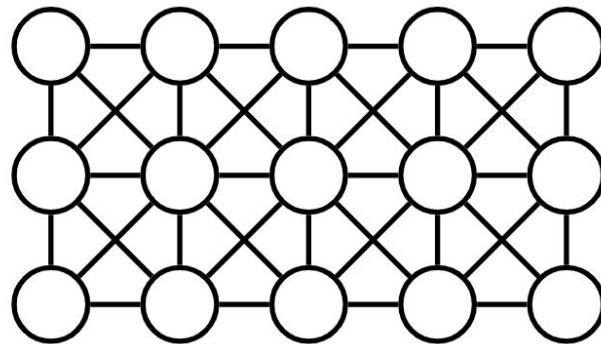
- RNN: a chain graph (physical system modelling)



recurrent neural  
network

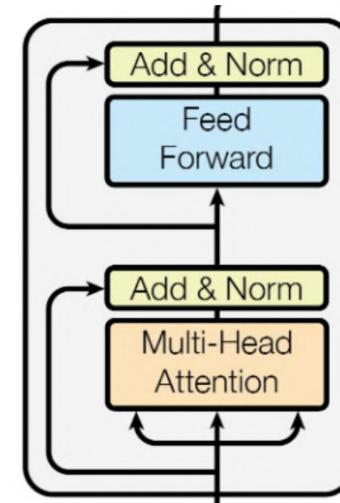
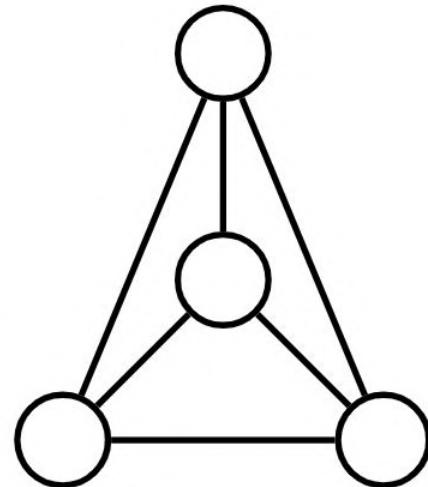
# Processing special graphs: grids

- CNN: a grid graph (PDE solving)



# Processing special graphs: fully-connected graphs

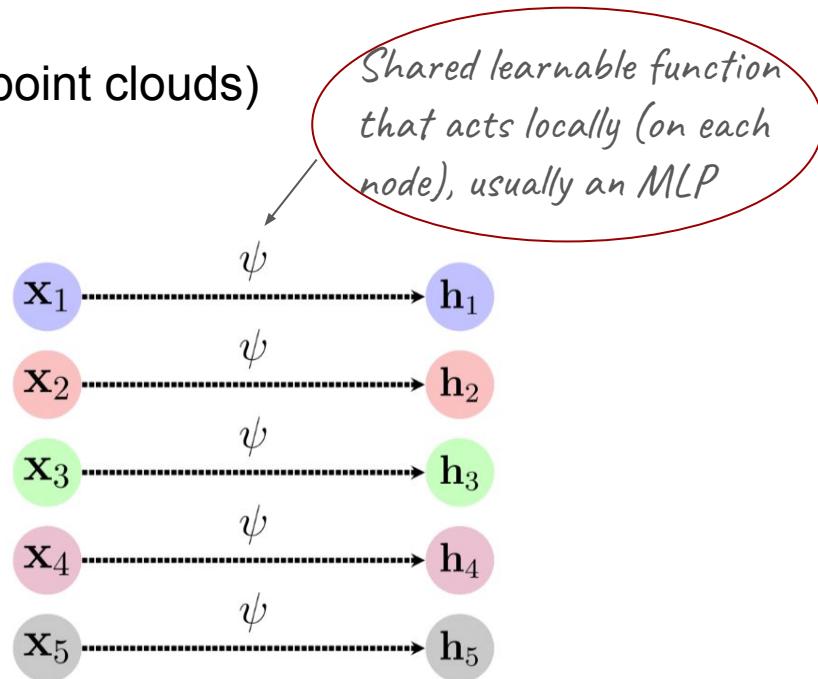
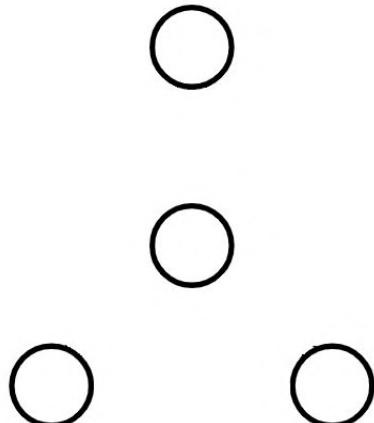
- Transformer encoder: a complete graph (text)



transformer encoder

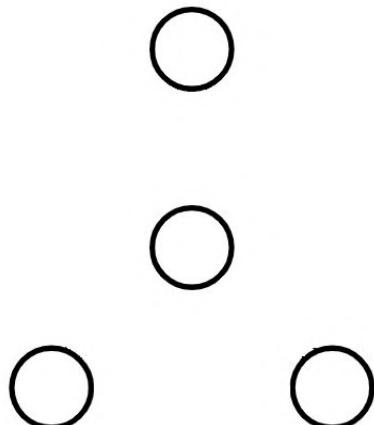
# Deep Sets

- Deep sets: a graph with no edges (point clouds)



# Deep Sets

- Deep sets: a graph with no edges (point clouds)



$$f(\mathbf{X}) = \phi \left( \bigoplus_{i \in \mathcal{V}} \psi(\mathbf{x}_i) \right)$$

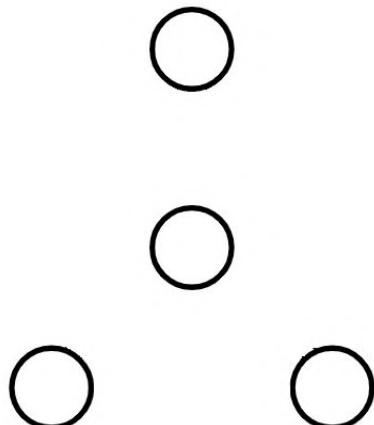
MLPs

The diagram illustrates the computation of a deep set function. Four arrows point from the text "MLPs" to the four circles representing points  $\mathbf{x}_i$ . These points are then combined via a summation operation (indicated by the symbol  $\bigoplus$ ) to produce the final output  $f(\mathbf{X})$  through the function  $\phi$ .

*Crucial! Permutation-invariant transformation  
for all vertices (sum, avg)*

# Deep Sets

- Deep sets: a graph with no edges (point clouds)



Invariant transformation      Equivariant transformation

$$f(\mathbf{X}) = \phi \left( \bigoplus_{i \in \mathcal{V}} \psi(\mathbf{x}_i) \right)$$

*Crucial! Permutation-invariant transformation  
for all vertices (sum, avg)*

# Beyond special cases: Geometry as a network input

- This is as far as we can get with special cases
- How about processing arbitrary graphs?

# Beyond special cases: Geometry as a network input

- This is as far as we can get with special cases
- How about processing arbitrary graphs?

Now the graph structure (geometry) becomes a part of the input!

# Beyond special cases: Geometry as a network input

- This is as far as we can get with special cases
- How about processing arbitrary graphs?

Now the graph structure (geometry) becomes a part of the input!

But we still assume that the structure is known.

However different general graphs can be processed in the same dataset.

**Break -?**

# Graph Neural Networks

# GNNs

- Learnable transformation of graph attributes (node, edge and global features)

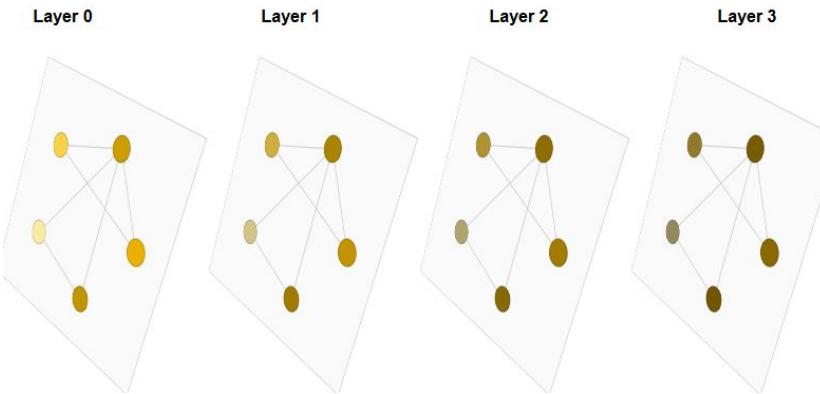


Image source: [GNN blogpost](#)

# GNNs

- Learnable transformation of graph attributes (node, edge and global features)
- The transformation is permutation invariant

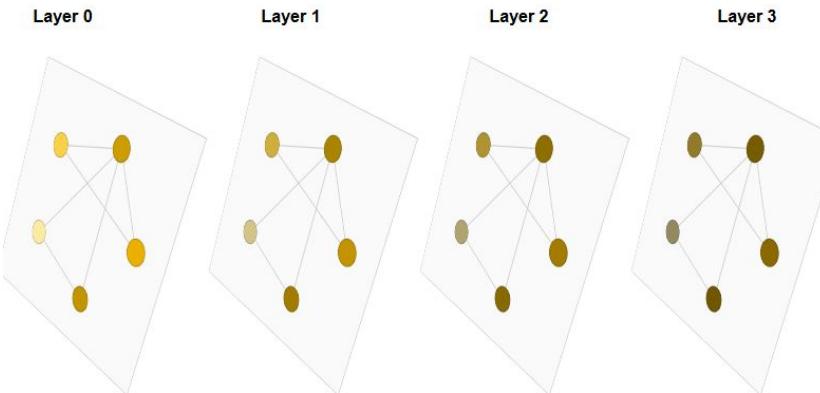


Image source: [GNN blogpost](#)

# GNNs

- Learnable transformation of graph attributes (node, edge and global features)
- The transformation is permutation invariant
- GNNs operate over graphs (both inputs are outputs are graphs)

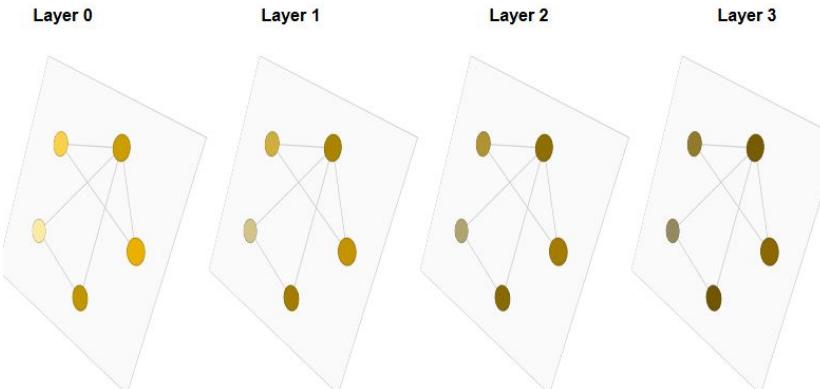


Image source: [GNN blogpost](#)

# GNNs

- Learnable transformation of graph attributes (node, edge and global features)
- The transformation is permutation invariant
- GNNs operate over graphs (both inputs are outputs are graphs)
- (Usually?) The transformations are applied without changing the structure of the input graph

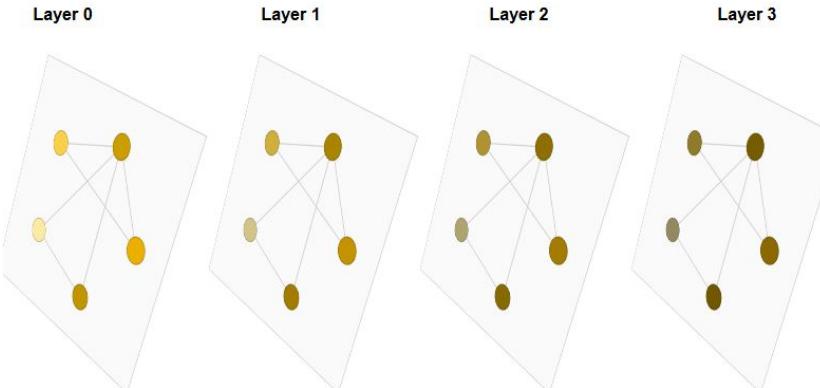
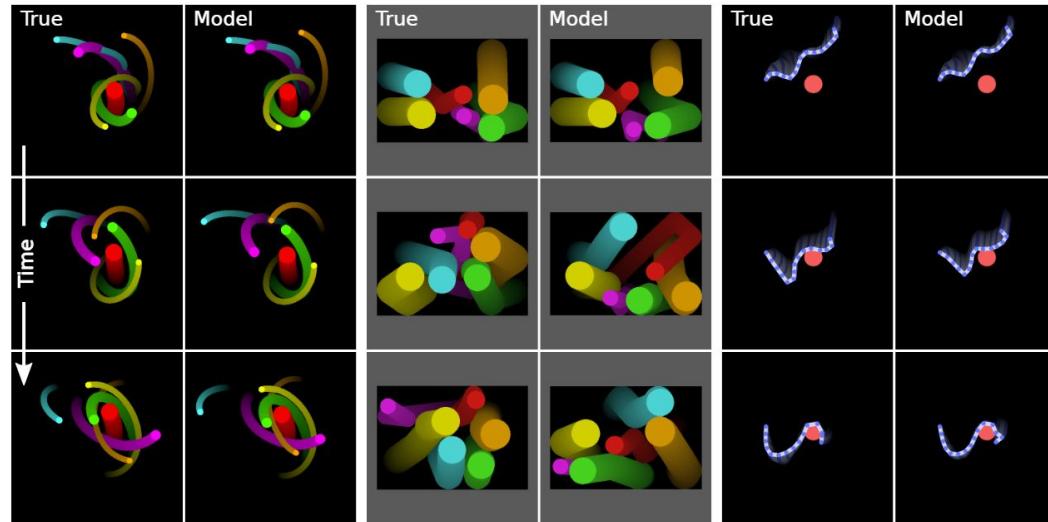


Image source: [GNN blogpost](#)

# One of the pioneering works: Interaction networks

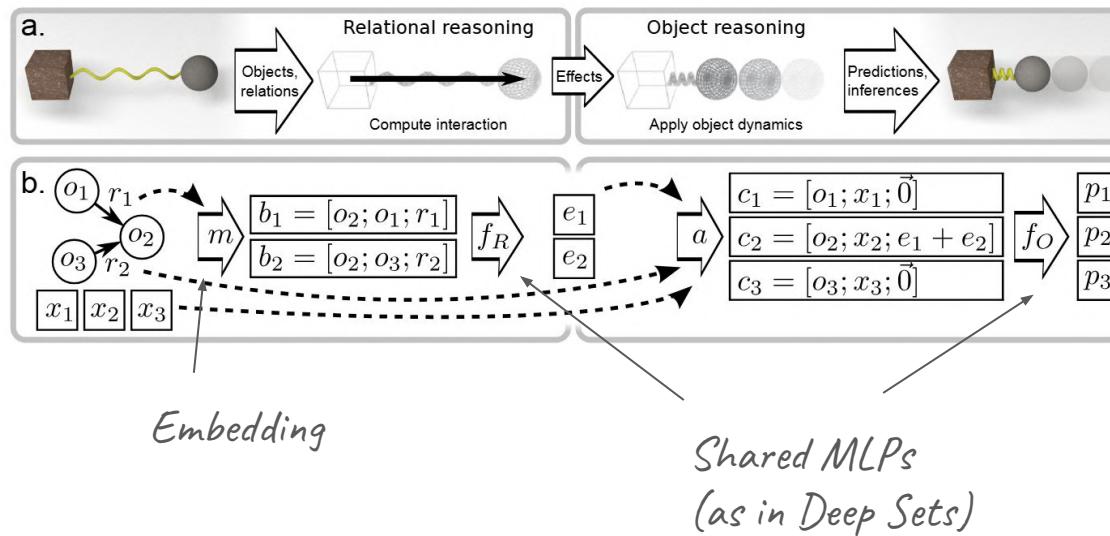
- Modelling of interactions in physical systems (predict the dynamics)



Interaction networks [paper](#) (and image source)

# One of the pioneering works: Interaction networks

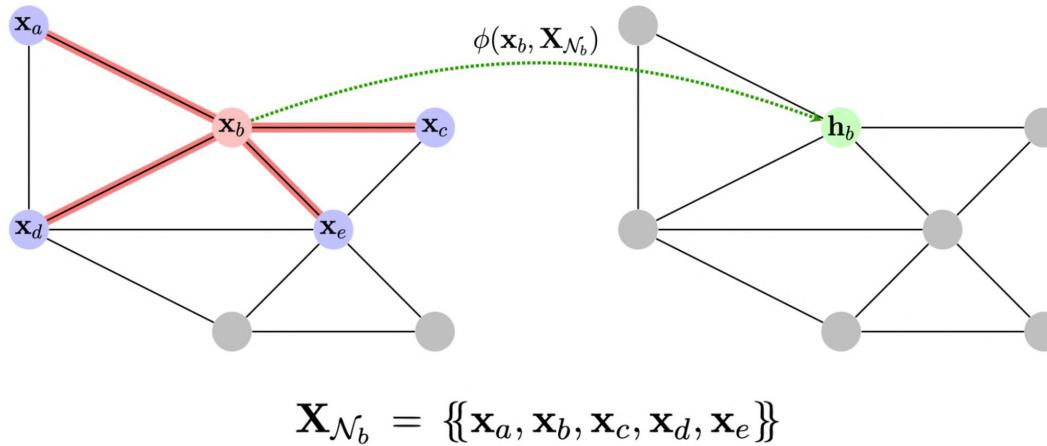
- Modelling of interactions in physical systems



Interaction networks [paper](#) (and image source)

# GNN layers

- Similar idea as in Deep Sets + dependency on the neighborhoods
- The update function is also shared between nodes (locality)



# GNN Blueprint

1. Nodes send messages to their neighbors

$$\psi(\mathbf{x}_i, \mathbf{x}_j)$$

2. Messages from the neighbors are aggregated

$$\bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)$$

3. Node features (attributes) are updated based on the received messages

$$\mathbf{h}_i = \phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j) \right)$$

**Note!** Edge and global features are ignored for simplicity.

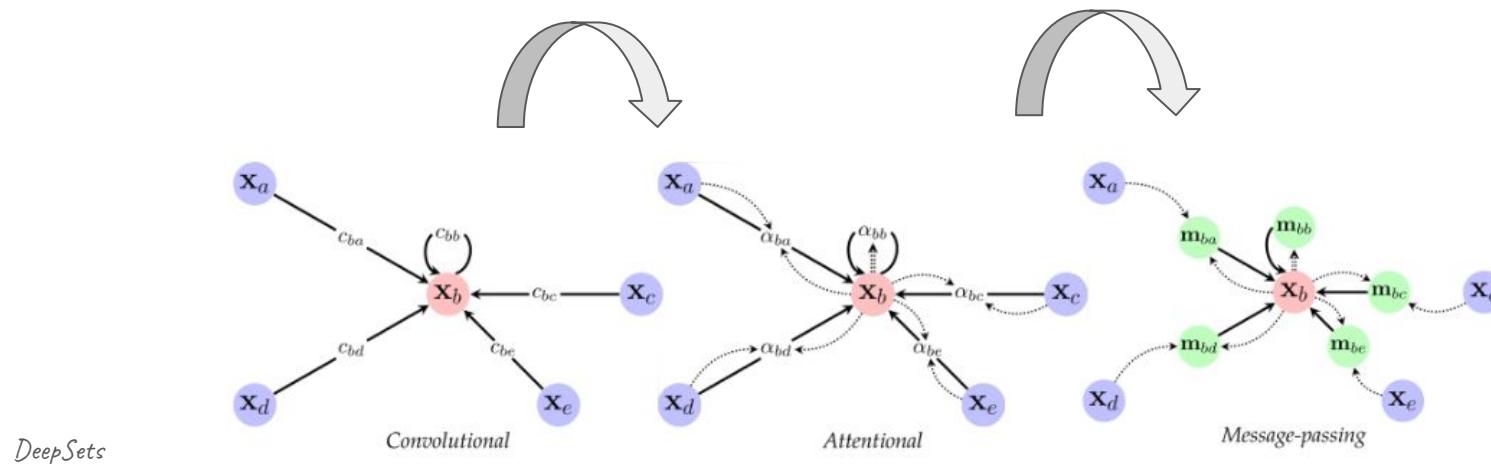
# GNN layers

*Permutation  
equivariant function*

$$\mathbf{F}(\mathbf{X}, \mathbf{A}) = \begin{bmatrix} - & \phi(\mathbf{x}_1, \mathbf{X}_{\mathcal{N}_1}) & - \\ - & \phi(\mathbf{x}_2, \mathbf{X}_{\mathcal{N}_2}) & - \\ & \vdots & \\ - & \phi(\mathbf{x}_n, \mathbf{X}_{\mathcal{N}_n}) & - \end{bmatrix}$$

*Local permutation-invariant function*

# Types of GNNs



$$\mathbf{h}_i = \phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij} \psi(\mathbf{x}_j) \right)$$
$$\mathbf{h}_i = \phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j) \psi(\mathbf{x}_j) \right)$$
$$\mathbf{h}_i = \phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j) \right)$$

The same structure, differences are in how messages are calculated and aggregated

# Types of GNNs

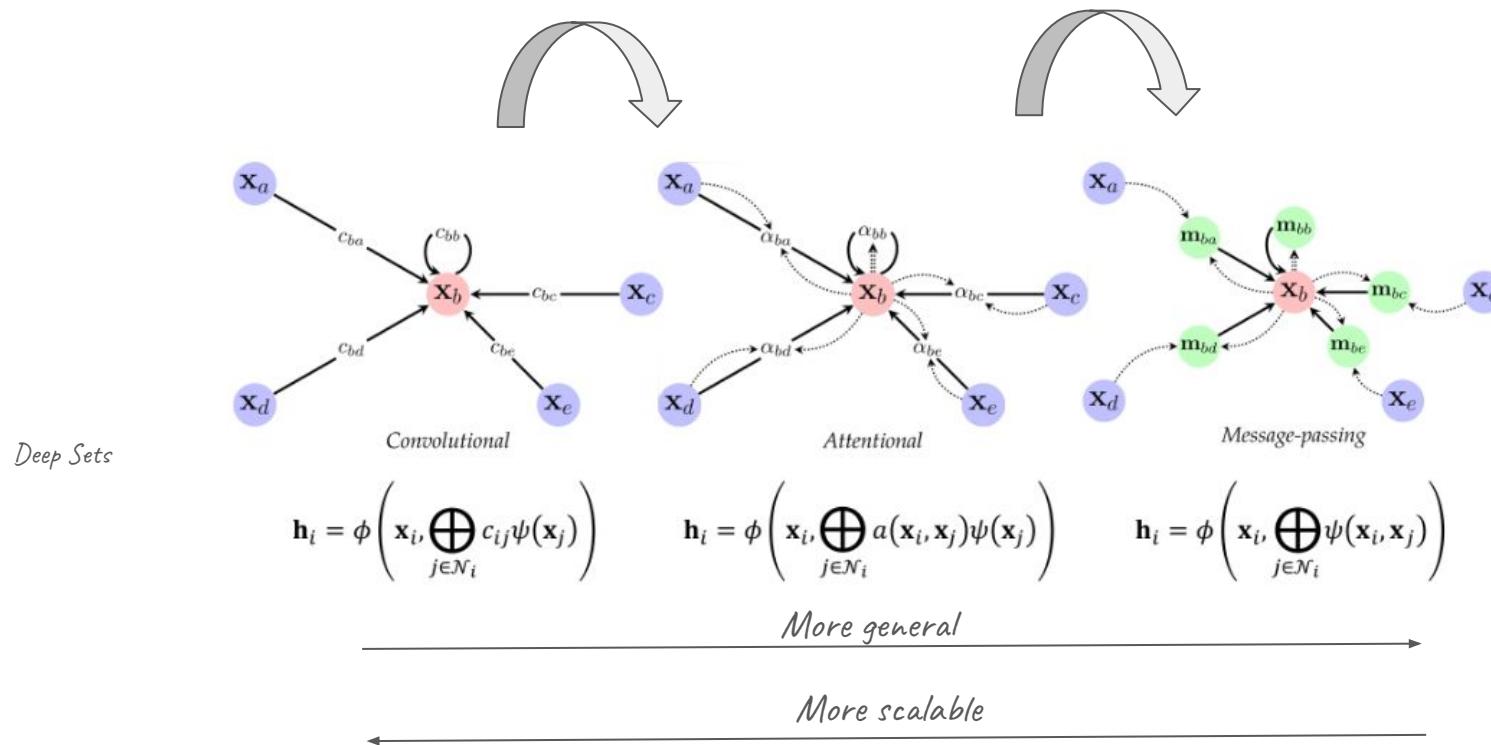


Image source: [GDL course, lecture 5](#)

