Geometric deep learning

- Geometric deep learning involves developing algorithms and models for analyzing **data structured in non-euclidean domains** (graphs, networks, or manifolds)

- It combines principles from graph theory, differential geometry, and topology to enable learning and processing of complex shapes and structures that are difficult to represent in traditional deep learning frameworks.

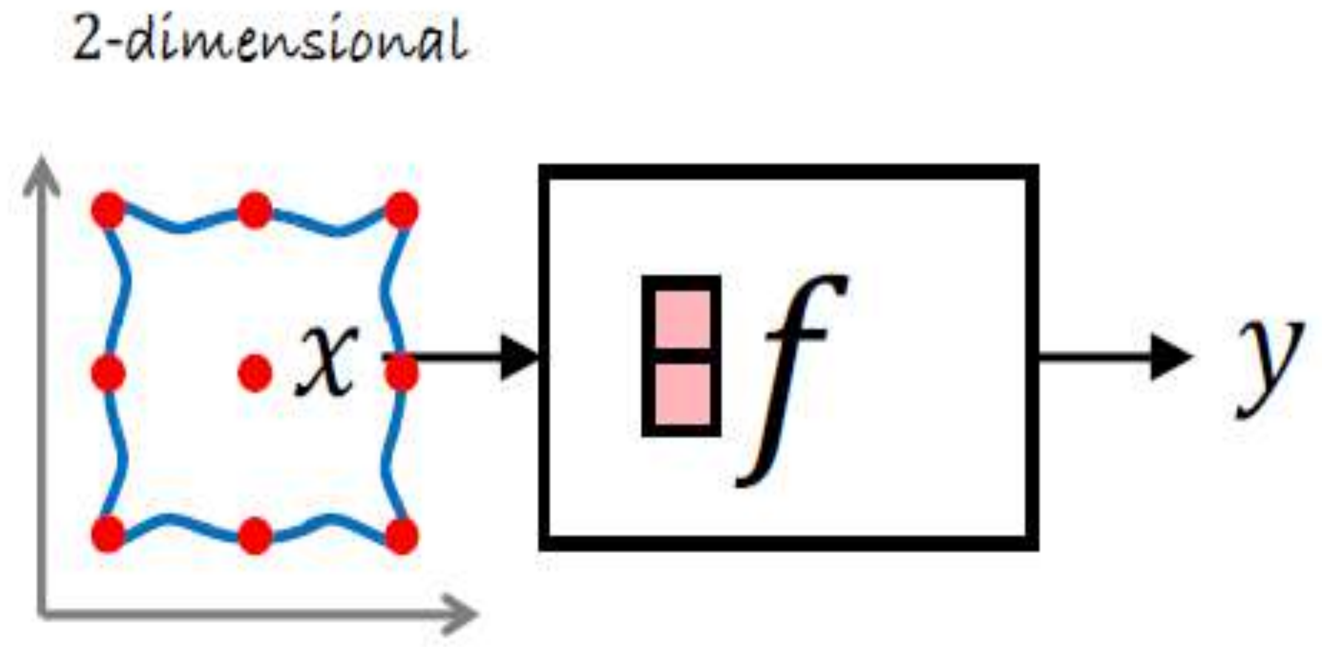
- One important direction is the understanding of the geometry of the data and the **design of meaningful priors** to decrease the sample complexity of the learning

Class 1

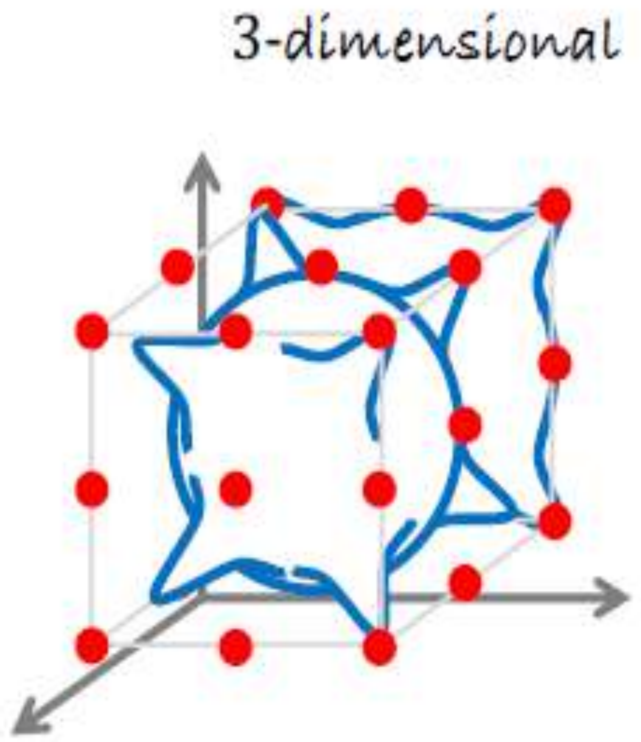
- Curse of dimensionality and geometric deep learning
- Fighting the curse with priors, examples
- Implicit regularization
- Symmetry

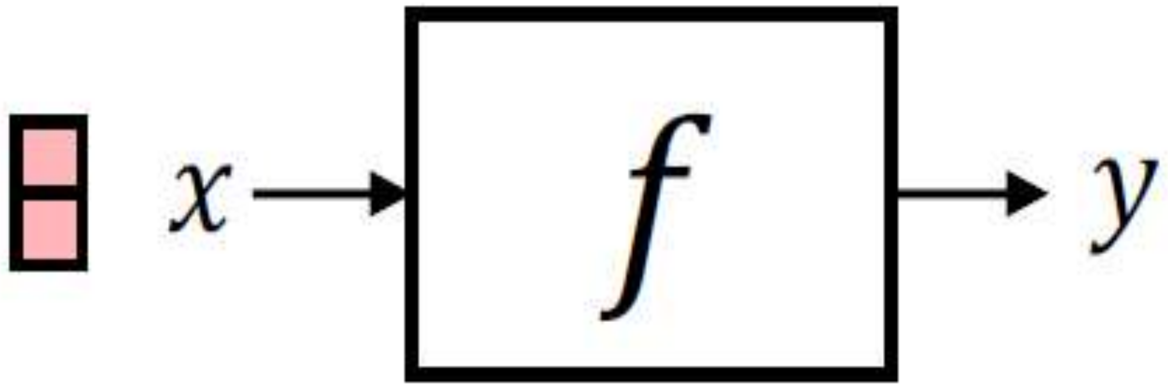
See: https://geometricdeeplearning.com/

Curse of dimensionality

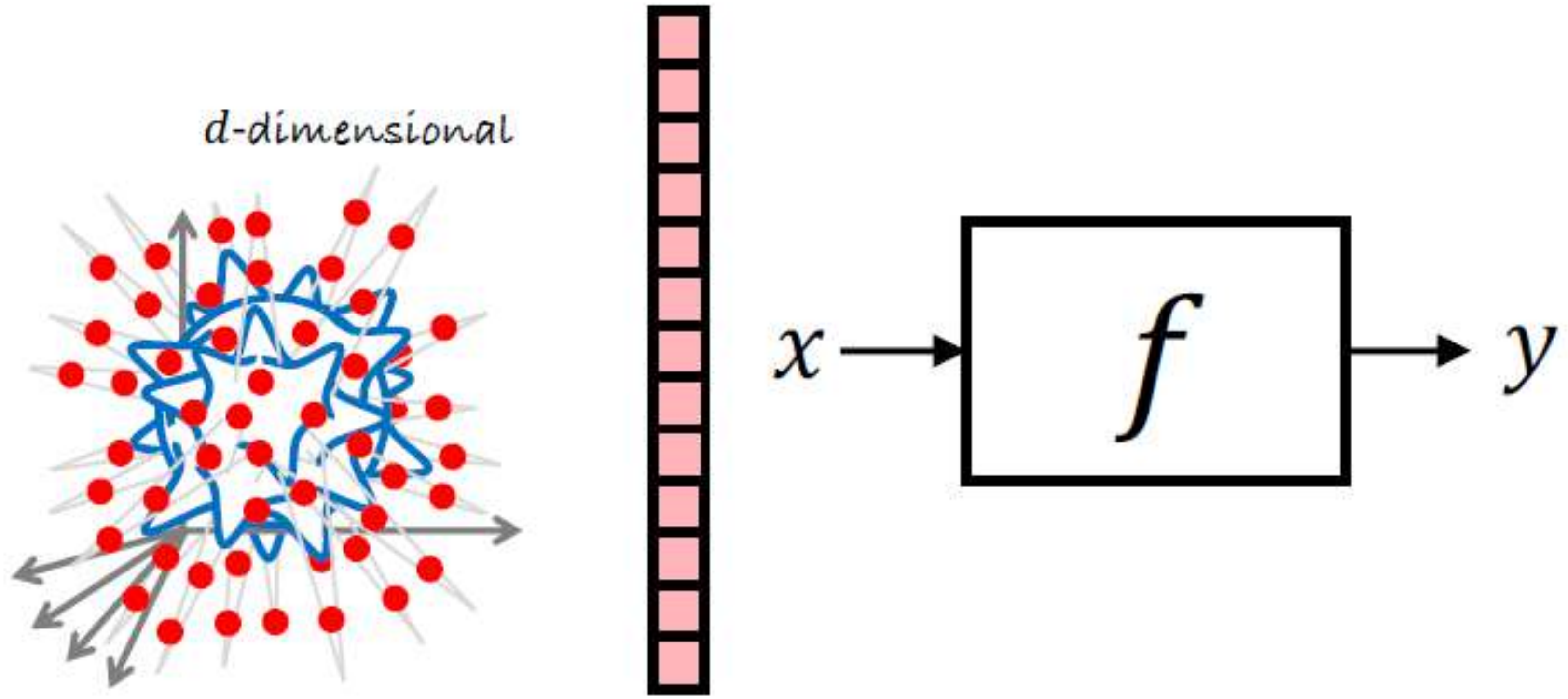


Curse of dimensionality





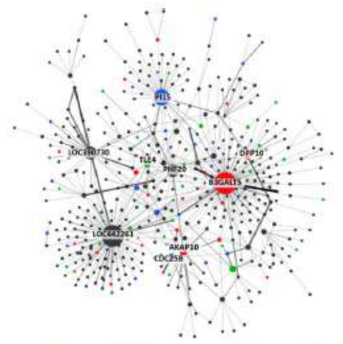
Curse of dimensionality

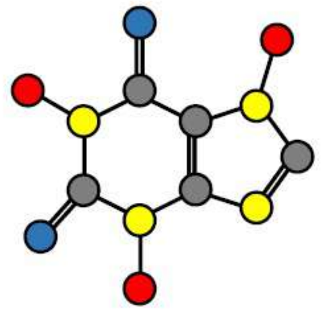


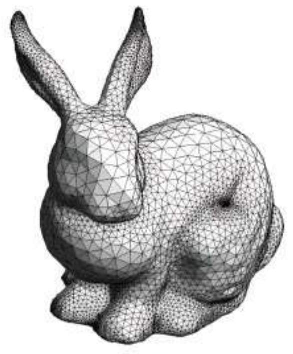
$O(\varepsilon^{-d})$ samples

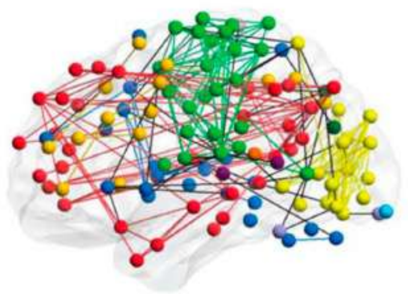
Curse of dimensionality is everywhere

Social networks


Interaction networks


Molecules


Meshes

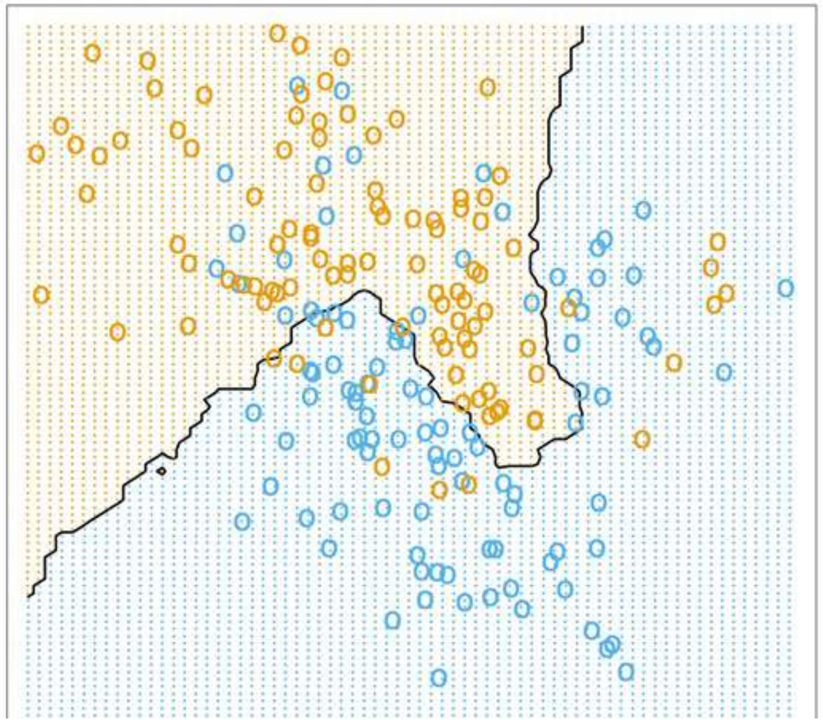

Functional networks

How do you estimate a function? Example with NNC

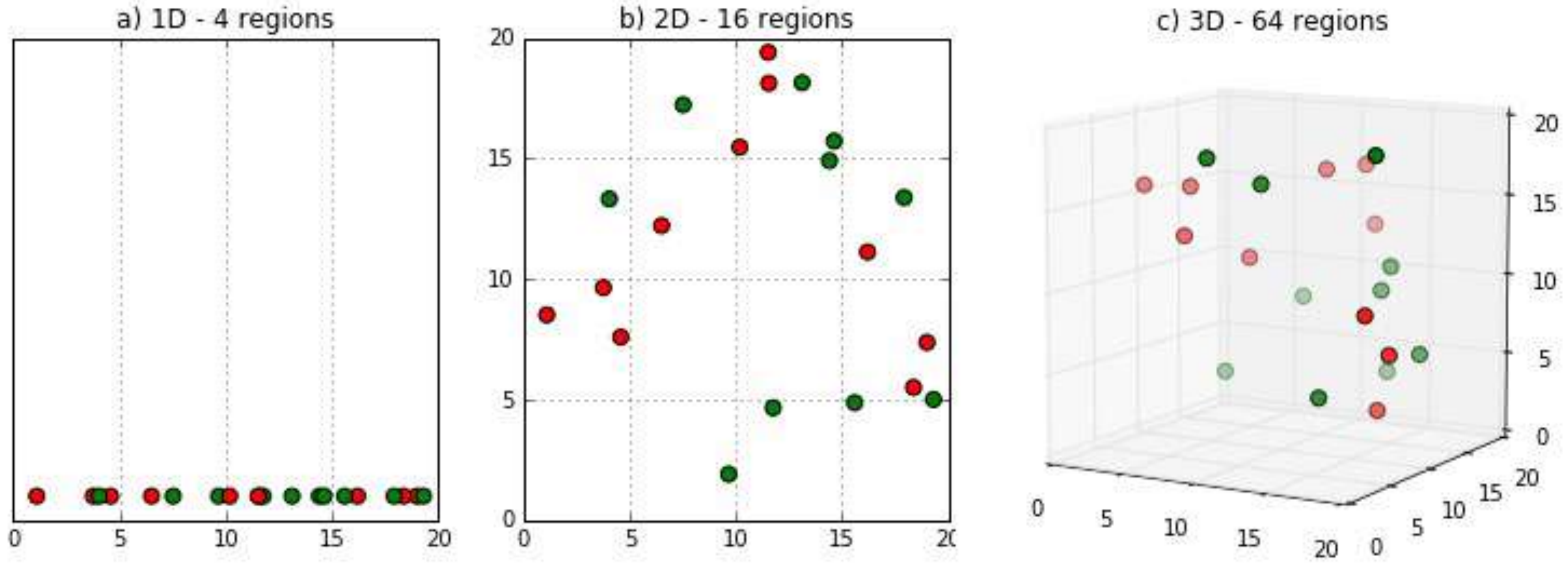
Supervised classification or regression often rely on local averages:

- Classification : you know the classes of n points from your learning database, you can classify a new point x by computing the most represented class in the neighborhood of x.

15-Nearest Neighbor Classifier



However…

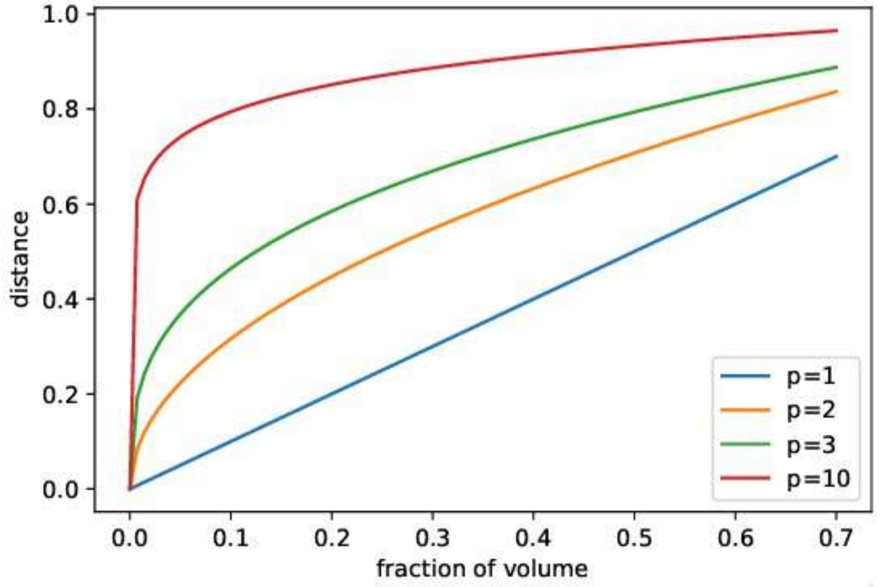


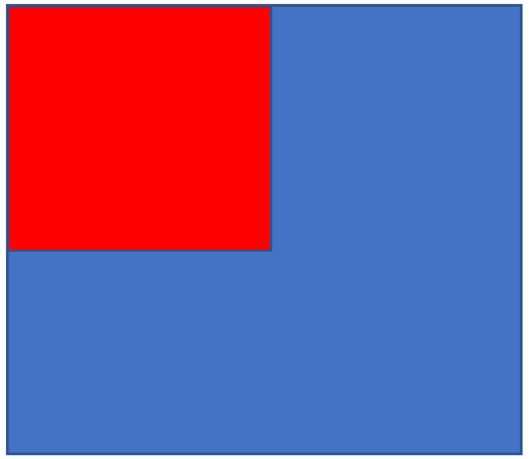
High dimensional spaces are empty

Assume your data lives in $[0,1]^p$. To capture a neighborhood which represents a fraction s of the hypercube volume, you need the edge length to be $s^{1/p}$

Area of the hypercube of length s

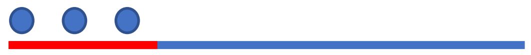
- $s = 0.1$, $p = 10$, $s^{1/p} = 0.63$
- $s = 0.01$, $p = 10$, $s^{1/p} = 0.8$



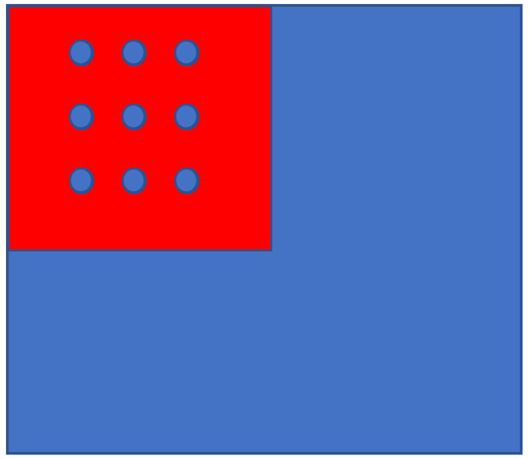


Neighborhoods are no longer local

Curse of dimensionality



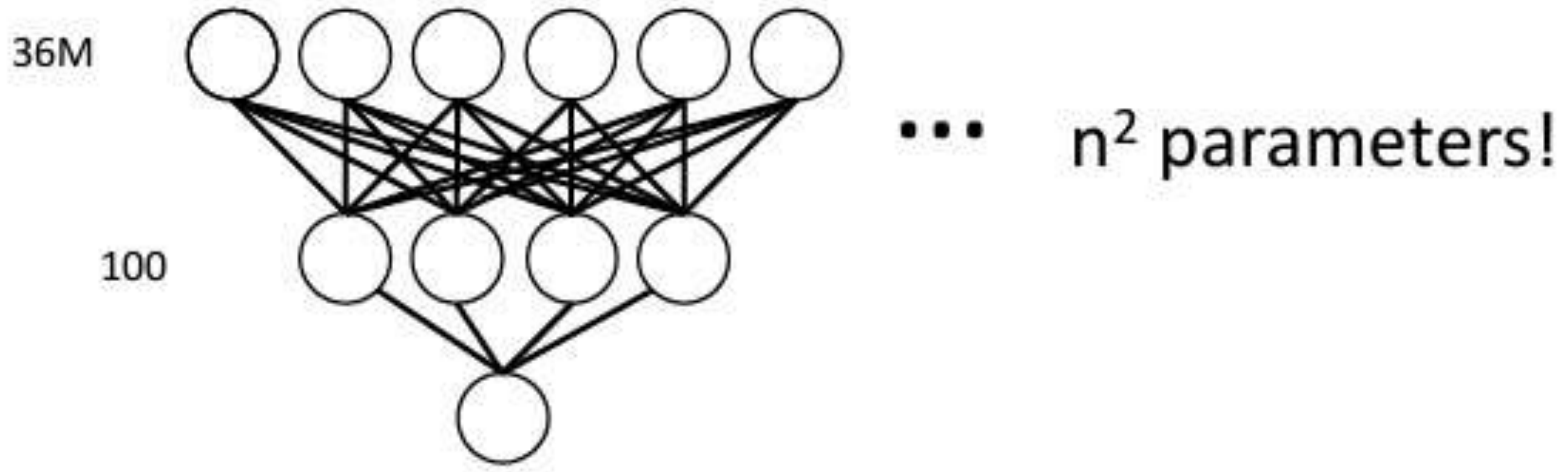
To approximate a (Lipschitz) continuous function $f\colon \mathbb{R}^d \to \mathbb{R}$ with ϵ accuracy one needs $O(\epsilon^{-d})$ samples



Input image resolution = 12 Mpixel * 3 channels = 36M elements
With $\epsilon \sim 0.1$, we need $10^{36000000}$ samples (10^{78} to 10^{82} atoms in the known, observable universe)

Same sample density (same precision): we need P^d samples

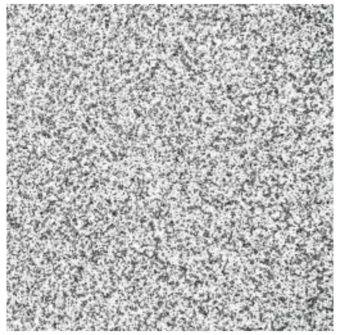
Curse of dimensionality: networks



- Input image resolution = 12 Mpixel * 3 channels = 36M elements
- MLP with one hidden layer of width=100 has **3.6B parameters**

Do we need all parameters? This is true if the image space has all possible images

but also



But the statistics of the images is translation and scale invariant

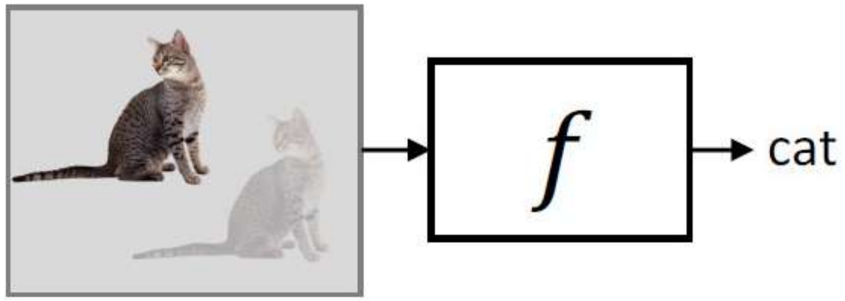
How can we fix this problem?

Look at data manifold and adapt approximation:

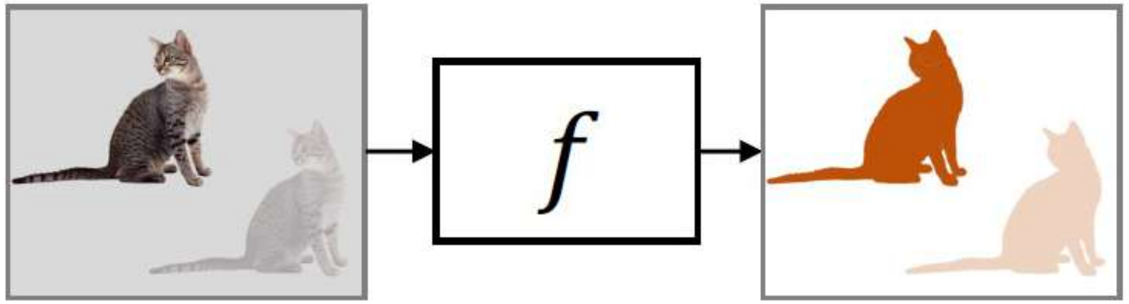
Exploit the geometry of the data to come up with new notions of regularity to use to choose better functional spaces to target.

Fight against the curse: priors on the function space. Equivariance

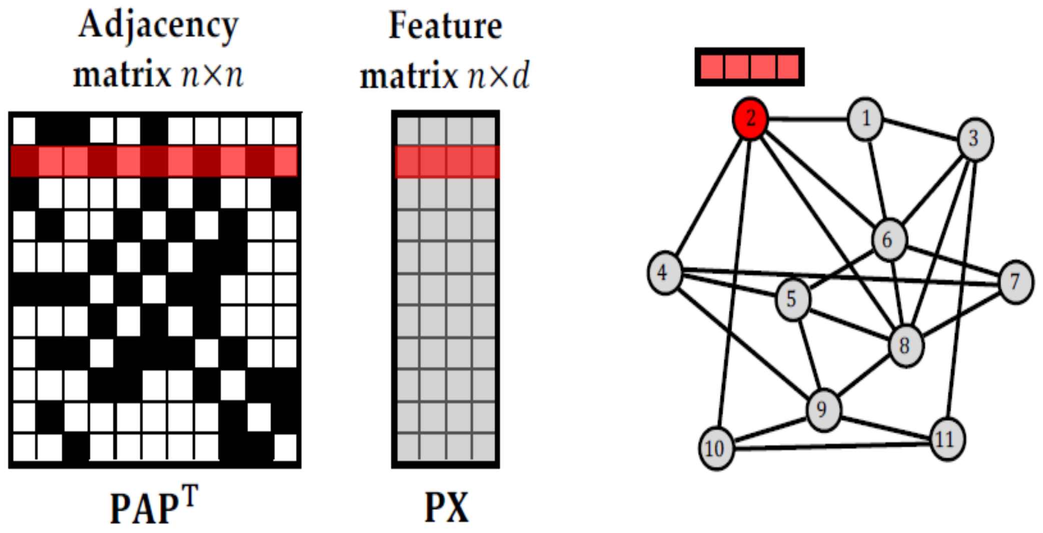
$\mathfrak{G}$-invariance $f(\rho(\mathfrak{g})x) = f(x)$



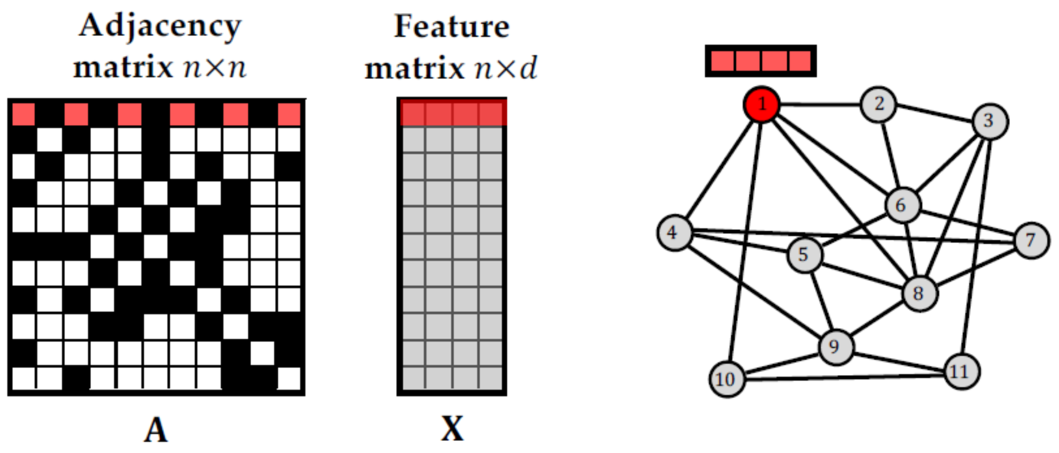
$\mathfrak{G}$-equivariance $f(\rho(\mathfrak{g})x) = \rho(\mathfrak{g})f(x)$



Example with graphs, graph isomorphism



arbitrary ordering of nodes



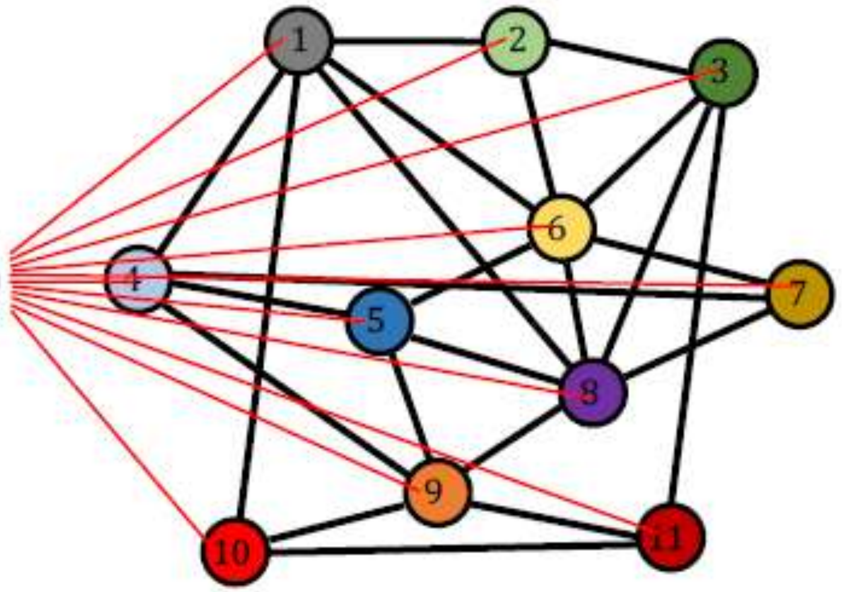
arbitrary ordering of nodes

It is a very good idea to look for permutation invariant functions!

Invariant Graph Functions

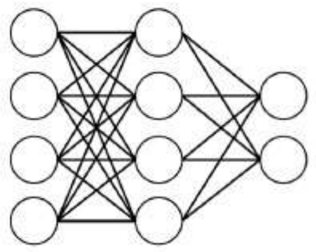
permutation-invariant

$$f\left(\mathbf{PX}, \mathbf{PAP}^\top\right) = f(\mathbf{X}, \mathbf{A})$$

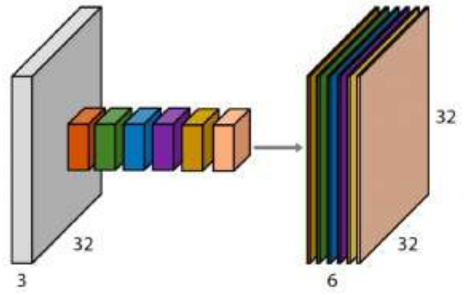


How do you contruct a permutation Invariant layer?