# Graph Convolutional network

- Generalizes convolutions for arbitrary structure of graphs beyond grids
- Graph Convolutions are based on spectral perspective of convolutions
- The convolution operation is equivalent to the multiplication in Fourier domain

$$\mathbf{h}_i = \phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij} \psi(\mathbf{x}_j) \right)$$

# Graph Convolutional network

- Generalizes convolutions for arbitrary structure of graphs beyond grids
- Graph Convolutions are based on spectral perspective of convolutions
- The convolution operation is equivalent to the multiplication in Fourier domain
  - Scales well with the size of the graph
  - Works well for homophilous graphs
  - Works when the graph is fixed

$$\mathbf{h}_i = \phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij} \psi(\mathbf{x}_j) \right)$$

# Graph attentional network (GAT)

- Utilizes attention to calculate weights of neighbors' messages

$$\mathbf{h}_i = \phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j) \psi(\mathbf{x}_j) \right)$$

# Graph attentional network (GAT)

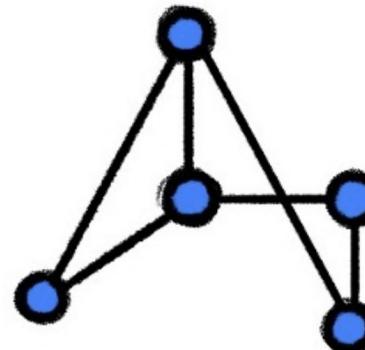
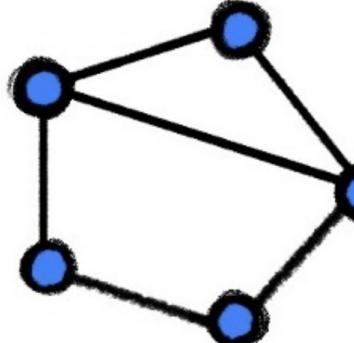
- Utilizes attention to calculate weights of neighbors' messages
- A bit more general than Graph Convolutional Networks
- Less scalable than Graph Convolutional Networks
- Can 'infer' a graph structure, e.g. if the graph is unknown we may pass a fully connected graph. In theory, GAT can learn to put near-zero weights for some of the neighbor messages.

$$\mathbf{h}_i = \phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j) \psi(\mathbf{x}_j) \right)$$

# Detour: Expressive power of GNNs

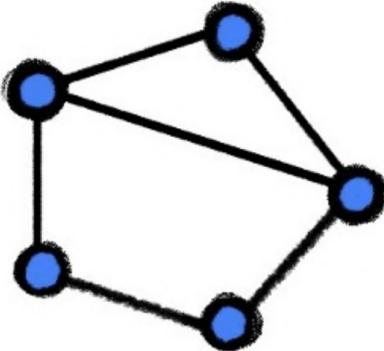
# Isomorphic graphs

- Two graphs are isomorphic if there is one-to-one correspondence between their vertices and edges that preserves the connectivity structure of the graphs.

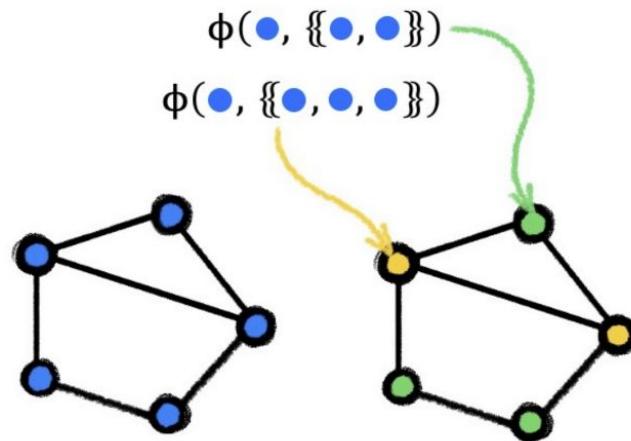


# 1-WL (Weisfeiler-Lehman) test

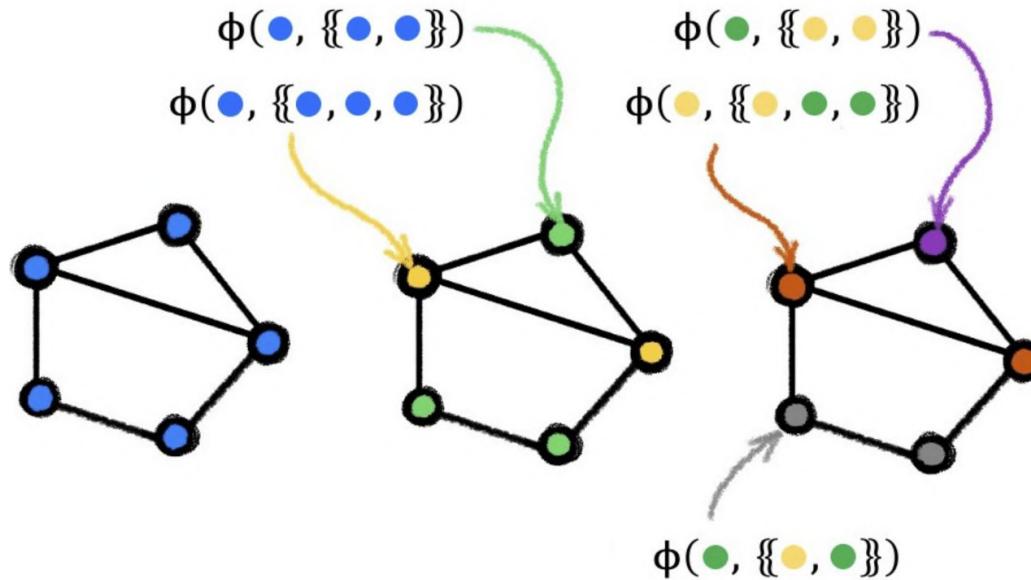
- The test for two graph being **isomorphic**
- Implemented by iteratively tracking neighborhoods that arise and changing their colors (colors = hashes)



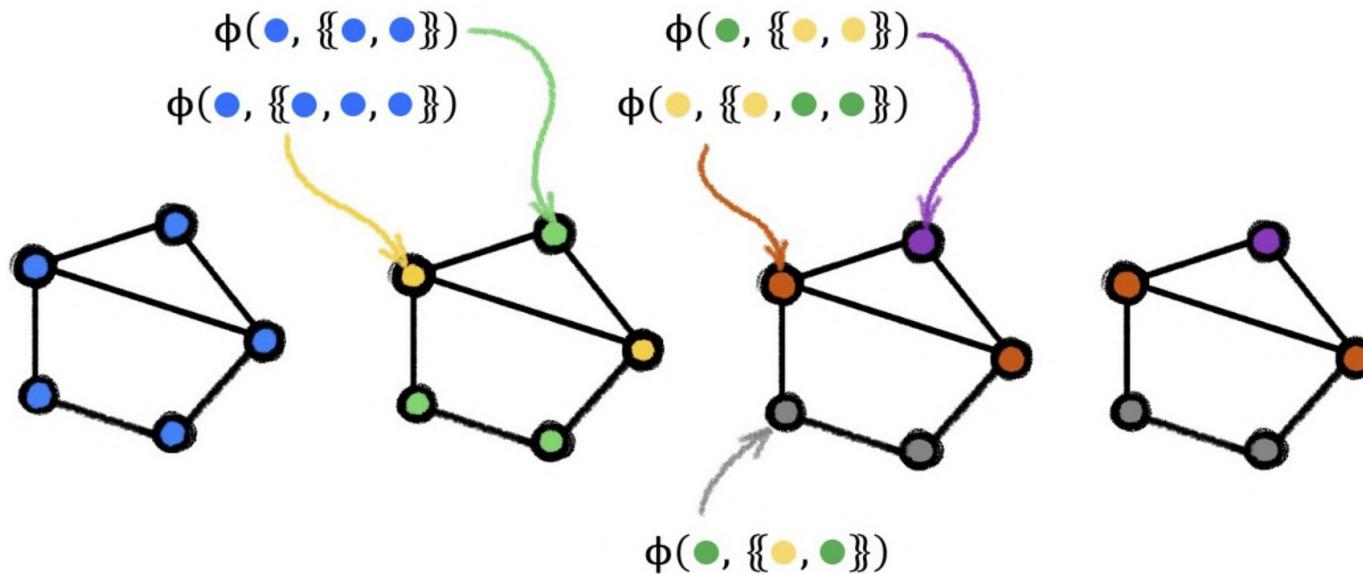
# 1-WL (Weisfeiler-Lehman) test



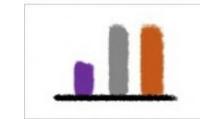
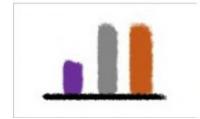
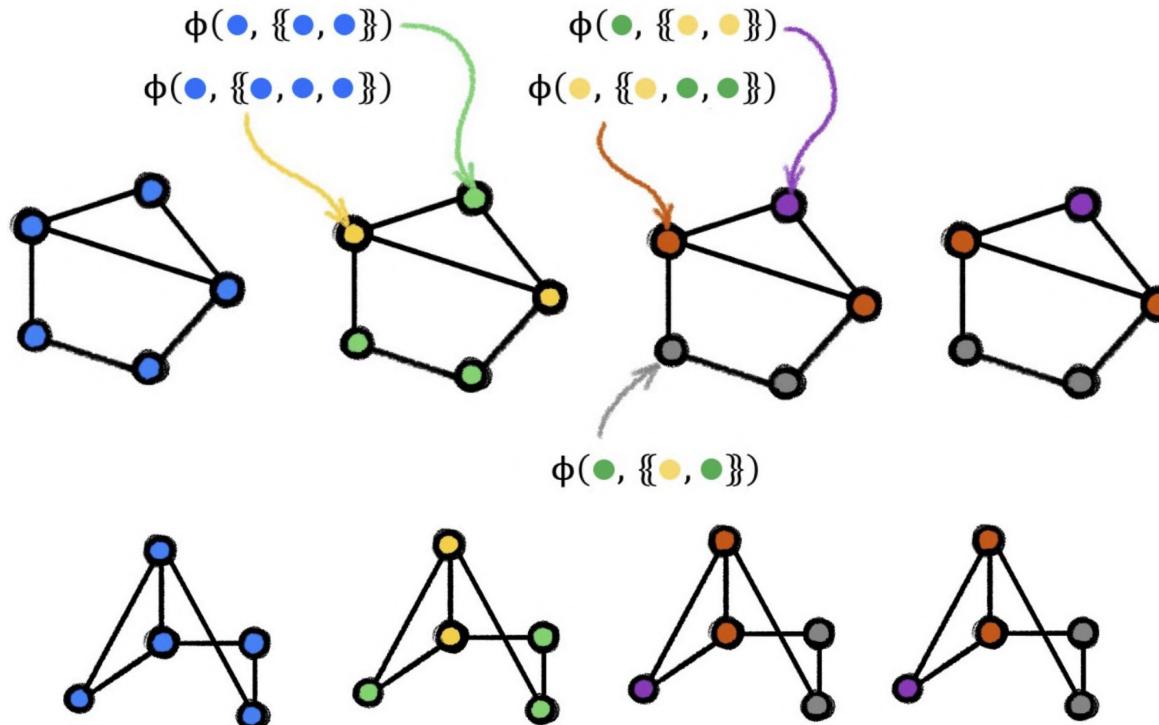
# 1-WL (Weisfeiler-Lehman) test



# 1-WL (Weisfeiler-Lehman) test

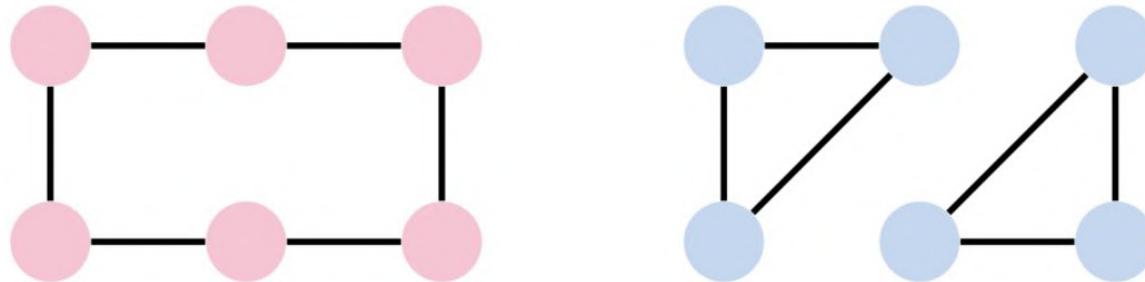


# 1-WL (Weisfeiler-Lehman) test



# 1-WL test failures

- The test cannot distinguish these two graphs



# Expressive power of GNNs

- GNNs over discrete features can be as only as expressive as 1-WL

---

## Algorithm 1: WL-1 algorithm (Weisfeiler & Lehmann, 1968)

---

**Input:** Initial node coloring  $(h_1^{(0)}, h_2^{(0)}, \dots, h_N^{(0)})$

**Output:** Final node coloring  $(h_1^{(T)}, h_2^{(T)}, \dots, h_N^{(T)})$

$t \leftarrow 0;$

**repeat**

**for**  $v_i \in \mathcal{V}$  **do**

$h_i^{(t+1)} \leftarrow \text{hash} \left( \sum_{j \in \mathcal{N}_i} h_j^{(t)} \right);$

$t \leftarrow t + 1;$

**until** stable node coloring is reached;

---

# Expressive power of GNNs

- GNNs over discrete features can be as only as expressive as 1-WL
- There is no guarantee that we even reach 1-WL test performance
- Graph Isomorphic Network propose maximally expressive GNNs

$$\mathbf{h}_u = \phi \left( (1 + \epsilon) \mathbf{h}_u + \sum_{v \in \mathcal{N}_v} \mathbf{h}_v \right)$$

# Expressive power of GNNs

- GNNs over discrete features can be as only as expressive as 1-WL
- There is no guarantee that we even reach 1-WL test performance
- Graph Isomorphic Network propose maximally expressive GNNs
- Since 1-WL test has failure modes -> GNNs inherit these problems

# Expressive power of GNNs

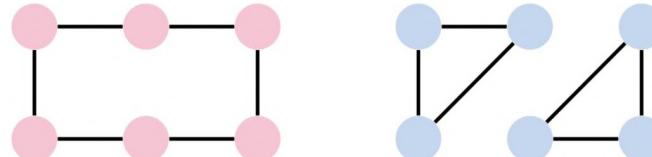
- GNNs over discrete features can be as only as expressive as 1-WL
- There is no guarantee that we even reach 1-WL test performance
- Graph Isomorphic Network propose maximally expressive GNNs
- Since 1-WL test has failure modes -> GNNs inherit these problems

**Remedy:** input to the GNN information, it is not able to compute

# Expressive power of GNNs

- GNNs over discrete features can be as only as expressive as 1-WL
- There is no guarantee that we even reach 1-WL test performance
- Graph Isomorphic Network propose maximally expressive GNNs
- Since 1-WL test has failure modes -> GNNs inherit these problems

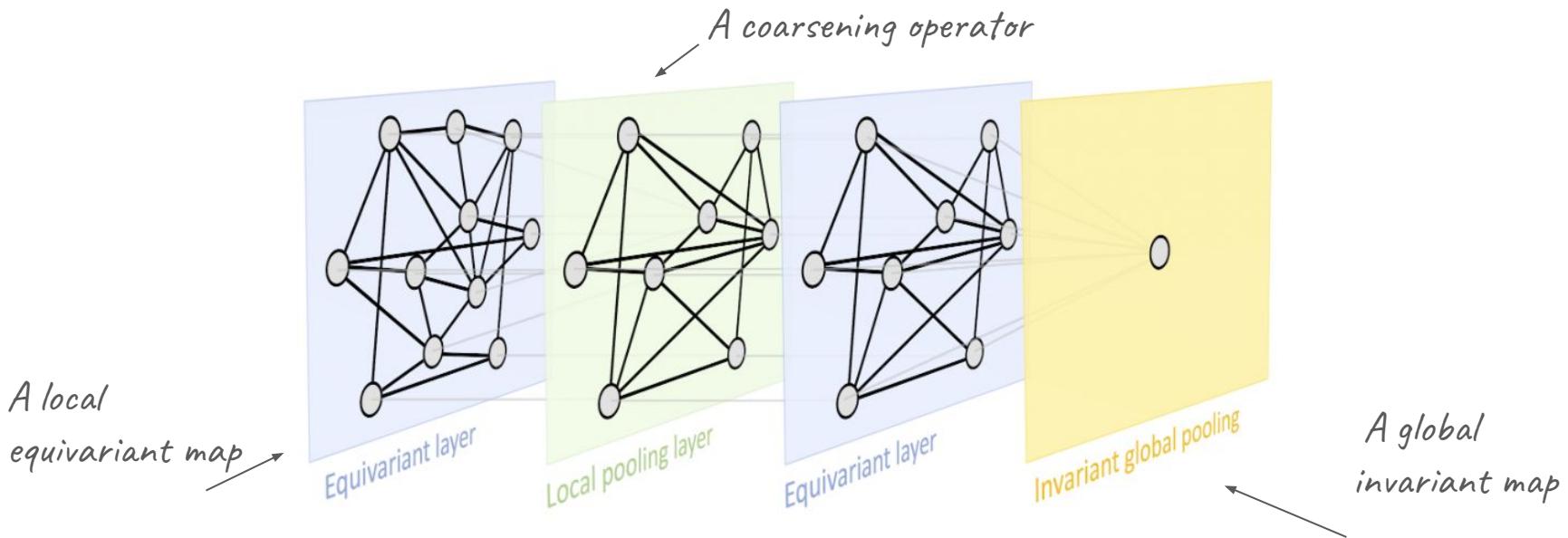
*How many clusters are in the graph?*



*What is the largest cycle?*

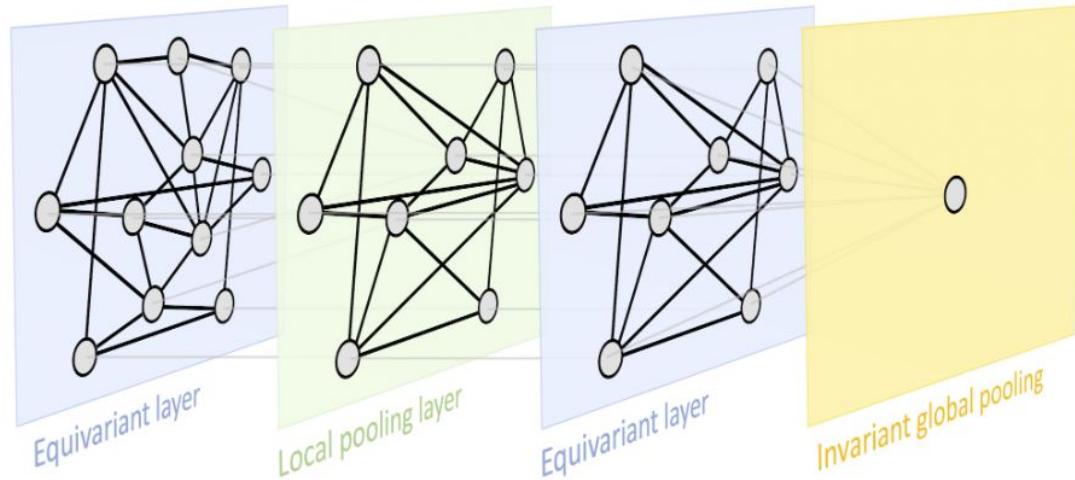
# Connecting back to GDL

# Geometric Deep Learning Blueprint



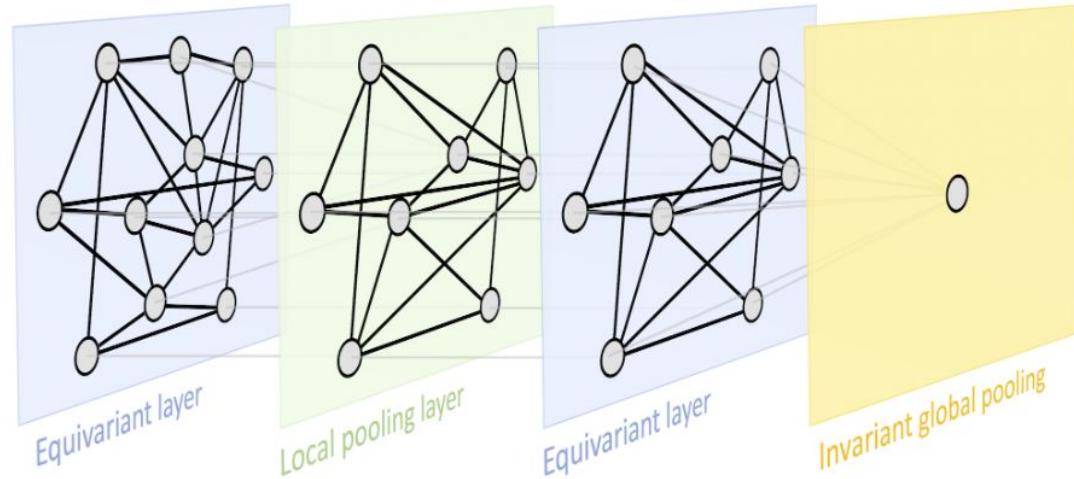
# Geometric Deep Learning Blueprint: CNNs

- Convolutional layers equivariant to **translations**
- Local pooling coarsening of the **grid**
- Global pooling (translation invariant) such as mean / sum