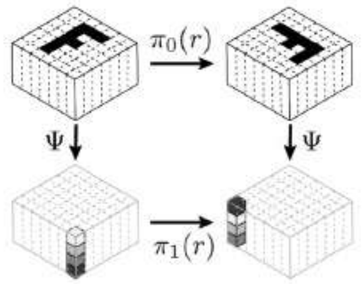
Other examples



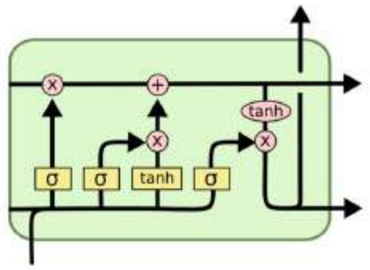
Perceptrons
Function regularity



CNNs
Translation

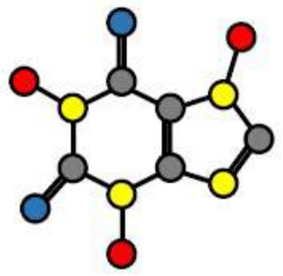


Group-CNNs
Translation+Rotation,
Global groups



LSTMs
Time warping

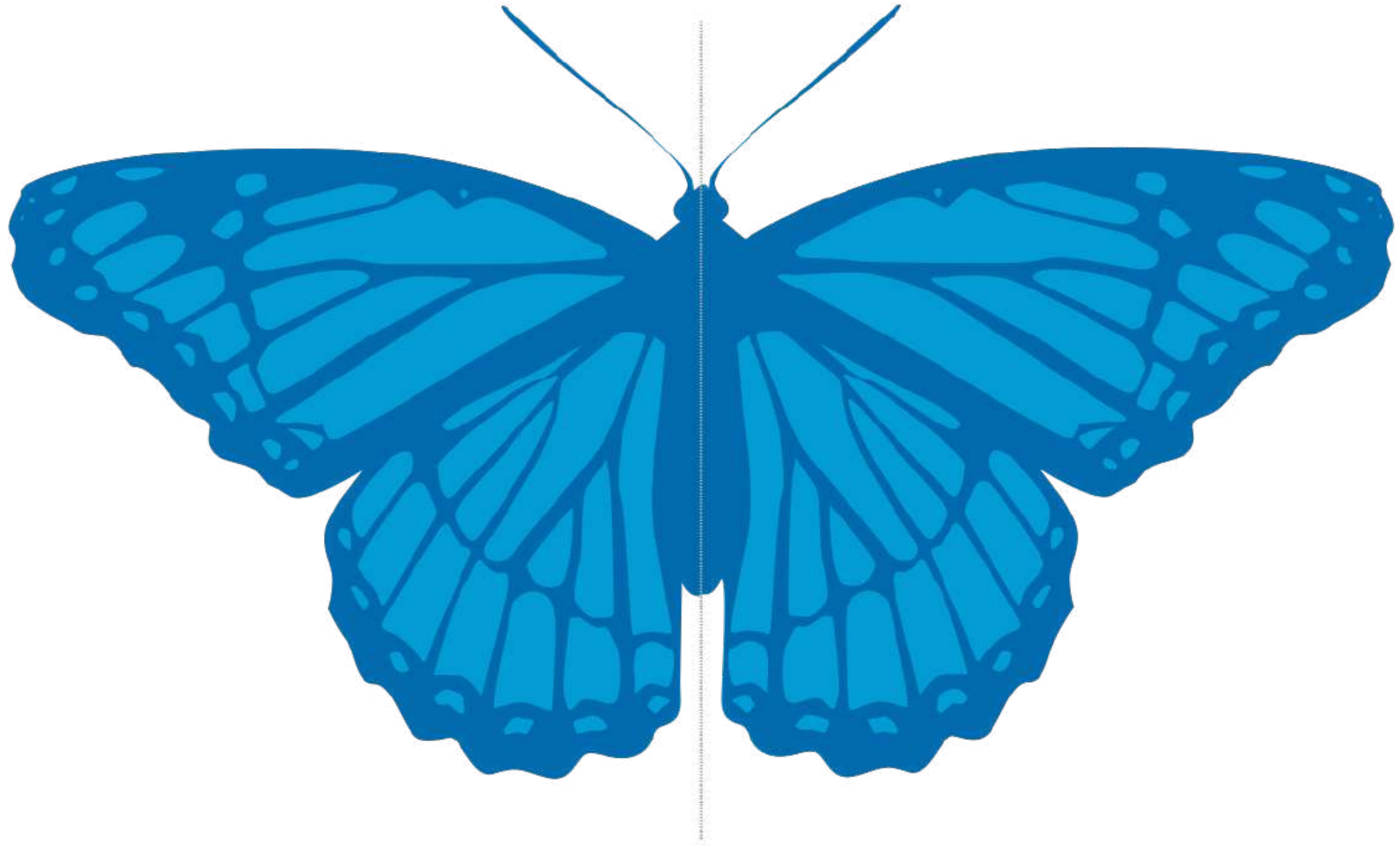
DeepSets / Transformers
Permutation



GNNs
Permutation

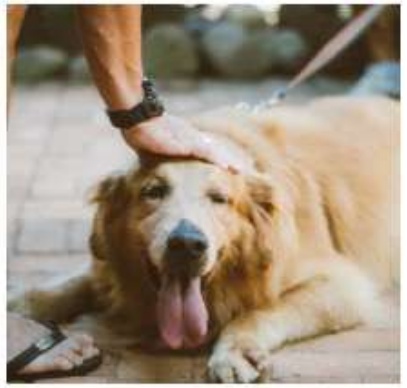
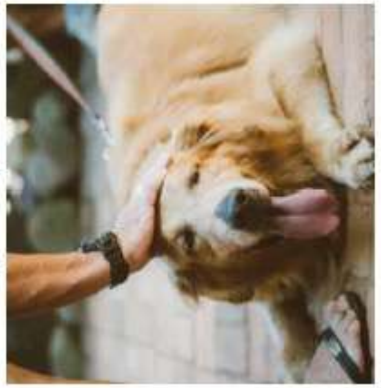
Intrinsic CNNs
Isometry / Gauge choice

Symmetry/Invariance



Symmetries of the Label Function

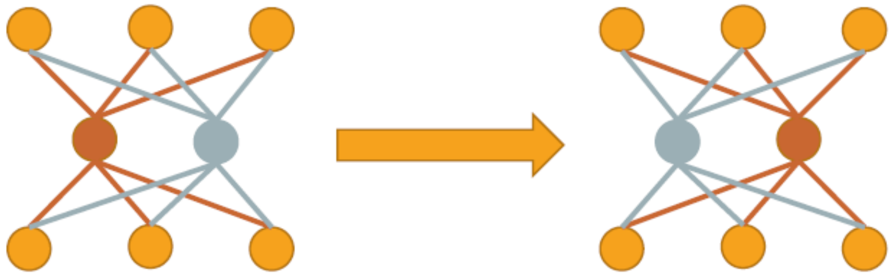
- Let $\mathcal{X}$ denote the input space and $\mathcal{Y}$ the label space
- Let $L : \mathcal{X} \rightarrow \mathcal{Y}$ be the ground-truth label function
- A transformation $\mathfrak{g} : \mathcal{X} \rightarrow \mathcal{X}$ is a symmetry if $L \circ \mathfrak{g} = L$

L() = L() = "dog"

Symmetries of the Parameterization

- Let $\mathcal{X}$ denote the input space, $\mathcal{Y}$ the label space, and $\mathcal{W}$ the weight space
- Let $\mathrm{f} : \mathcal{X} \times \mathcal{W} \rightarrow \mathcal{Y}$ denote a model (e.g. neural network)
- A transformation $\mathrm{g} : \mathcal{W} \rightarrow \mathcal{W}$ is a symmetry of the parameterization if

$$f(x, \mathrm{g}\, w) = f(x, w) \quad \text{for all } \mathrm{x} \in \mathcal{X} \text{ and } \mathrm{w} \in \mathcal{W}$$



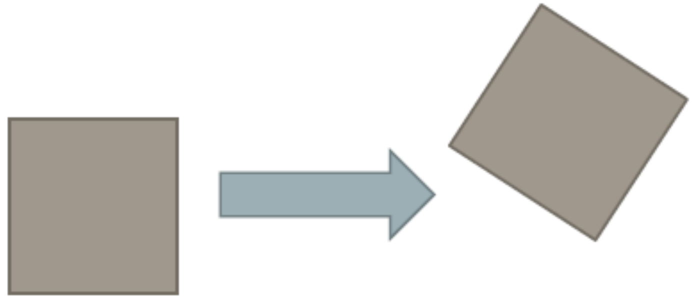
Swapping the incoming and outgoing connections of two neurons in the same layer does not change the input-output map $f(\cdot\,, \mathrm{w})$

What is the structure that allows to bridge symmetries of parameterization and data as equivalent?

Symmetries of the data: transformations, groups

Example: Euclidean planar motions acting on $\mathbb{R}^2$:

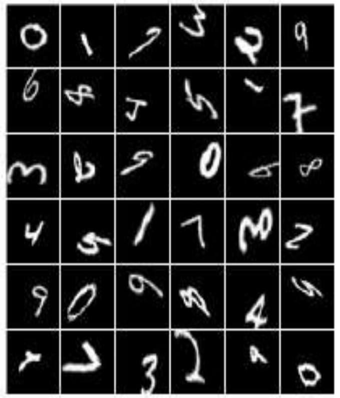
$$((\theta, t_{\mathrm{x}}, \mathrm{t_y}), (\mathrm{x}, \mathrm{y})) \mapsto \begin{bmatrix} \cos\theta & \sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

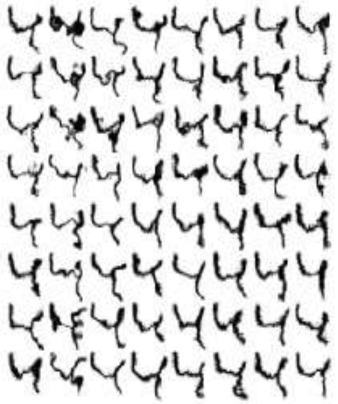


Transformations groups

- compact: rotation
- locally compact: translation, scaling (multiplication), affine
  e.g., Translation group: linear operator $T_\tau : \mathcal{X} \to \mathcal{X}$

$$T_\tau x(p) = x(p - \tau), \quad \forall p, \tau \in \mathbb{R}, x \in \mathcal{X}$$





- non-locally compact: diffeomorphisms, local/global deformations
  e.g. given a smooth map $d : \mathbb{R} \to \mathbb{R}$: linear operator $D_d : \mathcal{X} \to \mathcal{X}$

$$D_d x(p) = x(d(p)), \quad \forall p \in \mathbb{R}, x \in \mathcal{X}$$

- non-group transformations
  - e.g. 3D rotations, illumination

Group properties

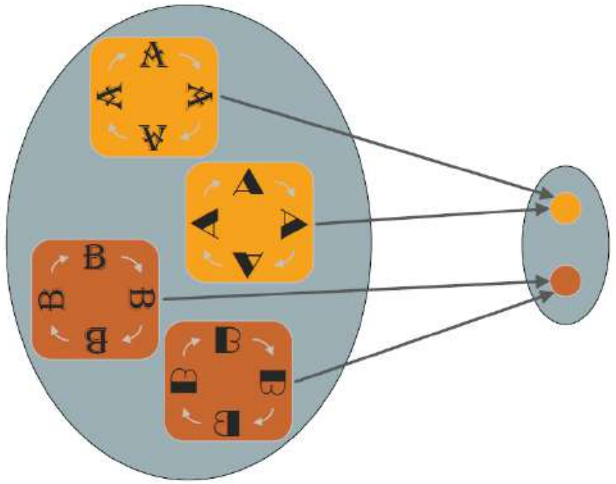
A *group* is a set $\mathfrak{G}$ with a binary operation denoted $\mathfrak{g}\mathfrak{h}$ satisfying the following properties:

- *Associativity:* $(\mathfrak{g}\mathfrak{h})\mathfrak{k} = \mathfrak{g}(\mathfrak{h}\mathfrak{k})$ for all $\mathfrak{g}, \mathfrak{h}, \mathfrak{k} \in \mathfrak{G}$
- *Identity*. there exists a unique $\mathfrak{e} \in \mathfrak{G}$ satisfying
$$\mathfrak{g}\mathfrak{e} = \mathfrak{e}\mathfrak{g} = \mathfrak{g}$$
- *Inverse:* for each $\mathfrak{g} \in \mathfrak{G}$ there is a unique inverse $\mathfrak{g}^{-1} \in \mathfrak{G}$,

  such that $\mathfrak{g}\mathfrak{g}^{-1} = \mathfrak{g}^{-1}\mathfrak{g} = \mathfrak{e}$
- *Closure*: for every $\mathfrak{g}, \mathfrak{h} \in \mathfrak{G}$, we have $\mathfrak{g}\mathfrak{h} \in \mathfrak{G}$

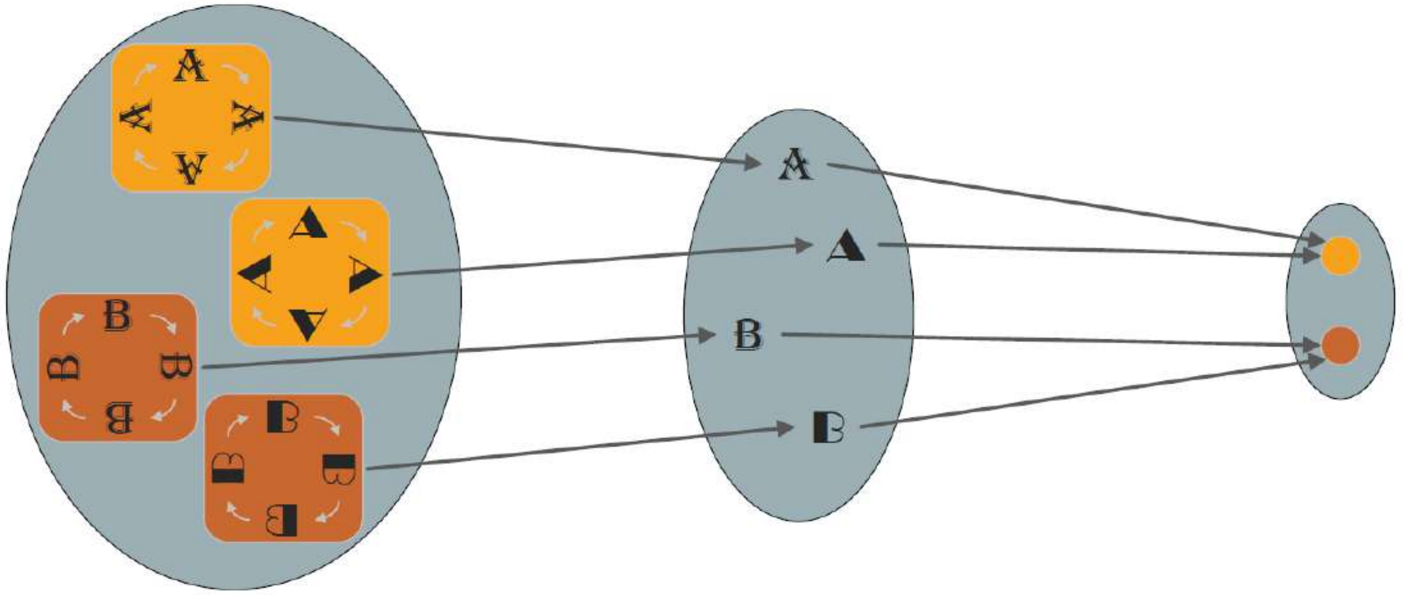
Invariant representations

Orbits & equivalence relations



$$O_x = \{gx \mid x \in X,\, g \in G\}$$

Invariant Representations



Invariant representations

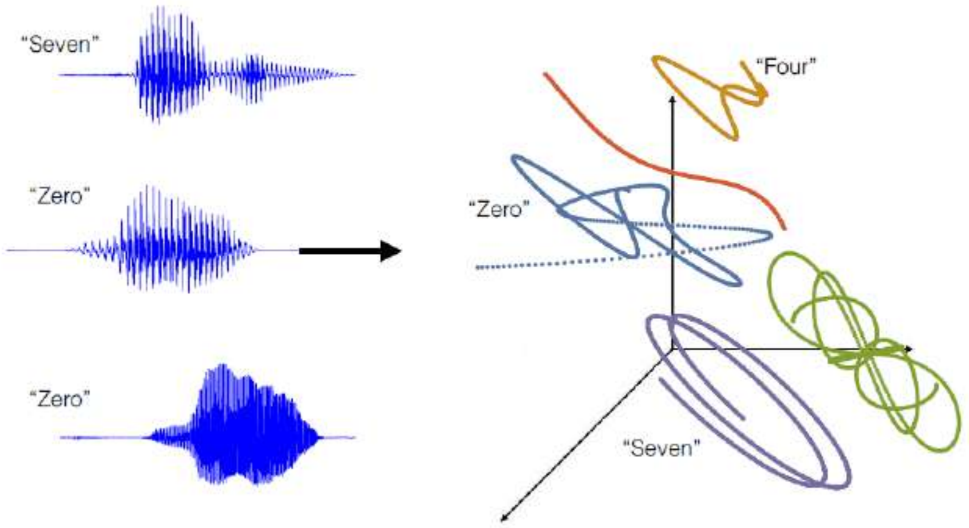
Orbit $O_s = \{gs \in \mathcal{S} | \, g \in G\} \in \mathcal{S}$

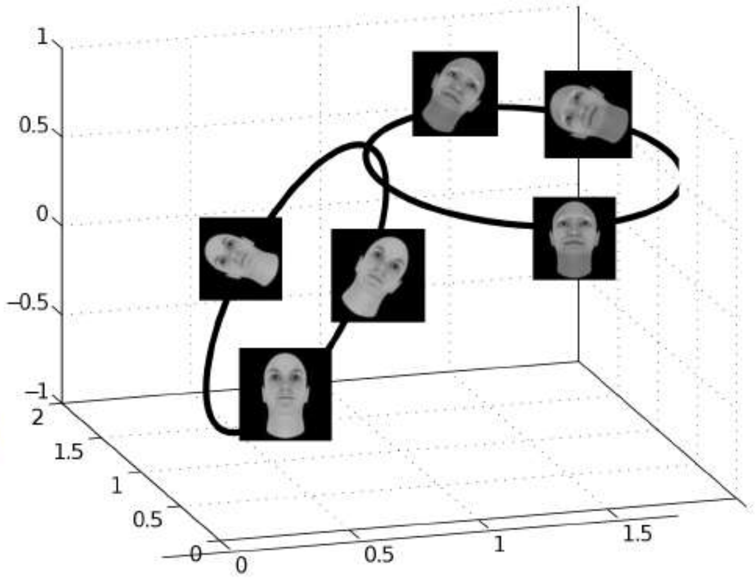
- Transformations as actions of a group element $g \in G$ on s

$$s' = gs(x) = s(g^{-1}x), x \in \mathbb{R}$$

- Equivalence relation, partition of $\mathcal{S}$

$$s \sim s' \iff \exists g \in G : s' = gs, \quad \forall s, s' \in \mathcal{S}$$





Invariant and selective representations

Invariance:

$$x' \sim x \Rightarrow \Phi(x') = \Phi(x), \quad \forall x, x' \in \mathcal{X}$$

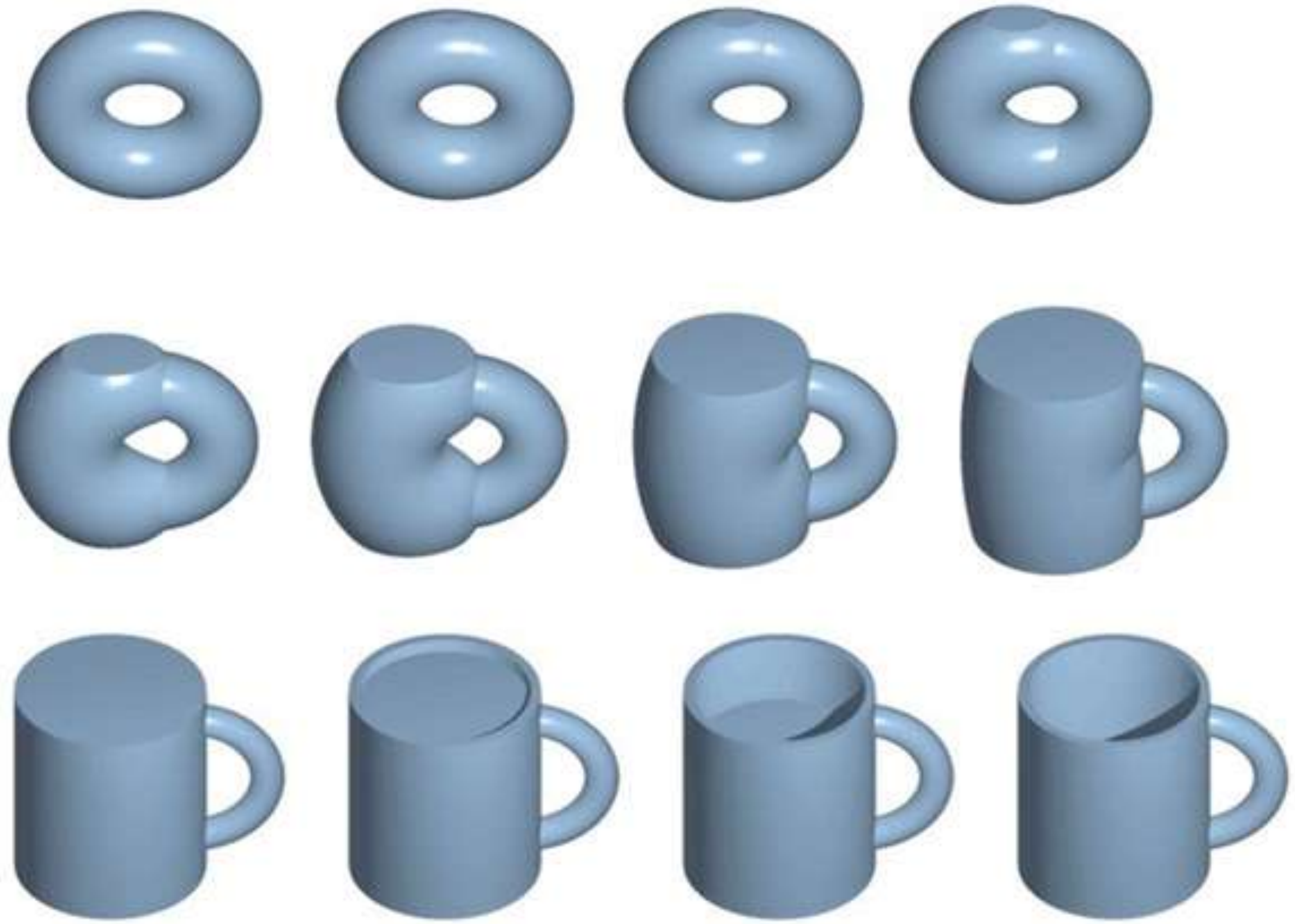
Selectivity:

$$x' \sim x \Leftarrow \Phi(x') = \Phi(x), \quad \forall x, x' \in \mathcal{X}$$

How to define the equivalence classes $x' \sim x$?

Equivalence under the action of a transformation $x' = gx, \quad g \in G$

How much invariance?



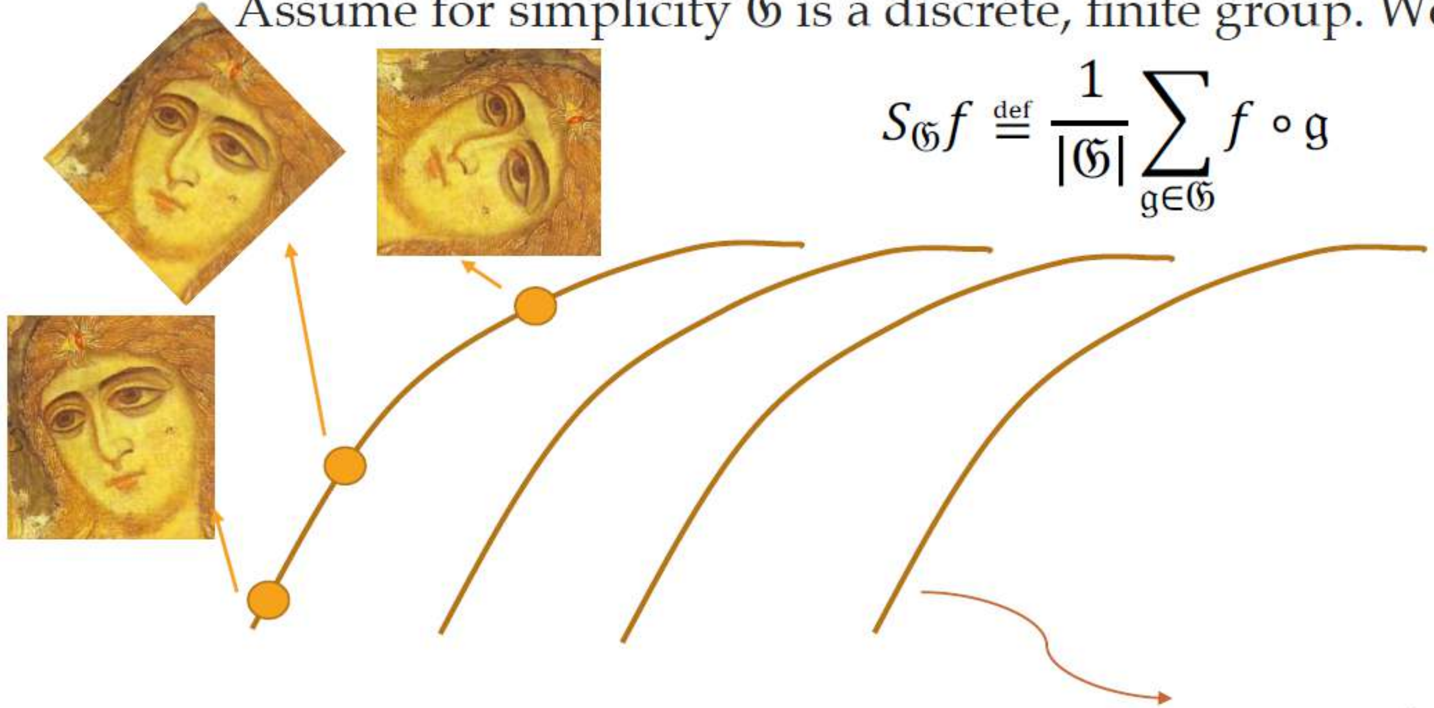
Other interesting geometric priors?

- Invariant/equivariant prior
- Scale separation: multiscale, localized filters
- Compositionality
- Combining invariance and scale separation

Invariant Function Classes

- We may first consider an abstract (non-algorithmic) option, using $\mathfrak{G}$-*smoothing operators:*
- Assume for simplicity $\mathfrak{G}$ is a discrete, finite group. We define

$$S_{\mathfrak{G}} f \overset{\text{def}}{=} \frac{1}{|\mathfrak{G}|} \sum_{\mathfrak{g} \in \mathfrak{G}} f \circ \mathfrak{g}$$



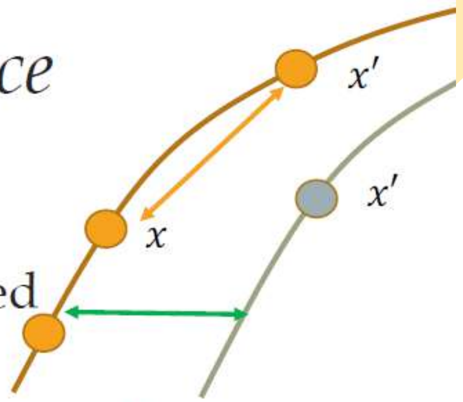
$$S_{\mathfrak{G}} f(x) = \frac{1}{|\mathfrak{G}|} \sum_{\mathfrak{g} \in \mathfrak{G}} f(\mathfrak{g}.x)$$

$S_{\mathfrak{G}}$ thus averages a function over group orbits.

$\boxed{\text{Observe that } S_{\mathfrak{G}} f^* = f^*.}$

Group Orbits $\mathfrak{G}.x = \{\mathfrak{g}.x ; \mathfrak{g} \in \mathfrak{G}\}$

On the Sample Complexity of Learning under Invariance



- We can consider, as before, the Lipschitz class $\mathcal{F}$, and its $\mathfrak{G}$-smoothed version.

$$f \in \mathcal{F}: |f(x) - f(x')| \leq \beta \|x - x'\|; \;\; f \in S_{\mathfrak{G}}\mathcal{F}: |f(x) - f(x')| \leq \beta \, inf_{\mathrm{g}} \|x - \mathrm{g}.x'\|$$

- **Theorem [BVB'21]:** Using a $\mathfrak{G}$-*invariant kernel ridge regression, the generalization error of learning a Lipschitz,* $\mathfrak{G}$-*invariant function* f^* *satisfies*

$$\mathbb{E}\mathcal{R}(\tilde{f}) \lesssim (|\mathfrak{G}|\, n)^{\frac{-1}{d}}$$

  - Group size $|\mathfrak{G}|$ can be exponential in dimension (local translations, or $\mathcal{S}_d$).

[Mei, Misiakiewicz, Montanari, 21][Bietti, Venturi, **B.**'21][Bousquet & Von Luxburg'04]

Invariant kernels and Convolutional representations

Convolution layer representation (with average pooling),

$$\Phi(x) = \left(\sum_g s(\langle x, gt_1 \rangle), \ldots, \sum_g s(\langle x, gt_T \rangle) \right), \quad x \in \mathcal{X}.$$

The **associated kernel** can be written as

$$K(x,x) = \langle \Phi(x), \Phi(x') \rangle = \sum_t \left(\sum_g s(\langle x, gt_1 \rangle) \cdot \sum_g s(\langle x', gt_1 \rangle) \right)$$

Using linearity

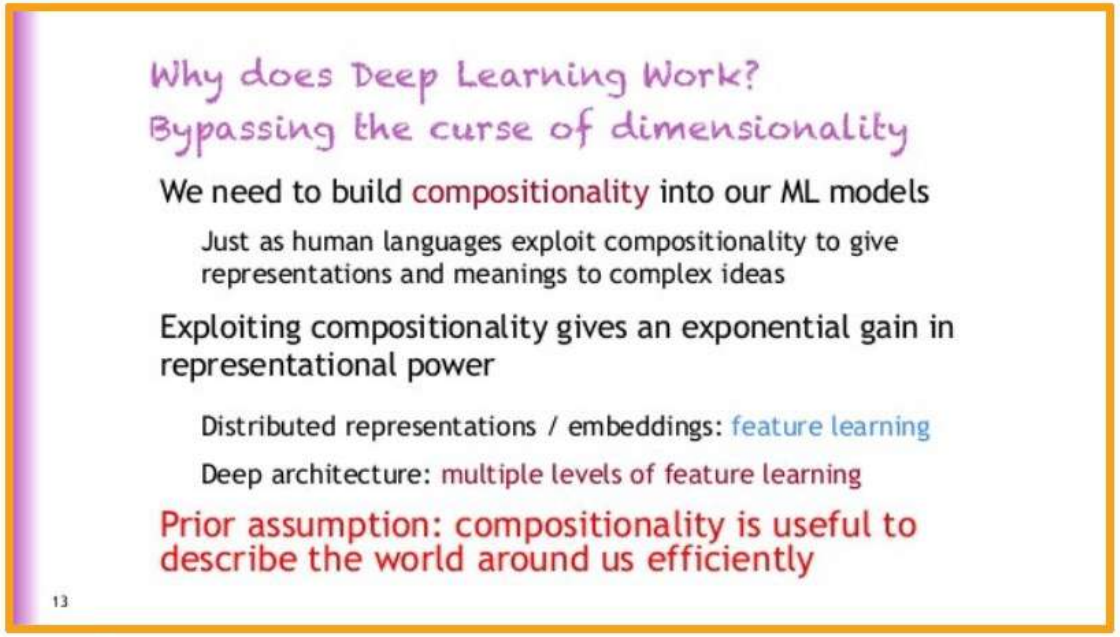
$$K(x,x') = \sum_{g}\sum_{g'} \underbrace{\left(\sum_{t} s(\langle x, gt_1\rangle)\cdot s(\langle x', g't_1\rangle)\right)}_{K'(x,x')}.$$

Convolution kernels

$$K(x,x') = \sum_{g}\sum_{g'} K'(gx, g'x')$$

- Kernel of the above form are a special case of **convolution kernels**.
- If $g_1,\ldots,g_G$ form a **group** then the kernel is invariant $K(x,x') = K(x,gx')$, i.e. $\Phi(x) = \Phi(gx)$.

Deep Learning "Inductive Bias": Compositionality



Why does Deep Learning Work?

Bypassing the curse of dimensionality

We need to build compositionality into our ML models

Just as human languages exploit compositionality to give representations and meanings to complex ideas

Exploiting compositionality gives an exponential gain in representational power

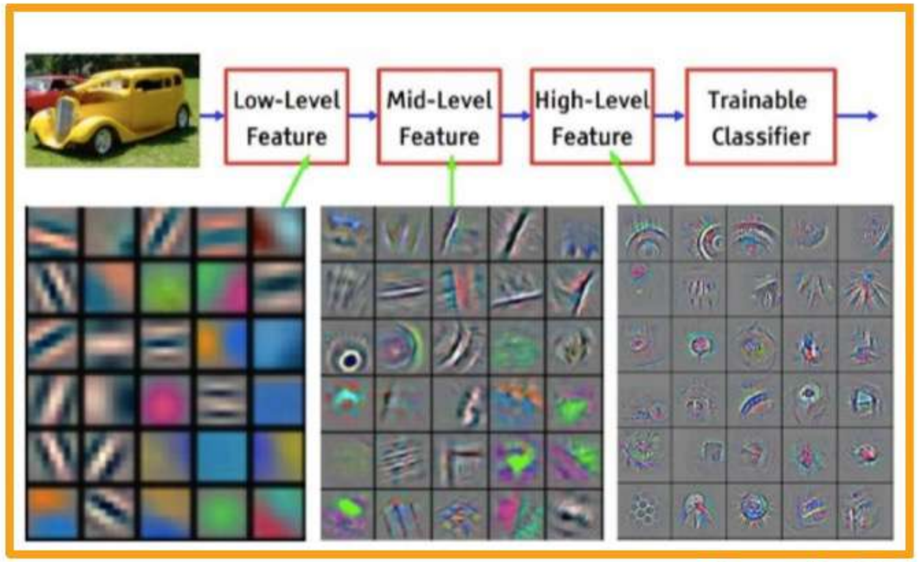
Distributed representations / embeddings: feature learning

Deep architecture: multiple levels of feature learning

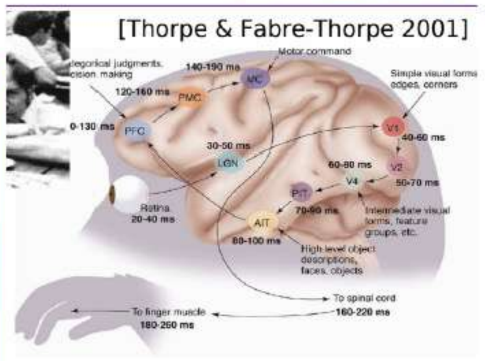
Prior assumption: compositionality is useful to describe the world around us efficiently

13

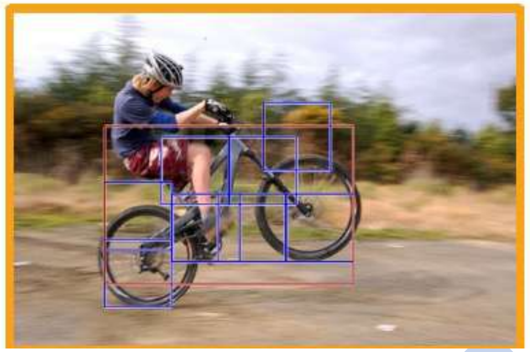
Yoshua Bengio



Yann LeCun


[Thorpe & Fabre-Thorpe 2001]
Motor command
egorical judgments, cision making
140-190 ms
MC
Simple visual forms edges, corners
120-160 ms
PMC
0-130 ms
PFC
V1
40-60 ms
30-50 ms
LGN
60-80 ms
V2
V4
50-70 ms
PIT
Retina 20-40 ms
70-90 ms
Intermediate visual forms, feature groups, etc.
AIT
80-100 ms
High level object descriptions, faces, objects
To spinal cord
To finger muscle 180-260 ms
160-220 ms

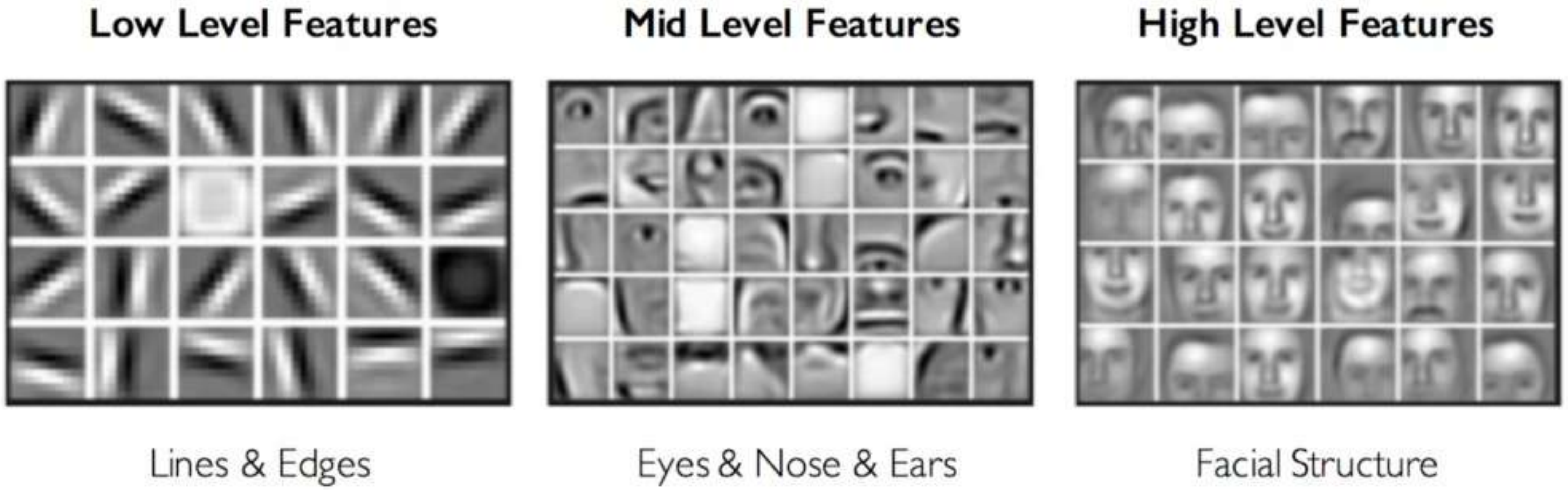
How to formalize this intuition?