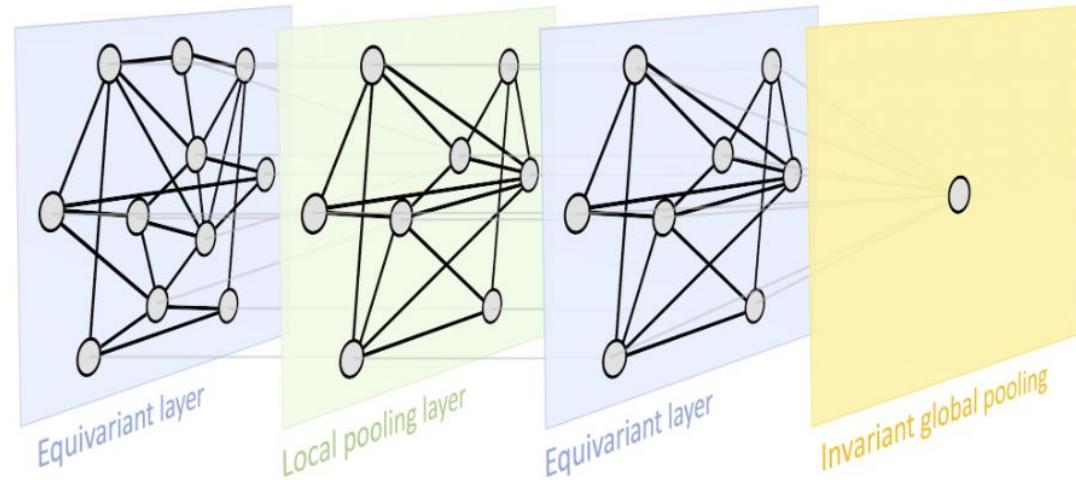
# Geometric Deep Learning Blueprint: GNNs

- Permutation-equivariant GNN layers
- Local pooling: coarsening of the graph
- Global pooling such as mean / sum



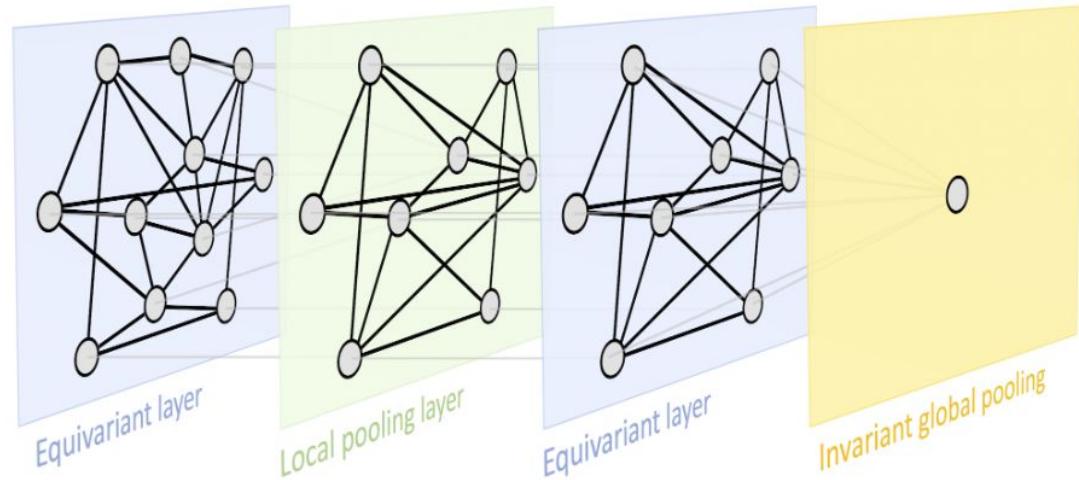
# Geometric Deep Learning Blueprint: beyond

- Equivariant layers wrt an **arbitrary group**
- Local pooling:  
coarsening of the **domain**
- Global pooling such as  
mean / sum



# Geometric Deep Learning Blueprint: beyond

- Equivariant layers wrt an **arbitrary group**
- Local pooling:  
coarsening of the **domain**
- Global pooling such as  
mean / sum



Next time!

# Summary

# Lecture recap

- Graphs and invariance / equivariance to permutations
  - Graph terminology
  - Motivating examples for graph applications
  - Desired properties of neural networks for graph-structured data

# Lecture recap

- Graphs and invariance / equivariance to permutations
  - Graph terminology
  - Motivating examples for graph applications
  - Desired properties of neural networks for graph-structured data
- Graph Neural Networks
  - Special cases
  - Main building blocks
  - Expressivity

# Lecture recap

- Graphs and invariance / equivariance to permutations
  - Graph terminology
  - Motivating examples for graph applications
  - Desired properties of neural networks for graph-structured data
- Graph Neural Networks
  - Special cases
  - Main building block
  - Expressivity
- Geometric Deep Learning Blueprint

# Exercise recap

- Calculate graph Laplacian, explain the connection with GCN
- Check if two graphs are isomorphic
- Small data exploration task
- Step-by-step implementation of Graph Attentional Layer for GAT
- Optional implementation of GCN layer (optional, for comparison)

# Recommended reading

- Graph representation Learning [Book](#): especially Ch. 2.3 and Ch.7.
- Geometric Deep Learning [Book](#): Ch. 3.5, Ch. 4.1, Ch. 5

# Geometric Deep Learning

---

Erik Schultheis, Kate Haitsiukevich, Çağlar Hızlı, Alison Pouplin, Vikas Garg

7.11.2024

## **Part V: Invariance through group smoothing**

---

# Label function invariant under group action

## Invariant function class

Assume we want to learn a function  $f^*$  with

$$\forall x \in \mathcal{X}, g \in G : f^*(g.x) = f^*(x)$$

⇒ Function is constant within each orbit.

## Example

$G = \mathbb{Z}/4\mathbb{Z} = C(4)$  cyclic group,  
acting as 90° rotations:

$$f^* \left( \begin{smallmatrix} \text{airplane} \\ 1 \end{smallmatrix} \right) = f^* \left( \begin{smallmatrix} \text{airplane} \\ 2 \end{smallmatrix} \right) = f^* \left( \begin{smallmatrix} \text{airplane} \\ 3 \end{smallmatrix} \right) \\ = f^* \left( \begin{smallmatrix} \text{airplane} \\ 4 \end{smallmatrix} \right) = \text{"airplane"}$$

# Label function invariant under group action

## Invariant function class

Assume we want to learn a function  $f^*$  with

$$\forall x \in \mathcal{X}, g \in G : f^*(g.x) = f^*(x)$$

⇒ Function is constant within each orbit.

How can we construct a hypothesis class that respects this constraint?

## Example

$G = \mathbb{Z}/4\mathbb{Z} = C(4)$  cyclic group,  
acting as 90° rotations:

$$f^* (\text{airplane}) = f^* (\text{airplane}) = f^* (\text{airplane}) \\ = f^* (\text{airplane}) = \text{"airplane"}$$

## Group smoothing operator

### Group smoothing operator

Let  $f : \mathcal{X} \rightarrow \mathcal{Y}$ , and  $G$  be a finite group with an action  $\cdot$  on  $\mathcal{X}$ . Define **group smoothing** through

$$\Sigma_G f(x) := |G|^{-1} \sum_{g \in G} f(g \cdot x).$$

⇒ Average over all elements within an orbit.

# Group smoothing operator

## Group smoothing operator

Let  $f : \mathcal{X} \rightarrow \mathcal{Y}$ , and  $G$  be a finite group with an action  $\cdot$  on  $\mathcal{X}$ . Define **group smoothing** through

$$\Sigma_G f(x) := |G|^{-1} \sum_{g \in G} f(g \cdot x).$$

⇒ Average over all elements within an orbit.

Acts as identity on invariant functions:

$$\Sigma_G f^* = f^*.$$

⇒  $\Sigma_G$  is a projection.

## Example

$G = \mathbb{Z}/4\mathbb{Z} = C(4)$  cyclic group, acting as 90° rotations:

$$\begin{aligned}\Sigma_G f(\text{↗}) &= \frac{1}{4} \left( f(\text{↗}) + f(\text{↖}) \right. \\ &\quad \left. + f(\text{↙}) + f(\text{↖}) \right)\end{aligned}$$

# Invariant hypothesis class through group smoothing

## Smoothed hypothesis class

Let  $\mathcal{F} \subset \{f : \mathcal{X} \rightarrow \mathcal{Y}\}$  be a hypothesis class, define

$$\Sigma_G \mathcal{F} := \{\Sigma_G f : f \in \mathcal{F}\}$$

## Example

$\mathcal{X} = \{\text{space of } 3 \times 3 \text{ images}\}$ ,  
 $\mathcal{F} = \{\text{linear functions}\} = \mathbb{R}^{3 \times 3}$   
 $\Sigma_G \mathcal{F} = ?$

# Invariant hypothesis class through group smoothing

## Smoothed hypothesis class

Let  $\mathcal{F} \subset \{f : \mathcal{X} \rightarrow \mathcal{Y}\}$  be a hypothesis class, define

$$\Sigma_G \mathcal{F} := \{\Sigma_G f : f \in \mathcal{F}\}$$

## Example

$\mathcal{X} = \{\text{space of } 3 \times 3 \text{ images}\}$ ,  
 $\mathcal{F} = \{\text{linear functions}\} = \mathbb{R}^{3 \times 3}$

$$\Sigma_G \mathcal{F} = ?$$

$$\Sigma_G f(\mathbf{x}) = 0.25 \sum_{g \in G} \langle \mathbf{f}, P_g \mathbf{x} \rangle,$$

where  $P_g$ : permutation matrix corresponding to rotation.

# Invariant hypothesis class through group smoothing

## Smoothed hypothesis class

Let  $\mathcal{F} \subset \{f : \mathcal{X} \rightarrow \mathcal{Y}\}$  be a hypothesis class, define

$$\Sigma_G \mathcal{F} := \{\Sigma_G f : f \in \mathcal{F}\}$$

## Example

$\mathcal{X} = \{\text{space of } 3 \times 3 \text{ images}\}$ ,  
 $\mathcal{F} = \{\text{linear functions}\} = \mathbb{R}^{3 \times 3}$

$$\Sigma_G \mathcal{F} = ?$$

$$\Sigma_G f(\mathbf{x}) = 0.25 \sum_{g \in G} \langle P_g^{-1} \mathbf{f}, \mathbf{x} \rangle,$$

where  $P_g$ : permutation matrix corresponding to rotation.

# Invariant hypothesis class through group smoothing

## Smoothed hypothesis class

Let  $\mathcal{F} \subset \{f : \mathcal{X} \rightarrow \mathcal{Y}\}$  be a hypothesis class, define

$$\Sigma_G \mathcal{F} := \{\Sigma_G f : f \in \mathcal{F}\}$$

## Example

$\mathcal{X} = \{\text{space of } 3 \times 3 \text{ images}\}$ ,  
 $\mathcal{F} = \{\text{linear functions}\} = \mathbb{R}^{3 \times 3}$

$$\Sigma_G \mathcal{F} = ?$$

$$\Sigma_G f(\mathbf{x}) = \langle 0.25 \sum_{g \in G} P_g^{-1} \mathbf{f}, \mathbf{x} \rangle,$$

where  $P_g$ : permutation matrix corresponding to rotation.

# Invariant hypothesis class through group smoothing

## Smoothed hypothesis class

Let  $\mathcal{F} \subset \{f : \mathcal{X} \rightarrow \mathcal{Y}\}$  be a hypothesis class, define

$$\Sigma_G \mathcal{F} := \{\Sigma_G f : f \in \mathcal{F}\}$$

## Example

$\mathcal{X} = \{\text{space of } 3 \times 3 \text{ images}\}$ ,  
 $\mathcal{F} = \{\text{linear functions}\} = \mathbb{R}^{3 \times 3}$

$\Sigma_G \mathcal{F} = \{\text{rot-sym linear functions}\}$

$$\Sigma_G f(\mathbf{x}) = \langle 0.25 \sum_{g \in G} P_g^{-1} \mathbf{f}, \mathbf{x} \rangle,$$

where  $P_g$ : permutation matrix corresponding to rotation.

# Rotation-invariant $3 \times 3$ linear functions

## Original Basis

The  $3 \times 3$  linear functions  
are generated by a basis

$$\mathcal{F} = \text{span}\{\begin{matrix} \blacksquare & \square \\ \square & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \square \\ \square & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \square \\ \square & \blacksquare \end{matrix},$$
$$\begin{matrix} \blacksquare & \square \\ \square & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \square \\ \square & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \square \\ \square & \blacksquare \end{matrix},$$
$$\begin{matrix} \blacksquare & \square \\ \square & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \square \\ \square & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \square \\ \square & \blacksquare \end{matrix}\}$$

## Rotation-invariant $3 \times 3$ linear functions

## Original Basis

The  $3 \times 3$  linear functions  
are generated by a basis

$$\mathcal{F} = \text{span}\{\begin{matrix} \text{■} & \text{□} & \text{□} \\ \text{□} & \text{■} & \text{□} \\ \text{□} & \text{□} & \text{■} \end{matrix}\}$$

## Orbits

Basis vectors fall into three orbits

$$G(\square) = \{\square, \square, \square, \square, \square\}$$

$$G(\square) = \{\square, \square, \square, \square\}$$

$$G(\blacksquare) = \{\blacksquare\}$$

## Rotation-invariant $3 \times 3$ linear functions

## Original Basis

The  $3 \times 3$  linear functions  
are generated by a basis

$$\mathcal{F} = \text{span}\{\begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}\}$$

## Orbits

Basis vectors fall into three orbits

$$G(\square) = \{\square, \square, \square, \square, \square\}$$

$$G(\text{■}) \equiv \{\text{■}, \text{■}, \text{■}, \text{■}\}$$

$$G(\blacksquare) = \{\blacksquare\}$$

## Reduced Basis

The rotation-invariant functions have a three-dimensional basis

$$\Sigma_G \mathcal{F} = \text{span}\{\begin{smallmatrix} \text{H} & \text{X} \\ \text{X} & \text{H} \end{smallmatrix}, \begin{smallmatrix} \text{X} & \text{X} \\ \text{X} & \text{X} \end{smallmatrix}, \begin{smallmatrix} \text{■} & \text{■} \\ \text{■} & \text{■} \end{smallmatrix}\}$$

# Rotation-invariant $3 \times 3$ linear functions

## Original Basis

The  $3 \times 3$  linear functions are generated by a basis

$$\mathcal{F} = \text{span}\{\begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}\}$$

## Orbits

Basis vectors fall into three orbits

$$G(\begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}) = \{\begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}\}$$

$$G(\begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}) = \{\begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}\}$$

$$G(\begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}) = \{\begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}\}$$

## Reduced Basis

The rotation-invariant functions have a three-dimensional basis

$$\Sigma_G \mathcal{F} = \text{span}\{\begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}, \begin{matrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{matrix}\}$$

⇒ Rotation-invariance reduced dimensionality by a factor of 3

## $G$ -smoothing is an orthogonal projection

### Group action as permutation

Reminder: For a one-dimensional signal

$x \in \mathcal{X}(\Omega, \mathbb{R})$  over a finite domain  $\Omega$ , the group action is given through:

$$g.x(\omega) = x(g.\omega).$$

# $G$ -smoothing is an orthogonal projection

## Group action as permutation

Reminder: For a one-dimensional signal

$x \in \mathcal{X}(\Omega, \mathbb{R})$  over a finite domain  $\Omega$ , the group action is given through:

$$g.x(\omega) = x(g.\omega).$$

⇒ If we pack  $x$  into a vector  $\mathbf{x} \in \mathbb{R}^{|\Omega|}$ , then this corresponds to a permutation

$$g.\mathbf{x} = P_g \mathbf{x}.$$


## $G$ -smoothing is an orthogonal projection

### Group action as permutation

Reminder: For a one-dimensional signal  $x \in \mathcal{X}(\Omega, \mathbb{R})$  over a finite domain  $\Omega$ , the group action is given through:

$$g.x(\omega) = x(g.\omega).$$

⇒ If we pack  $x$  into a vector  $\mathbf{x} \in \mathbb{R}^{|\Omega|}$ , then this corresponds to a permutation

$$g.\mathbf{x} = P_g \mathbf{x}.$$

1	2	3
4	5	6
7	8	9

## $G$ -smoothing is an orthogonal projection

### Group action as permutation

Reminder: For a one-dimensional signal  $x \in \mathcal{X}(\Omega, \mathbb{R})$  over a finite domain  $\Omega$ , the group action is given through:

$$g.x(\omega) = x(g.\omega).$$

⇒ If we pack  $x$  into a vector  $\mathbf{x} \in \mathbb{R}^{|\Omega|}$ , then this corresponds to a permutation

$$g.\mathbf{x} = P_g \mathbf{x}.$$

$$\mathbf{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{pmatrix}$$

## $G$ -smoothing is an orthogonal projection

### Group action as permutation

Reminder: For a one-dimensional signal  $x \in \mathcal{X}(\Omega, \mathbb{R})$  over a finite domain  $\Omega$ , the group action is given through:

$$g.x(\omega) = x(g.\omega).$$

⇒ If we pack  $x$  into a vector  $\mathbf{x} \in \mathbb{R}^{|\Omega|}$ , then this corresponds to a permutation

$$g.\mathbf{x} = P_g \mathbf{x}.$$

1	2	3
4	5	6
7	8	9



$$\mathbf{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{pmatrix}$$

7	4	1
8	5	2
9	6	3

$$P_g \mathbf{x} = \begin{pmatrix} 7 \\ 4 \\ 1 \\ 8 \\ 5 \\ 2 \\ 9 \\ 6 \\ 3 \end{pmatrix}$$

## $G$ -smoothing is an orthogonal projection

$$\begin{aligned}|G|\langle \Sigma_G \mathbf{x}, \mathbf{y} \rangle &= \left\langle \sum_{g \in G} P_g \mathbf{x}, \mathbf{y} \right\rangle && (\text{def.}) \\&= \sum_{g \in G} \langle P_g \mathbf{x}, \mathbf{y} \rangle && (\text{linearity}) \\&= \sum_{g \in G} \langle \mathbf{x}, P_g^T \mathbf{y} \rangle && (\text{adjoint}) \\&= \sum_{g \in G} \langle \mathbf{x}, P_{g^{-1}} \mathbf{y} \rangle && (P_g^T = P_g^{-1}) \\&= \sum_{g' \in G} \langle \mathbf{x}, P_{g'} \mathbf{y} \rangle && (\text{change of var.}) \\&= \langle \mathbf{x}, \sum_{g' \in G} P_{g'} \mathbf{y} \rangle = |G| \langle \mathbf{x}, \Sigma_G \mathbf{y} \rangle\end{aligned}$$

## $G$ -smoothing is an orthogonal projection

$\Sigma_G$  is self-adjoint:  $\langle \Sigma_G f, g \rangle = \langle f, \Sigma_G g \rangle$

This property allows to decompose inner products

$$\begin{aligned}\langle f, g \rangle &= \langle \Sigma_G f + (I - \Sigma_G) f, \Sigma_G g + (I - \Sigma_G) g \rangle \\&= \langle \Sigma_G f, \Sigma_G g \rangle + \langle \Sigma_G f, (I - \Sigma_G) g \rangle + \langle (I - \Sigma_G) f, \Sigma_G g \rangle + \langle (I - \Sigma_G) f, (I - \Sigma_G) g \rangle \\&= \langle \Sigma_G f, \Sigma_G g \rangle + \langle f, (\Sigma_G - \Sigma_G^2) g \rangle + \langle (\Sigma_G - \Sigma_G^2) f, g \rangle + \langle (I - \Sigma_G) f, (I - \Sigma_G) g \rangle \\&= \langle \Sigma_G f, \Sigma_G g \rangle + \langle (I - \Sigma_G) f, (I - \Sigma_G) g \rangle\end{aligned}$$

## $G$ -smoothing is an orthogonal projection

$\Sigma_G$  is self-adjoint:  $\langle \Sigma_G f, g \rangle = \langle f, \Sigma_G g \rangle$

This property allows to decompose inner products

$$\begin{aligned}\langle f, g \rangle &= \langle \Sigma_G f + (I - \Sigma_G) f, \Sigma_G g + (I - \Sigma_G) g \rangle \\&= \langle \Sigma_G f, \Sigma_G g \rangle + \langle \Sigma_G f, (I - \Sigma_G) g \rangle + \langle (I - \Sigma_G) f, \Sigma_G g \rangle + \langle (I - \Sigma_G) f, (I - \Sigma_G) g \rangle \\&= \langle \Sigma_G f, \Sigma_G g \rangle + \langle f, (\Sigma_G - \Sigma_G^2) g \rangle + \langle (\Sigma_G - \Sigma_G^2) f, g \rangle + \langle (I - \Sigma_G) f, (I - \Sigma_G) g \rangle \\&= \langle \Sigma_G f, \Sigma_G g \rangle + \langle (I - \Sigma_G) f, (I - \Sigma_G) g \rangle\end{aligned}$$

and Euclidian distances

$$\|f - g\|^2 = \|\Sigma_G f - \Sigma_G g\|^2 + \|(I - \Sigma_G) f - (I - \Sigma_G) g\|^2$$

## $G$ -smoothing cannot increase $L_2$ error

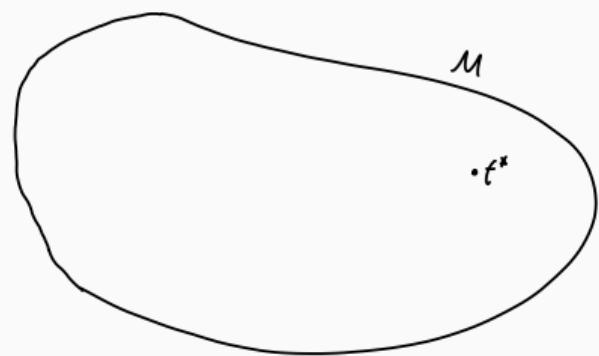
$$\begin{aligned}\|f - f^*\|^2 &= \|\Sigma_G f - \Sigma_G f^*\|^2 + \|(I - \Sigma_G)f - (I - \Sigma_G)f^*\|^2 \\&= \|\Sigma_G f - f^*\|^2 + \|(I - \Sigma_G)f\|^2 \\&\geq \|\Sigma_G f - f^*\|^2\end{aligned}$$

## $G$ -smoothing cannot increase $L_2$ error

$$\begin{aligned}\|f - f^*\|^2 &= \|\Sigma_G f - \Sigma_G f^*\|^2 + \|(I - \Sigma_G)f - (I - \Sigma_G)f^*\|^2 \\ &= \|\Sigma_G f - f^*\|^2 + \|(I - \Sigma_G)f\|^2 \\ &\geq \|\Sigma_G f - f^*\|^2\end{aligned}$$

$\implies$

$$\inf_{f \in \mathcal{F}} \|f - f^*\|^2 \geq \inf_{f \in \Sigma_G \mathcal{F}} \|f - f^*\|^2$$