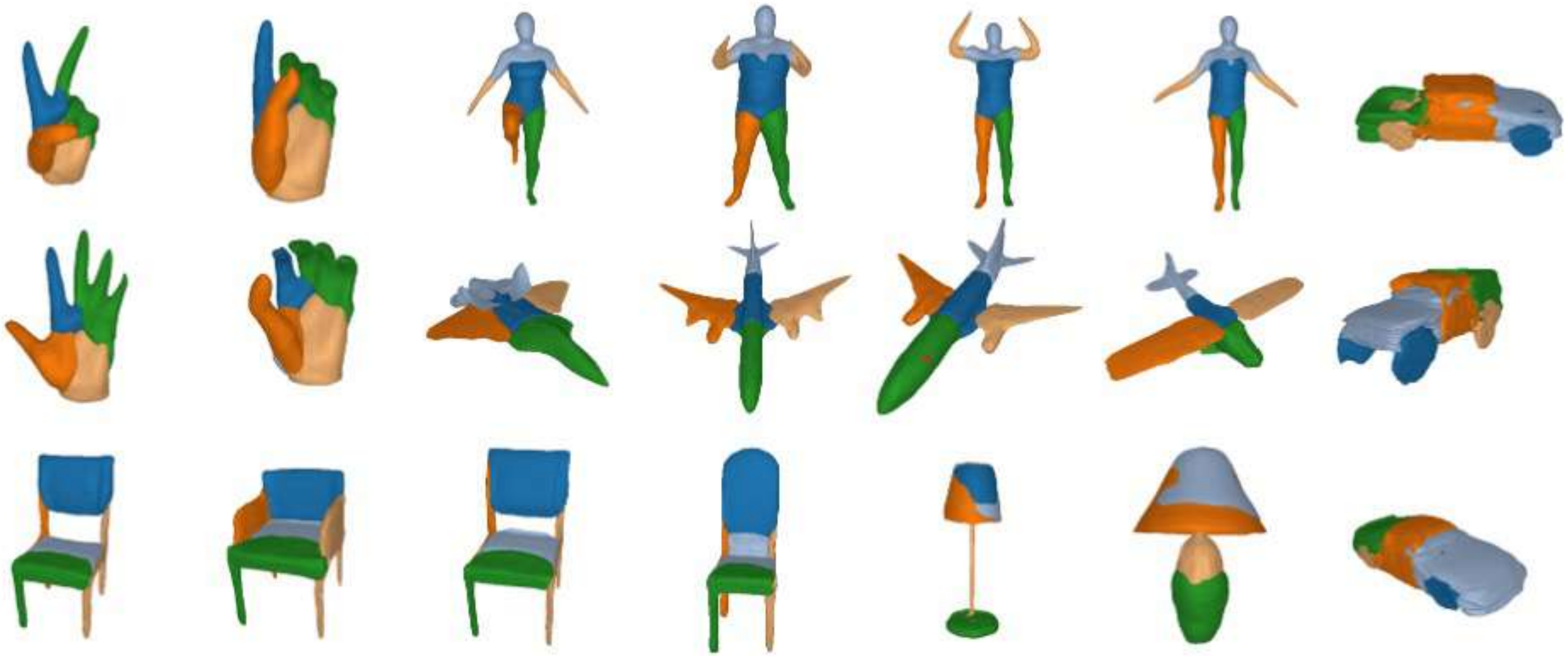


Deformable Parts Model, Felzenswalb et al.

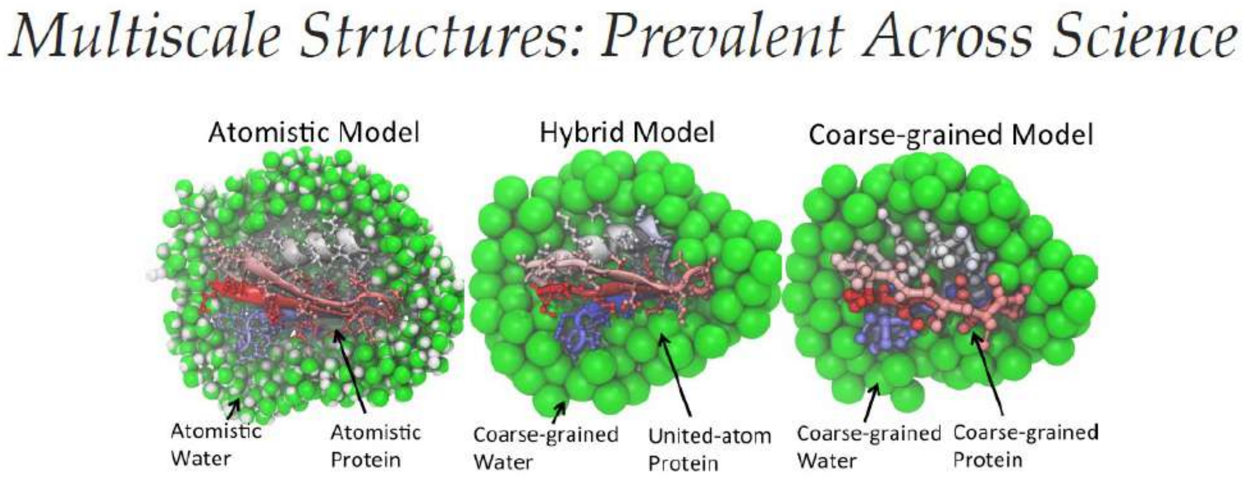
Compositionality prior

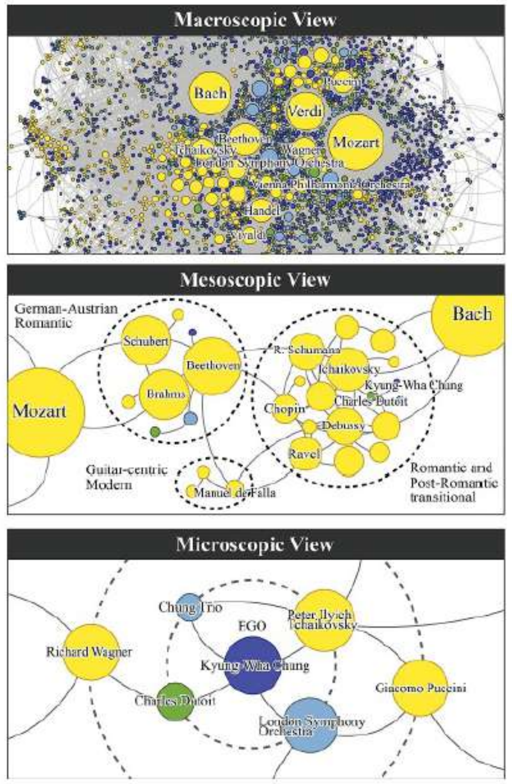
- What is the computational advantage?

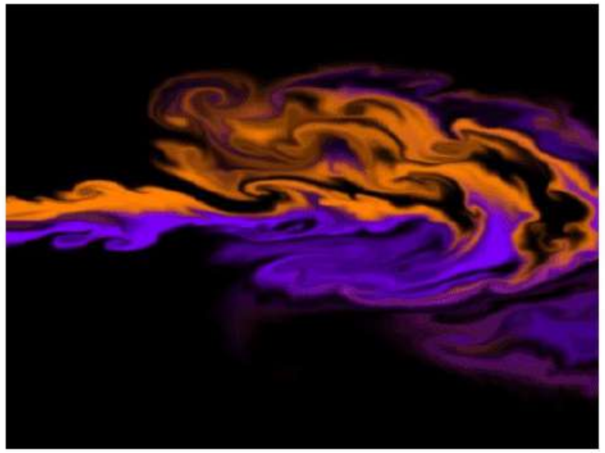


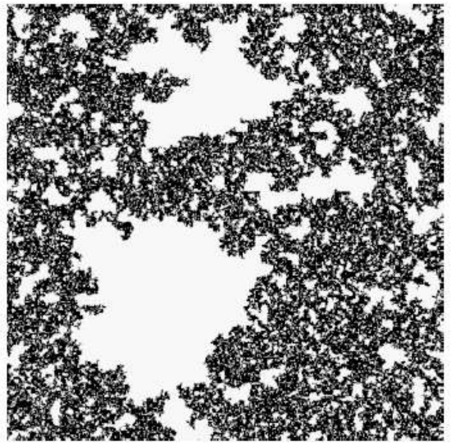


Scale prior/levels of description

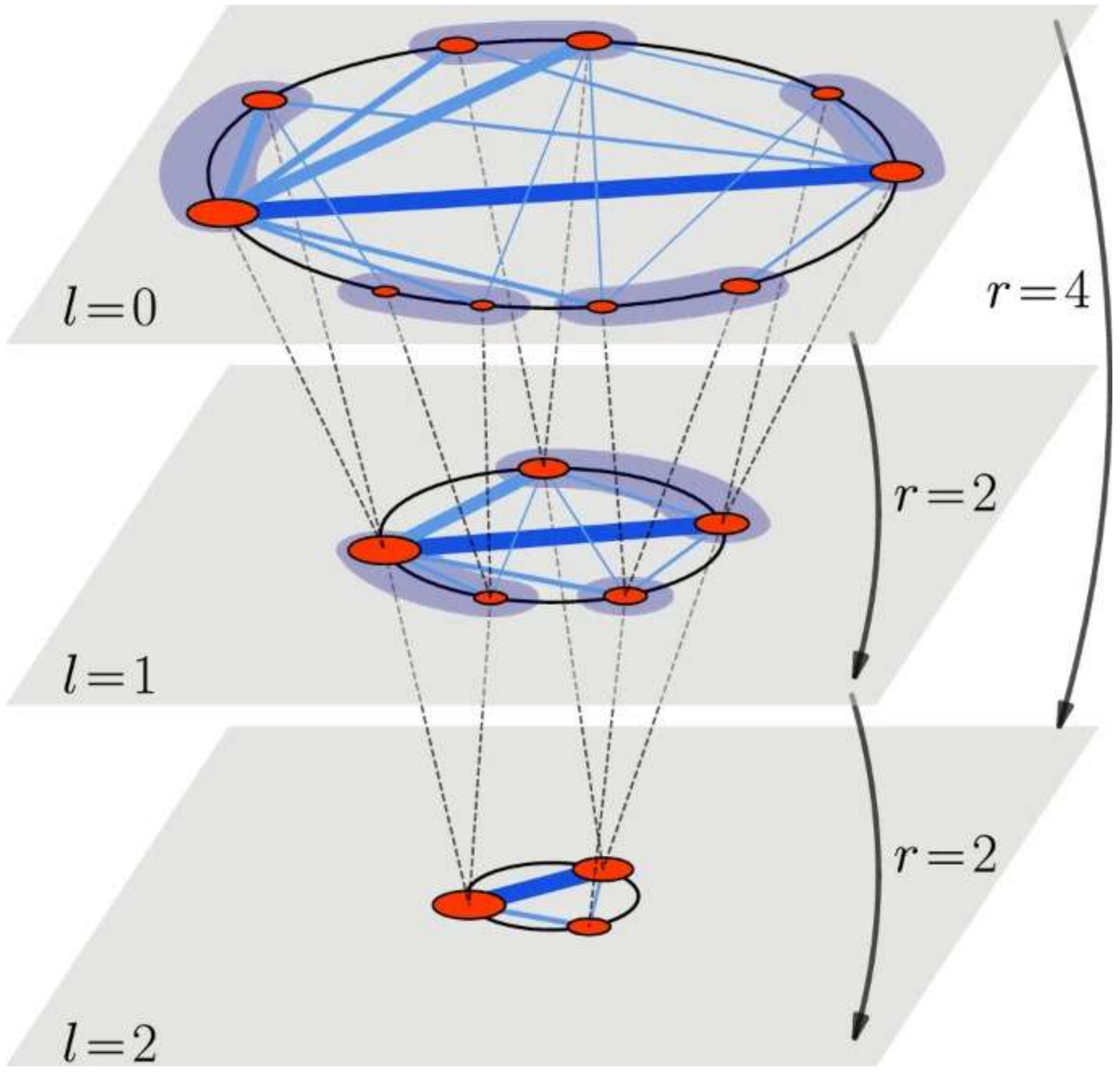




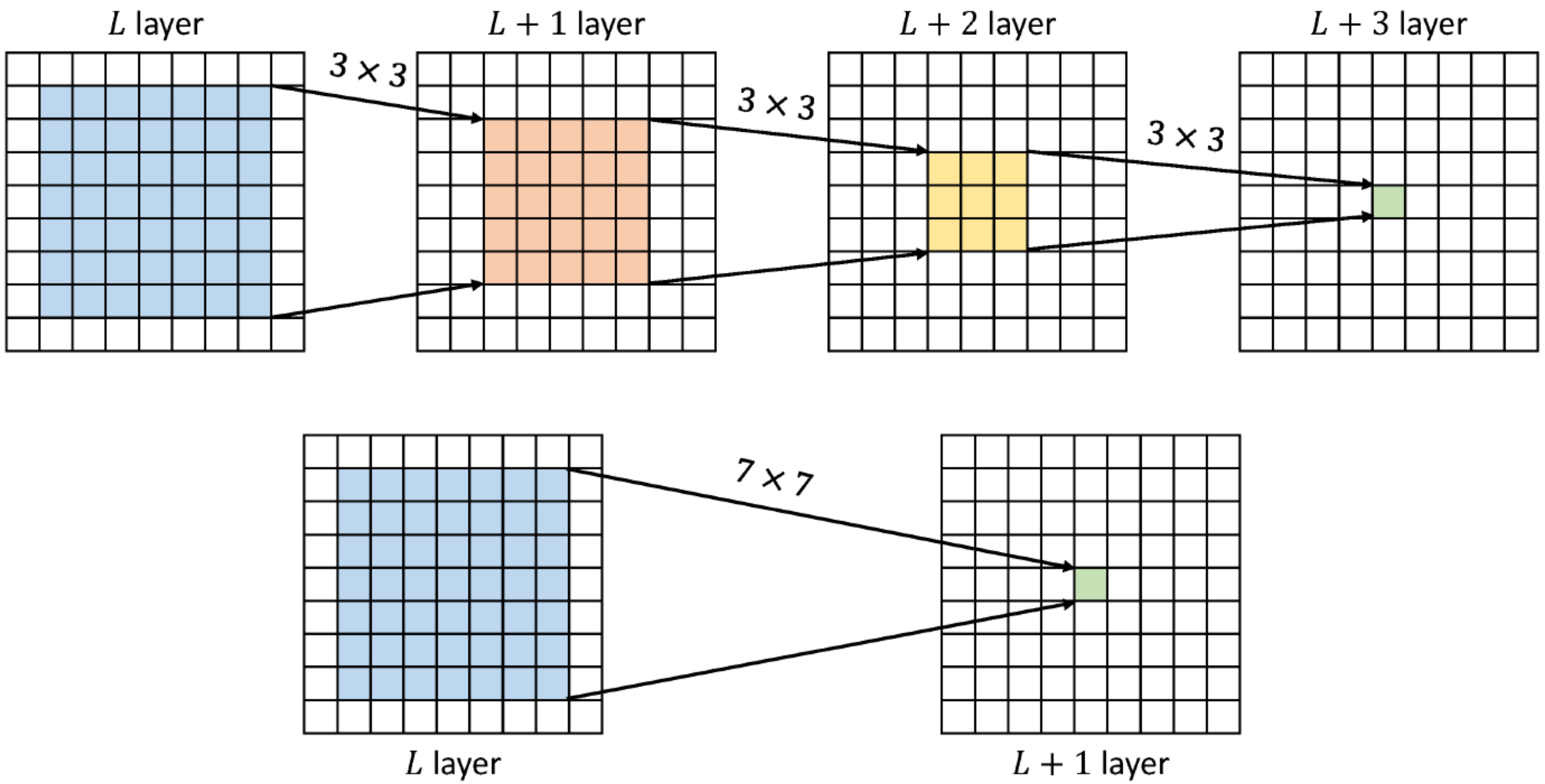




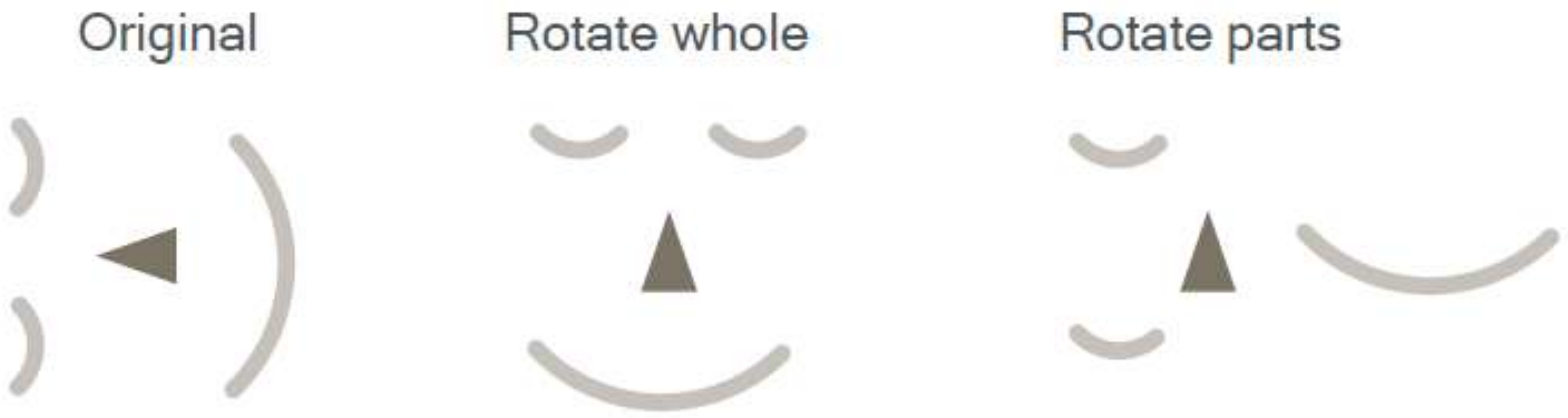
Renormalization group in Physics



Choosing the right scale



Scale and compositionality

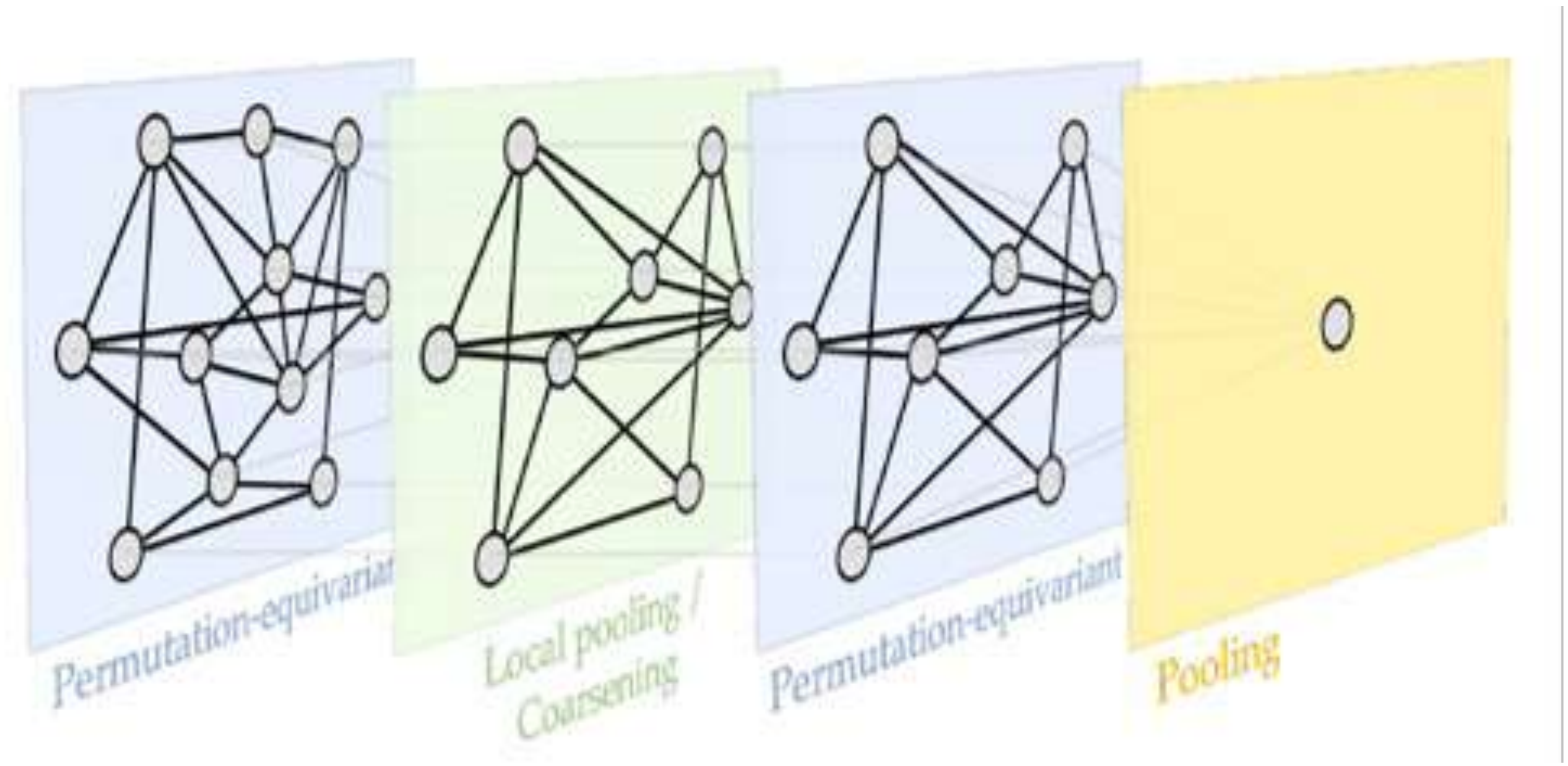


How?

- This suggests a constructive approach to build rich invariants with multiscale structure, with building blocks:
  - *Linear $\mathfrak{G}$-equivariant layer* $B : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega', \mathcal{C}')$, satisfying $B(\mathfrak{g}.x) = \mathfrak{g}.B(x)$ for all $\mathfrak{g} \in \mathfrak{G}$ and $x \in \mathcal{X}(\Omega, \mathcal{C})$.
  - *Nonlinearity* $\sigma : \mathcal{C} \to \mathcal{C}'$ applied element-wise as $\big(\boldsymbol{\sigma}(x)\big)(u) = \sigma(x(u))$.
  - *Local pooling (coarsening)* $P : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega', \mathcal{C})$, such that $\Omega' \subseteq \Omega$.
  - *$\mathfrak{G}$-invariant layer (global pooling)* $A : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{Y}$, satisfying $A(\mathfrak{g}.x) = A(x)$for all $\mathfrak{g} \in \mathfrak{G}$ and $x \in \mathcal{X}(\Omega, \mathcal{C})$.

- These blocks can be defined under very mild conditions on the geometric domain Ω.

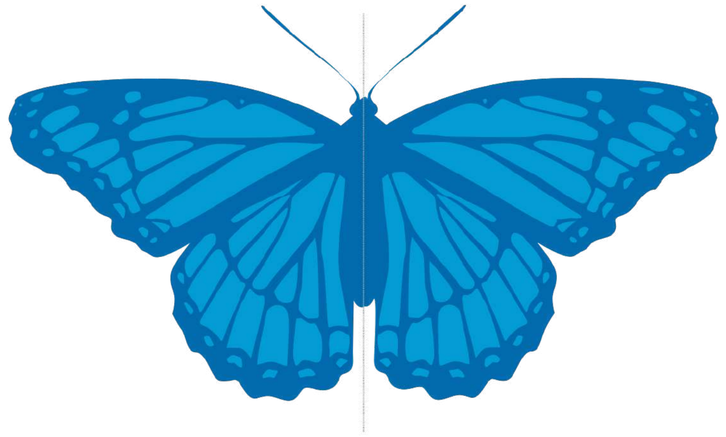
An example with graphs



Takeaway

- Geometric Priors enable a novel class of hypothesis spaces that together may tame the curse of dimensionality.
- Symmetries must be combined with scale separation.
- The language of group invariance and equivariance provides a principled architecture compatible with these priors, while preserving large approximation power.
- Theory side: what precise assumptions on non-linear coarsening operators?

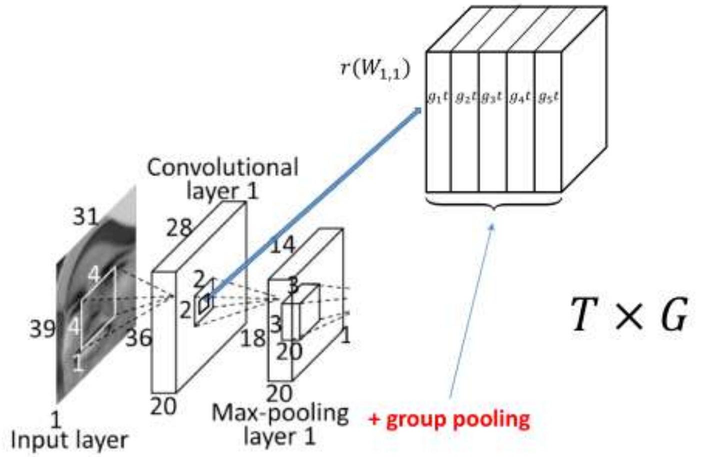
Class 1/2: Learning symmetries



What if the symmetry is not evident?

Rationale

Standard CNNs use translational convolution. A reason why they need such big training sets might be that the translational invariance is not the optimal choice to produce the best invariant representation. We want to learn the symmetry from the data.



Message: **the statistics of the data might be invariant to a broader set of symmetries then translations and to learn them will reduce the sample complexity.**

Let $\mathcal{G}$ a group. Let X set is a collection of orbits of stimuli:

$$O_t = \{t' \mid t' = gt,\ g \in \mathcal{G},\ t \in \mathcal{X}\}, \qquad X = (O_{t_1}, \cdots, O_{t_M})$$

How do we learn the group of symmetry?

Main observation to built the regularization term

If $W = \{g_1 t, \cdots, g_{|\mathcal{G}|} t\},\ t \in \mathbb{R}^d,\ g \in \mathcal{G},\ \mathbb{R}^{d\times d}$

$$(W^T W)_{i,j} = \langle w_i, w_j\rangle = \langle g_i t, g_j t\rangle = \langle t, g_i^{-1} g_j t\rangle$$

Let $\mathcal{G} = \{e, g\}$, $O_t = \{t, gt\}$:

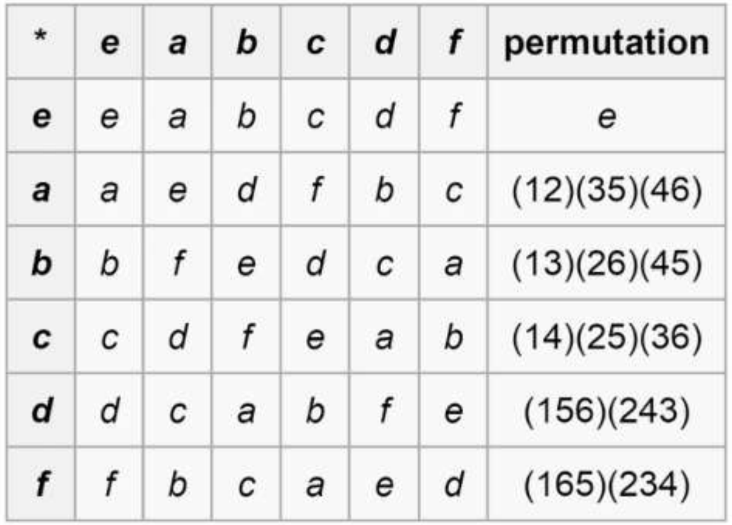
$$W^T W = \begin{bmatrix} \langle t, e^{-1} et\rangle & \langle t, e^{-1} gt\rangle \\ \langle t, g^{-1} et\rangle & \langle t, g^{-1} gt\rangle \end{bmatrix} = \begin{bmatrix} \langle t, et\rangle & \langle t, gt\rangle \\ \langle t, gt\rangle & \langle t, et\rangle \end{bmatrix}$$

The Gramian's **columns are permutations of a single vector**.

1. The problem can be chosen to be low dimensional.
2. We map the problem from $\mathcal{G}$ to the permutation group.

Explanation: group multiplication tables and Cayley theorem

Let $\mathcal{G} = \{e, a, b, c, d, e, f\}$.



*	e	a	b	c	d	f	permutation
e	e	a	b	c	d	f	e
a	a	e	d	f	b	c	(12)(35)(46)
b	b	f	e	d	c	a	(13)(26)(45)
c	c	d	f	e	a	b	(14)(25)(36)
d	d	c	a	b	f	e	(156)(243)
f	f	b	c	a	e	d	(165)(234)

Theorem (Cayley)

Every group $\mathcal{G}$ is isomorphic to a subgroup of the symmetric group acting on $\mathcal{G}$.

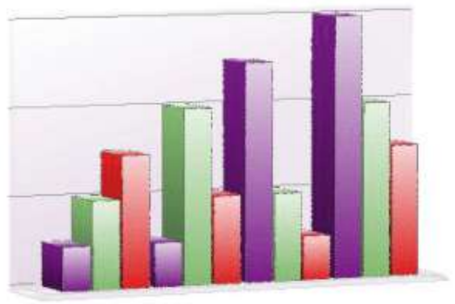
Separating permutation orbits

We need to test if the Gramian columns $G_{:,i}$ are permuted version of the same vector. A choice is to consider the distribution function of the vector values:

$$\mu_\lambda(G_{:,i}) = \sum_{k=1}^{|\mathcal{G}|} \delta(G_{k,i} - \lambda)$$

and impose equality of the distribution:

$$\|\mu(G_{:,i}) - \mu(G_{:,j})\|_2^2 = \sum_{k,l=1}^{|\mathcal{G}|} \int d\lambda \, (\mu_\lambda(G_{k,i}) - \mu_\lambda(G_{:,j}))^2 = 0$$



Separating permutation orbits

For all pairs of Gramian columns we apply:

$$\|\mu(G_{:,i}) - \mu(G_{:,j})\|_2^2 \;\forall\, i,j \;\Leftrightarrow\; \sum_{ikjl=1}^{|\mathcal{G}|} (1 - |\mathcal{G}|\delta_{ij})\delta(w_i^T w_k - w_j^T w_l) = 0$$

or if we approximate δ by its smooth form (letting $\sigma \to 0$)

$$\sum_{ikjl=1}^{|\mathcal{G}|} (1 - |\mathcal{G}|\delta_{ij}) e^{-\frac{(w_i^T w_k - w_j^T w_l)^2}{\sigma^2}} = 0.$$

More compact:

$$r(W) = \tau^T g_\sigma(C\text{vec}(W^T W)) = 0$$

Enforcing the Gramian constraint

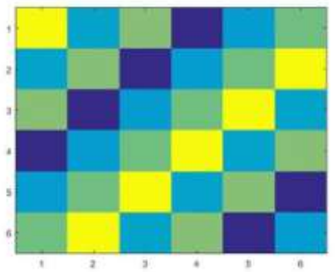
Theorem (Symmetry regularization)

Let matrix $W \in \mathbb{R}^{d \times |\mathcal{G}|}$ and $r : \mathbb{R}^{|\mathcal{G}| \times |\mathcal{G}|} \to \mathbb{R}_+$ with

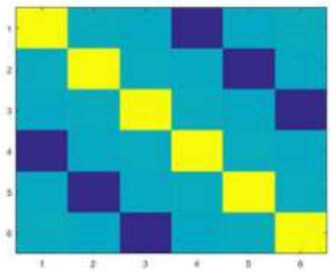
$$r(W) = \tau^T g_\sigma(C\,\mathrm{vec}(W^T W))$$

C, τ is a constant matrix and vector and function g_σ acts as the elementwise gaussian function. If $r(W) = 0$, then $W^T W$ is a permuted matrix and viceversa.

W



$W^T W$



Alternative strategy. Google softmax

```
import sys
!git clone https://github.com/google-research/fast-soft-sort/
softsort_path = './fast-soft-sort'
sys.path.append(softsort_path)

from fast_soft_sort.pytorch_ops import soft_sort
```

Testing the algorithm

- Test set: randomly permuted orbits of vectors w.r.t. Cyclic group ($|\mathcal{G}| = 6$), Dihedral group ($|\mathcal{G}| = 12$), Pyritohedral group ($|\mathcal{G}| = 12$) in dimension 6.