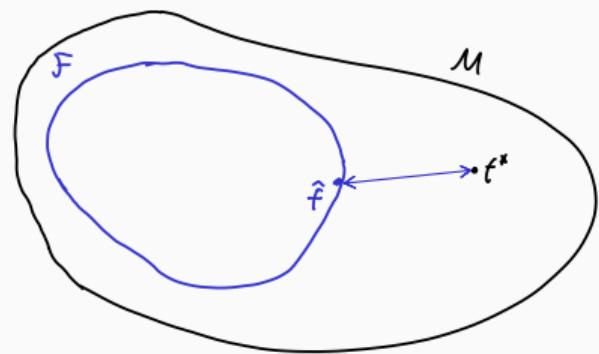


## $G$ -smoothing cannot increase $L_2$ error

$$\begin{aligned}\|f - f^*\|^2 &= \|\Sigma_G f - \Sigma_G f^*\|^2 + \|(I - \Sigma_G)f - (I - \Sigma_G)f^*\|^2 \\ &= \|\Sigma_G f - f^*\|^2 + \|(I - \Sigma_G)f\|^2 \\ &\geq \|\Sigma_G f - f^*\|^2\end{aligned}$$

$\implies$

$$\inf_{f \in \mathcal{F}} \|f - f^*\|^2 \geq \inf_{f \in \Sigma_G \mathcal{F}} \|f - f^*\|^2$$

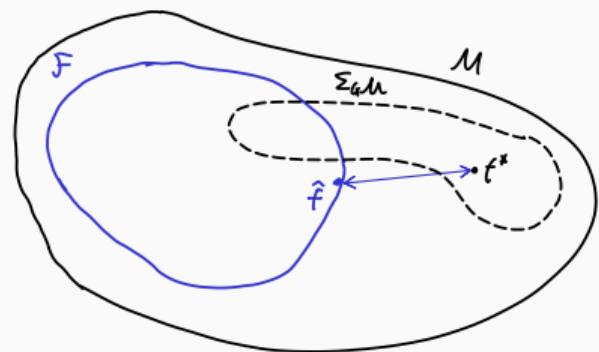


## $G$ -smoothing cannot increase $L_2$ error

$$\begin{aligned}\|f - f^*\|^2 &= \|\Sigma_G f - \Sigma_G f^*\|^2 + \|(I - \Sigma_G)f - (I - \Sigma_G)f^*\|^2 \\ &= \|\Sigma_G f - f^*\|^2 + \|(I - \Sigma_G)f\|^2 \\ &\geq \|\Sigma_G f - f^*\|^2\end{aligned}$$

$\implies$

$$\inf_{f \in \mathcal{F}} \|f - f^*\|^2 \geq \inf_{f \in \Sigma_G \mathcal{F}} \|f - f^*\|^2$$

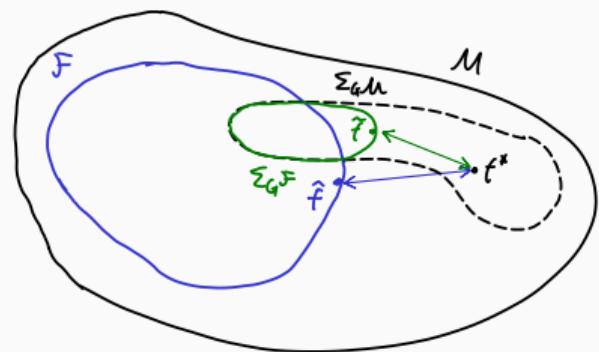


## $G$ -smoothing cannot increase $L_2$ error

$$\begin{aligned}\|f - f^*\|^2 &= \|\Sigma_G f - \Sigma_G f^*\|^2 + \|(I - \Sigma_G)f - (I - \Sigma_G)f^*\|^2 \\ &= \|\Sigma_G f - f^*\|^2 + \|(I - \Sigma_G)f\|^2 \\ &\geq \|\Sigma_G f - f^*\|^2\end{aligned}$$

$\implies$

$$\inf_{f \in \mathcal{F}} \|f - f^*\|^2 \geq \inf_{f \in \Sigma_G \mathcal{F}} \|f - f^*\|^2$$



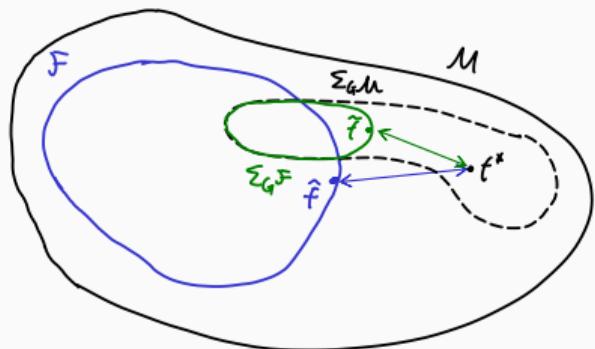
## $G$ -smoothing cannot increase $L_2$ error

$$\begin{aligned}\|f - f^*\|^2 &= \|\Sigma_G f - \Sigma_G f^*\|^2 + \|(I - \Sigma_G)f - (I - \Sigma_G)f^*\|^2 \\ &= \|\Sigma_G f - f^*\|^2 + \|(I - \Sigma_G)f\|^2 \\ &\geq \|\Sigma_G f - f^*\|^2\end{aligned}$$

$\implies$

$$\inf_{f \in \mathcal{F}} \|f - f^*\|^2 \geq \inf_{f \in \Sigma_G \mathcal{F}} \|f - f^*\|^2$$

If  $\Sigma_G \mathcal{F} \subset \mathcal{F}$ , we have equality.



## G-smoothing of Lipschitz functions

Let  $\mathcal{F}$  be the class of  $\beta$ -Lipschitz functions  
and  $G$  act isometrically,

$$|f(x) - f(x')| \leq \beta \|x - x'\| \|g.x - g.x'\| = \|x - x'\|.$$

## G-smoothing of Lipschitz functions

Let  $\mathcal{F}$  be the class of  $\beta$ -Lipschitz functions  
and  $G$  act isometrically,

$$|f(x) - f(x')| \leq \beta \|x - x'\| \|g.x - g.x'\| = \|x - x'\|.$$

$$|\tilde{f}(x) - \tilde{f}(x')| = \frac{1}{|G|} \left| \sum_g f(g.x) - \sum_g f(g.x') \right|$$

$$\forall h \in G : \quad \leq \frac{1}{|G|} \sum_g |f(g.x) - f(gh.x')|$$

$$\forall h \in G : \quad \leq \frac{\beta}{|G|} \sum_g \|g.x - hg.x'\|$$

$$\leq \beta \min_h \|x - h.x'\|$$

**Isometric property is necessary?**

Imagine  $\Omega = \mathbb{R}^2$ ,  $G = SO(2)$ ,

$$g.x = \begin{cases} \|x\| = 1 : R_g x \text{ } R_g \text{ 2d-rotation} \\ \|x\| \neq 1 : x \end{cases}$$

What is the "smoothed" version of  
 $f(x, y) = x$ ?

## G-smoothing of Lipschitz functions

Let  $\mathcal{F}$  be the class of  $\beta$ -Lipschitz functions  
and  $G$  act isometrically,

$$|f(x) - f(x')| \leq \beta \|x - x'\| \|g.x - g.x'\| = \|x - x'\|.$$

$$|\tilde{f}(x) - \tilde{f}(x')| = \frac{1}{|G|} \left| \sum_g f(g.x) - \sum_g f(g.x') \right|$$

$$\forall h \in G : \quad \leq \frac{1}{|G|} \sum_g |f(g.x) - f(gh.x')|$$

$$\forall h \in G : \quad \leq \frac{\beta}{|G|} \sum_g \|g.x - hg.x'\|$$

$$\leq \beta \min_h \|x - h.x'\|$$

**Isometric property is necessary?**

Imagine  $\Omega = \mathbb{R}^2$ ,  $G = SO(2)$ ,

$$g.x = \begin{cases} \|x\| = 1 : R_g x \text{ } R_g \text{ 2d-rotation} \\ \|x\| \neq 1 : x \end{cases}$$

What is the "smoothed" version of  
 $f(x, y) = x$ ?

A non-continuous function

$$\Sigma_G f(1, 1) = 0, \Sigma_G f(1, 1 + \epsilon) = 1$$

# Sample complexity with G-smoothing

## Ridge Regression

Using a  $G$ -invariant kernel ridge regression, the generalization error of learning a Lipschitz,  $G$ -invariant function  $f^*$  satisfies

$$\mathbb{E}[(f * (X) - \hat{f}(X))^2] < \mathcal{O}((|G|n)^{-1/d}),$$

for  $d$ -dimensions.

# Sample complexity with G-smoothing

## Ridge Regression

Using a  $G$ -invariant kernel ridge regression, the generalization error of learning a Lipschitz,  $G$ -invariant function  $f^*$  satisfies

$$\mathbb{E}[(f * (X) - \hat{f}(X))^2] < \mathcal{O}((|G|n)^{-1/d}),$$

for  $d$ -dimensions.

- Group size  $|G|$  could be exponential in  $d$ ,  $|G| \sim \alpha^d$ , does not prevent  $n^{-1/d}$
- Implementation very inefficient for large  $|G|$

## **Part VI: Group Representations**

---

## Group Representations: Linear Group Actions

- We want to be able to encode our objects of interest as vectors (of real numbers).

## Group Representations: Linear Group Actions

- We want to be able to encode our objects of interest as vectors (of real numbers).
- $\implies$  Group action:  $G \times \mathbb{R}^m \longrightarrow \mathbb{R}^m$

## Group Representations: Linear Group Actions

- We want to be able to encode our objects of interest as vectors (of real numbers).
- $\implies$  Group action:  $G \times \mathbb{R}^m \longrightarrow \mathbb{R}^m$
- What is the simplest way we can interact with vectors?

## Group Representations: Linear Group Actions

- We want to be able to encode our objects of interest as vectors (of real numbers).
- $\implies$  Group action:  $G \times \mathbb{R}^m \longrightarrow \mathbb{R}^m$
- What is the simplest way we can interact with vectors?
- Linear transformations.

## Group Representations: Linear Group Actions

- We want to be able to encode our objects of interest as vectors (of real numbers).
- $\implies$  Group action:  $G \times \mathbb{R}^m \longrightarrow \mathbb{R}^m$
- What is the simplest way we can interact with vectors?
- Linear transformations.

# Group Representations: Linear Group Actions

- We want to be able to encode our objects of interest as vectors (of real numbers).
- $\implies$  Group action:  $G \times \mathbb{R}^m \rightarrow \mathbb{R}^m$
- What is the simplest way we can interact with vectors?
- Linear transformations.

## Group Representation

A  $d$ -dimensional representation of a group  $G$  is a mapping

$$R : G \longrightarrow \mathrm{GL}(d) \subset \mathbb{R}^{d \times d}$$

that preserves the group structure (group *homomorphism*)

$$\forall g, h \in G : R(g \circ h) = R(g)R(h)$$

Implies group action

$$\forall g \in G, v \in \mathbb{R}^d : g.v \mapsto R(g)v$$

## Some examples of group representations

Given a group  $G$ , can you think of a  
1-dimensional representation?

## Some examples of group representations

### Trivial representation

Given a group  $G$ , can you think of a 1-dimensional representation?

$$G \ni g \mapsto 1$$

## Some examples of group representations

### Trivial representation

Given a group  $G$ , can you think of a 1-dimensional representation?

$$G \ni g \mapsto 1$$

### Faithful representation

A rep. is called *faithful* if it is injective.

# Some examples of group representations

## Trivial representation

Given a group  $G$ , can you think of a 1-dimensional representation?

$$G \ni g \mapsto 1$$

## Faithful representation

A rep. is called *faithful* if it is injective.

## Two-dimensional rotations

$$r_\theta \mapsto \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

## Permutation group

The permutation group  $S(n)$  can be represented in  $\mathbb{R}^n$  with permutation matrices, e.g.,

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

# Some examples of group representations

## Trivial representation

Given a group  $G$ , can you think of a 1-dimensional representation?

$$G \ni g \mapsto 1$$

## Faithful representation

A rep. is called *faithful* if it is injective.

## Two-dimensional rotations

$$r_\theta \mapsto \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

## Permutation group

The permutation group  $S(n)$  can be represented in  $\mathbb{R}^n$  with permutation matrices, e.g.,

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

**Note:** Transforming adjacency matrices is *not* a group representation:  
 $g.A = P(g)AP(g)^T$

## Representing Translations

A simple group action for two-dimensional translations  $T_2 := (\mathbb{R}^2, +)$ :

$$\mathbf{t} \in T_2, \mathbf{s} \in \mathbb{R}^2; (\mathbf{t}, \mathbf{s}) \mapsto \mathbf{t} + \mathbf{s}.$$

**Not a representation** ( $\forall \mathbf{t} : R(\mathbf{t})\mathbf{0} = \mathbf{0}$ )

## Representing Translations

A simple group action for two-dimensional translations  $T_2 := (\mathbb{R}^2, +)$ :

$$\mathbf{t} \in T_2, \mathbf{s} \in \mathbb{R}^2; (\mathbf{t}, \mathbf{s}) \mapsto \mathbf{t} + \mathbf{s}.$$

Composition:

$$\begin{pmatrix} e^u & 0 \\ 0 & e^v \end{pmatrix} \cdot \begin{pmatrix} e^p & 0 \\ 0 & e^q \end{pmatrix} = \begin{pmatrix} e^{u+p} & 0 \\ 0 & e^{v+q} \end{pmatrix}$$

**Not a representation** ( $\forall \mathbf{t} : R(\mathbf{t})\mathbf{0} = \mathbf{0}$ )

We can easily construct a two-dimensional representation through

$$(u, v) \mapsto \begin{pmatrix} \exp(u) & 0 \\ 0 & \exp(v) \end{pmatrix}$$

## Representing Translations

A simple group action for two-dimensional translations  $T_2 := (\mathbb{R}^2, +)$ :

$$\mathbf{t} \in T_2, \mathbf{s} \in \mathbb{R}^2; (\mathbf{t}, \mathbf{s}) \mapsto \mathbf{t} + \mathbf{s}.$$

**Not a representation** ( $\forall \mathbf{t} : R(\mathbf{t})\mathbf{0} = \mathbf{0}$ )

We can easily construct a two-dimensional representation through

$$(u, v) \mapsto \begin{pmatrix} \exp(u) & 0 \\ 0 & \exp(v) \end{pmatrix}$$

Composition:

$$\begin{pmatrix} e^u & 0 \\ 0 & e^v \end{pmatrix} \cdot \begin{pmatrix} e^p & 0 \\ 0 & e^q \end{pmatrix} = \begin{pmatrix} e^{u+p} & 0 \\ 0 & e^{v+q} \end{pmatrix}$$

Action:

$$\begin{pmatrix} e^u & 0 \\ 0 & e^v \end{pmatrix} \cdot \begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} e^u s \\ e^v t \end{pmatrix}$$

## Representing Translations

A simple group action for two-dimensional translations  $T_2 := (\mathbb{R}^2, +)$ :

$$\mathbf{t} \in T_2, \mathbf{s} \in \mathbb{R}^2; (\mathbf{t}, \mathbf{s}) \mapsto \mathbf{t} + \mathbf{s}.$$

**Not a representation** ( $\forall \mathbf{t} : R(\mathbf{t})\mathbf{0} = \mathbf{0}$ )

We can easily construct a two-dimensional representation through

$$(u, v) \mapsto \begin{pmatrix} \exp(u) & 0 \\ 0 & \exp(v) \end{pmatrix}$$

Composition:

$$\begin{pmatrix} e^u & 0 \\ 0 & e^v \end{pmatrix} \cdot \begin{pmatrix} e^p & 0 \\ 0 & e^q \end{pmatrix} = \begin{pmatrix} e^{u+p} & 0 \\ 0 & e^{v+q} \end{pmatrix}$$

Action:

$$\begin{pmatrix} e^u & 0 \\ 0 & e^v \end{pmatrix} \cdot \begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} e^u s \\ e^v t \end{pmatrix}$$

**does not match desired group action**

## Embed group action into larger space to enable linear representation

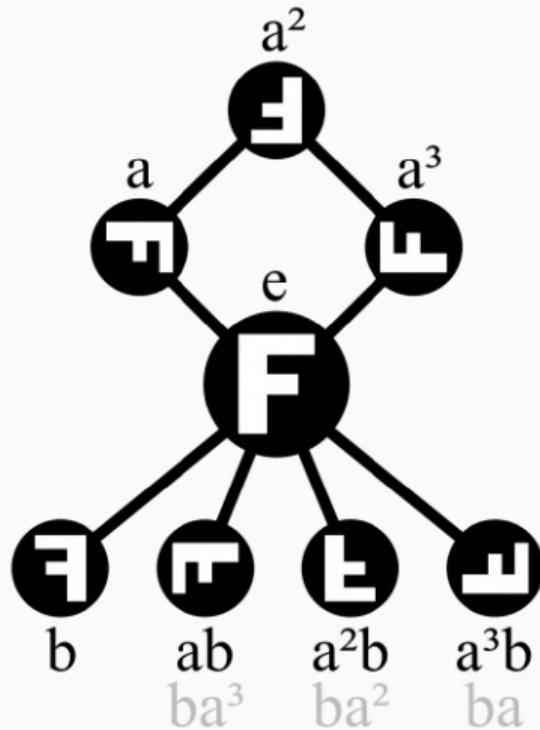
Expand target space to  $\mathbb{R}^2 \times \{1\} \subset \mathbb{R}^3$ .

$$(u, v) \mapsto \begin{pmatrix} 1 & 0 & u \\ 0 & 1 & v \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & u \\ 0 & 1 & v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + u \\ y + v \\ 1 \end{pmatrix}$$

## Constructing representations of D4

- Observation: Specifying representation of generator is enough.
- For D4: rotation by  $90^\circ$   $a$ , horizontal flip  $b$
- $D_4 = \{e, a, a^2, a^3, b, ab, a^2b, a^3b\}$



## Constructing representations of D4: Necessarily Conditions

Let's assume  $R(a) = A$ ,  $R(b) = B$ .

**What do we know about these matrices?**

## Constructing representations of D4: Necessarily Conditions

Let's assume  $R(a) = A$ ,  $R(b) = B$ .

**What do we know about these matrices?**

$$I = R(e) = R(a^4) = A^4$$

## Constructing representations of D4: Necessarily Conditions

Let's assume  $R(a) = A$ ,  $R(b) = B$ .

**What do we know about these matrices?**

$$I = R(e) = R(a^4) = A^4 \implies \lambda \text{ eigenvalue of } A \Leftrightarrow \lambda^4 = 1$$

## Constructing representations of D4: Necessarily Conditions

Let's assume  $R(a) = A$ ,  $R(b) = B$ .

**What do we know about these matrices?**

$$I = R(e) = R(a^4) = A^4 \implies \lambda \text{ eigenvalue of } A \Leftrightarrow \lambda^4 = 1$$

$$I = R(e) = R(b^2) = B^2 \implies \mu \text{ eigenvalue of } B \Leftrightarrow \mu^2 = 1$$

## Constructing representations of D4: Necessarily Conditions

Let's assume  $R(a) = A$ ,  $R(b) = B$ .

**What do we know about these matrices?**

$$I = R(e) = R(a^4) = A^4 \implies \lambda \text{ eigenvalue of } A \Leftrightarrow \lambda^4 = 1$$

$$I = R(e) = R(b^2) = B^2 \implies \mu \text{ eigenvalue of } B \Leftrightarrow \mu^2 = 1$$

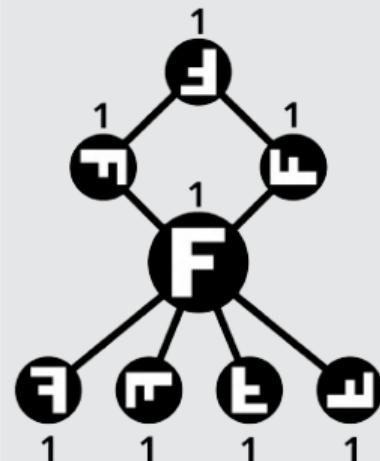
Note: Eigenvalues might be complex, so  $\lambda \in \{1, -1, i, -i\}$ ,  $\mu \in \{1, -1\}$

# Constructing one-dimensional representations of D4

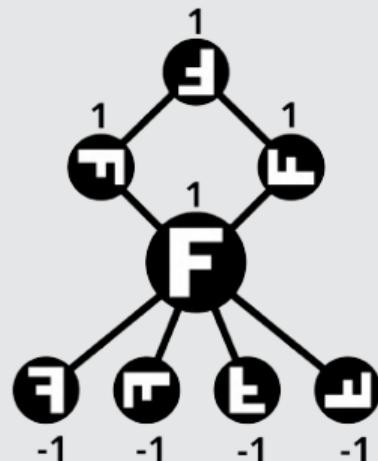
## Candidates

$A = (\lambda), B = (\mu)$ , real matrices  $\implies \lambda \in 1, -1$ .

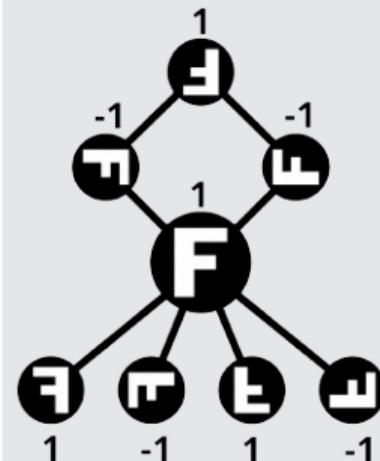
**A1:**  $A = 1, B = 1$



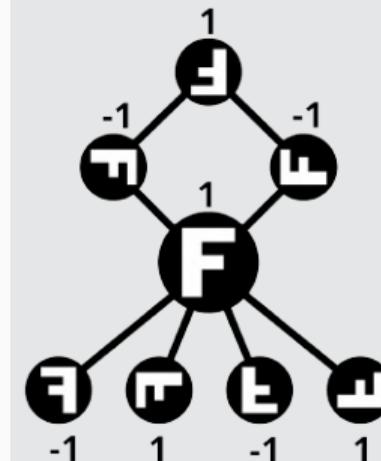
**A2:**  $A = 1, B = -1$



**B1:**  $A = -1, B = 1$



**B2:**  $A = -1, B = -1$



**In two dimensions, there is a natural representation of D4**

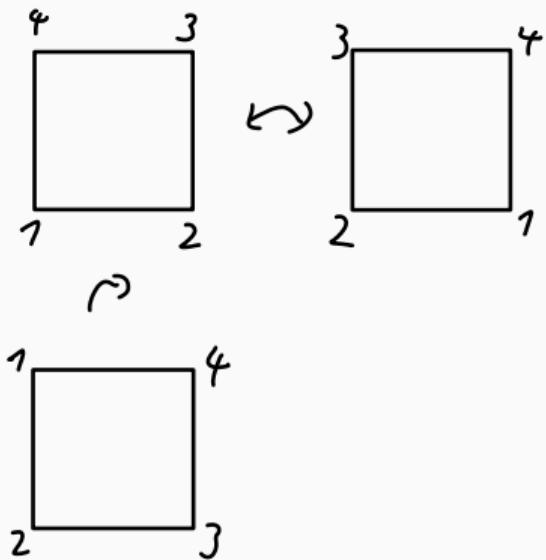
**E: Two dimensions**

Use geometric intuition do write down  
representation:

In two dimensions, there is a natural representation of D4

### E: Two dimensions

Use geometric intuition do write down representation:



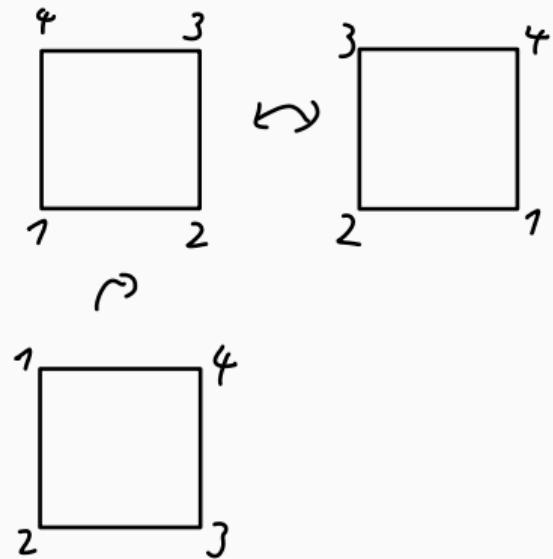
In two dimensions, there is a natural representation of D4

### E: Two dimensions

Use geometric intuition do write down representation:

$$R(a) = \begin{pmatrix} \cos 90 & -\sin 90 \\ \sin 90 & \cos 90 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

$$R(b) = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$



In  $\mathbb{C}$ , we can diagonalize this representation

This representation uses the complex Eigenvalues  $i, -i$ .