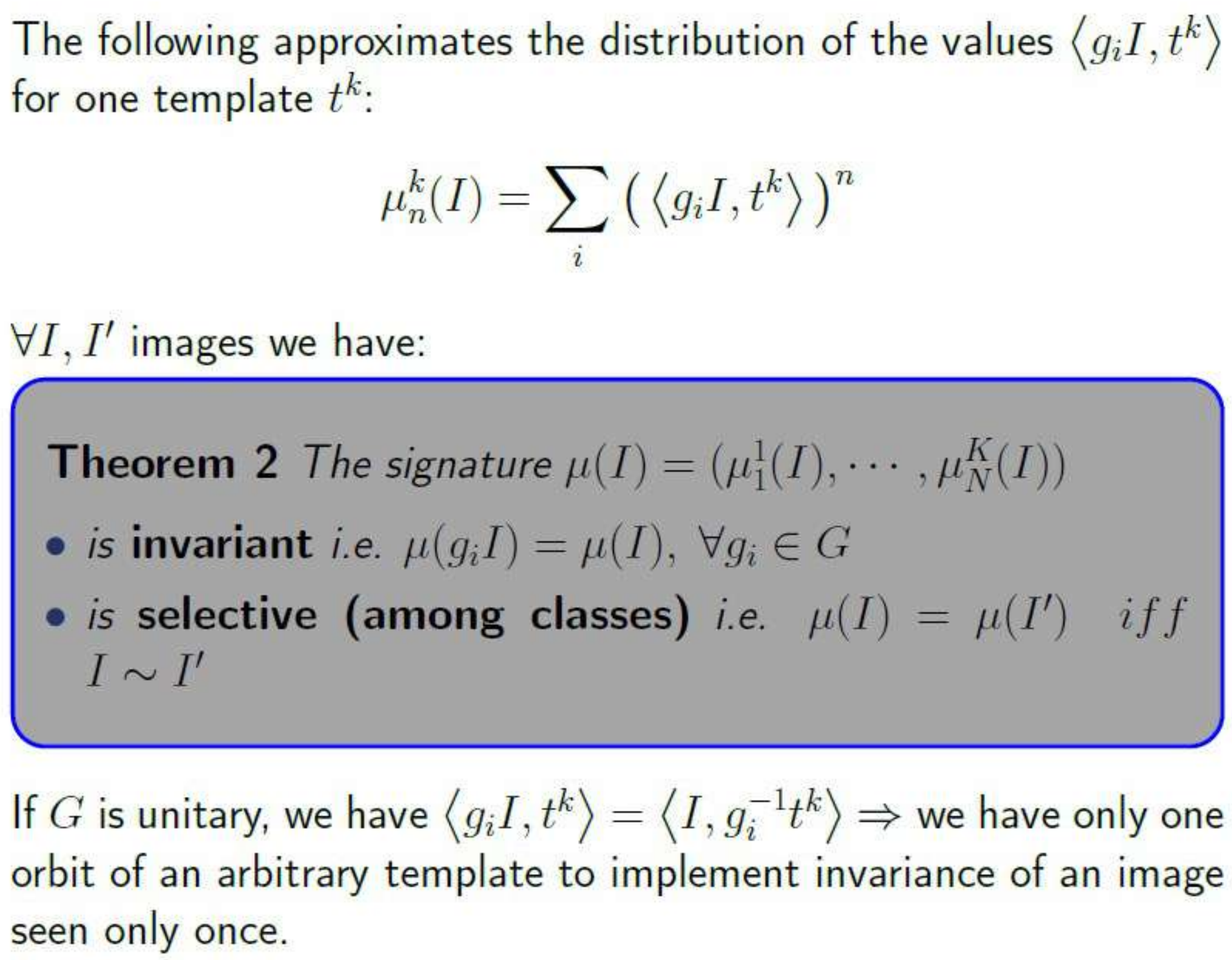
X: Randomly permuted orbits dataset

- We calculate the intra and infra orbit elements distance with a ground truth orbit vs learned orbit. If the learned weights are an orbit of $\mathcal{G}$ then:
  1. $\|\Phi(x) - \Phi(x')\| = 0 \;\Rightarrow\; O_x = O_{x'}$
  2. $\|\Phi(x) - \Phi(x')\| \neq 0 \;\Rightarrow\; O_x \neq O_{x'}$

  To test invariance and selectivity we plot the distances in a confusion matrix and distance distributions.

Example1: illumination invariance (project?)

Recap



The following approximates the distribution of the values $\langle g_i I, t^k \rangle$ for one template t^k:

$$\mu_n^k(I) = \sum_i \left(\langle g_i I, t^k \rangle \right)^n$$

$\forall I, I'$ images we have:

Theorem 2 *The signature* $\mu(I) = (\mu_1^1(I), \cdots, \mu_N^K(I))$

- *is* **invariant** *i.e.* $\mu(g_i I) = \mu(I),\ \forall g_i \in G$
- *is* **selective (among classes)** *i.e.* $\mu(I) = \mu(I') \quad iff \quad I \sim I'$

If G is unitary, we have $\langle g_i I, t^k \rangle = \langle I, g_i^{-1} t^k \rangle \Rightarrow$ we have only one orbit of an arbitrary template to implement invariance of an image seen only once.

Framework for illumination transformations

$*$ **Contrast Functions**: continuous, monotonic, and positive functions acting on an image.

We assume the contrast function $e : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^+$ has the following properties:

1. $e(x, 0) = 1$
2. $e(x, \zeta)e(x, \zeta') = e(x, \zeta \circ \zeta'), \;\; \zeta, \zeta' \in \mathbb{R}_+$

where $\circ$ is a group composition. The transformation of an image I is:

$$I_\zeta(x) \equiv I(x, \zeta) = e(x, \zeta) I(x, 0).$$

and similarly for template t. We have the following signature for an image under change in illumination:

$$\mu_n^k(I) = \int_0^\infty \left(\langle I_0, t_\zeta^k \rangle \right)^n d\zeta$$

Protocol

$*$ **Invariance**: We calculated the euclidean distance between images and their transformations. For invariance we tested:

$$||\mu(I_0) - \mu(I_\zeta)||^2 \sim 0$$

where we fixed 1 template, with $\mu(I) \in \mathbb{R}^{n \times k}$, and averaged the euclidean distance over images and their transformations.

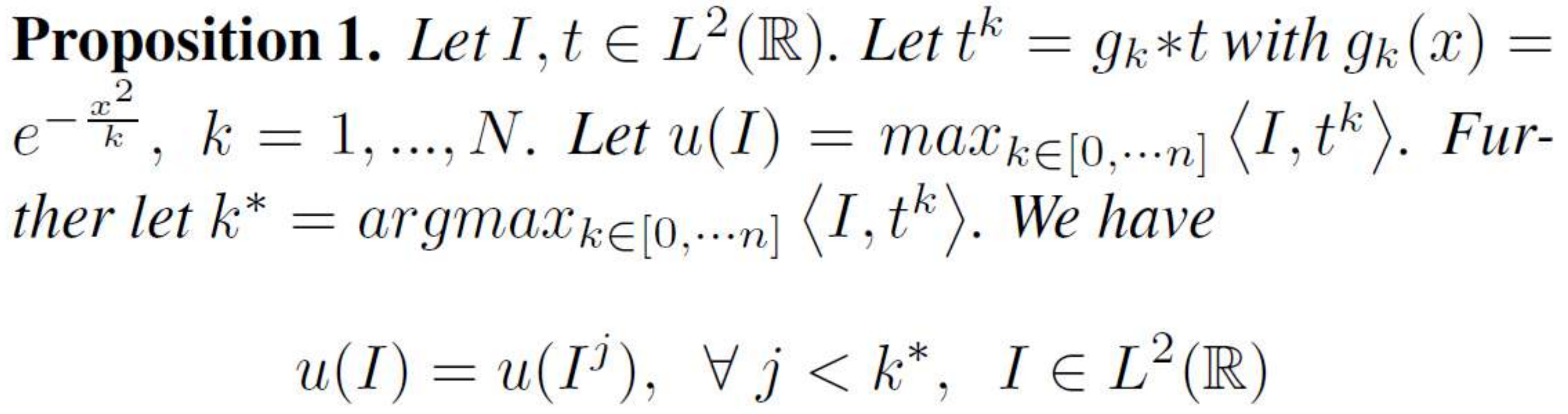
$*$ **Selectivity**: We calculated the euclidean difference between different images. For selectivity we tested:

$$||\mu(I_0^i) - \mu(I_0^j)||^2 \nsim 0, \ i \neq j$$

$*$ **Control**: We defined a "fake orbit" as a set of randomly selected images from the datasets.

For each of the datasets, we let I_0 be a face under one illumination condition and I_ζ be all other illumination conditions of the face. Our set of templates $\{t_\zeta^k\}$, contained faces under all illumination conditions with $t^k \neq I$.

Blur invariance? Others?



Proposition 1. *Let* $I, t \in L^2(\mathbb{R})$. *Let* $t^k = g_k * t$ *with* $g_k(x) = e^{-\frac{x^2}{k}}$, $k = 1, ..., N$. *Let* $u(I) = max_{k\in[0,\cdots n]} \langle I, t^k \rangle$. *Further let* $k^* = argmax_{k\in[0,\cdots n]} \langle I, t^k \rangle$. *We have*

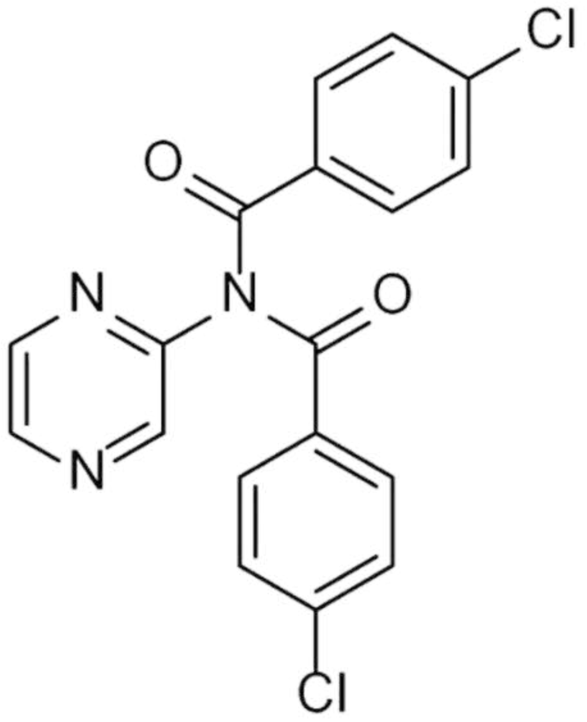
$$u(I) = u(I^j), \;\; \forall \, j < k^*, \;\; I \in L^2(\mathbb{R})$$

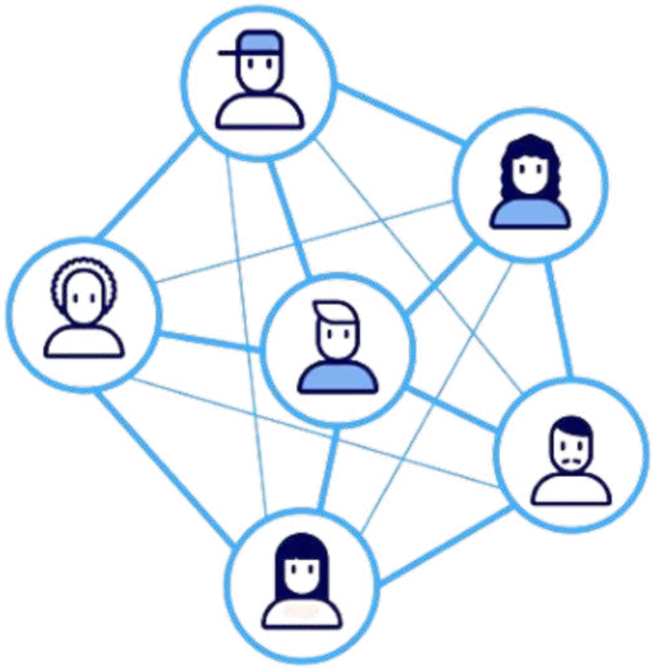
Can you prove it?

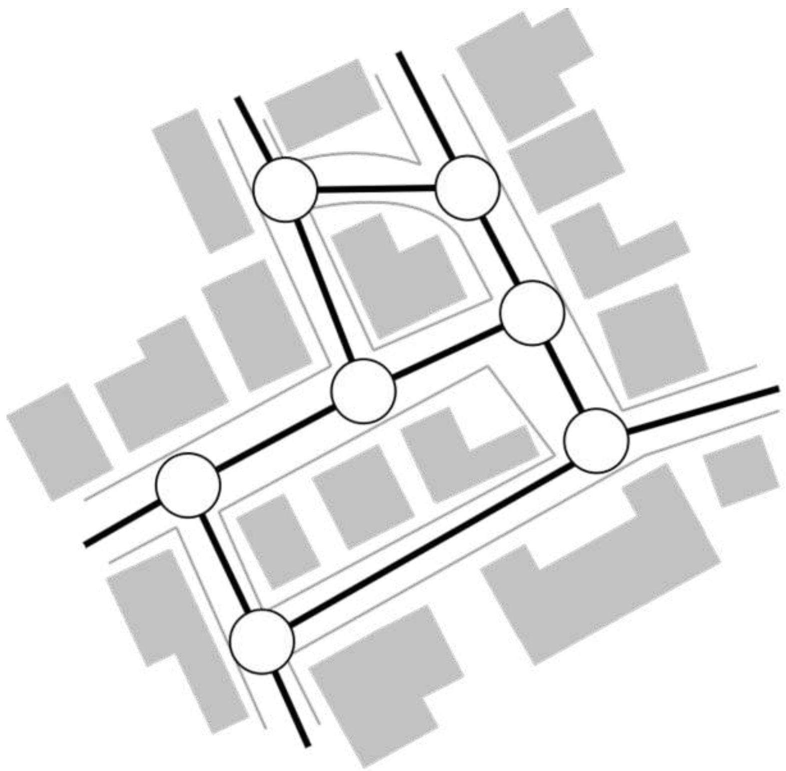
Can you think (and try) other types of invariances that will help to generate architectures better reflecting the priors on data?

Short intro to graph neural networks

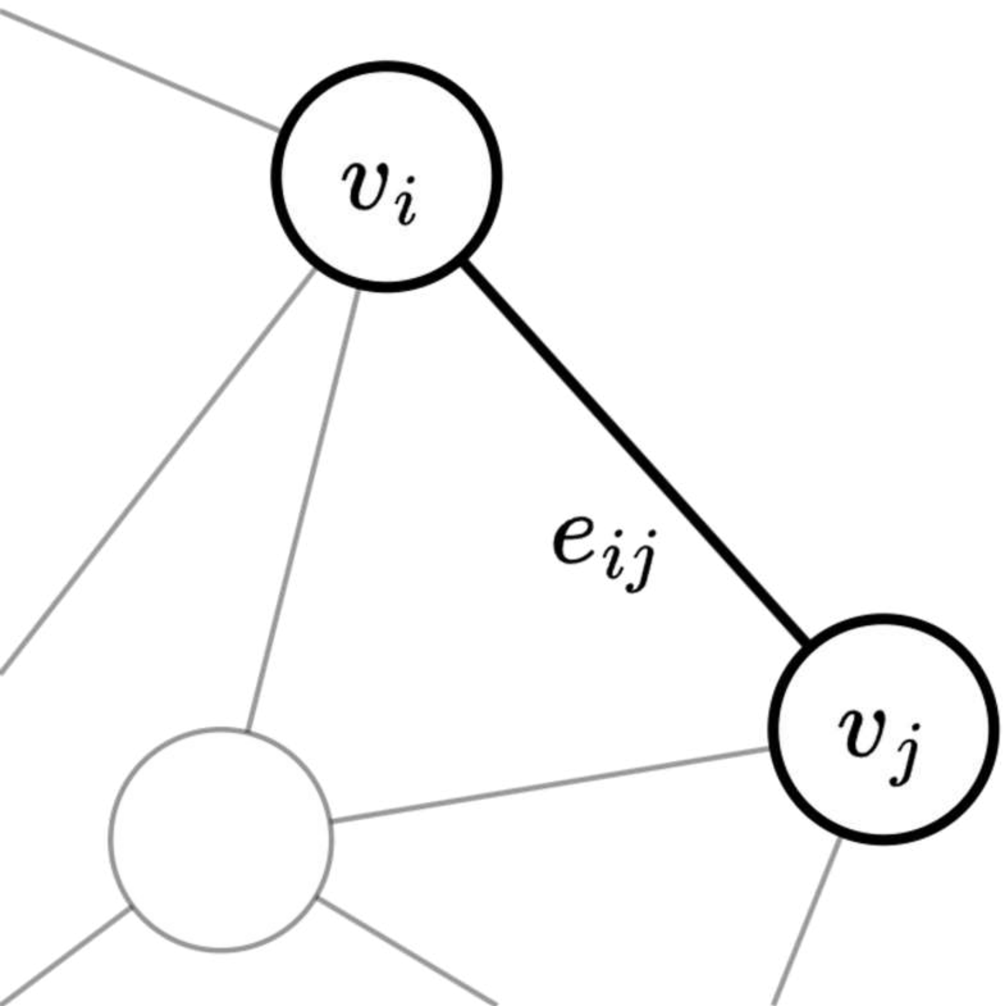
Graph structures in the real world





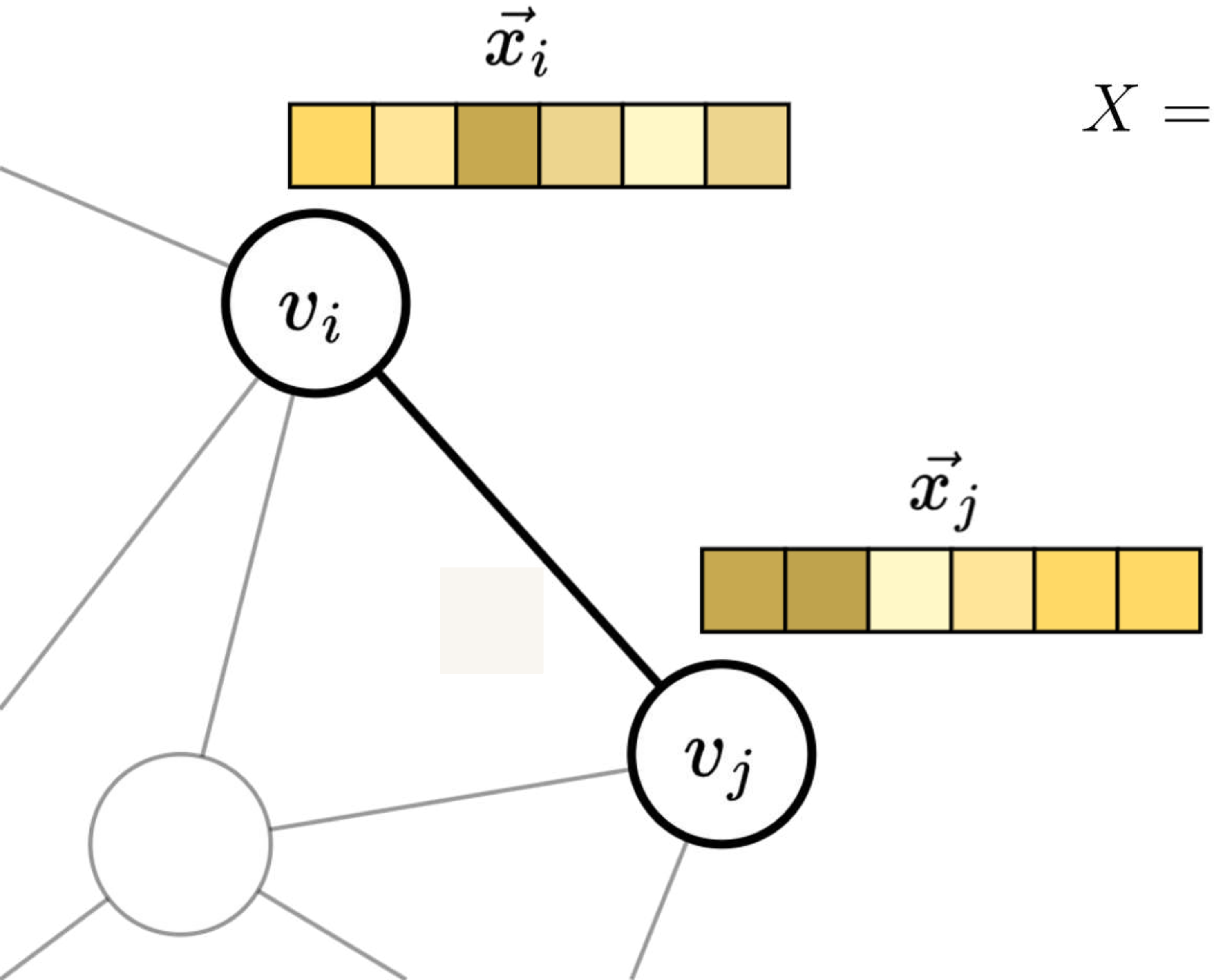


$$\mathcal{G} = (V, E)$$



Adjacency matrix of $\mathcal{G}$

$$A_{ij} = \begin{cases} 0 & \text{if } (i,j) \notin E \\ 1 & \text{if } (i,j) \in E \end{cases}$$



$$X = \begin{pmatrix} \vec{x}_1 \\ \vdots \\ \vec{x}_i \\ \vdots \\ \vec{x}_n \end{pmatrix}$$ **node features** of $\mathcal{G}$

for example:

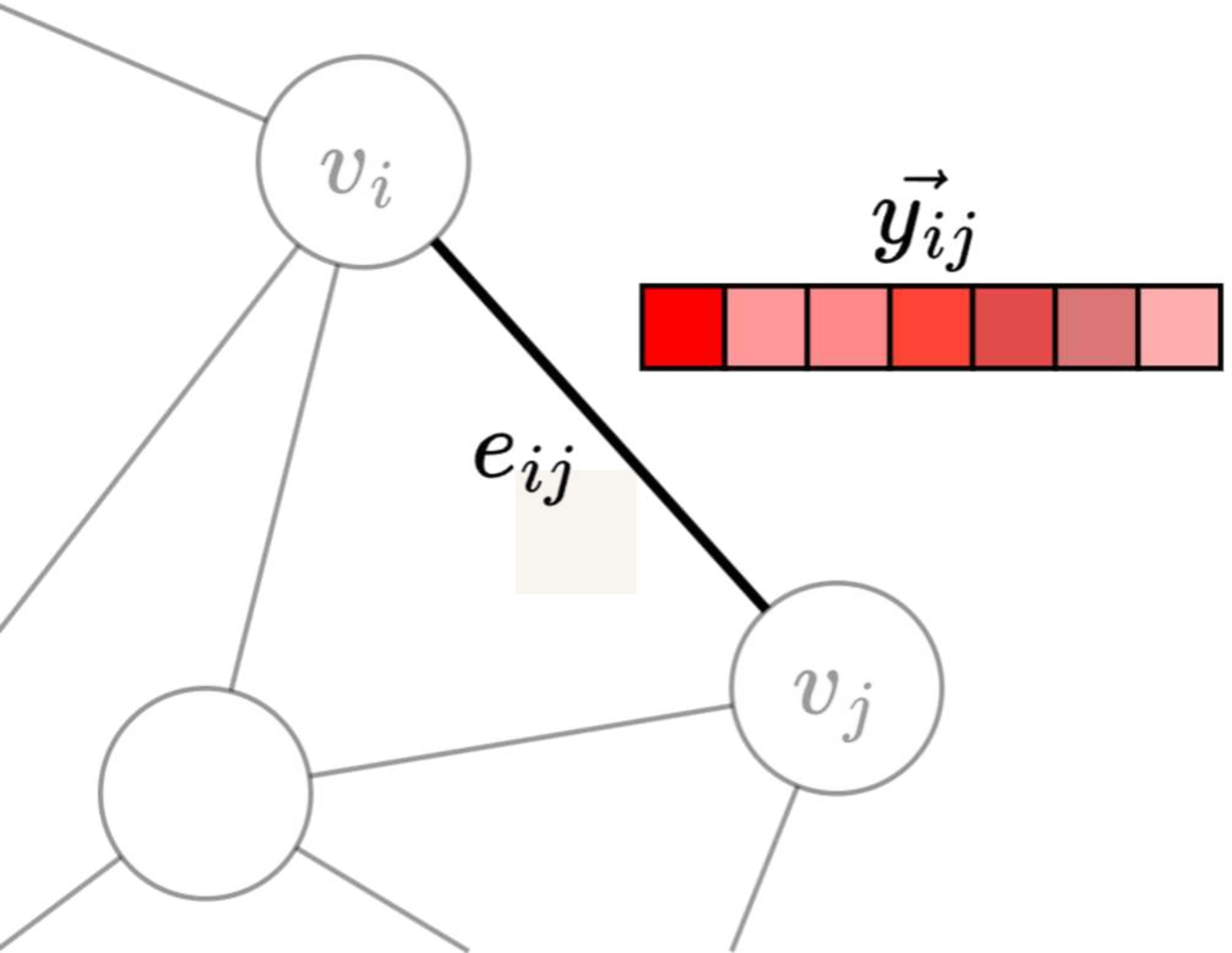
- properties of the atom
- data of the account
- number/type of crossroads

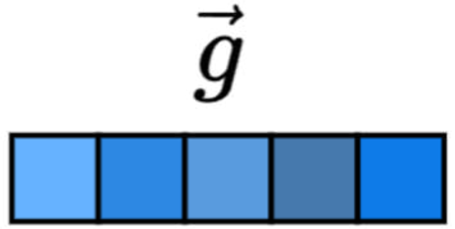
$$Y_{ij} = \vec{y_{ij}}$$

edge features of $\mathcal{G}$

for example:

- type of bond
- date of friend request, relationship
- average traffic, number of car parks





global features of $\mathcal{G}$

for example:

- known properties
- year
- day of the week

Tasks on graphs

Global tasks

tasks on the **whole graph**.
Both classification and regression.

Examples:

- predicting properties of a molecule
- predicting the hobby shared by all the people
- predicting the likelihood of a car accident

Tasks on graphs

Node tasks

tasks at the level of the **single node**.
Both classification and regression.

Examples:

- predicting if an atom is part of an aromatic ring
- predicting political stance of each users on a topic
- classifying intersections based on predicted traffic

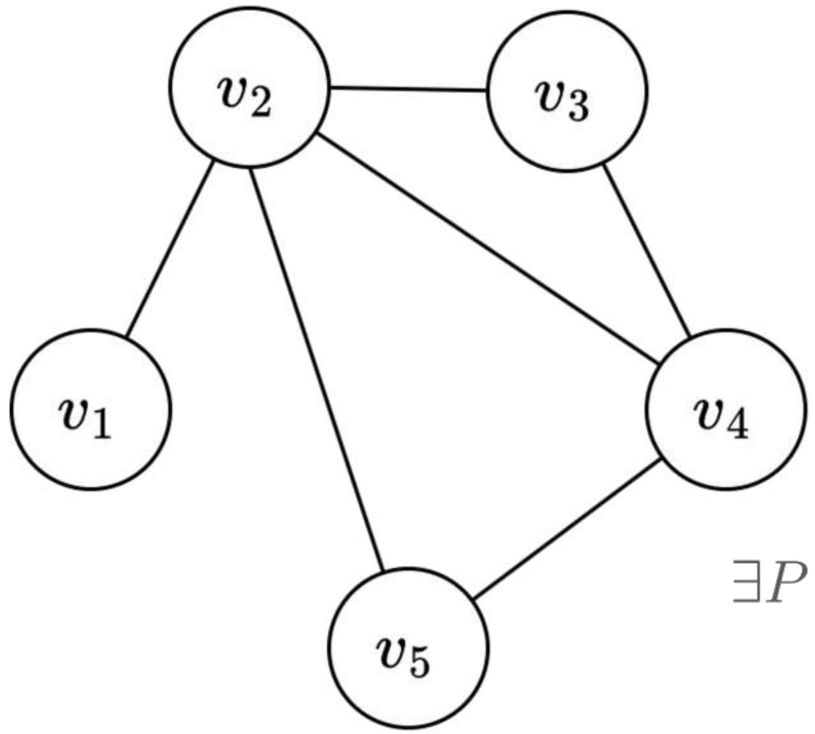
Tasks on graphs

Edge tasks

tasks at the level of the **single edge.**
Both classification and regression.

Examples:

- predicting which bond is easier to break
- predicting how likely two users will become friends
- predicting how busy a road will be in an hour

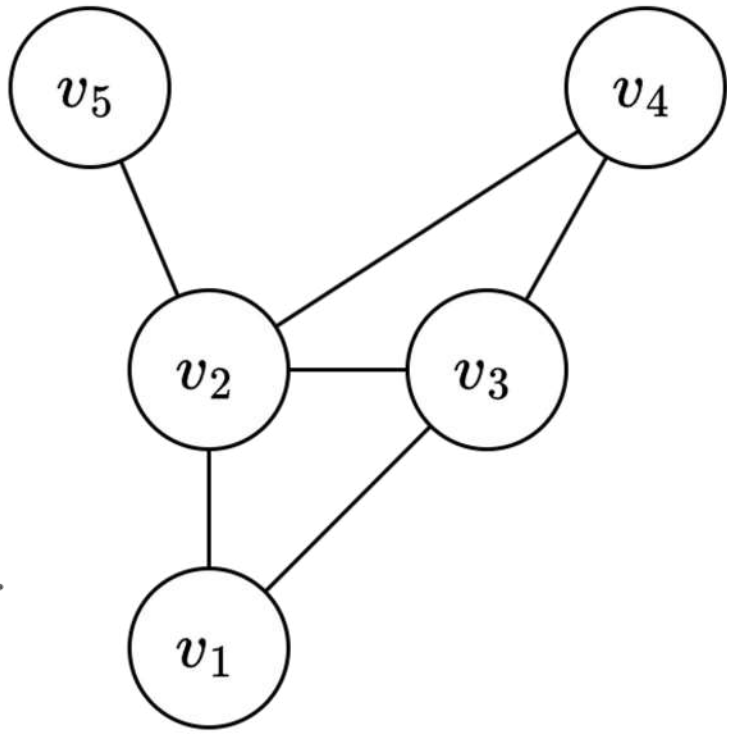


$$\cong$$

$$\Longleftrightarrow$$

$\exists P$ permutation matrix s.t.

$$PA_1P^{-1} = A_2$$



$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$\forall P$ permutation matrix

invariance in the context of global tasks

$$F_G(A, X, Y, \vec{g}) = F_G(PAP^{-1}, PX, PYP^{-1}, \vec{g}) \quad \in \mathbb{R} \text{ or } \in \mathbb{R}^d$$

equivariance in the context of node or edge tasks

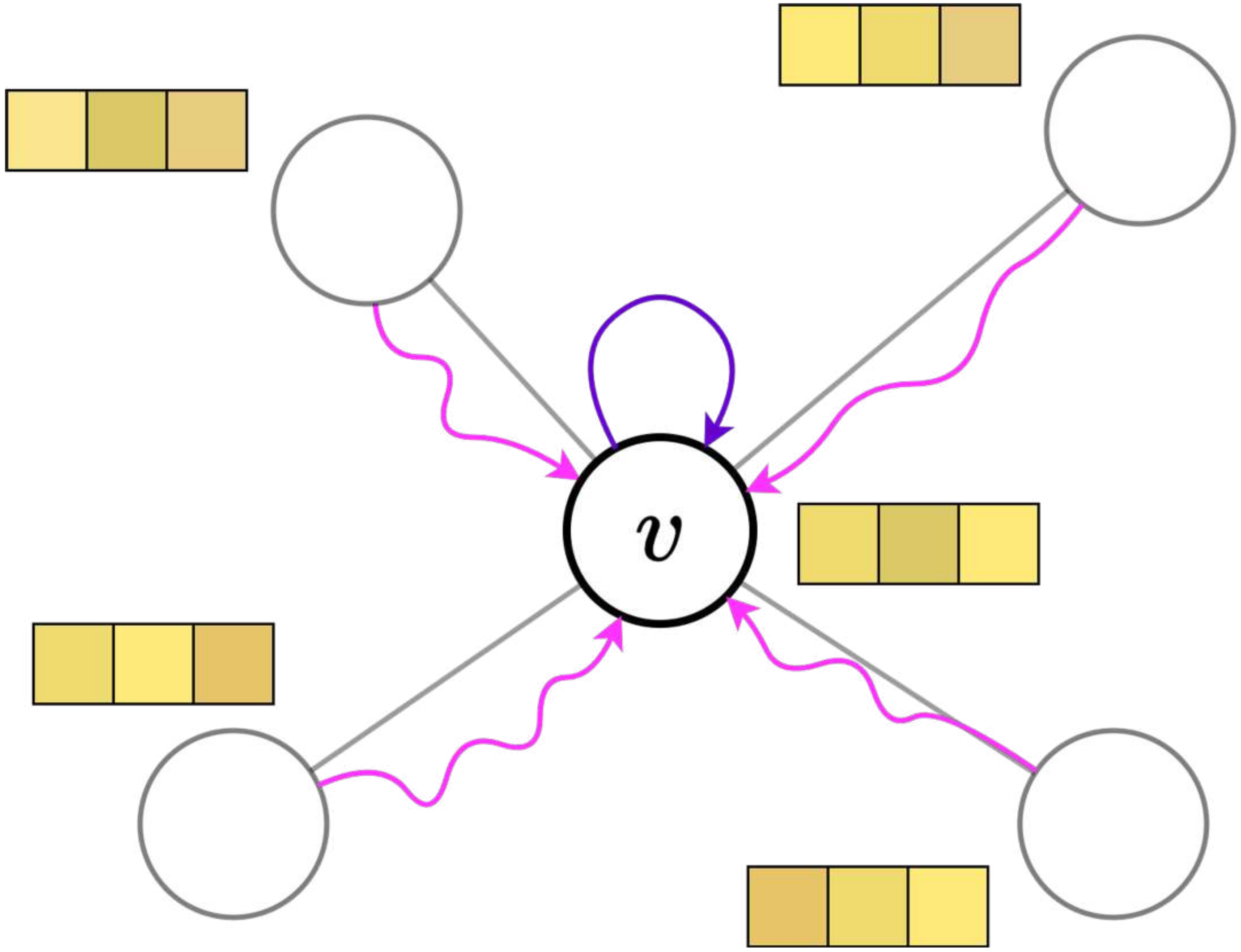
$$\underset{\in \mathbb{R}^n \text{ or } \in \mathbb{R}^{n \cdot d}}{F_V(A, X, Y, \vec{g})} = P^{-1} F_V(PAP^{-1}, PX, PYP^{-1}, \vec{g})$$

$$\underset{\in \mathcal{M}_{\mathbb{R}}(n,n) \text{ or } \in \mathcal{M}_{\mathbb{R}^d}(n,n)}{F_E(A, X, Y, \vec{g})} = P^{-1} F_E(PAP^{-1}, PX, PYP^{-1}, \vec{g}) P$$

in the context of node/global tasks

input: A , X

- **Aggregating** over $\mathcal{N}_v$
- **Updating** $\vec{x_v}$



Convolutional approach

$$\vec{x_v} = \phi\left(\vec{x_v}, \bigoplus_{u \in \mathcal{N}_v} c_{vu}\psi(\vec{x_u})\right)$$

Aggregation

must be a **permutation invariant** operator

- mean
- max
- sum

$$\vec{x_v} = \phi\left(\vec{x_v}, \bigoplus_{u\in\mathcal{N}_v} c_{vu}\psi(\vec{x_u})\right)$$

Update

$$\phi(\vec{x},\vec{z}) = \sigma(W\vec{x} + U\vec{z} + \vec{b})$$

learnable

coefficient
in the case of summation
can be

$$c_{uv} = \frac{1}{\sqrt{deg(u)\cdot deg(v)}}$$

transformation of
the features

$$\psi(\vec{x}) = W\vec{x} + \vec{b}$$

learnable

Attentional approach

$$\vec{x_v} = \phi\left(\vec{x_v}, \bigoplus_{u \in \mathcal{N}_v} a(\vec{x_v}, \vec{x_u})\psi(\vec{x_u})\right)$$

$\uparrow$

coefficients are **learned** via NN and softmax normalized

$a(\vec{x_v}, \vec{x_u}) \in \mathbb{R}$

Message-passing approach

$$\vec{x_v} = \phi\left(\vec{x_v}, \bigoplus_{u \in \mathcal{N}_v} \psi(\vec{x_v}, \vec{x_u})\right)$$

$\uparrow$

ψ here is a **learnable message function**

$\psi(\vec{x_v}, \vec{x_u}) \in \mathbb{R}^m$

How would you implement an invariant Graph model?