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = 0.5 \begin{pmatrix} -i & i \\ 1 & 1 \end{pmatrix} \begin{pmatrix} -i & 0 \\ 0 & i \end{pmatrix} \begin{pmatrix} i & 1 \\ -i & 1 \end{pmatrix}$$

In  $\mathbb{C}$ , we can diagonalize this representation

This representation uses the complex Eigenvalues  $i, -i$ .

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = 0.5 \begin{pmatrix} -i & i \\ 1 & 1 \end{pmatrix} \begin{pmatrix} -i & 0 \\ 0 & i \end{pmatrix} \begin{pmatrix} i & 1 \\ -i & 1 \end{pmatrix}$$

Could we have another representation using  $+i, +i$  as Eigenvalues?

In  $\mathbb{C}$ , we can diagonalize this representation

This representation uses the complex Eigenvalues  $i, -i$ .

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = 0.5 \begin{pmatrix} -i & i \\ 1 & 1 \end{pmatrix} \begin{pmatrix} -i & 0 \\ 0 & i \end{pmatrix} \begin{pmatrix} i & 1 \\ -i & 1 \end{pmatrix}$$

Could we have another representation using  $+i, +i$  as Eigenvalues?

Yes

$$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = 0.5 \begin{pmatrix} i & -i \\ 1 & 1 \end{pmatrix} \begin{pmatrix} i & 0 \\ 0 & i \end{pmatrix} \begin{pmatrix} 1 & i \\ -1 & i \end{pmatrix}$$

In  $\mathbb{C}$ , we can diagonalize this representation

This representation uses the complex Eigenvalues  $i, -i$ .

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = 0.5 \begin{pmatrix} -i & i \\ 1 & 1 \end{pmatrix} \begin{pmatrix} -i & 0 \\ 0 & i \end{pmatrix} \begin{pmatrix} i & 1 \\ -i & 1 \end{pmatrix}$$

Could we have another representation using  $+i, +i$  as Eigenvalues?

Yes

$$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = 0.5 \begin{pmatrix} i & -i \\ 1 & 1 \end{pmatrix} \begin{pmatrix} i & 0 \\ 0 & i \end{pmatrix} \begin{pmatrix} 1 & i \\ -1 & i \end{pmatrix}$$

This is just two times B1 stacked on the diagonal!

## Representations can be concatenated

Let  $A$  and  $B$  be two representations of  $G$  with dimension  $d_A$  and  $d_B$ . Then  $A \oplus B$ , the *direct sum* of  $A$  and  $B$ , defined through

$$(A \oplus B)(g) = \begin{pmatrix} A(g) & 0 \\ 0 & B(g) \end{pmatrix} \in \mathrm{GL}(d_A + d_B),$$

is a representation of  $G$ .

## Representations can be concatenated

Let  $A$  and  $B$  be two representations of  $G$  with dimension  $d_A$  and  $d_B$ . Then  $A \oplus B$ , the *direct sum* of  $A$  and  $B$ , defined through

$$(A \oplus B)(g) = \begin{pmatrix} A(g) & 0 \\ 0 & B(g) \end{pmatrix} \in \mathrm{GL}(d_A + d_B),$$

is a representation of  $G$ .

### Example

Describing a spinning particle.

- Velocity:  $E$  (**vector**)
- Mass:  $A_1$  (**scalar**)
- Spin:  $A_2$  (**pseudoscalar**)

Full system:  $E \oplus A_1 \oplus A_2$

## Representations are preserved under change of basis

Let  $J \in \mathrm{GL}(d)$ , and  $R$  be a  $d$ -dimensional representation of group  $G$ .  
Then  $R_J : G \longrightarrow \mathrm{GL}(d)$ ,  $g \mapsto JR(g)J^{-1}$  is a representation of  $G$ .

## Representations are preserved under change of basis

Let  $J \in \mathrm{GL}(d)$ , and  $R$  be a  $d$ -dimensional representation of group  $G$ .

Then  $R_J : G \longrightarrow \mathrm{GL}(d)$ ,  $g \mapsto JR(g)J^{-1}$  is a representation of  $G$ .

$$R_J(g \circ h) = JR(g \circ h)J^{-1} \quad (\text{Definition of } R_J)$$

$$= JR(g)R(h)J^{-1} \quad (R \text{ is rep.})$$

$$= JR(g)J^{-1}JR(h)J^{-1} \quad (\text{inserting identity})$$

$$= R_J(g)R_J(h). \quad (\text{Definition of } R_J)$$

## Representations are preserved under change of basis

Let  $J \in \mathrm{GL}(d)$ , and  $R$  be a  $d$ -dimensional representation of group  $G$ .

Then  $R_J : G \longrightarrow \mathrm{GL}(d)$ ,  $g \mapsto JR(g)J^{-1}$  is a representation of  $G$ .

$$R_J(g \circ h) = JR(g \circ h)J^{-1} \quad (\text{Definition of } R_J)$$

$$= JR(g)R(h)J^{-1} \quad (R \text{ is rep.})$$

$$= JR(g)J^{-1}JR(h)J^{-1} \quad (\text{inserting identity})$$

$$= R_J(g)R_J(h). \quad (\text{Definition of } R_J)$$

Such representations are called **equivalent**.

## Representations are preserved under change of basis

Let  $J \in \mathrm{GL}(d)$ , and  $R$  be a  $d$ -dimensional representation of group  $G$ .

Then  $R_J : G \longrightarrow \mathrm{GL}(d)$ ,  $g \mapsto JR(g)J^{-1}$  is a representation of  $G$ .

$$\begin{aligned} R_J(g \circ h) &= JR(g \circ h)J^{-1} && \text{(Definition of } R_J\text{)} \\ &= JR(g)R(h)J^{-1} && (R \text{ is rep.}) \\ &= JR(g)J^{-1}JR(h)J^{-1} && \text{(inserting identity)} \\ &= R_J(g)R_J(h). && \text{(Definition of } R_J\text{)} \end{aligned}$$

Such representations are called **equivalent**.

In particular  $A \oplus B$  and  $B \oplus A$  are equivalent.

## Reducible representations

A representation is called **reducible** if it has a non-trivial *invariant subspace*:

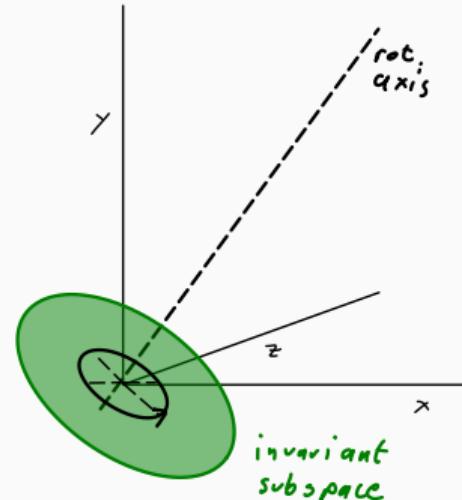
$$\forall g \in G, v \in V \subset \mathbb{R}^d : R(g)v \in V.$$

## Reducible representations

A representation is called **reducible** if it has a non-trivial *invariant subspace*:

$$\forall g \in G, v \in V \subset \mathbb{R}^d : R(g)v \in V.$$

2D rotation in 3D space:



## Reducible representations

A representation is called **reducible** if it has a non-trivial *invariant subspace*:

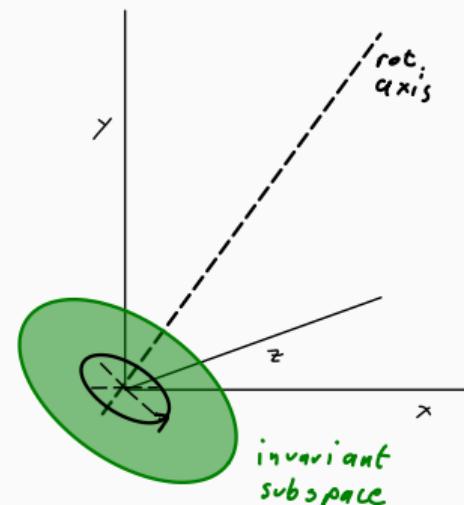
$$\forall g \in G, v \in V \subset \mathbb{R}^d : R(g)v \in V.$$

This implies that it is equivalent to a representation with upper-triangular matrices:

$$\exists J : J^{-1}R(g)J = \begin{pmatrix} A & B \\ 0 & C \end{pmatrix}$$

with  $C \in \mathrm{GL}(\dim V)$ .

2D rotation in 3D space:



## Reducible representations

A representation is called **reducible** if it has a non-trivial *invariant subspace*:

$$\forall g \in G, v \in V \subset \mathbb{R}^d : R(g)v \in V.$$

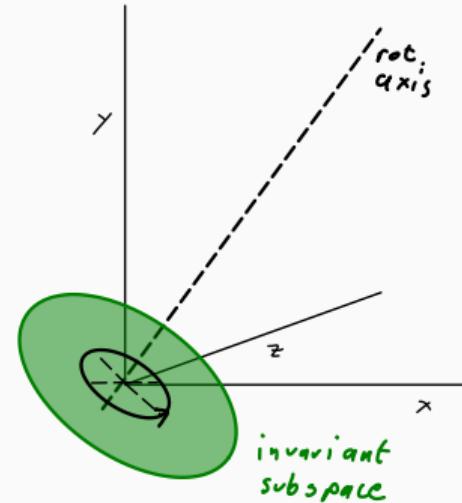
This implies that it is equivalent to a representation with upper-triangular matrices:

$$\exists J : J^{-1}R(g)J = \begin{pmatrix} A & B \\ 0 & C \end{pmatrix}$$

with  $C \in \mathrm{GL}(\dim V)$ .

- If  $B = 0$ , it is called **decomposable**.

2D rotation in 3D space:



## Reducible representations

A representation is called **reducible** if it has a non-trivial *invariant subspace*:

$$\forall g \in G, v \in V \subset \mathbb{R}^d : R(g)v \in V.$$

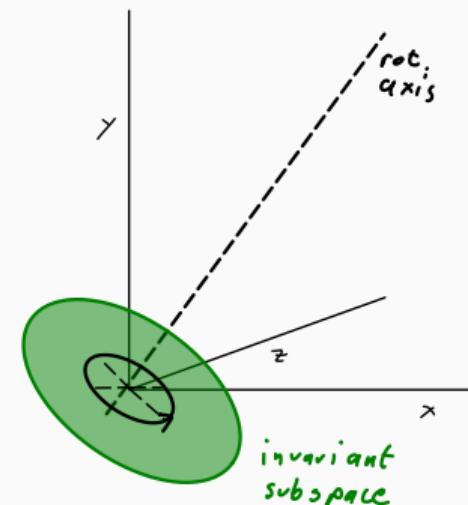
This implies that it is equivalent to a representation with upper-triangular matrices:

$$\exists J : J^{-1}R(g)J = \begin{pmatrix} A & B \\ 0 & C \end{pmatrix}$$

with  $C \in \mathrm{GL}(\dim V)$ .

- If  $B = 0$ , it is called **decomposable**.
- Otherwise, the representation is **irreducible (indecomposable)**.

2D rotation in 3D space:



## Reducible representations

A representation is called **reducible** if it has a non-trivial *invariant subspace*:

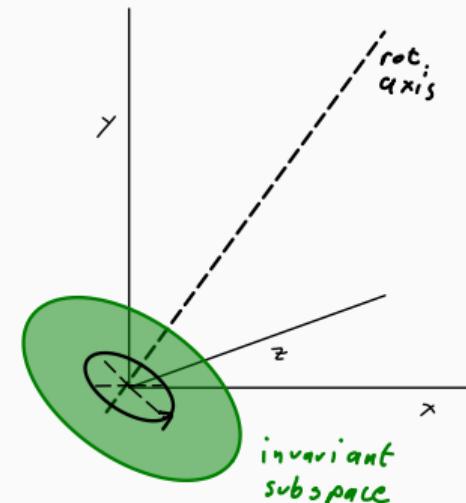
$$\forall g \in G, v \in V \subset \mathbb{R}^d : R(g)v \in V.$$

This implies that it is equivalent to a representation with upper-triangular matrices:

$$\exists J : J^{-1}R(g)J = \begin{pmatrix} A & B \\ 0 & C \end{pmatrix}$$

with  $C \in \mathrm{GL}(\dim V)$ .

2D rotation in 3D space:



- If  $B = 0$ , it is called **decomposable**.
- Otherwise, the representation is **irreducible (indecomposable)**.
- For finite groups these are equivalent (Maschke's theorem).

## The number of irreps is limited by the group size

A result in group theory (“Main theorem”<sup>1</sup>) tells us that the set of all irreducible representations (**irreps**)  $R_1, \dots, R_k$  of a group  $G$  fulfills

$$|G| = \sum_{i=1}^k \dim(R_k)^2$$

---

<sup>1</sup>Artin. (2010). *Algebra, 2nd edition*

## The number of irreps is limited by the group size

A result in group theory (“Main theorem”<sup>1</sup>) tells us that the set of all irreducible representations (**irreps**)  $R_1, \dots, R_k$  of a group  $G$  fulfills

$$|G| = \sum_{i=1}^k \dim(R_i)^2$$

For our previous example:

$$\underbrace{\dim(A_1)^2}_1 + \underbrace{\dim(A_2)^2}_1 + \underbrace{\dim(B_1)^2}_1 + \underbrace{\dim(B_2)^2}_1 + \underbrace{\dim(E)^2}_4 = 8 = |D_4|$$

---

<sup>1</sup>Artin. (2010). *Algebra, 2nd edition*

## The number of irreps is limited by the group size

A result in group theory (“Main theorem”<sup>1</sup>) tells us that the set of all irreducible representations (**irreps**)  $R_1, \dots, R_k$  of a group  $G$  fulfills

$$|G| = \sum_{i=1}^k \dim(R_i)^2$$

For our previous example:

$$\underbrace{\dim(A_1)^2}_1 + \underbrace{\dim(A_2)^2}_1 + \underbrace{\dim(B_1)^2}_1 + \underbrace{\dim(B_2)^2}_1 + \underbrace{\dim(E)^2}_4 = 8 = |D_4|$$

⇒ We found *all* irreps.

---

<sup>1</sup>Artin. (2010). *Algebra, 2nd edition*

## The regular representation

Let  $G$  be a group,  $\mathcal{X}(G, \mathbb{R}) \cong \mathbb{R}^{|G|}$  the space of signals on that group. The *regular* representation of  $G$  is given through

$$R(g)\hat{e}_h = \hat{e}_{gh},$$

i.e., it maps one-hot encoding for group element  $h$  to one-hot vectors for  $gh$ .

## The regular representation

Let  $G$  be a group,  $\mathcal{X}(G, \mathbb{R}) \cong \mathbb{R}^{|G|}$  the space of signals on that group. The *regular* representation of  $G$  is given through

$$R(g)\hat{e}_h = \hat{e}_{gh},$$

i.e., it maps one-hot encoding for group element  $h$  to one-hot vectors for  $gh$ .

Example:  $90^\circ$  rotations

$$\begin{matrix} 0 & \curvearrowright \\ \curvearrowleft & 0 \end{matrix}$$

$$\begin{matrix} 1 & \cdot \\ \curvearrowright & 0 \end{matrix}$$

## The regular representation

Let  $G$  be a group,  $\mathcal{X}(G, \mathbb{R}) \cong \mathbb{R}^{|G|}$  the space of signals on that group. The *regular* representation of  $G$  is given through

$$R(g)\hat{e}_h = \hat{e}_{gh},$$

i.e., it maps one-hot encoding for group element  $h$  to one-hot vectors for  $gh$ .

Example:  $90^\circ$  rotations

$$1 \xrightarrow{\text{r}} \begin{smallmatrix} 0 \\ 1 \end{smallmatrix}$$

$$\curvearrowright$$

$$0 \cdot \xrightarrow{\text{r}} 0$$

## Regular Representation for Cyclic Shifts

For the group of cyclic shifts  $C(n) = \{e, g, g^2, \dots, g^{n-1}\}$ ,  
how does the regular representation look like?

---

## Regular Representation for Cyclic Shifts

For the group of cyclic shifts  $C(n) = \{e, g, g^2, \dots, g^{n-1}\}$ ,  
how does the regular representation look like?

$$R(g) = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

## Regular Representation for Cyclic Shifts

For the group of cyclic shifts  $C(n) = \{e, g, g^2, \dots, g^{n-1}\}$ ,  
how does the regular representation look like?

$$R(g) = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Since shifts commute, these matrices are *simultaneously  
diagonalizable*.

---

## Regular Representation for Cyclic Shifts

For the group of cyclic shifts  $C(n) = \{e, g, g^2, \dots, g^{n-1}\}$ ,  
how does the regular representation look like?

$$R(g) = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Since shifts commute, these matrices are *simultaneously diagonalizable*.

If  $R(g) = JDJ^{-1}$ , then  $R(g^k) = JD^k J^{-1}$

---

## Regular Representation for Cyclic Shifts

For the group of cyclic shifts  $C(n) = \{e, g, g^2, \dots, g^{n-1}\}$ ,  
how does the regular representation look like?

$$R(g) = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Since shifts commute, these matrices are *simultaneously diagonalizable*.

If  $R(g) = JDJ^{-1}$ , then  $R(g^k) = JD^k J^{-1}$

There are  $n$  1-dimensional irreps. in  $\mathbb{C}$

---

## Regular Representation for Cyclic Shifts

For the group of cyclic shifts  $C(n) = \{e, g, g^2, \dots, g^{n-1}\}$ ,  
how does the regular representation look like?

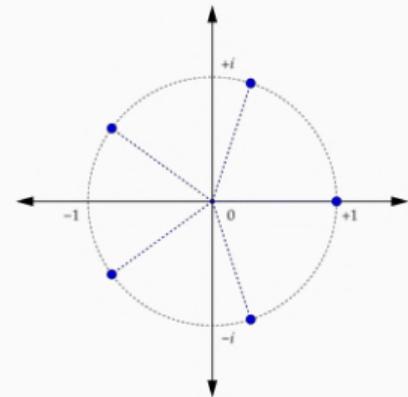
$$R(g) = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Since shifts commute, these matrices are *simultaneously diagonalizable*.

If  $R(g) = JDJ^{-1}$ , then  $R(g^k) = JD^k J^{-1}$

There are  $n$  1-dimensional irreps. in  $\mathbb{C}$

Roots of unit  $\sqrt[n]{1}$



## Summary

- Representations are linear group actions  $g.v = R(g)v$ ,  $R(g) \in \mathrm{GL}(d)$
- Can be built up from *irreducible representations*
- *Regular representation* acts on the group itself

## References

---

- [1] Michael Artin. **Algebra, 2nd edition.** Pearson, 2010.
- [2] Alberto Bietti, Luca Venturi, and Joan Bruna. “**On the Sample Complexity of Learning under Geometric Stability**”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 18673–18684.

# Geometric Deep Learning

---

Erik Schultheis, Kate Haitsiukevich, Çağlar Hızlı, Alison Pouplin, Vikas Garg

7.11.2024

## **Part VII: Convolutions**

---

## A note on terminology

Caveat: Convolution operation

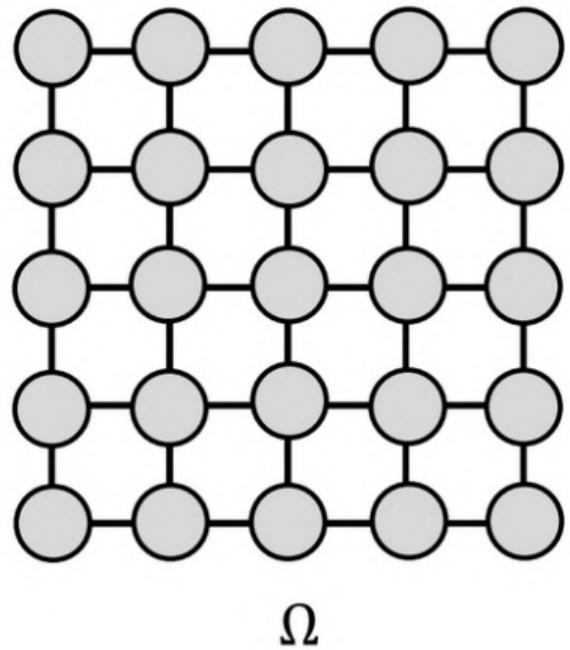
$$(f \star g)(t) = \int_0^T f(\tau)g(T - \tau),$$

but in ML we call *cross-correlation* convolution

$$(f \star g)(t) = \int_0^T f(\tau)g(\tau).$$

# Grids vs Graphs

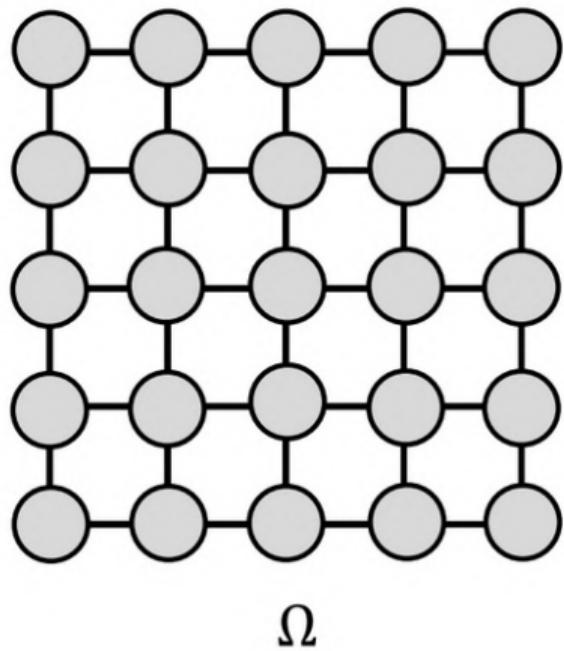
What makes the geometric domain of a grid different from that of a graph?



# Grids vs Graphs

What makes the geometric domain of a grid different from that of a graph?

The symmetry group. Each node has well-defined top/bottom/left/right neighbour  $\Rightarrow$  not *permutation-invariant*

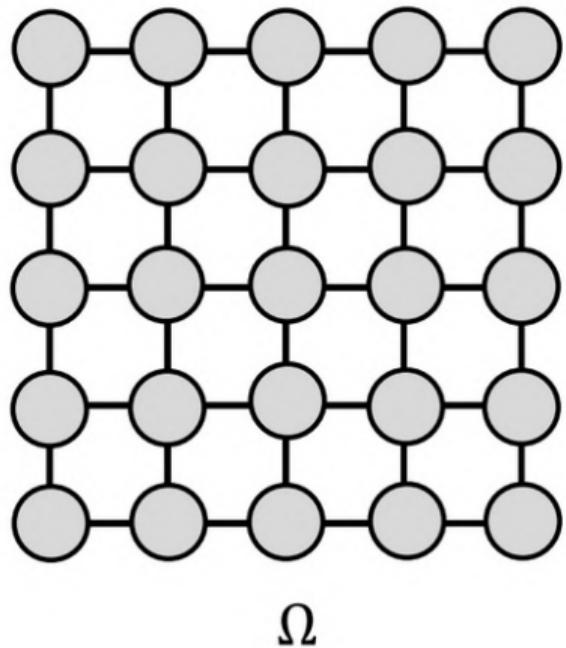


# Grids vs Graphs

What makes the geometric domain of a grid different from that of a graph?

The symmetry group. Each node has well-defined top/bottom/left/right neighbour  $\Rightarrow$  not *permutation-invariant*

But *shift-invariant* (periodic boundary)



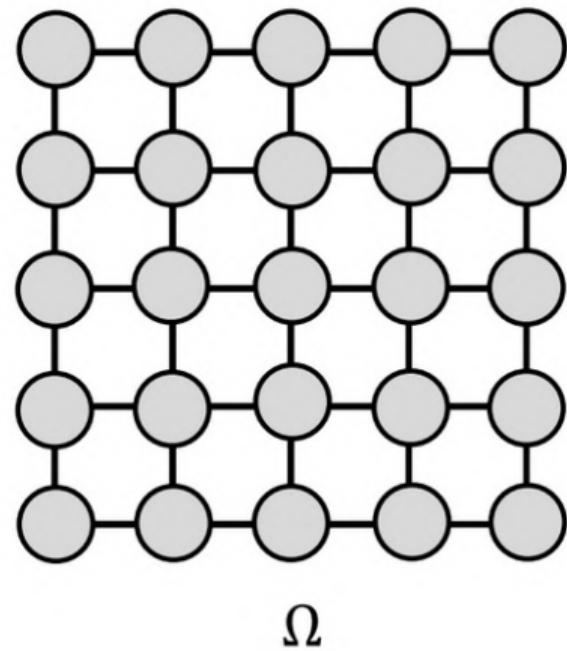
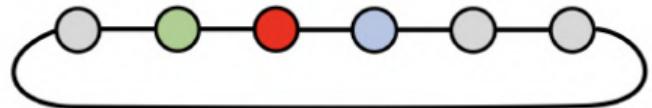
# Grids vs Graphs

What makes the geometric domain of a grid different from that of a graph?

The symmetry group. Each node has well-defined top/bottom/left/right neighbour  $\Rightarrow$  not *permutation-invariant*

But *shift-invariant* (periodic boundary)

For simplicity: limit to one dimension



## Formal setup

Let  $G = C(n)$  be the cyclic group of order  $n$ ,  
and  $\Omega = 1, \dots, n$  be the domain of sequences  
of length  $n$  with periodic boundary conditions



## Formal setup

Let  $G = C(n)$  be the cyclic group of order  $n$ ,  
and  $\Omega = 1, \dots, n$  be the domain of sequences  
of length  $n$  with periodic boundary conditions

**What are linear equivariant mappings?**



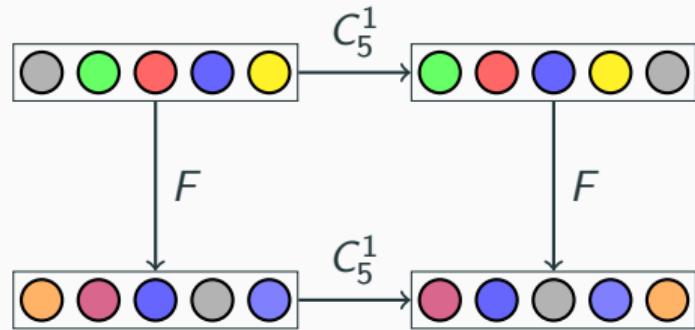
## Formal setup

Let  $G = C(n)$  be the cyclic group of order  $n$ ,  
and  $\Omega = 1, \dots, n$  be the domain of sequences  
of length  $n$  with periodic boundary conditions

**What are linear equivariant mappings?**

We want  $CF\mathbf{a} = FC\mathbf{a}$ ,

$C$  shift-matrix,  $\mathbf{a}$  activations/inputs,  $F$   
equivariant mapping



## Eigendecomposition of shift-matrices

The regular representation of  $C(n)$  is given by circulant matrices

$$C_n^k := \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}^k$$

## Eigendecomposition of shift-matrices

The regular representation of  $C(n)$  is given by circulant matrices

$$C_n^k := \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}^k$$

We know:

- Over  $\mathbb{C}$ , the representation is decomposable

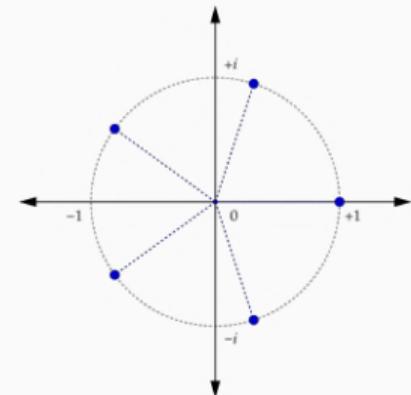
## Eigendecomposition of shift-matrices

The regular representation of  $C(n)$  is given by circulant matrices

$$C_n^k := \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}^k$$

We know:

- Over  $\mathbb{C}$ , the representation is decomposable
- The irreps of  $C(n)$  are the roots of unity  $\psi_n^k = \exp(2\pi ik/n)$



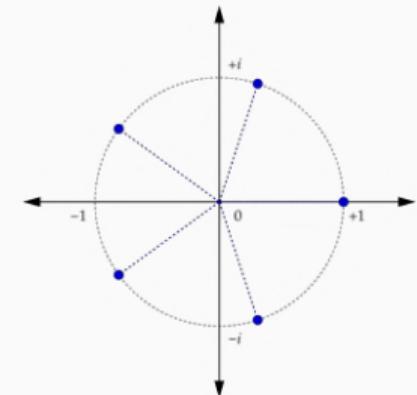
## Eigendecomposition of shift-matrices

The regular representation of  $C(n)$  is given by circulant matrices

$$C_n^k := \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}^k$$

We know:

- Over  $\mathbb{C}$ , the representation is decomposable
- The irreps of  $C(n)$  are the roots of unity  $\psi_n^k = \exp(2\pi ik/n)$
- $\Rightarrow C_n^k = J^{-1}DJ$  for *diagonal*  $D$  with entries  $\psi_n^k$



## Gaussian elimination

To find  $J$ , we can solve

$$(C_n^1 - \exp(2\pi ik/n)I)\mathbf{v} = 0$$

using Gaussian elimination.

## Gaussian elimination

To find  $J$ , we can solve

$$(C_n^1 - \exp(2\pi ik/n)I)\mathbf{v} = 0$$

using Gaussian elimination.

$$\begin{pmatrix} -\psi_n^k & 1 & 0 & \dots & 0 \\ 0 & -\psi_n^k & 1 & \dots & 0 \\ 0 & 0 & -\psi_n^k & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 1 & 0 & 0 & 0 & -\psi_n^k \end{pmatrix}$$

## Gaussian elimination

To find  $J$ , we can solve

$$(C_n^1 - \exp(2\pi ik/n)I)\mathbf{v} = 0$$

using Gaussian elimination.

$$\left( \begin{array}{ccccc} -\psi_n^k & 1 & 0 & \dots & 0 \\ 0 & -\psi_n^k & 1 & \dots & 0 \\ 0 & 0 & -\psi_n^k & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 1 & 0 & 0 & 0 & -\psi_n^k \end{array} \right) \left| \begin{array}{l} \cdot \psi_n^{n-k} \\ \text{-row 1} \end{array} \right.$$

## Gaussian elimination

To find  $J$ , we can solve

$$(C_n^1 - \exp(2\pi ik/n)I)\mathbf{v} = 0$$

using Gaussian elimination.

$$\begin{pmatrix} -\psi_n^k & 1 & 0 & \dots & 0 \\ 0 & -\psi_n^k & 1 & \dots & 0 \\ 0 & 0 & -\psi_n^k & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & \psi_n^{n-k} & 0 & 0 & -\psi_n^k \end{pmatrix}$$

## Gaussian elimination

To find  $J$ , we can solve

$$(C_n^1 - \exp(2\pi ik/n)I)\mathbf{v} = 0$$

using Gaussian elimination.

$$\begin{pmatrix} -\psi_n^k & 1 & 0 & \dots & 0 \\ 0 & -\psi_n^k & 1 & \dots & 0 \\ 0 & 0 & -\psi_n^k & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & \psi_n^{n-2k} & 0 & -\psi_n^k \end{pmatrix}$$

## Gaussian elimination

To find  $J$ , we can solve

$$(C_n^1 - \exp(2\pi ik/n)I)\mathbf{v} = 0$$

using Gaussian elimination.

$$\begin{pmatrix} -\psi_n^k & 1 & 0 & \dots & 0 \\ 0 & -\psi_n^k & 1 & \dots & 0 \\ 0 & 0 & -\psi_n^k & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & 0 & \psi_n^{n-(n-2)k} & -\psi_n^k \end{pmatrix}$$

## Gaussian elimination

To find  $J$ , we can solve

$$(C_n^1 - \exp(2\pi ik/n)I)\mathbf{v} = 0$$

using Gaussian elimination.

$$\begin{pmatrix} -\psi_n^k & 1 & 0 & \dots & 0 \\ 0 & -\psi_n^k & 1 & \dots & 0 \\ 0 & 0 & -\psi_n^k & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & 0 & \psi_n^{2k} & -\psi_n^k \end{pmatrix}$$

## Gaussian elimination

To find  $J$ , we can solve

$$(C_n^1 - \exp(2\pi ik/n)I)\mathbf{v} = 0$$

using Gaussian elimination.

$$\begin{pmatrix} -\psi_n^k & 1 & 0 & \dots & 0 \\ 0 & -\psi_n^k & 1 & \dots & 0 \\ 0 & 0 & -\psi_n^k & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

## Resolving the Eigenvectors

$$\begin{pmatrix} -\psi_n^k & 1 & 0 & \dots & 0 & 0 \\ 0 & -\psi_n^k & 1 & \dots & 0 & 0 \\ 0 & 0 & -\psi_n^k & \ddots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & -\psi_n^k & 1 \\ 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-1} \\ v_n \end{pmatrix}$$

Set  $v_n = 1$ , we can start solving:

$$-\psi_n^k v_{n-1} + 1 = 0 \qquad \Rightarrow \qquad \psi_n^{-k} = v_{n-1}$$

## Resolving the Eigenvectors

$$\begin{pmatrix} -\psi_n^k & 1 & 0 & \dots & 0 & 0 \\ 0 & -\psi_n^k & 1 & \dots & 0 & 0 \\ 0 & 0 & -\psi_n^k & \ddots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & -\psi_n^k & 1 \\ 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-1} \\ v_n \end{pmatrix}$$

Set  $v_n = 1$ , we can start solving:

$$\begin{aligned} -\psi_n^k v_{n-1} + 1 &= 0 &\implies \psi_n^{-k} &= v_{n-1} \\ -\psi_n^k v_{n-2} + v_{n-1} &= 0 &\implies \psi_n^{-2k} &= v_{n-2}. \end{aligned}$$

## The Fourier Basis

Therefore, the  $k$ th eigenvector is given by:

$$\mathbf{v}_k = (\psi_n^{n-k}, \psi_n^{n-2k}, \dots, \psi_n^k, 1)^T.$$

# The Fourier Basis

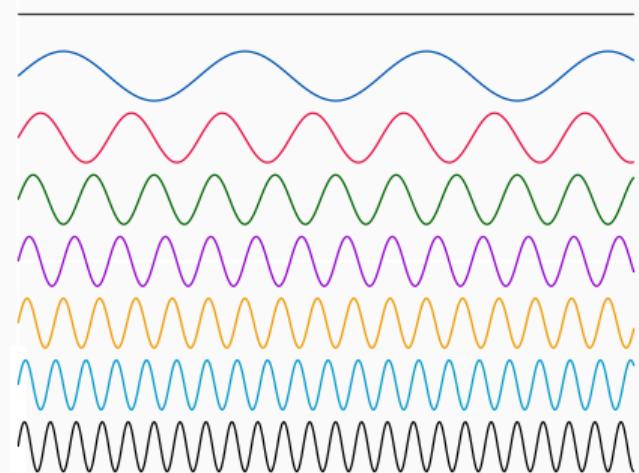
Therefore, the  $k$ th eigenvector is given by:

$$\mathbf{v}_k = (\psi_n^{n-k}, \psi_n^{n-2k}, \dots, \psi_n^k, 1)^T.$$

Let  $J = [\mathbf{v}_0, \dots, \mathbf{v}_{n-1}]^T$ ,  $u$  an arbitrary vector.

Change-of-basis results in **discrete Fourier transform**

$$\begin{aligned}(Ju)_k &= \langle v_k, u \rangle = \sum_t \exp(2\pi i \frac{k}{n} \cdot t) u_t \\&= \sum_t \cos(2\pi \frac{k}{n} \cdot t) u_t + i \sin(2\pi \frac{k}{n} \cdot t) u_t\end{aligned}$$



## Convolutions are shift-equivariant

Fourier-transform diagonalizes *all* shifts simultaneously:

$$C_n^1 = J^{-1}DJ \implies C_n^k = J^{-1}D^k J$$

## Convolutions are shift-equivariant

Fourier-transform diagonalizes *all* shifts simultaneously:

$$C_n^1 = J^{-1}DJ \implies C_n^k = J^{-1}D^k J$$

Thus, we can use arbitrary combinations

$$F = \sum \alpha_k C_n^k = J^{-1}(\sum \alpha_k D^k)J$$

$$\begin{pmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \dots \\ \alpha_{n-1} & \alpha_0 & \alpha_1 & \dots \\ \alpha_{n-2} & \alpha_{n-1} & \alpha_0 & \dots \\ \vdots & \vdots & \vdots & \ddots \\ \alpha_1 & \alpha_2 & \alpha_3 & \dots \end{pmatrix}$$

General convolution.

## Convolutions are shift-equivariant

Fourier-transform diagonalizes *all* shifts simultaneously:

$$C_n^1 = J^{-1}DJ \implies C_n^k = J^{-1}D^k J$$

Thus, we can use arbitrary combinations

$$F = \sum \alpha_k C_n^k = J^{-1} \left( \sum \alpha_k D^k \right) J$$

Then

$$FC_n^1 \mathbf{x} = C_n^1 F \mathbf{x},$$

so *any* convolution commutes with the shift operator.

$$\begin{pmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \dots \\ \alpha_{n-1} & \alpha_0 & \alpha_1 & \dots \\ \alpha_{n-2} & \alpha_{n-1} & \alpha_0 & \dots \\ \vdots & \vdots & \vdots & \ddots \\ \alpha_1 & \alpha_2 & \alpha_3 & \dots \end{pmatrix}$$

General convolution.

## Shift-equivariant functions are convolutions

Assume the converse; for matrix  $F$ :

$$\forall \mathbf{x} : FC_n^1 \mathbf{x} = C_n^1 F \mathbf{x} \Leftrightarrow FJ^{-1}DJ\mathbf{x} = J^{-1}DJF\mathbf{x}$$

## Shift-equivariant functions are convolutions

Assume the converse; for matrix  $F$ :

$$\forall \mathbf{x} : FC_n^1 \mathbf{x} = C_n^1 F \mathbf{x} \Leftrightarrow FJ^{-1}DJ\mathbf{x} = J^{-1}DJF\mathbf{x}$$

Set  $F = J^{-1}F'J$ , and  $\mathbf{x} = J^{-1}\mathbf{y}$ , then

$$\forall \mathbf{x} : J^{-1}F'DJ\mathbf{x} = J^{-1}DFJ\mathbf{x} \Leftrightarrow \forall \mathbf{y} : F'D\mathbf{y} = DF\mathbf{y}$$

## Shift-equivariant functions are convolutions

Assume the converse; for matrix  $F$ :

$$\forall \mathbf{x} : FC_n^1 \mathbf{x} = C_n^1 F \mathbf{x} \Leftrightarrow FJ^{-1}DJ\mathbf{x} = J^{-1}DJF\mathbf{x}$$

Set  $F = J^{-1}F'J$ , and  $\mathbf{x} = J^{-1}\mathbf{y}$ , then

$$\forall \mathbf{x} : J^{-1}F'DJ\mathbf{x} = J^{-1}DFJ\mathbf{x} \Leftrightarrow \forall \mathbf{y} : F'D\mathbf{y} = DF\mathbf{y}$$

Pick  $\mathbf{x} = \hat{\mathbf{e}}_1$ :

$$F'(d_{11}, 0, \dots)^\top = D(f_{11}, f_{21}, \dots)^\top \Leftrightarrow (f_{11}d_{11}, 0, \dots)^\top = (d_{11}f_{11}, d_{22}f_{21}, \dots)^\top \quad (1)$$

Thus, equality requires  $f_{21}, f_{32}, \dots = 0$

**Every convolution is shift-equivariant, and any shift-equivariant linear function is a convolution**

## Recap

$$\begin{array}{c} \text{x} \quad \quad \quad \overbrace{\mathbf{C}(\theta)}^{\theta_{11} + \theta_{13} + \theta_{22} + \theta_{31} + \theta_{33}} \quad \text{x} * \theta \\ \begin{matrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix} \quad \begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix} \quad \begin{matrix} 1 & 4 & 3 & 4 & 1 \\ 1 & 2 & 4 & 3 & 3 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 & 0 \end{matrix} \end{array}$$

Interpretation: Given some filter  $F$ , we apply *all* possible group-elements (shift) and record all corresponding responses.

## Recap

$$\begin{array}{c} \text{x} \\ \left[ \begin{array}{ccccccc} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \right] \\ \star \\ \left[ \begin{array}{ccc} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{array} \right] \\ = \\ \left[ \begin{array}{ccccc} 1 & 4 & 3 & 4 & 1 \\ 1 & 2 & 4 & 3 & 3 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 & 0 \end{array} \right] \end{array}$$

$\overbrace{\theta_{11} + \theta_{13} + \theta_{22} + \theta_{31} + \theta_{33}}$

Interpretation: Given some filter  $F$ , we apply *all* possible group-elements (shift) and record all corresponding responses.

Can we generalize this to arbitrary groups?

## Part VIII: Steerable Kernels

---

## A first construction for an equivariant linear layer

### Standard linear layer

Select a filter  $f \in \mathcal{X}(\Omega, \mathcal{C})$ , output is given by inner product

$$f(x) = \int_{\Omega} \langle f(\omega), x(\omega) \rangle d\omega.$$

Stacking  $c$  filters defines the layer with  $c$  output channels.

# A first construction for an equivariant linear layer

## Standard linear layer

Select a filter  $f \in \mathcal{X}(\Omega, \mathcal{C})$ , output is given by inner product

$$f(x) = \int_{\Omega} \langle f(\omega), x(\omega) \rangle d\omega.$$

Stacking  $c$  filters defines the layer with  $c$  output channels.

## Equivariant linear layer

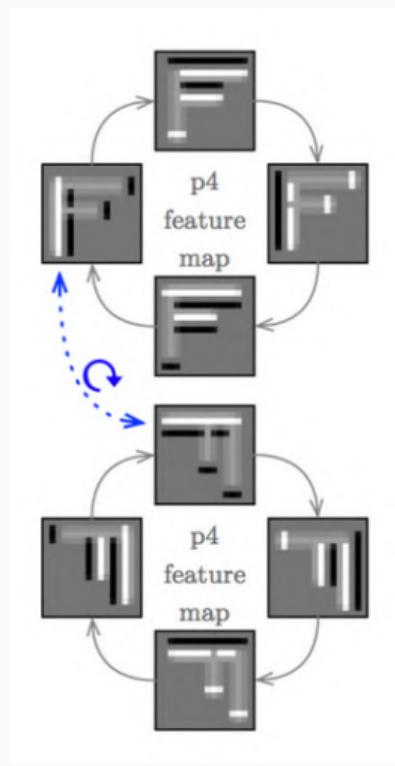
Let  $R$  be a representation of the action of  $G$  on  $\mathcal{X}$ . Then, for any filter  $f$  and group element  $g$ , define

$$f_g(x) = \int_{\Omega} \langle g.f(\omega), x(\omega) \rangle d\omega.$$

Stack all such activations, resulting in  $|G|c$  output channels.

## Example: P4 symmetry

- 90° rotations, translations
- rotated input rotated in rotated output, and **cyclic shift** in feature maps
- translated input results in translated output



## The resulting operation is equivariant

Let  $h \in G$  be a group element. Then, for a single filter  $f$  we have

$$f_g(h.x) = \int_{\Omega} \langle g.f(\omega), h.x(\omega) \rangle d\omega$$

## The resulting operation is equivariant

Let  $h \in G$  be a group element. Then, for a single filter  $f$  we have

$$\begin{aligned} f_g(h.x) &= \int_{\Omega} \langle g.f(\omega), h.x(\omega) \rangle d\omega \\ &= \int_{\Omega} \langle f(g^{-1}\omega), x(h^{-1}\omega) \rangle d\omega \end{aligned}$$

## The resulting operation is equivariant

Let  $h \in G$  be a group element. Then, for a single filter  $f$  we have

$$\begin{aligned}f_g(h.x) &= \int_{\Omega} \langle g.f(\omega), h.x(\omega) \rangle d\omega \\&= \int_{\Omega} \langle f(g^{-1}\omega), x(h^{-1}\omega) \rangle d\omega \\&= \int_{\Omega} \langle f(g^{-1}h\omega'), x(\omega') \rangle d\omega'\end{aligned}$$

- Change of variables  $\omega \rightarrow h^{-1}\omega =: \omega'$  assumes  $d\omega = d\omega'$  (otherwise, we shouldn't call  $G$  a symmetry of  $\Omega$ ).

## The resulting operation is equivariant

Let  $h \in G$  be a group element. Then, for a single filter  $f$  we have

$$\begin{aligned} f_g(h.x) &= \int_{\Omega} \langle g.f(\omega), h.x(\omega) \rangle d\omega \\ &= \int_{\Omega} \langle f(g^{-1}\omega), x(h^{-1}\omega) \rangle d\omega \\ &= \int_{\Omega} \langle f(g^{-1}h\omega'), x(\omega') \rangle d\omega' \\ &= \int_{\Omega} \langle (\mathbf{g^{-1}h})^{-1}.f(\omega'), x(\omega') \rangle d\omega' \end{aligned}$$

- Change of variables  $\omega \rightarrow h^{-1}\omega =: \omega'$  assumes  $d\omega = d\omega'$  (otherwise, we shouldn't call  $G$  a symmetry of  $\Omega$ ).

## The resulting operation is equivariant

Let  $h \in G$  be a group element. Then, for a single filter  $f$  we have

$$\begin{aligned} f_g(h.x) &= \int_{\Omega} \langle g.f(\omega), h.x(\omega) \rangle d\omega \\ &= \int_{\Omega} \langle f(g^{-1}\omega), x(h^{-1}\omega) \rangle d\omega \\ &= \int_{\Omega} \langle f(g^{-1}h\omega'), x(\omega') \rangle d\omega' \\ &= \int_{\Omega} \langle (g^{-1}h)^{-1}.f(\omega'), x(\omega') \rangle d\omega' \\ &= f_{\textcolor{red}{h^{-1}g}}(x) \end{aligned}$$

- Change of variables  $\omega \rightarrow h^{-1}\omega =: \omega'$  assumes  $d\omega = d\omega'$  (otherwise, we shouldn't call  $G$  a symmetry of  $\Omega$ ).

## The resulting operation is equivariant

Let  $h \in G$  be a group element. Then, for a single filter  $f$  we have

$$\begin{aligned} f_g(h.x) &= \int_{\Omega} \langle g.f(\omega), h.x(\omega) \rangle d\omega \\ &= \int_{\Omega} \langle f(g^{-1}\omega), x(h^{-1}\omega) \rangle d\omega \\ &= \int_{\Omega} \langle f(g^{-1}h\omega'), x(\omega') \rangle d\omega' \\ &= \int_{\Omega} \langle (g^{-1}h)^{-1}.f(\omega'), x(\omega') \rangle d\omega' \\ &= f_{h^{-1}g}(x) \end{aligned}$$

- Change of variables  $\omega \rightarrow h^{-1}\omega =: \omega'$  assumes  $d\omega = d\omega'$  (otherwise, we shouldn't call  $G$  a symmetry of  $\Omega$ ).
- The mapping  $h : G \longrightarrow G$ ,  $g \mapsto hg$  is bijective. Represented by permutation matrix  $P_h$ .

## The resulting operation is equivariant

Let  $h \in G$  be a group element. Then, for a single filter  $f$  we have

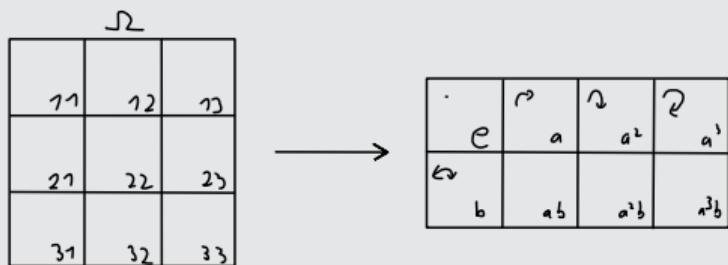
$$\begin{aligned} f_g(h.x) &= \int_{\Omega} \langle g.f(\omega), h.x(\omega) \rangle d\omega \\ &= \int_{\Omega} \langle f(g^{-1}\omega), x(h^{-1}\omega) \rangle d\omega \\ &= \int_{\Omega} \langle f(g^{-1}h\omega'), x(\omega') \rangle d\omega' \\ &= \int_{\Omega} \langle (g^{-1}h)^{-1}.f(\omega'), x(\omega') \rangle d\omega' \\ &= f_{h^{-1}g}(x) \end{aligned}$$

- Change of variables  $\omega \rightarrow h^{-1}\omega =: \omega'$  assumes  $d\omega = d\omega'$  (otherwise, we shouldn't call  $G$  a symmetry of  $\Omega$ ).
- The mapping  $h : G \longrightarrow G$ ,  $g \mapsto hg$  is bijective. Represented by permutation matrix  $P_h$ .  
⇒ Collect  $\mathbf{f}_G(x) = (f_g(x))_{g \in G} \in \mathbb{R}^{|G|}$ ,  
then  $\mathbf{f}_G(h.x) = P_h^{-1}\mathbf{f}_G(x)$ .  
⇒  $\mathbf{f}_G$  transforms according to the regular representation.

# Group-convolution maps signals to signals over the group itself

$$f_G : \mathcal{X}(\Omega, \mathcal{C}) \longrightarrow \mathcal{X}(G, \mathcal{C})$$

## D4-Convolution



## T2-Convolution

Original (image) domain:  $\Omega \cong (\mathbb{Z}/n\mathbb{Z})^2$

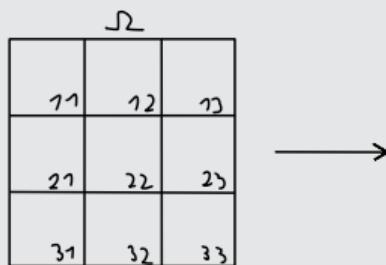
Translation group:

$$T_2(n) \cong C(n)^2 \cong (\mathbb{Z}/n\mathbb{Z})^2$$

## Group-convolution maps signals to signals over the group itself

$$f_G : \mathcal{X}(\Omega, \mathcal{C}) \longrightarrow \mathcal{X}(G, \mathcal{C})$$

### D4-Convolution



### T2-Convolution

Original (image) domain:  $\Omega \cong (\mathbb{Z}/n\mathbb{Z})^2$

Translation group:

$$T_2(n) \cong C(n)^2 \cong (\mathbb{Z}/n\mathbb{Z})^2$$

This construction requires the output to have  $c|G|$  channels, which becomes infeasible for large groups.

# Steerable Kernels

## Steerable Mapping (Intertwiner)

Let  $G$  be a group,  $\Omega, \Psi$  two geometric domains.

A linear mapping  $L$  from  $\mathcal{X}(\Omega, \mathcal{C})$  to  $\mathcal{X}(\Omega', \mathcal{C}')$  is called (linearly) **steerable** if there exist group actions (representations)  $\mu, \nu$  such that

$$\forall x \in \mathcal{X}(\Omega, \mathcal{C}), g \in G : \nu(g).L(x) = L(\mu(g).x)$$

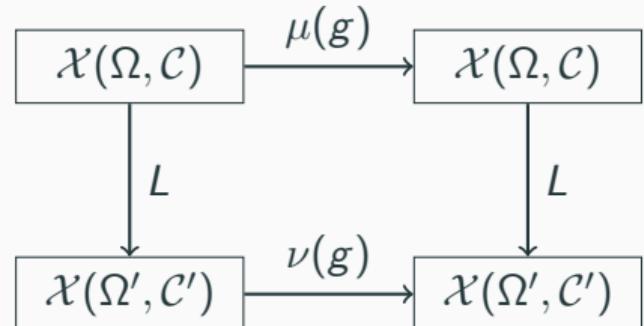
# Steerable Kernels

## Steerable Mapping (Intertwiner)

Let  $G$  be a group,  $\Omega, \Psi$  two geometric domains.

A linear mapping  $L$  from  $\mathcal{X}(\Omega, \mathcal{C})$  to  $\mathcal{X}(\Omega', \mathcal{C}')$  is called (linearly) **steerable** if there exist group actions (representations)  $\mu, \nu$  such that

$$\forall x \in \mathcal{X}(\Omega, \mathcal{C}), g \in G : \nu(g).L(x) = L(\mu(g).x)$$



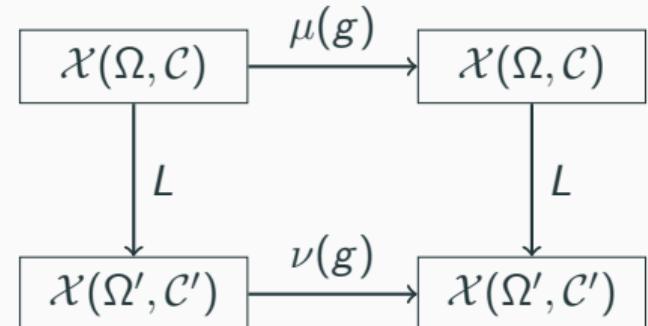
# Steerable Kernels

## Steerable Mapping (Intertwiner)

Let  $G$  be a group,  $\Omega, \Psi$  two geometric domains.

A linear mapping  $L$  from  $\mathcal{X}(\Omega, \mathcal{C})$  to  $\mathcal{X}(\Omega', \mathcal{C}')$  is called (linearly) **steerable** if there exist group actions (representations)  $\mu, \nu$  such that

$$\forall x \in \mathcal{X}(\Omega, \mathcal{C}), g \in G : \nu(g).L(x) = L(\mu(g).x)$$



Note: If  $\nu$  is the trivial representation, then  $L$  is *invariant* ( $\nu(g) = \text{id}$ ).

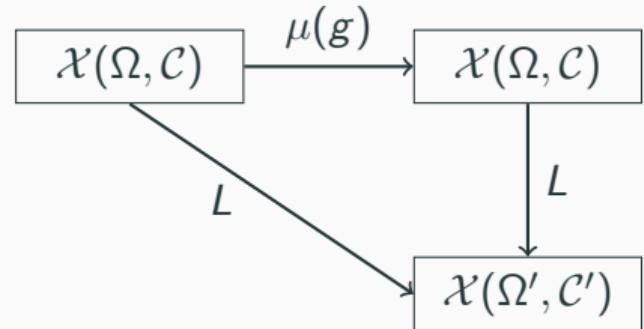
# Steerable Kernels

## Steerable Mapping (Intertwiner)

Let  $G$  be a group,  $\Omega, \Psi$  two geometric domains.

A linear mapping  $L$  from  $\mathcal{X}(\Omega, \mathcal{C})$  to  $\mathcal{X}(\Omega', \mathcal{C}')$  is called (linearly) **steerable** if there exist group actions (representations)  $\mu, \nu$  such that

$$\forall x \in \mathcal{X}(\Omega, \mathcal{C}), g \in G : \nu(g).L(x) = L(\mu(g).x)$$



Note: If  $\nu$  is the trivial representation, then  $L$  is *invariant* ( $\nu(g) = \text{id}$ ).

# Equivariance implies invariant mapping

## Invariance of weights

For  $\nu(g) \in \mathrm{GL}(m)$ ,  $\mu(g) \in \mathrm{GL}(n)$

$$\forall g, x : \nu(g).L(x) = L(\mu(g).x)$$

$$\Leftrightarrow \forall g : \nu(g) \circ L = L \circ \mu(g)$$

# Equivariance implies invariant mapping

## Invariance of weights

For  $\nu(g) \in \mathrm{GL}(m)$ ,  $\mu(g) \in \mathrm{GL}(n)$

$$\forall g, x : \nu(g).L(x) = L(\mu(g).x)$$

$$\Leftrightarrow \forall g : \nu(g) \circ L = L \circ \mu(g)$$

$$\Leftrightarrow \forall g : L = \nu(g)^{-1} \circ L \circ \mu(g)$$

# Equivariance implies invariant mapping

## Invariance of weights

For  $\nu(g) \in \mathrm{GL}(m)$ ,  $\mu(g) \in \mathrm{GL}(n)$

$$\forall g, x : \nu(g).L(x) = L(\mu(g).x)$$

$$\Leftrightarrow \forall g : \nu(g) \circ L = L \circ \mu(g)$$

$$\Leftrightarrow \forall g : L = \nu(g)^{-1} \circ L \circ \mu(g)$$

The weight matrix  $L$  is invariant under this particular group action.

# Equivariance implies invariant mapping

## Invariance of weights

For  $\nu(g) \in \mathrm{GL}(m)$ ,  $\mu(g) \in \mathrm{GL}(n)$

$$\forall g, x : \nu(g).L(x) = L(\mu(g).x)$$

$$\Leftrightarrow \forall g : \nu(g) \circ L = L \circ \mu(g)$$

$$\Leftrightarrow \forall g : L = \nu(g)^{-1} \circ L \circ \mu(g)$$

## Group averaging

Can reuse our group-averaging to construct equivariant weights from arbitrary weights  $\tilde{L}$ :

$$L = \sum_{g \in G} \nu(g)^{-1} \tilde{L} \mu(g)$$

## Convolutions through weight-sharing

Let  $\mu(k) = C_3^k, \nu(k) = C_3^k$ , select  $\tilde{L} = \delta_{11}$ .

Group action:

$$(C_3^1)^{-1} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} C_3^1$$

## Convolutions through weight-sharing

Let  $\mu(k) = C_3^k, \nu(k) = C_3^k$ , select  $\tilde{L} = \delta_{11}$ .

Group action:

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

## Convolutions through weight-sharing

Let  $\mu(k) = C_3^k, \nu(k) = C_3^k$ , select  $\tilde{L} = \delta_{11}$ .

Group action:

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

## Convolutions through weight-sharing

Let  $\mu(k) = C_3^k, \nu(k) = C_3^k$ , select  $\tilde{L} = \delta_{11}$ .

Group action:

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

## Convolutions through weight-sharing

Let  $\mu(k) = C_3^k, \nu(k) = C_3^k$ , select  $\tilde{L} = \delta_{11}$ .

Group action:

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

## Convolutions through weight-sharing

Let  $\mu(k) = C_3^k, \nu(k) = C_3^k$ , select  $\tilde{L} = \delta_{11}$ .

Group action:

$$(C_3^1)^{-1} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} C_3^1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

In general: Move to the right by  $k$  places, then move down  $k$  places.

## Convolutions through weight-sharing

In total, we get the following nine group-averages (color indicates original matrix):

$$\begin{pmatrix} \textcolor{blue}{1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & \textcolor{orange}{1} & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & \textcolor{red}{1} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 1 \\ \textcolor{red}{1} & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \textcolor{blue}{1} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & \textcolor{orange}{1} \\ 1 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \textcolor{orange}{1} & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & \textcolor{red}{1} & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \textcolor{blue}{1} \end{pmatrix}$$

## Convolutions through weight-sharing

In total, we get the following nine group-averages (color indicates original matrix):

$$\begin{pmatrix} \textcolor{blue}{1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & \textcolor{orange}{1} & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & \textcolor{red}{1} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 1 \\ \textcolor{red}{1} & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \textcolor{blue}{1} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & \textcolor{orange}{1} \\ 1 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \textcolor{orange}{1} & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & \textcolor{red}{1} & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \textcolor{blue}{1} \end{pmatrix}$$

**Three distinct matrices → three orbits that serve as basis vectors**

## Steerable Filter Banks

- If  $L$  and  $L'$  are linearly steerable (w.r.t.  $\mu, \nu$ ), then so is  $\alpha L + L'$  for  $\alpha \in \mathbb{R}$ .

## Steerable Filter Banks

- If  $L$  and  $L'$  are linearly steerable (w.r.t.  $\mu, \nu$ ), then so is  $\alpha L + L'$  for  $\alpha \in \mathbb{R}$ .

$$\begin{aligned} & (\alpha L + L')(\mu(g).x) \\ &= \alpha L(\mu(g).x) + L'(\mu(g).x) \end{aligned}$$

## Steerable Filter Banks

- If  $L$  and  $L'$  are linearly steerable (w.r.t.  $\mu, \nu$ ), then so is  $\alpha L + L'$  for  $\alpha \in \mathbb{R}$ .

$$\begin{aligned}(\alpha L + L')(\mu(g).x) \\= \alpha L(\mu(g).x) + L'(\mu(g).x) \\= \alpha \nu(g).L(x) + \nu(g).L'(x)\end{aligned}$$

## Steerable Filter Banks

- If  $L$  and  $L'$  are linearly steerable (w.r.t.  $\mu, \nu$ ), then so is  $\alpha L + L'$  for  $\alpha \in \mathbb{R}$ .

$$\begin{aligned} & (\alpha L + L')(\mu(g).x) \\ &= \alpha L(\mu(g).x) + L'(\mu(g).x) \\ &= \alpha \nu(g).L(x) + \nu(g).L'(x) \\ &= \nu(g)(\alpha L(x) + L'(x)) \end{aligned}$$

## Steerable Filter Banks

- If  $L$  and  $L'$  are linearly steerable (w.r.t.  $\mu, \nu$ ), then so is  $\alpha L + L'$  for  $\alpha \in \mathbb{R}$ .

$$\begin{aligned} & (\alpha L + L')(\mu(g).x) \\ &= \alpha L(\mu(g).x) + L'(\mu(g).x) \\ &= \alpha \nu(g).L(x) + \nu(g).L'(x) \\ &= \nu(g)(\alpha L(x) + L'(x)) \\ &= \nu(g)(\alpha L + L')(x) \end{aligned}$$

## Steerable Filter Banks

- If  $L$  and  $L'$  are linearly steerable (w.r.t.  $\mu, \nu$ ), then so is  $\alpha L + L'$  for  $\alpha \in \mathbb{R}$ .
- $\Rightarrow \mathcal{L} := \{L : L \text{ is } \mu, \nu \text{ steerable}\}$  is a vector space.

$$\begin{aligned} & (\alpha L + L')(\mu(g).x) \\ &= \alpha L(\mu(g).x) + L'(\mu(g).x) \\ &= \alpha \nu(g).L(x) + \nu(g).L'(x) \\ &= \nu(g)(\alpha L(x) + L'(x)) \\ &= \nu(g)(\alpha L + L')(x) \end{aligned}$$

## Steerable Filter Banks

- If  $L$  and  $L'$  are linearly steerable (w.r.t.  $\mu, \nu$ ), then so is  $\alpha L + L'$  for  $\alpha \in \mathbb{R}$ .
- $\Rightarrow \mathcal{L} := \{L : L \text{ is } \mu, \nu \text{ steerable}\}$  is a vector space.
- There exists a *basis*, in which we can expand the filters.

$$\begin{aligned} & (\alpha L + L')(\mu(g).x) \\ &= \alpha L(\mu(g).x) + L'(\mu(g).x) \\ &= \alpha \nu(g).L(x) + \nu(g).L'(x) \\ &= \nu(g)(\alpha L(x) + L'(x)) \\ &= \nu(g)(\alpha L + L')(x) \end{aligned}$$

## Steerable Filter Banks

- If  $L$  and  $L'$  are linearly steerable (w.r.t.  $\mu, \nu$ ), then so is  $\alpha L + L'$  for  $\alpha \in \mathbb{R}$ .
- $\Rightarrow \mathcal{L} := \{L : L \text{ is } \mu, \nu \text{ steerable}\}$  is a vector space.
- There exists a *basis*, in which we can expand the filters.

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{aligned} & (\alpha L + L')(\mu(g).x) \\ &= \alpha L(\mu(g).x) + L'(\mu(g).x) \\ &= \alpha \nu(g).L(x) + \nu(g).L'(x) \\ &= \nu(g)(\alpha L(x) + L'(x)) \\ &= \nu(g)(\alpha L + L')(x) \end{aligned}$$

## Steerable Filter Banks

- If  $L$  and  $L'$  are linearly steerable (w.r.t.  $\mu, \nu$ ), then so is  $\alpha L + L'$  for  $\alpha \in \mathbb{R}$ .
- $\Rightarrow \mathcal{L} := \{L : L \text{ is } \mu, \nu \text{ steerable}\}$  is a vector space.
- There exists a *basis*, in which we can expand the filters.

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{aligned} & (\alpha L + L')(\mu(g).x) \\ &= \alpha L(\mu(g).x) + L'(\mu(g).x) \\ &= \alpha \nu(g).L(x) + \nu(g).L'(x) \\ &= \nu(g)(\alpha L(x) + L'(x)) \\ &= \nu(g)(\alpha L + L')(x) \end{aligned}$$

In general, generate all orbits and orthogonalize (e.g., Gram-Schmidt)

The same strategy also works for trivial (invariant) representation of D4

### Trivial Representation A1

We have already seen this: rotation-invariant linear functions



What about non-trivial representations?

The same strategy also works for trivial (invariant) representation of D4

### Trivial Representation A1

We have already seen this: rotation-invariant linear functions



What about non-trivial representations?

# Constructing a basis for D4's B1 representation

## B1

$a$ : rotate  $90^\circ$ ,  $b$ : flip along y-axis.

B1 is a 1-dimensional representation with

$$\nu(a) = -1, \quad \nu(b) = 1$$

## Smoothing

# Constructing a basis for D4's B1 representation

## B1

$a$ : rotate  $90^\circ$ ,  $b$ : flip along y-axis.

B1 is a 1-dimensional representation with

$$\nu(a) = -1, \quad \nu(b) = 1$$

## Smoothing

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{E_{11}} + \underbrace{(-1)}_{\nu(a)} \cdot \underbrace{\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{R(a)E_{11}}$$

$$+ \underbrace{(1)}_{\nu(a^2)} \cdot \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{R(a^2)E_{11}} + \dots$$

# Constructing a basis for D4's B1 representation

## B1

$a$ : rotate 90°,  $b$ : flip along y-axis.

B1 is a 1-dimensional representation with

$$\nu(a) = -1, \quad \nu(b) = 1$$

## Smoothing



# Constructing a basis for D4's B1 representation

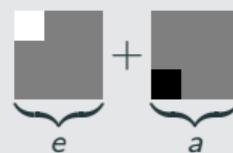
## B1

$a$ : rotate 90°,  $b$ : flip along y-axis.

B1 is a 1-dimensional representation with

$$\nu(a) = -1, \quad \nu(b) = 1$$

## Smoothing



# Constructing a basis for D4's B1 representation

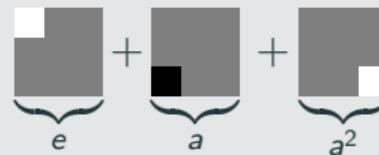
## B1

$a$ : rotate 90°,  $b$ : flip along y-axis.

B1 is a 1-dimensional representation with

$$\nu(a) = -1, \quad \nu(b) = 1$$

## Smoothing



# Constructing a basis for D4's B1 representation

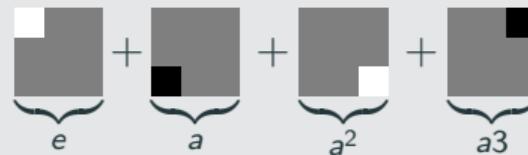
## B1

$a$ : rotate 90°,  $b$ : flip along y-axis.

B1 is a 1-dimensional representation with

$$\nu(a) = -1, \quad \nu(b) = 1$$

## Smoothing



# Constructing a basis for D4's B1 representation

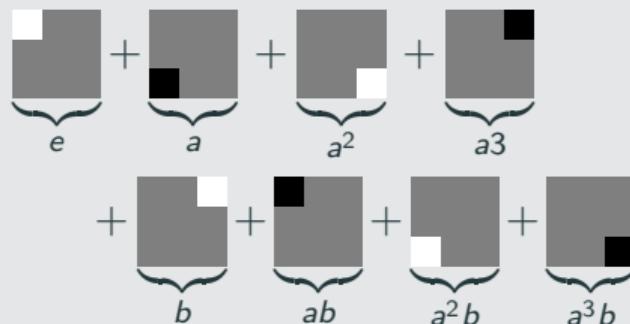
## B1

$a$ : rotate 90°,  $b$ : flip along y-axis.

B1 is a 1-dimensional representation with

$$\nu(a) = -1, \quad \nu(b) = 1$$

## Smoothing



# Constructing a basis for D4's B1 representation

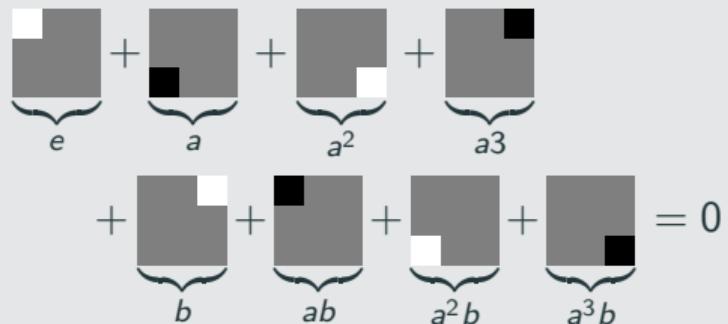
## B1

$a$ : rotate 90°,  $b$ : flip along y-axis.

B1 is a 1-dimensional representation with

$$\nu(a) = -1, \quad \nu(b) = 1$$

## Smoothing

$$\begin{aligned} e &+ a &+ a^2 &+ a^3 \\ + b &+ ab &+ a^2b &+ a^3b = 0 \end{aligned}$$


# Constructing a basis for D4's B1 representation

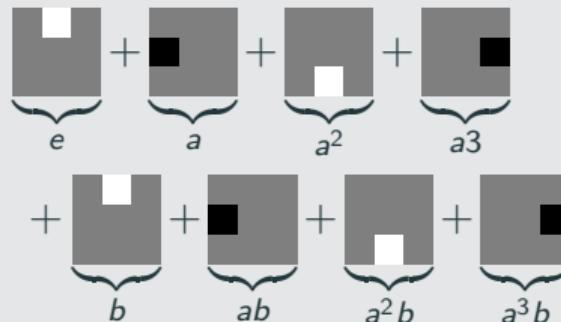
## B1

$a$ : rotate 90°,  $b$ : flip along y-axis.

B1 is a 1-dimensional representation with

$$\nu(a) = -1, \quad \nu(b) = 1$$

## Smoothing



# Constructing a basis for D4's B1 representation

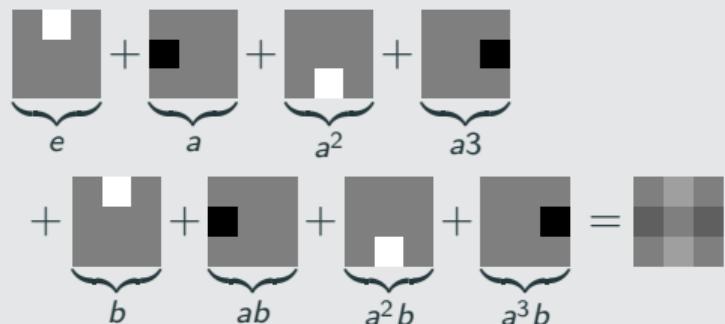
## B1

$a$ : rotate 90°,  $b$ : flip along y-axis.

B1 is a 1-dimensional representation with

$$\nu(a) = -1, \quad \nu(b) = 1$$

## Smoothing



# One-dimensional representations of D4

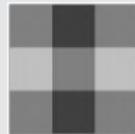
## Trivial Representation A1

We have already seen this: rotation-invariant linear functions



## B1

Rotation sign, reflection inv:



## B2

Rotation, reflection sign:



# One-dimensional representations of D4

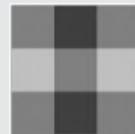
## Trivial Representation A1

We have already seen this: rotation-invariant linear functions



## B1

Rotation sign, reflection inv:



## B2

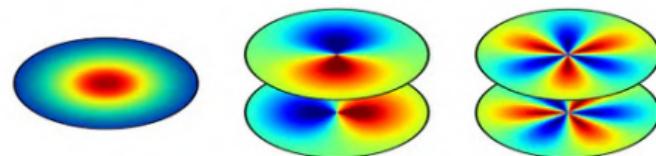
Rotation, reflection sign:



**What about A2?** (rotation invariant, reflection sign-change)

# Steerable kernels for SO(2)

rotation steerable kernels

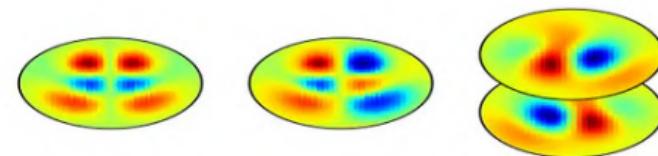


scalar  
↔  
scalar

scalar  
↔  
vector

scalar  
↔  
irrep order 3

reflection steerable kernels



scalar  
↔  
scalar

scalar  
↔  
pseudoscalar

scalar  
↔  
regular

# Homogenous spaces

## Definition (Transitive Action)

A group action is called **transitive** iff

$$\forall x, y \in \Omega : \exists g \in G \text{ s.t. } x = gy.$$

# Homogenous spaces

## Definition (Transitive Action)

A group action is called **transitive** iff

$$\forall x, y \in \Omega : \exists g \in G \text{ s.t. } x = gy.$$

## Definition (Homogenous space)

A geometric domain  $\Omega$  with a transitive group action is called a **homogenous space**.

# Homogenous spaces

## Definition (Transitive Action)

A group action is called **transitive** iff

$$\forall x, y \in \Omega : \exists g \in G \text{ s.t. } x = gy.$$

## Definition (Homogenous space)

A geometric domain  $\Omega$  with a transitive group action is called a **homogenous space**.

## Theorem (Convolutions are all you need)

*An equivariant map can always be written as a convolution-like integral*

## Part IX: Scale Separation

---

# Can we just use any convolution operation?

## Parameter Cost

A generic convolution on an  $n \times n$  image  
with  $c$  input and  $d$  output channels  
requires

$c \times n \times n \times d$  parameters.

# Can we just use any convolution operation?

## Parameter Cost

A generic convolution on an  $n \times n$  image with  $c$  input and  $d$  output channels requires

$$c \times n \times n \times d \text{ parameters.}$$

Scales quadratically with input size.

## Group-smoothing

For a single linear layer, a “global” convolution is just *group smoothing*.

## Downsampling

Rescale the input to be small enough that we can easily handle it.

**Loses detail information**

**Rigid**

## Local Patches

Use very sparse filter matrices so that only local neighbourhoods interact.

# Can we just use any convolution operation?

## Parameter Cost

A generic convolution on an  $n \times n$  image with  $c$  input and  $d$  output channels requires

$$c \times n \times n \times d \text{ parameters.}$$

Scales quadratically with input size.

## Group-smoothing

For a single linear layer, a “global” convolution is just *group smoothing*.

## Downsampling

Rescale the input to be small enough that we can easily handle it.

**Loses detail information**

**Rigid**

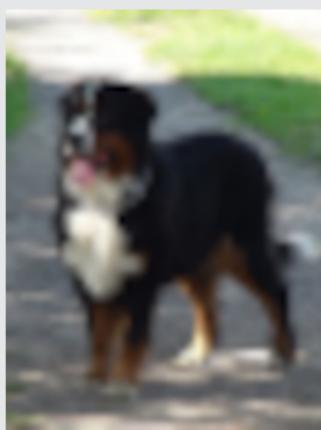
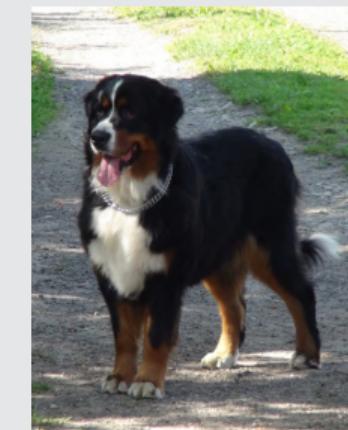
## Local Patches

Use very sparse filter matrices so that only local neighbourhoods interact.

**Cannot capture global structure**

# Scale separation prior

Sometimes, coarse scales are sufficient



Sometimes, local patches are sufficient



---

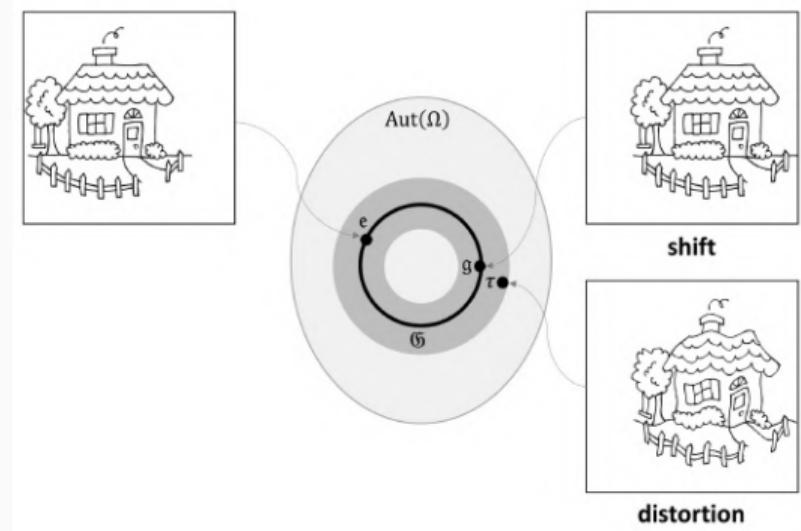
Antonio Kless, CC BY-SA 4.0, via Wikimedia Commons

---

Eiffel, public domain

# Deformation Stability

- Let  $G$  be translations, a subgroup of the diffeomorphisms  $\tau : \Omega \longrightarrow \Omega$ .
- Define  $c(\tau) := \sup \|\nabla \cdot \tau(u)\|$ . For  $g \in G$  we have  $c(g) = 0$ ;  $c$  measures the distance to the group
- We want **deformation stability**:  
$$\|f(x) - f(\tau.x)\| \leq c(\tau).$$



# Multiresolution analysis

- Decompose signal  $x \in \mathcal{X}(\Omega)$  into a lower resolution version  $\tilde{x} \in \mathcal{X}(\tilde{\Omega})$  and residuals.

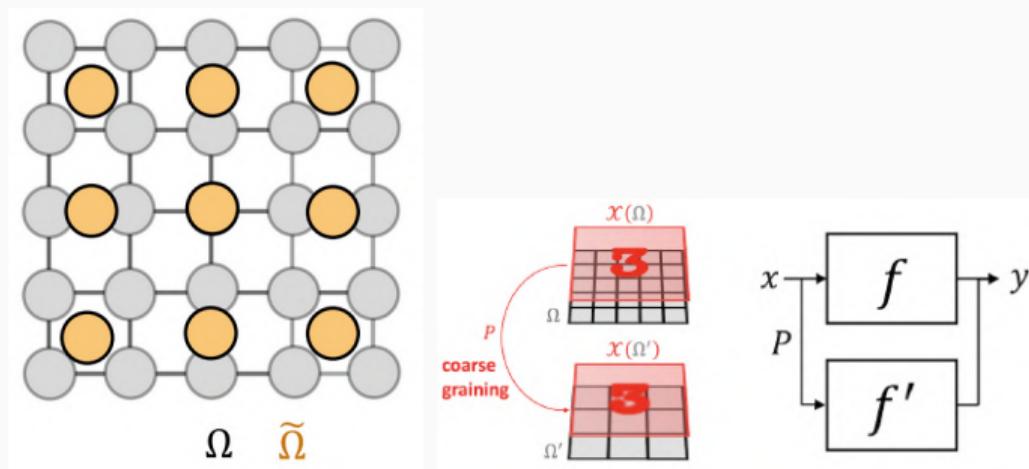


Image: Bronstein et al. (2021),

# Multiresolution analysis

- Decompose signal  $x \in \mathcal{X}(\Omega)$  into a lower resolution version  $\tilde{x} \in \mathcal{X}(\tilde{\Omega})$  and residuals.
- (Wavelet) filters localized in space

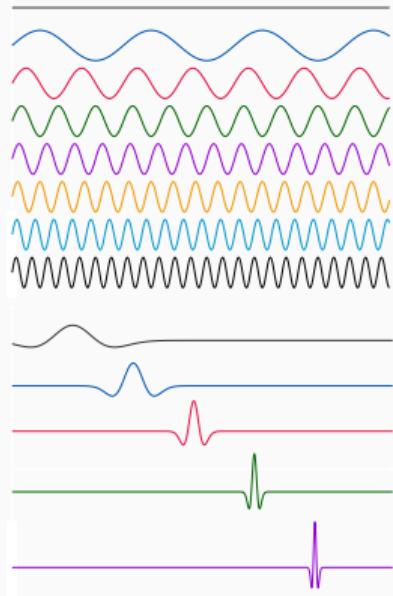
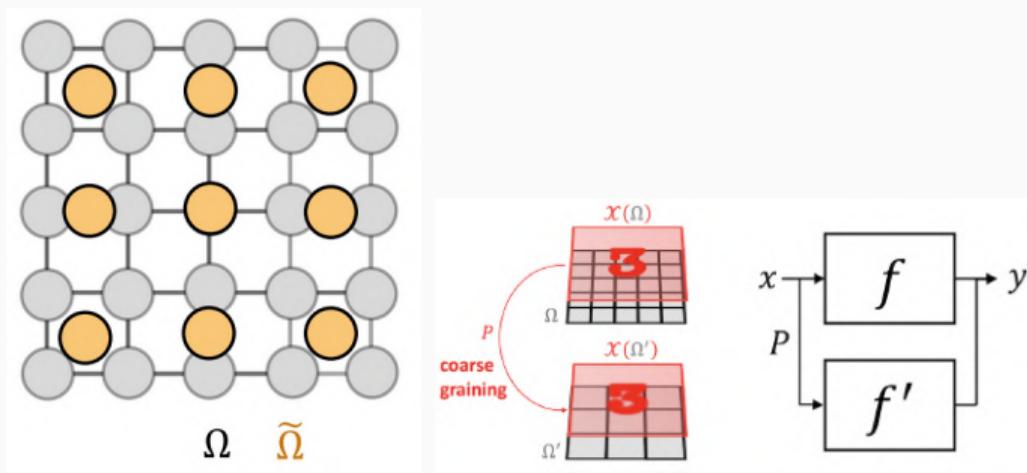
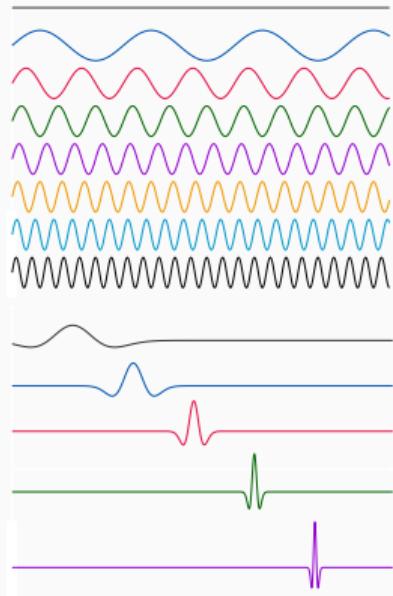
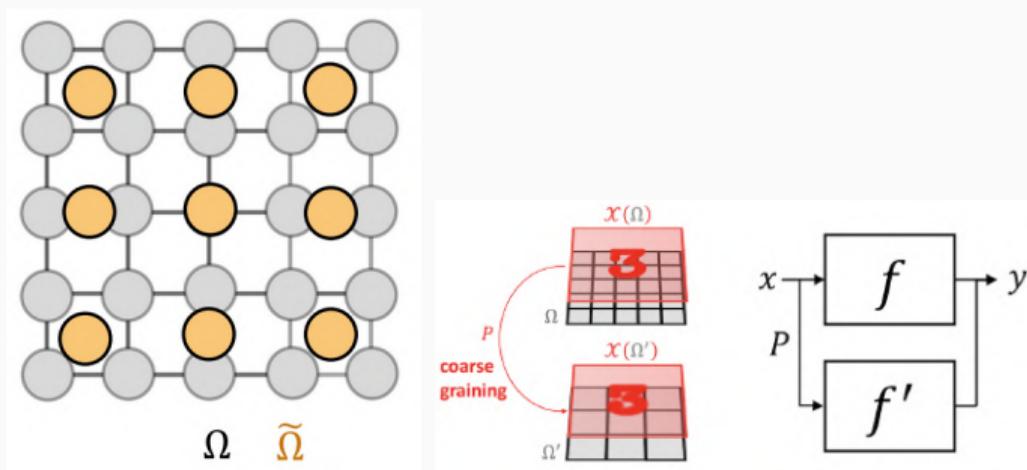


Image: Bronstein et al. (2021),

# Multiresolution analysis

- Decompose signal  $x \in \mathcal{X}(\Omega)$  into a lower resolution version  $\tilde{x} \in \mathcal{X}(\tilde{\Omega})$  and residuals.
- (Wavelet) filters localized in space
- Exhibit deformation stability<sup>1</sup>



# Compositionality

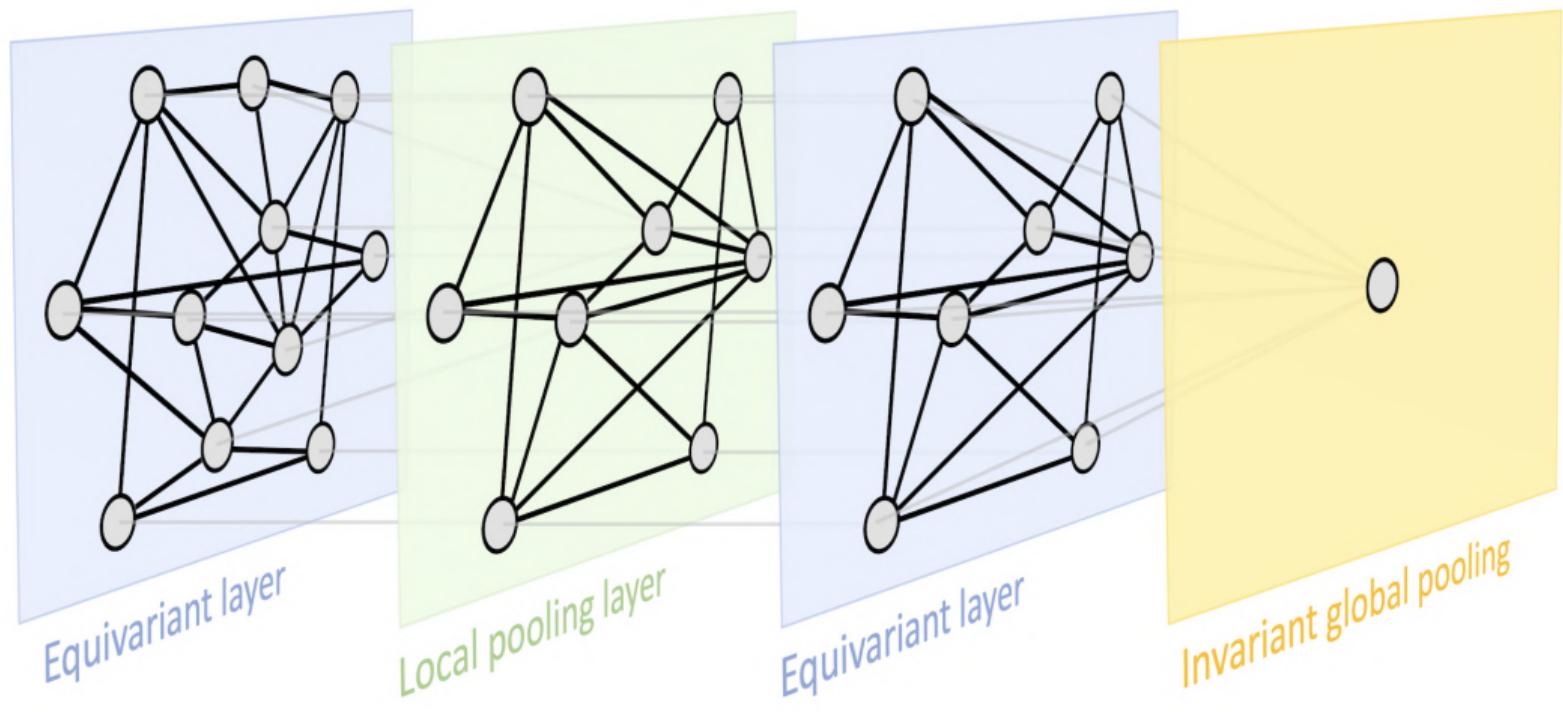
- The *deep* in deep learning
- Multiple layers of *local* connectivity can accumulate a *global* receptive field
- ⇒ **The GDL Blueprint**



## Part X: The GDL Blueprint

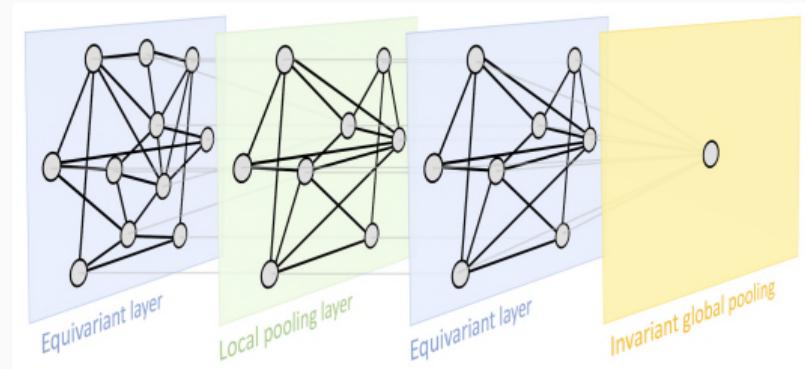
---

# The GDL Blueprint



# The GDL Blueprint

- Linear  $G$ -equivariant layer  
 $B : \mathcal{X}(\Omega, \mathcal{C}) \longrightarrow \mathcal{X}(\Omega', \mathcal{C}')$  s.t.  
 $g.B(x) = B(g.x)$
- Pointwise non-linearity
- Local pooling layer (coarsening)  
 $P : \mathcal{X}(\Omega, \mathcal{C}) \longrightarrow \mathcal{X}(\Omega', \mathcal{C})$  s.t.  $\Omega' \subset \Omega$
- $G$ -invariant layer (global pooling)  
 $A : \mathcal{X}(\Omega, \mathcal{C}) \longrightarrow \mathcal{Y}$  s.t.  $A(x) = A(g.x)$



## Many famous architectures follow this blueprint

<b>Architecture</b>	<b>Domain <math>\Omega</math></b>	<b>Group <math>G</math></b>
CNN	Grid	Translation
LSTM	1D-Grid	Time translation
<i>Spherical</i> CNN	Sphere	Rotation $SO(3)$
<i>Mesh</i> CNN	Manifold	Gauge $SO(2)$
GNN	Graph	Permutation $S(n)$
Deep Sets	Set	Permutation $S(n)$
Transformer	Complete Graph	Permutation $S(n)$

## References

## References

---

- [1] Michael M. Bronstein et al. **Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges**. 2021. arXiv: 2104.13478 [cs.LG]. URL: <https://arxiv.org/abs/2104.13478>.
- [2] Taco Cohen, Mario Geiger, and Maurice Weiler. “**A general theory of equivariant cnns on homogeneous spaces**”. In: *Advances in neural information processing systems* 32 (2019).
- [3] Taco S. Cohen and Max Welling. **Steerable CNNs**. 2016. arXiv: 1612.08498 [cs.LG]. URL: <https://arxiv.org/abs/1612.08498>.
- [4] Stéphane Mallat. “**Recursive interferometric representation**”. In: *Proc. of EUSICO conference, Danemark*. Vol. 3. 2010.
- [5] MD Zeiler. “**Visualizing and Understanding Convolutional Networks**”. In: