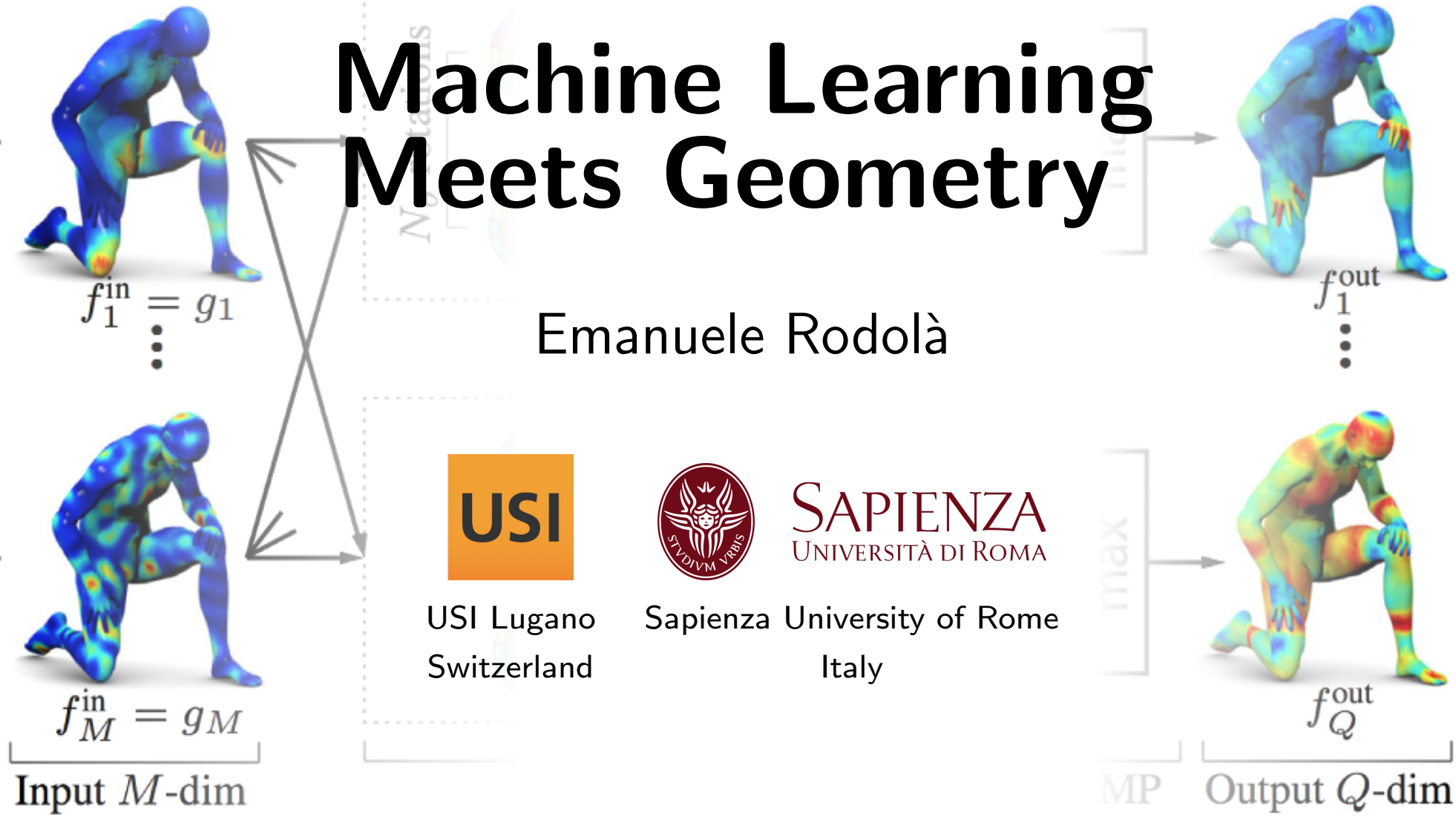


Machine Learning Meets Geometry

Emanuele Rodolà

USI Lugano
Switzerland

Sapienza University of Rome
Italy

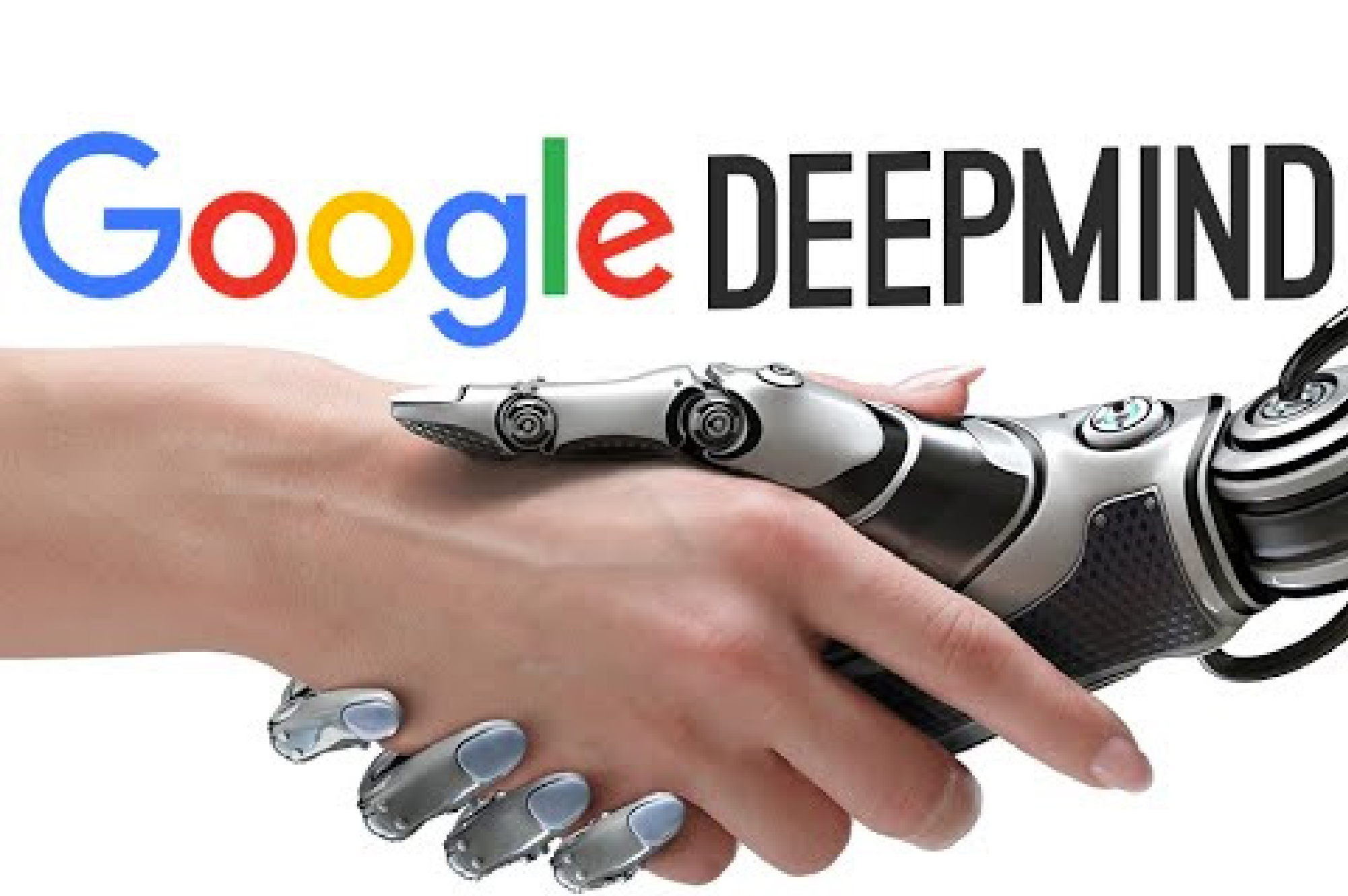
SGP Grad School, London, 02 July 2017

Science
$10
17 JULY 2015
sciencemag.org
AAAS

SPECIAL ISSUE

ARTIFICIAL INTELLIGENCE

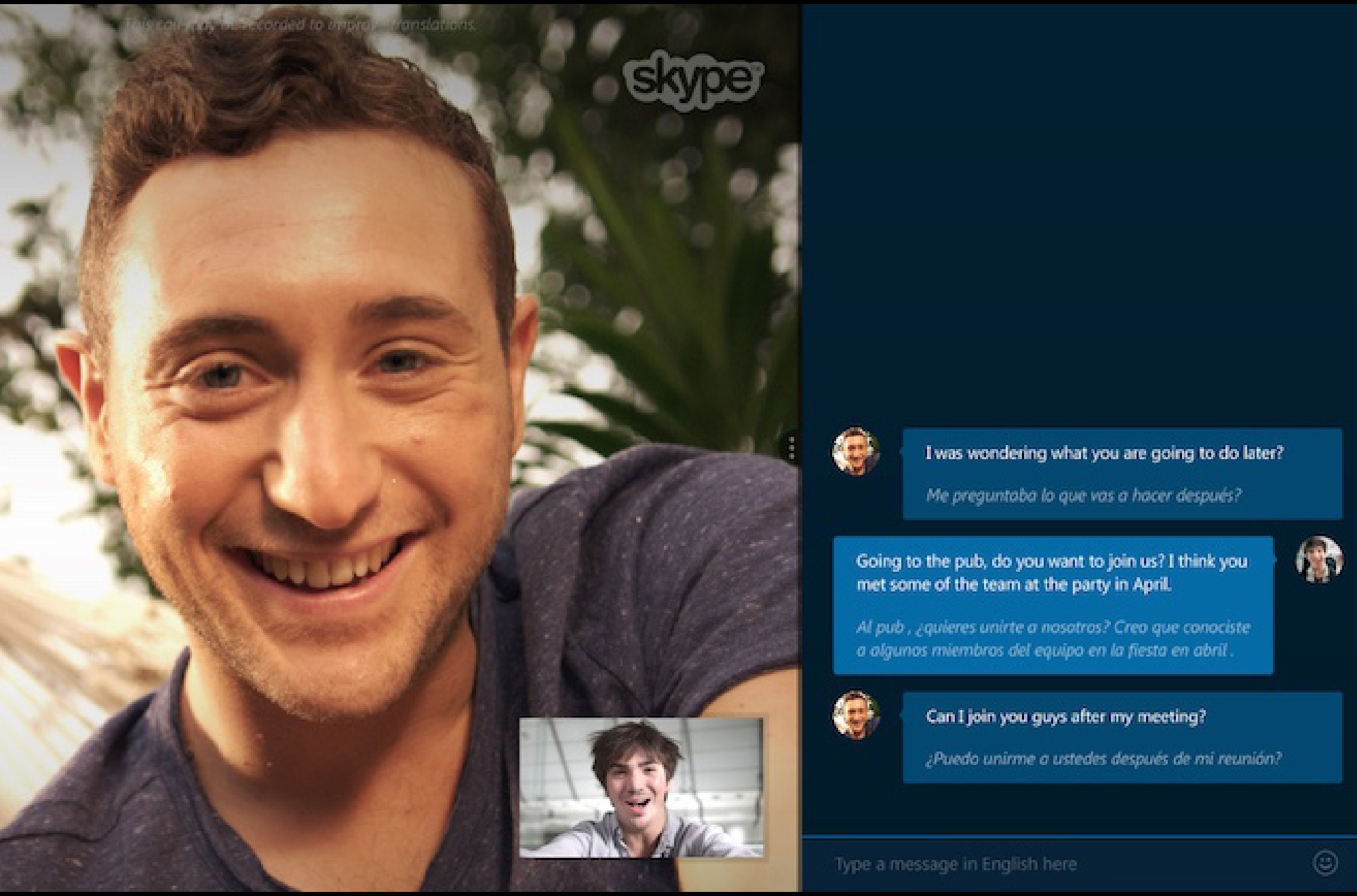
Facebook opens AI Research in 2014



Google acquires DeepMind in 2014

Google DeepMind AlphaGo beats human player

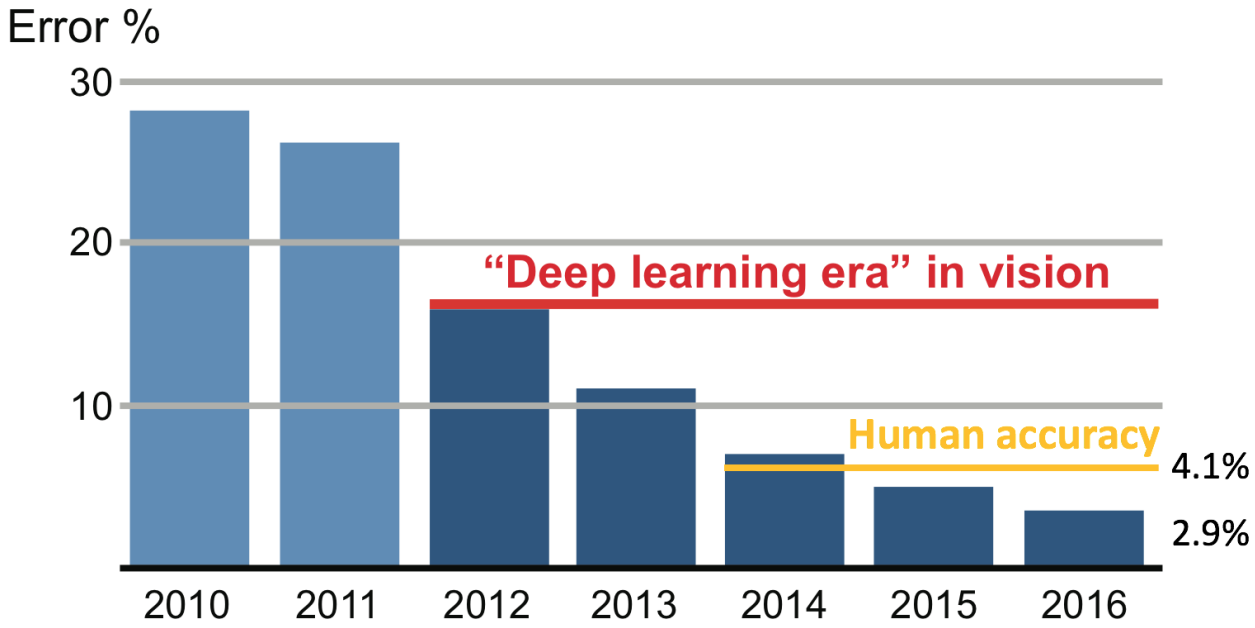
Tesla autonomous driving car



Microsoft Skype real-time voice translation 2016

DeepMind
AlphaGo
RNN / LSTM
Schmidhuber
First NIPS
IMAGENET
Microsoft
Speech recognition
Perceptron
Rosenblatt
Backprop
Werbos
CNN
LeCun
Autoencoder
LeCun, Hinton
facebook
AI Research
1958
1959
1982
1987
1995
1997
1998
1999
2006
2010
2012
2014
2015
2016
Visual cortex
Hubel&Wiesel
Neurocognitron
Fukushima
SVM
Vapnik
First GPU
amazon web services
ImageNet breakthrough
Krizhevsky
TensorFlow
OpenAI
TESLA
Autonomous cars
AI BIRTH
DARK AGES
RENAISSANCE

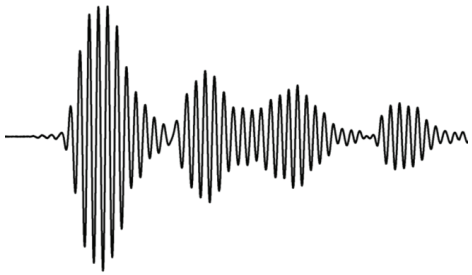
Breakthrough in image recognition



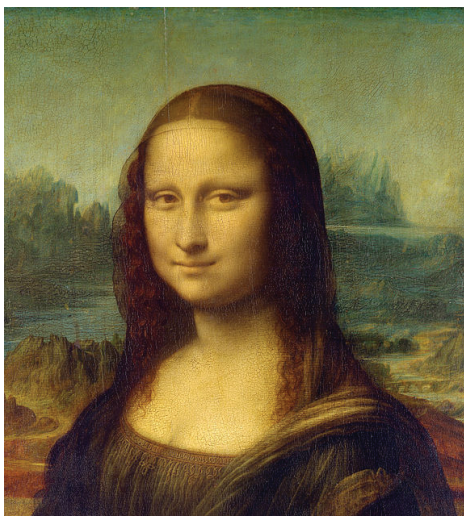
ImageNet ILSVRC Challenge

Doubt thou the stars are fire,
Doubt that the sun doth move,
Doubt truth to be a liar,
But never doubt I love...

Text



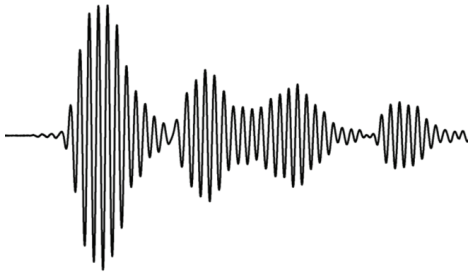
Audio signals



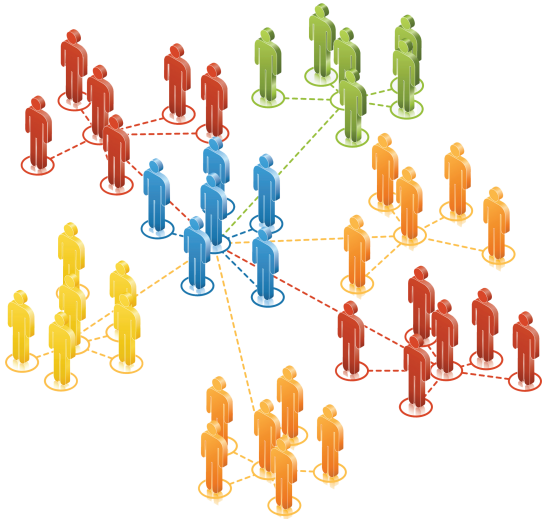
Images

Doubt thou the stars are fire,
Doubt that the sun doth move,
Doubt truth to be a liar,
But never doubt I love...

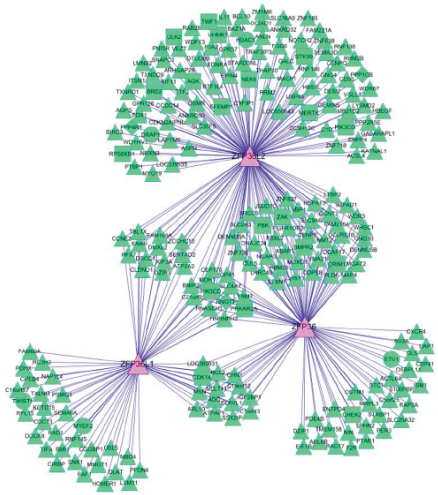
Text



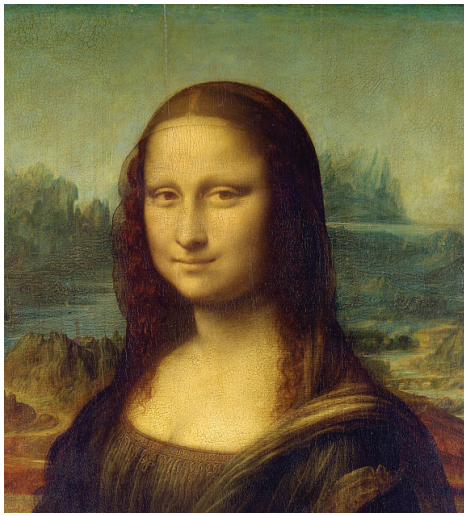
Audio signals



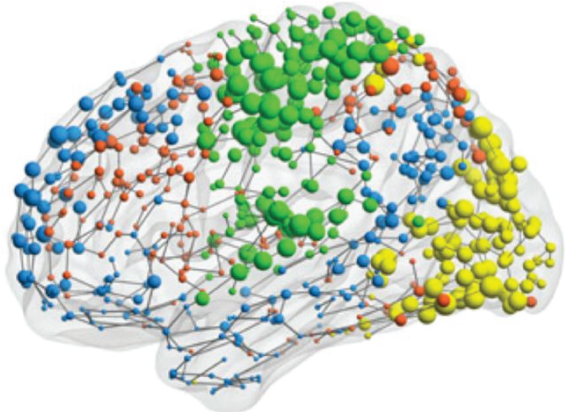
Social networks



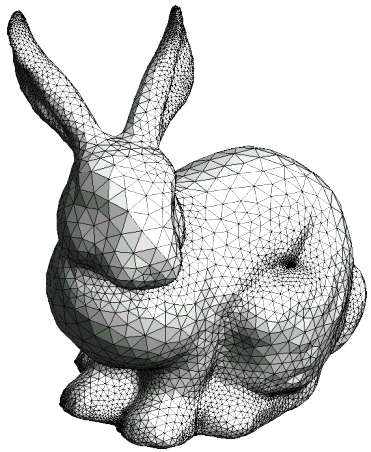
Regulatory networks



Images



Functional networks

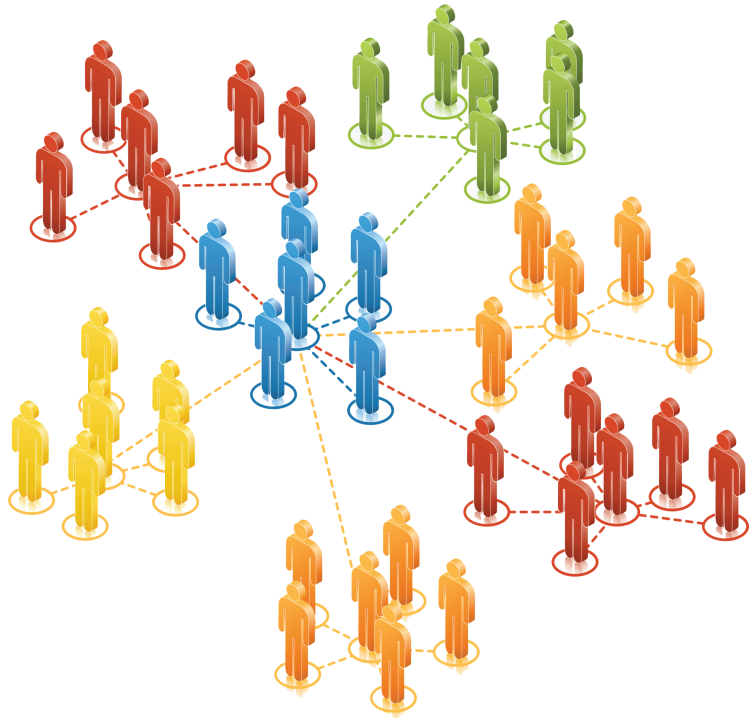


3D shapes

How to design deep nets on non-Euclidean domains and what to do with them?

Prototypical non-Euclidean objects

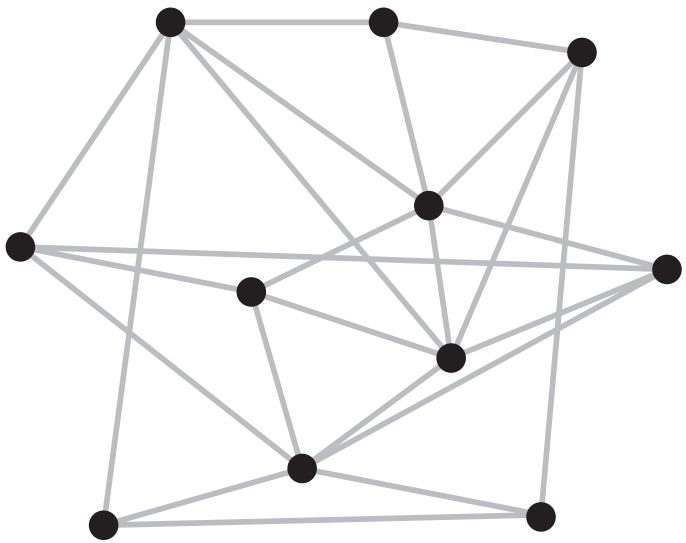
Manifolds



Graphs

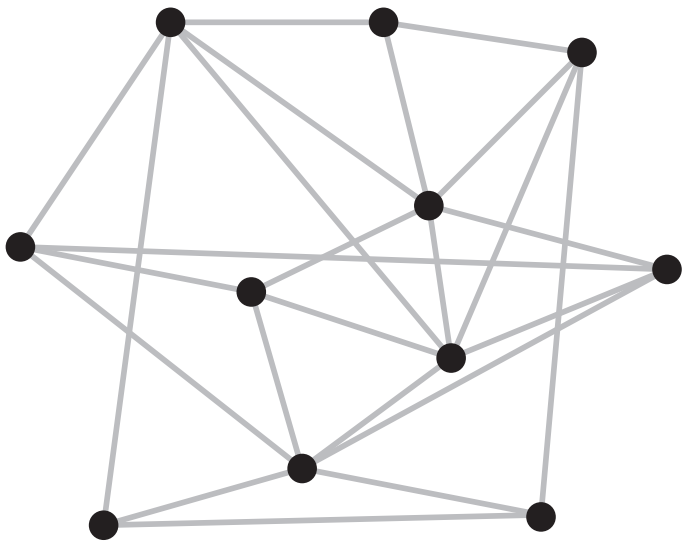
Domain structure vs Data on a domain

Domain structure vs Data on a domain

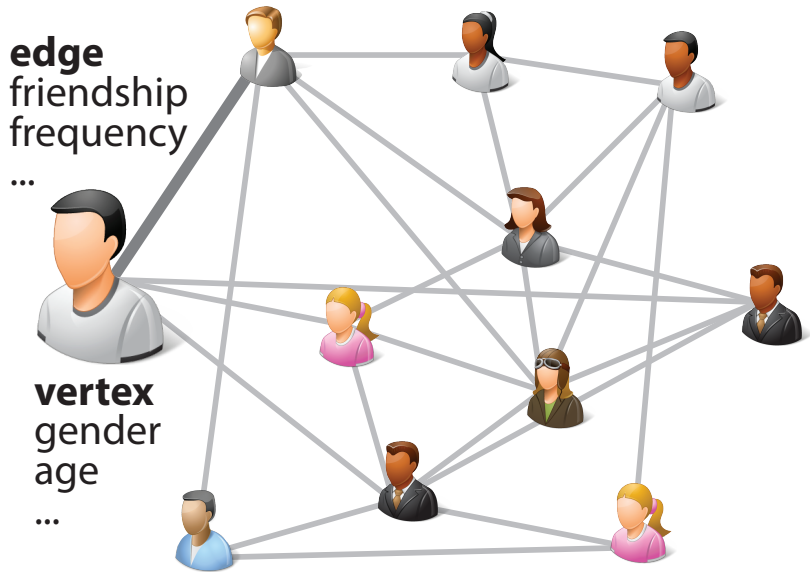


Domain structure

Domain structure vs Data on a domain

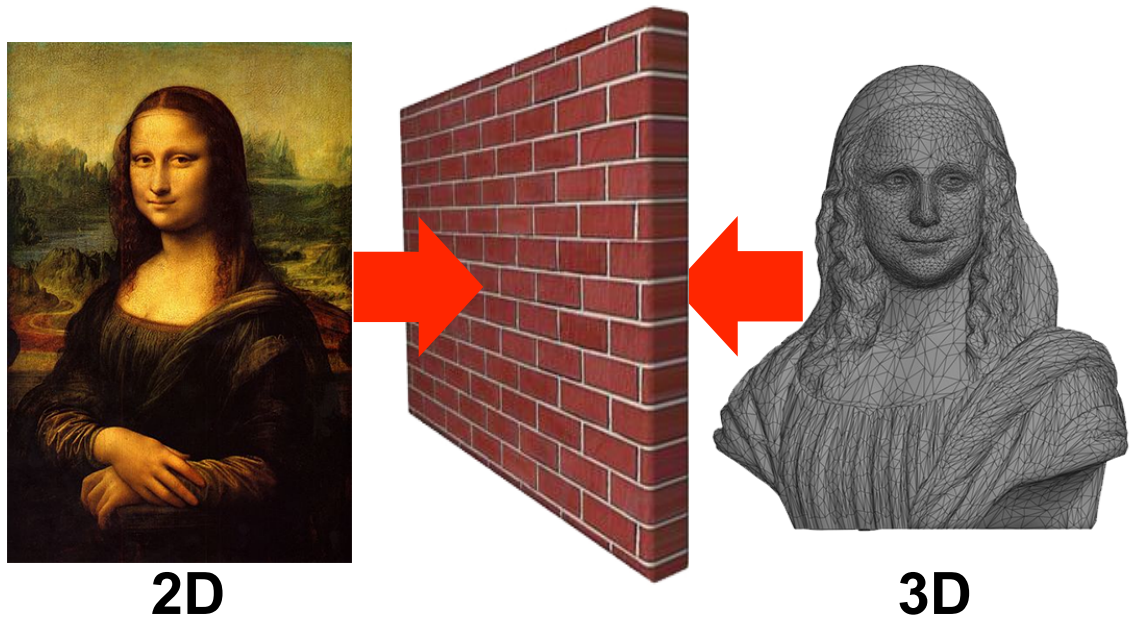


Domain structure



Data on a domain

Domain structure vs Data on a domain

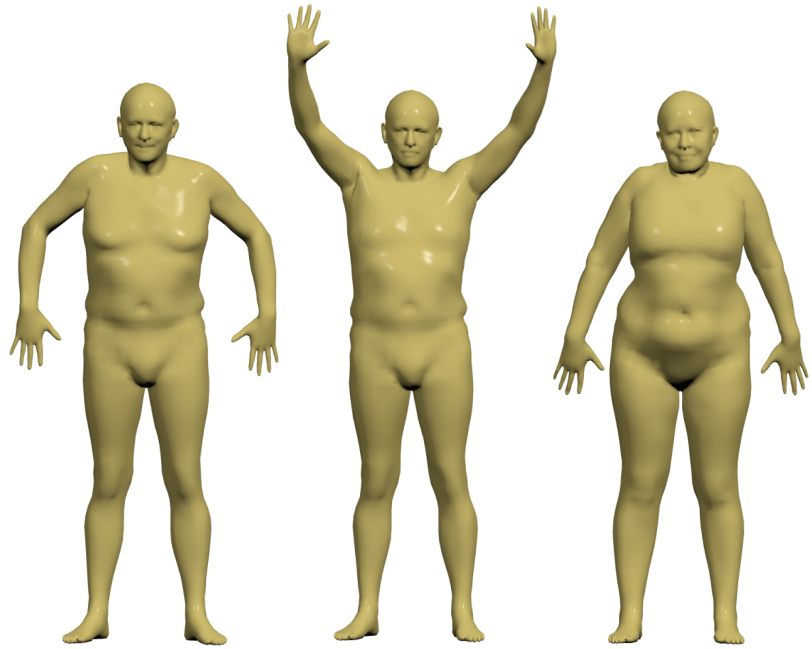


Fixed vs different domain

Social network
(fixed graph)

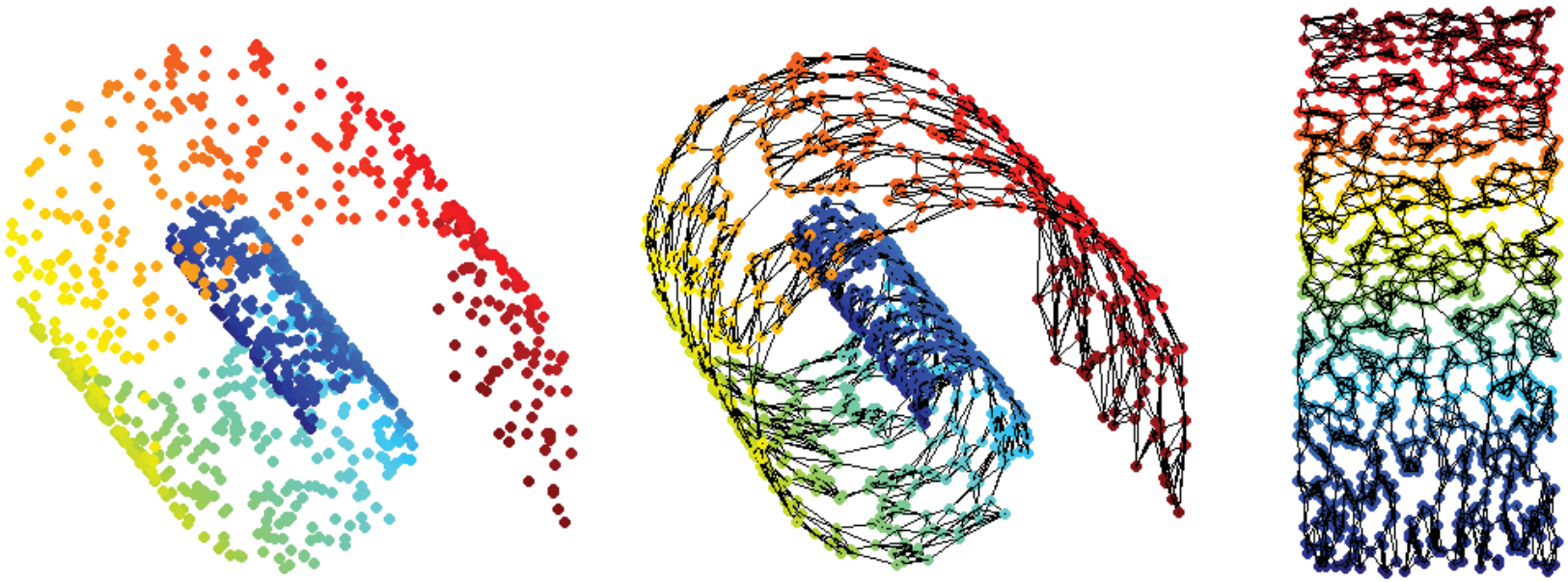
Fixed vs different domain

Social network
(fixed graph)



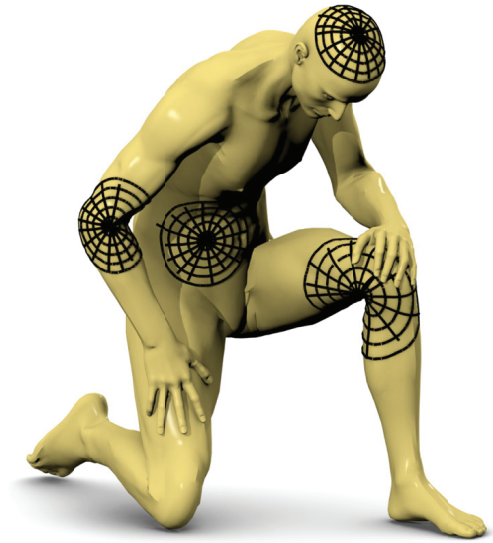
3D shapes
(different manifolds)

Geometric learning $\neq$ Manifold learning

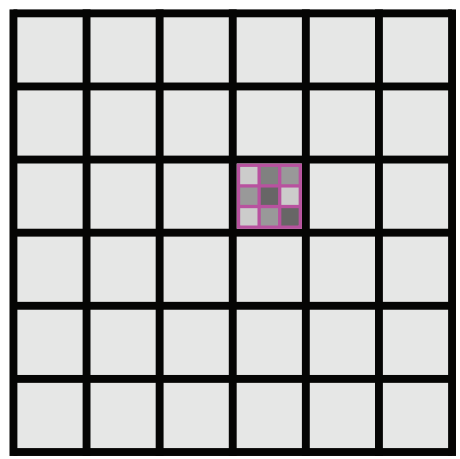


Different formulations of non-Euclidean CNNs

Spectral domain



Spatial domain



Embedding domain

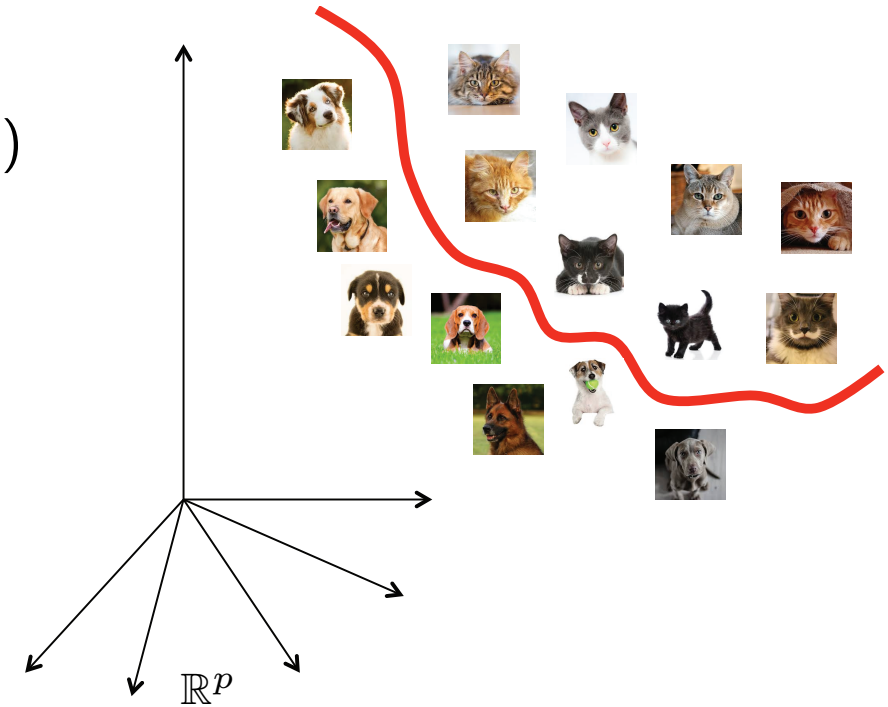
Background

Supervised learning

- Data vectors $\mathbf{f} \in \mathbb{R}^p$
  (e.g. for 512×512 images $p \approx 10^6$)
- Unknown classification functional
  $y : \mathbb{R}^p \to \{1, \ldots, L\}$ in L classes
- Training set

$$S = \{(\mathbf{f}_i \in \mathbb{R}^p,\, y_i = y(\mathbf{f}_i))\}_{i=1}^{T}$$

- Parametric model $y_{\boldsymbol{\Theta}}$ of y

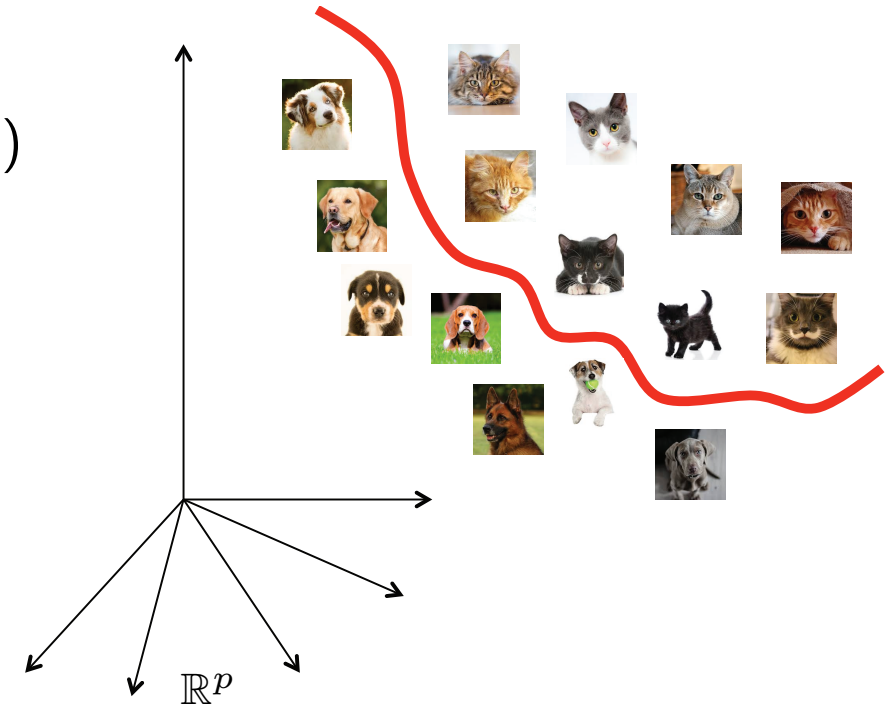


Supervised learning

- Data vectors $\mathbf{f} \in \mathbb{R}^p$
  (e.g. for 512×512 images $p \approx 10^6$)
- Unknown classification functional
  $y : \mathbb{R}^p \to \{1, \ldots, L\}$ in L classes
- Training set

$$S = \{(\mathbf{f}_i \in \mathbb{R}^p,\, y_i = y(\mathbf{f}_i))\}_{i=1}^{T}$$

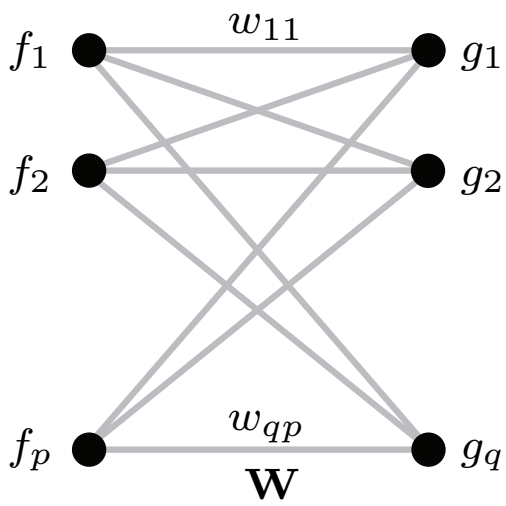
- Parametric model $y_{\boldsymbol{\Theta}}$ of y



Supervised learning: find optimal model parameters by minimizing the loss ℓ on the training set

$$\boldsymbol{\Theta}^* = \operatorname*{argmin}_{\boldsymbol{\Theta}} \sum_{i=1}^{T} \ell(y_{\boldsymbol{\Theta}}(\mathbf{f}_i), y_i)$$

Neural network (NN)



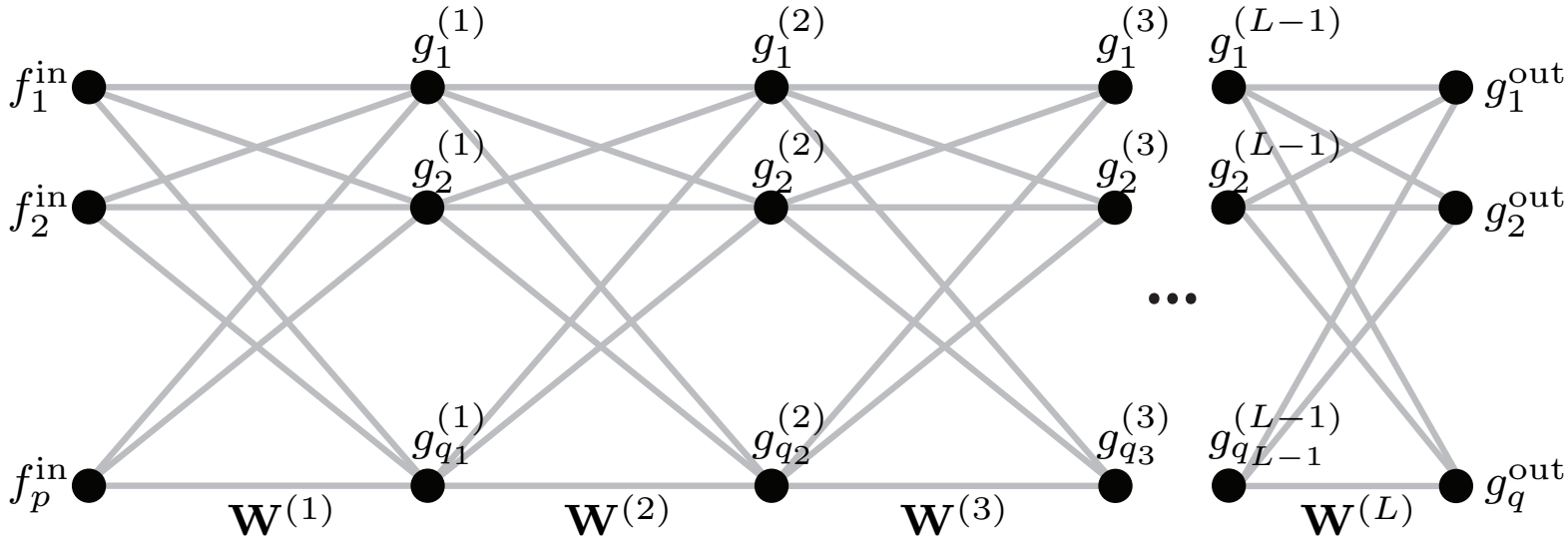
Single linear layer

Linear layer $$g_l = \xi\left(\sum_{l'=1}^{p} f_{l'} w_{l,l'}\right) \qquad \begin{matrix} l = 1, \ldots, q \\ l' = 1, \ldots, p \end{matrix}$$

Activation, e.g. $\xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Parameters layer weights $\mathbf{W}$ (including bias)

Neural network (NN)



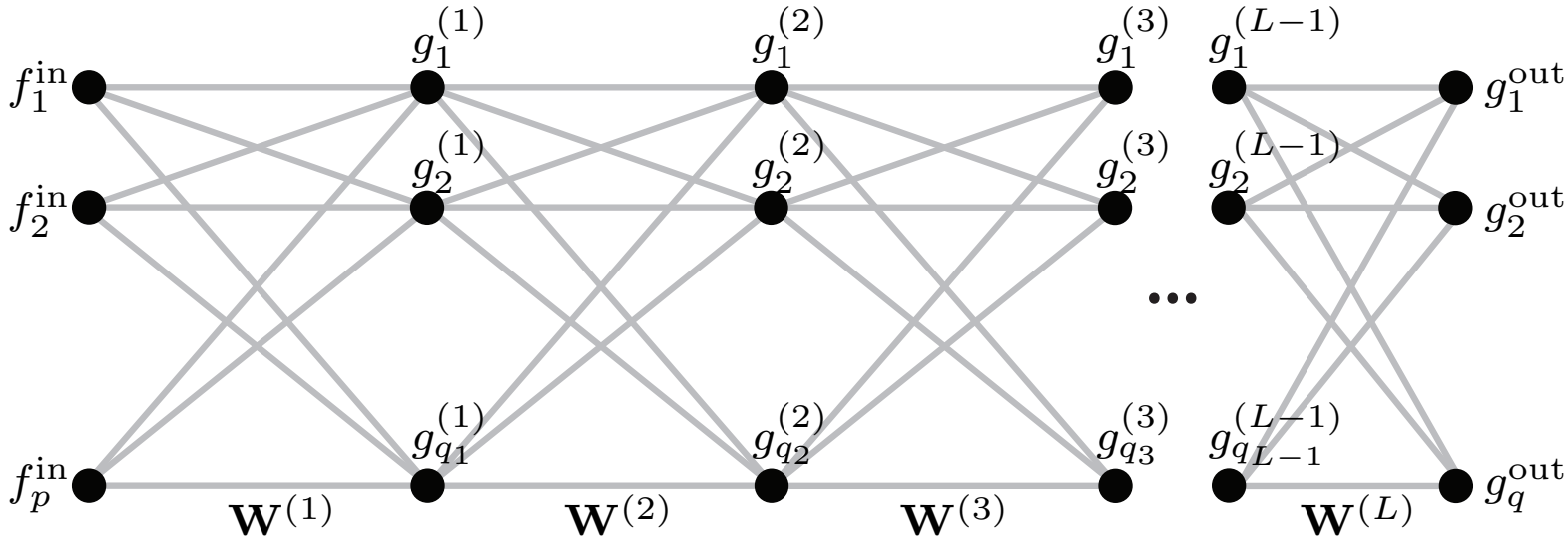
Deep neural network consisting of L layers

Linear layer $$g_l^{(k)} = \xi\left(\sum_{l'=1}^{q_{k-1}} g_{l'}^{(k-1)} w_{l,l'}^{(k)}\right) \qquad \begin{array}{l} l = 1, \ldots, q_k \\ l' = 1, \ldots, q_{k-1} \end{array}$$

Activation, e.g. $\xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Parameters weights of all layers $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}$ (including biases)

Neural network (NN)



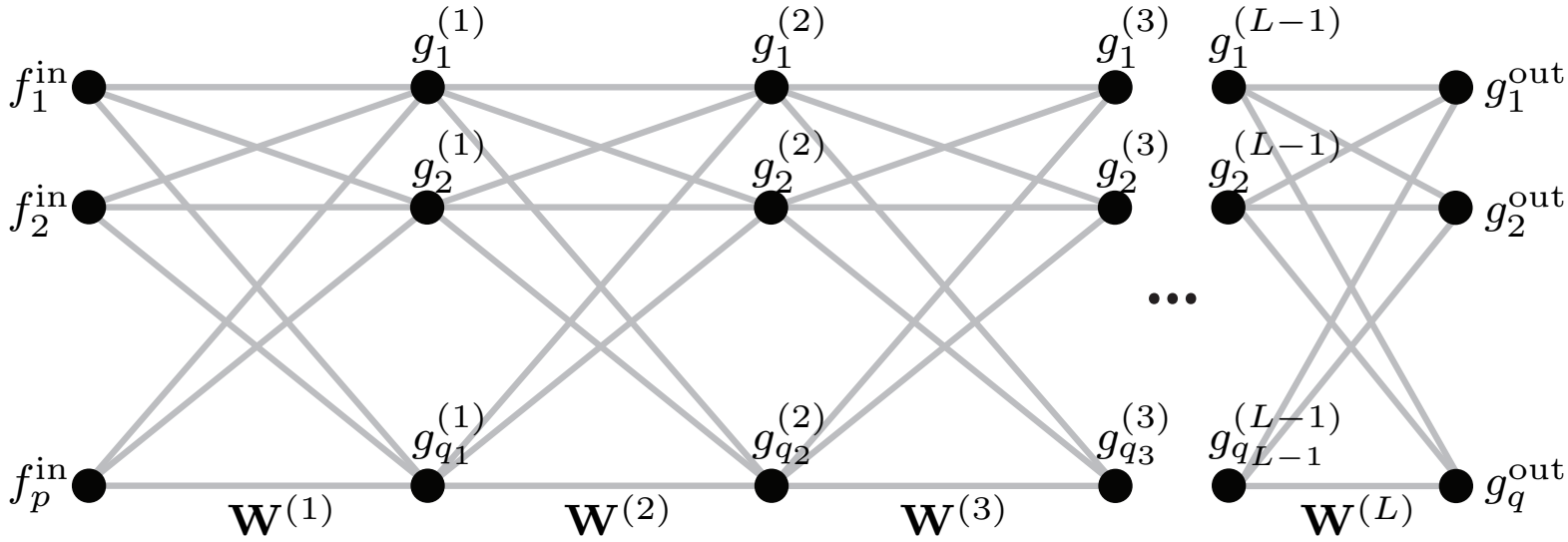
Deep neural network consisting of L layers

Linear layer $\quad \mathbf{g}^{(k)} = \xi\left(\mathbf{W}^{(k)}\mathbf{g}^{(k-1)}\right)$

Activation, e.g. $\quad \xi(x) = \max\{x, 0\}$ $\quad$ rectified linear unit (ReLU)

Parameters $\quad$ weights of all layers $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}$ (including biases)

Neural network (NN)



Deep neural network consisting of L layers

Net output $\mathbf{g}^{\text{out}} = \xi\left(\ldots \mathbf{W}^{(2)} \xi\left(\mathbf{W}^{(1)} \mathbf{f}^{\text{in}}\right)\right) = y_{(\mathbf{W}^{(1)},\ldots,\mathbf{W}^{(L)})}(\mathbf{f}^{\text{in}})$

Activation, e.g. $\xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Parameters weights of all layers $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}$ (including biases)

Neural nets as universal approximators

Universal Approximation Theorem Let ξ be a non-constant, bounded, and monotonically-increasing continuous activation function, $y : [0,1]^p \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times p}$ (including bias) s.t.

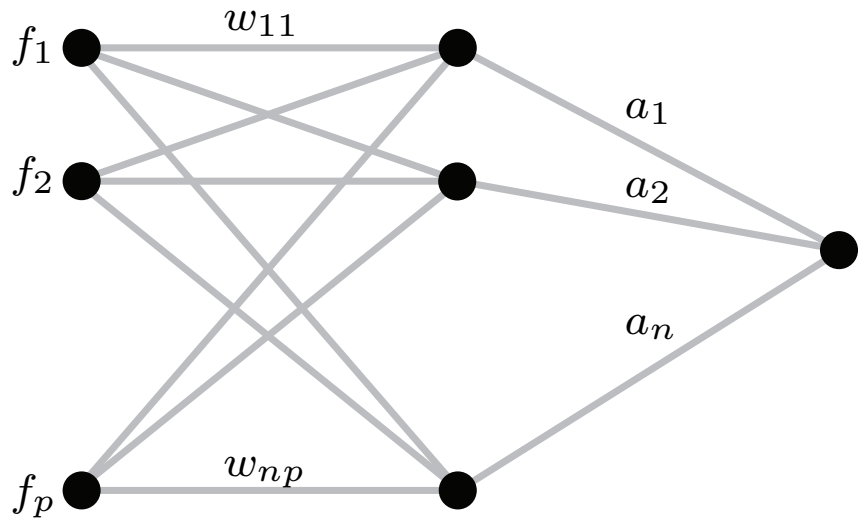
$$\left| \sum_{i=1}^{n} a_i \xi(\mathbf{w}_i^\top \mathbf{f}) - y(\mathbf{f}) \right| < \epsilon \qquad \forall \mathbf{f} \in [0,1]^p$$

Cybenko 1989; Hornik 1991

Neural nets as universal approximators

Universal Approximation Theorem Let ξ be a non-constant, bounded, and monotonically-increasing continuous activation function, $y : [0,1]^p \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times p}$ (including bias) s.t.

$$\left| \sum_{i=1}^{n} a_i \xi(\mathbf{w}_i^\top \mathbf{f}) - y(\mathbf{f}) \right| < \epsilon \qquad \forall \mathbf{f} \in [0,1]^p$$



Cybenko 1989; Hornik 1991

Neural nets as universal approximators

Universal Approximation Theorem Let ξ be a non-constant, bounded, and monotonically-increasing continuous activation function, $y : [0,1]^p \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times p}$ (including bias) s.t.

$$\left| \sum_{i=1}^{n} a_i \xi(\mathbf{w}_i^\top \mathbf{f}) - y(\mathbf{f}) \right| < \epsilon \qquad \forall \mathbf{f} \in [0,1]^p$$

☺ Any continuous function can be approximated arbitrarily well by a neural network with a single hidden layer

Cybenko 1989; Hornik 1991

Neural nets as universal approximators

Universal Approximation Theorem Let ξ be a non-constant, bounded, and monotonically-increasing continuous activation function, $y : [0,1]^p \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times p}$ (including bias) s.t.

$$\left| \sum_{i=1}^{n} a_i \xi(\mathbf{w}_i^\top \mathbf{f}) - y(\mathbf{f}) \right| < \epsilon \qquad \forall \mathbf{f} \in [0,1]^p$$

☺ Any continuous function can be approximated arbitrarily well by a neural network with a single hidden layer

☹ How to find the parameters?

Cybenko 1989; Hornik 1991

Neural nets as universal approximators

Universal Approximation Theorem Let ξ be a non-constant, bounded, and monotonically-increasing continuous activation function, $y : [0,1]^p \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times p}$ (including bias) s.t.

$$\left| \sum_{i=1}^{n} a_i \xi(\mathbf{w}_i^\top \mathbf{f}) - y(\mathbf{f}) \right| < \epsilon \qquad \forall \mathbf{f} \in [0,1]^p$$

- ☺ Any continuous function can be approximated arbitrarily well by a neural network with a single hidden layer
- ☹ How to find the parameters?
- ☹ Does it generalize well / overfit?

Cybenko 1989; Hornik 1991

Neural nets as universal approximators

Universal Approximation Theorem Let ξ be a non-constant, bounded, and monotonically-increasing continuous activation function, $y : [0,1]^p \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times p}$ (including bias) s.t.

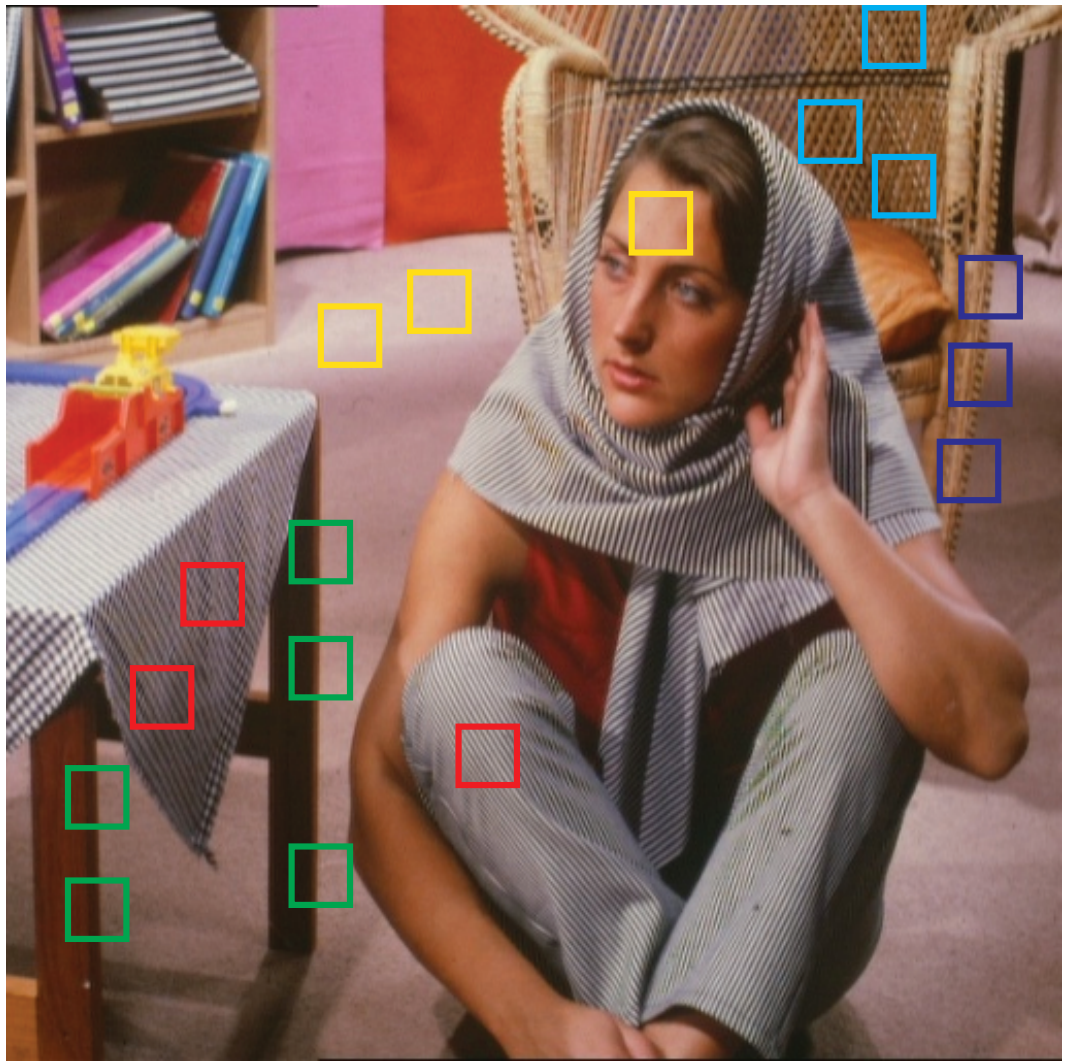
$$\left| \sum_{i=1}^{n} a_i \xi(\mathbf{w}_i^\top \mathbf{f}) - y(\mathbf{f}) \right| < \epsilon \qquad \forall \mathbf{f} \in [0,1]^p$$

- ☺ Any continuous function can be approximated arbitrarily well by a neural network with a single hidden layer
- ☹ How to find the parameters?
- ☹ Does it generalize well / overfit?
- ☹ Shallow nets work poorly for high-dimensional data s.a. images

Cybenko 1989; Hornik 1991

Take advantage of the structure of the data!

Stationarity and Self-similarity



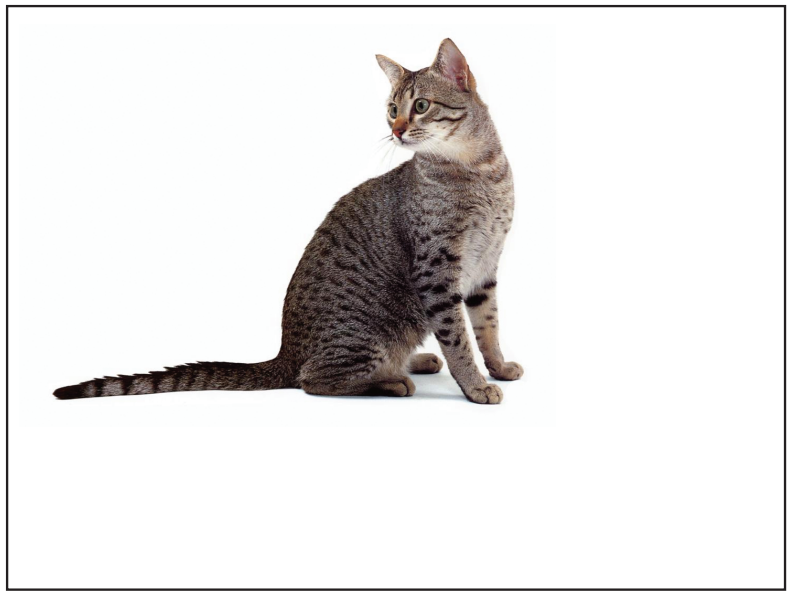
Data is self-similar across the domain

Locality

- Local features are represented by $r \times r$ compact support kernels.
- $\mathcal{O}(1)$ parameters per filter.

Karpathy 2016

Translation invariance (image classification tasks)



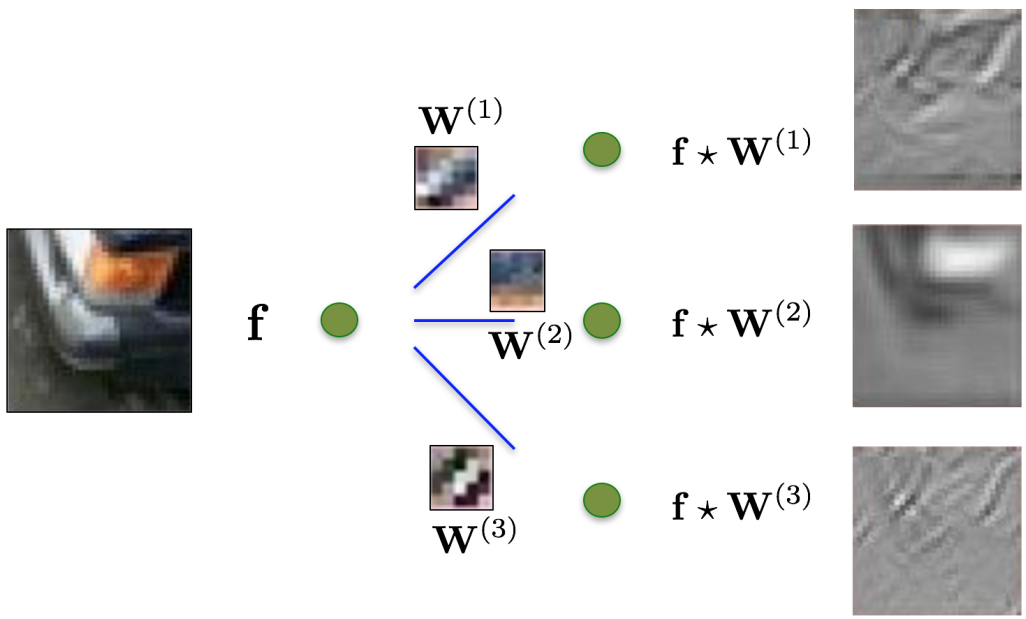
$$y(\mathcal{T}_v f) = y(f) \quad \forall f, v$$

where

- image is modeled as a function $f \in L^2([0,1]^2)$
- $\mathcal{T}_v f(x) = f(x - v)$ is a translation operator
- $v \in [0,1]^2$ is a translation vector
- $y : L^2([0,1]^2) \to \{1, \ldots, L\}$ is classification functional

Introducing local filters

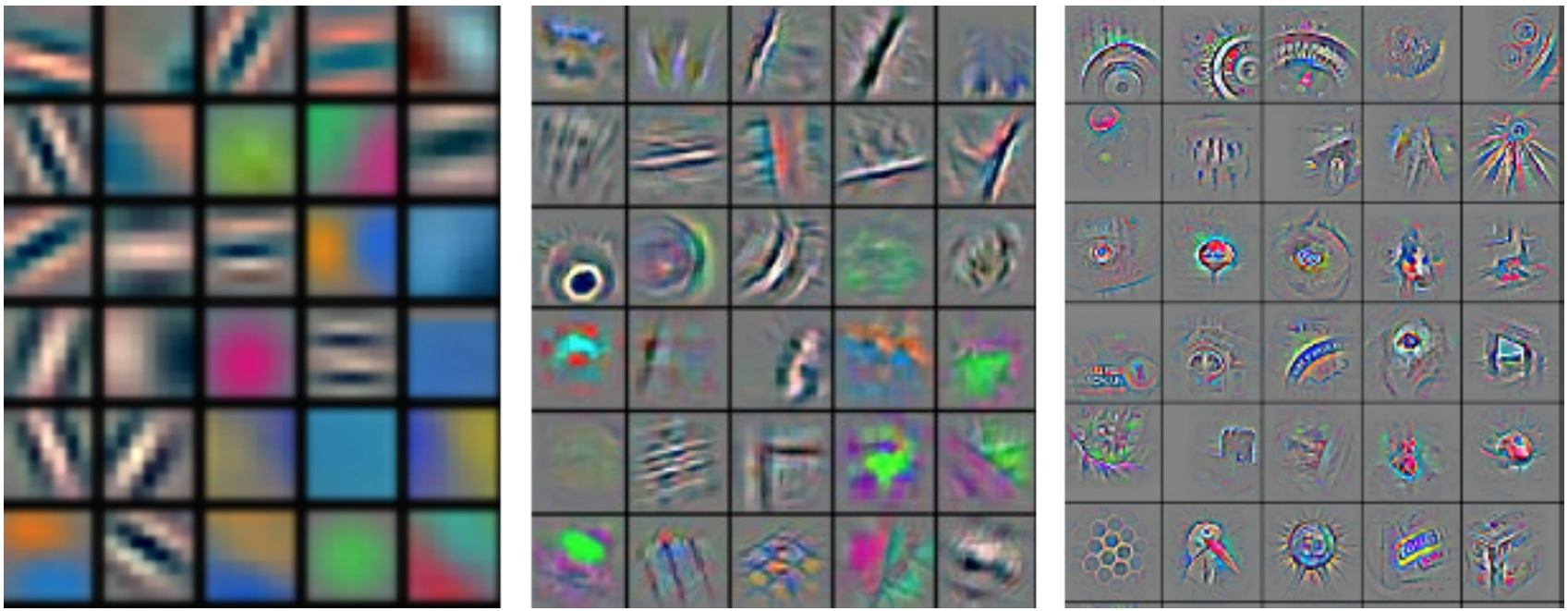
- Stationarity is implemented by shift-invariant convolutional operators.
- $\mathcal{O}(n \log n)$ computational complexity in general case (FFT).
- $\mathcal{O}(n)$ computational complexity for compact kernels.



Vasilache, LeCun, et al 2015

Hierarchy and Compositionality

Data is compositional: images, video, sound are formed of hierarchical local stationary patterns.

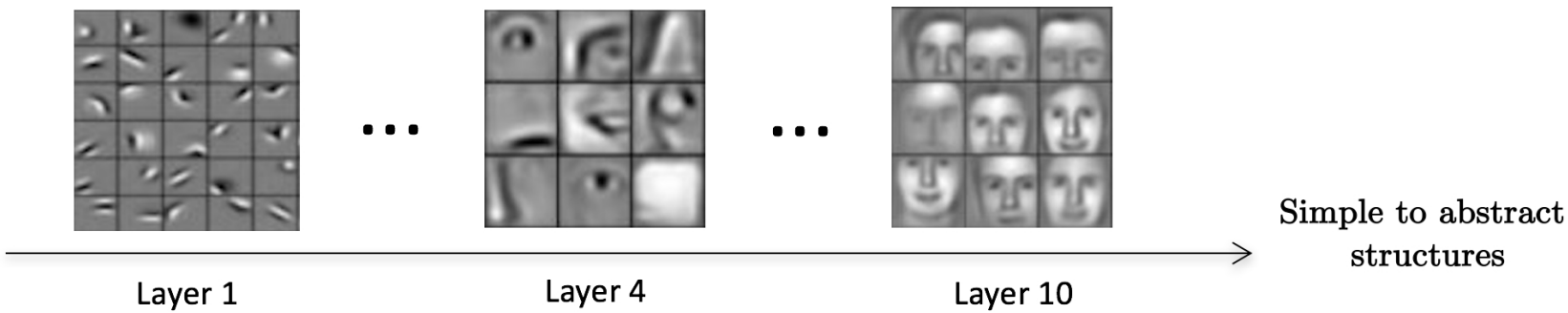


Typical features learned by a CNN becoming increasingly complex at deeper layers

Zeiler, Fergus 2013

Hierarchy and Compositionality

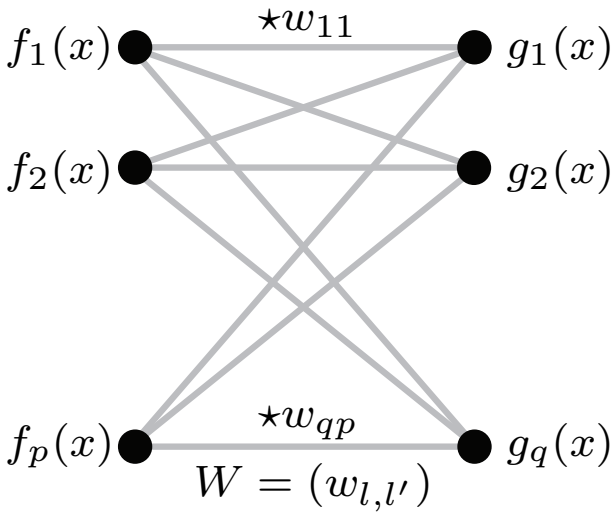
Data is compositional: images, video, sound are formed of hierarchical local stationary patterns.



Typical features learned by a CNN becoming increasingly complex at deeper layers

Zeiler, Fergus 2013

Convolutional neural network (CNN)



Single convolutional layer

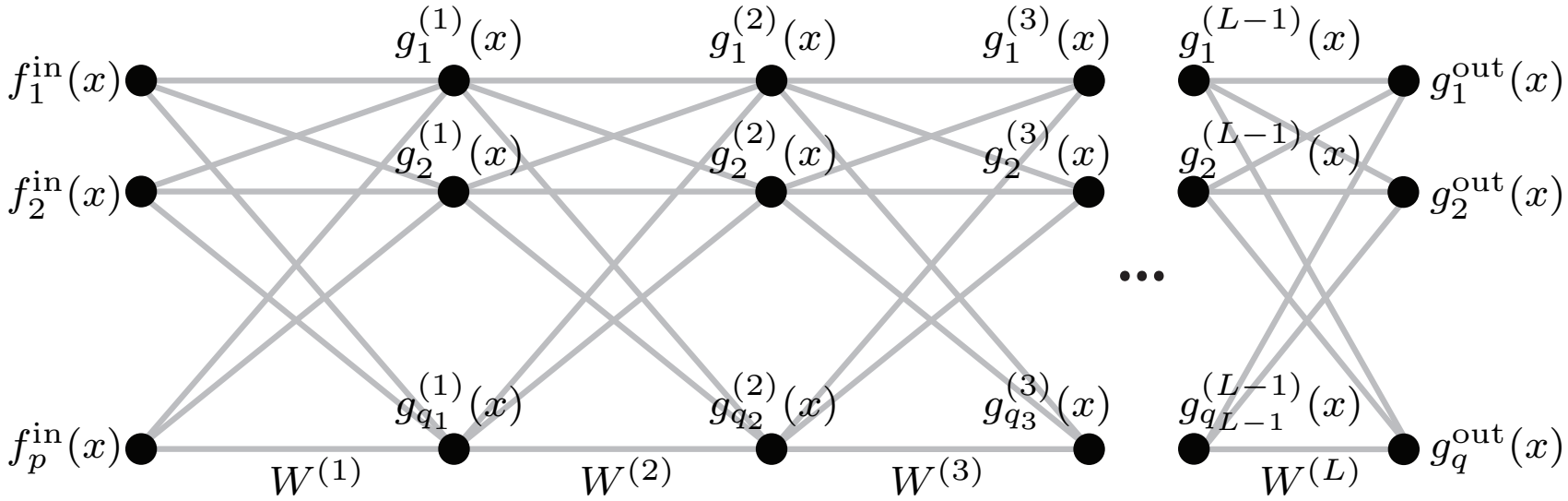
Conv. layer $$g_l(x) = \xi\left(\sum_{l'=1}^{p}(f_{l'} \star w_{l,l'})(x)\right) \quad \begin{matrix} l = 1, \ldots, q \\ l' = 1, \ldots, p \end{matrix}$$

Activation, e.g. $\xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Parameters filters W

LeCun et al. 1989

Convolutional neural network (CNN)



CNN consisting of L convolutional layers

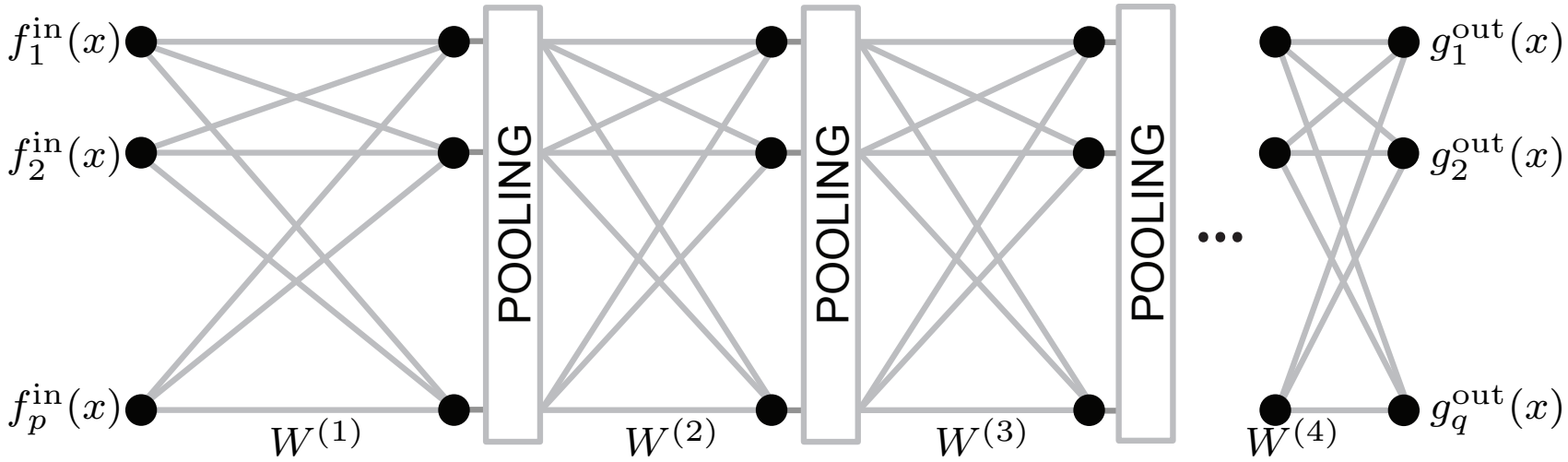
Conv. layer $$g_l^{(k)}(x) = \xi\left(\sum_{l'=1}^{q_{k-1}} (g_{l'}^{(k-1)} \star w_{l,l'}^{(k)})(x)\right) \quad \begin{matrix} l = 1, \dots, q_k \\ l' = 1, \dots, q_{k-1} \end{matrix}$$

Activation, e.g. $\xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Parameters filters of all layers $W^{(1)}, \dots, W^{(L)}$

LeCun et al. 1989

Convolutional neural network (CNN)



CNN consisting of L convolutional layers interleaved with pooling

Conv. layer $$g_l^{(k)}(x) = \xi\left(\sum_{l'=1}^{q_{k-1}} (g_{l'}^{(k-1)} \star w_{l,l'}^{(k)})(x)\right) \quad \begin{matrix} l = 1, \dots, q_k \\ l' = 1, \dots, q_{k-1} \end{matrix}$$

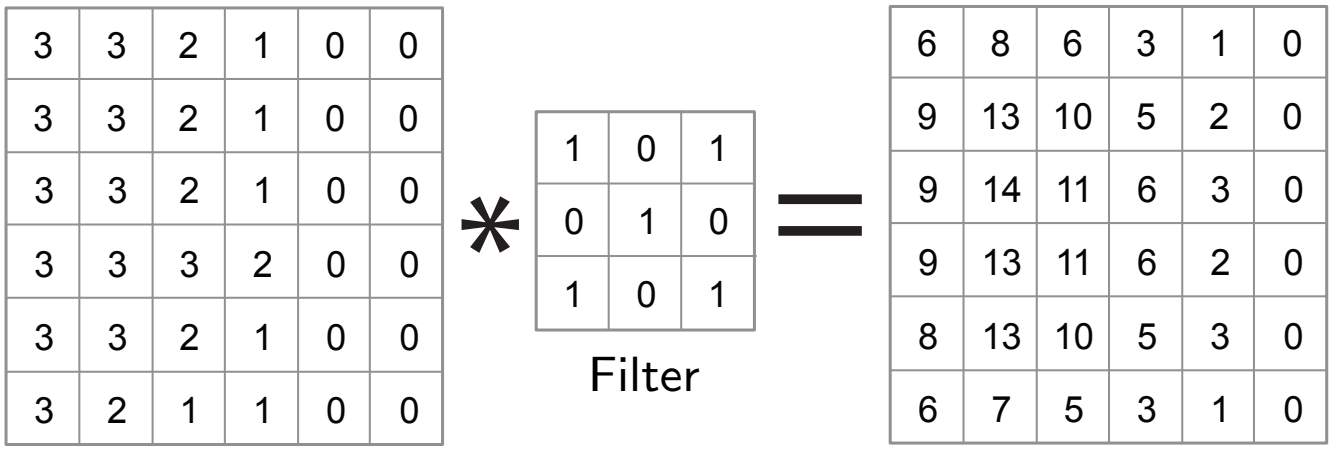
Activation, e.g. $\xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Parameters filters of all layers $W^{(1)}, \dots, W^{(L)}$

Pooling $g_l^{(k)}(x) = \|g_l^{(k-1)}(x') : x' \in \mathcal{N}(x)\|_p \quad p = 1, 2, \text{ or } \infty$

LeCun et al. 1989

Pooling

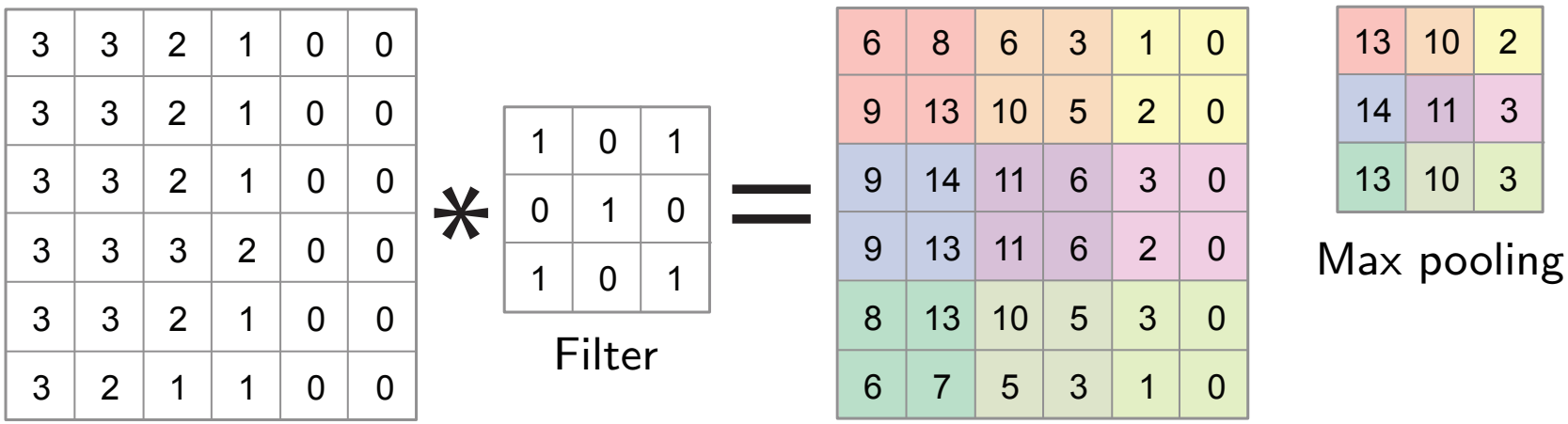


Input image

Filter

Feature map

Pooling


3 3 2 1 0 0
3 3 2 1 0 0
3 3 2 1 0 0
3 3 3 2 0 0
3 3 2 1 0 0
3 2 1 1 0 0
Input image
*
1 0 1
0 1 0
1 0 1
Filter
=
6 8 6 3 1 0
9 13 10 5 2 0
9 14 11 6 3 0
9 13 11 6 2 0
8 13 10 5 3 0
6 7 5 3 1 0
Feature map
13 10 2
14 11 3
13 10 3
Max pooling

Pooling

3	3	2	1	0	0
3	3	2	1	0	0
3	3	2	1	0	0
3	3	3	2	0	0
3	3	2	1	0	0
3	2	1	1	0	0

Input image

$*$

1	0	1
0	1	0
1	0	1

Filter

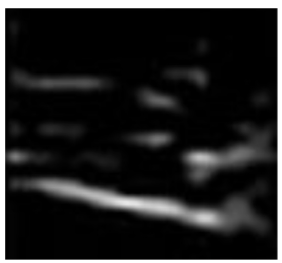
$=$

6	8	6	3	1	0
9	13	10	5	2	0
9	14	11	6	3	0
9	13	11	6	2	0
8	13	10	5	3	0
6	7	5	3	1	0

Feature map

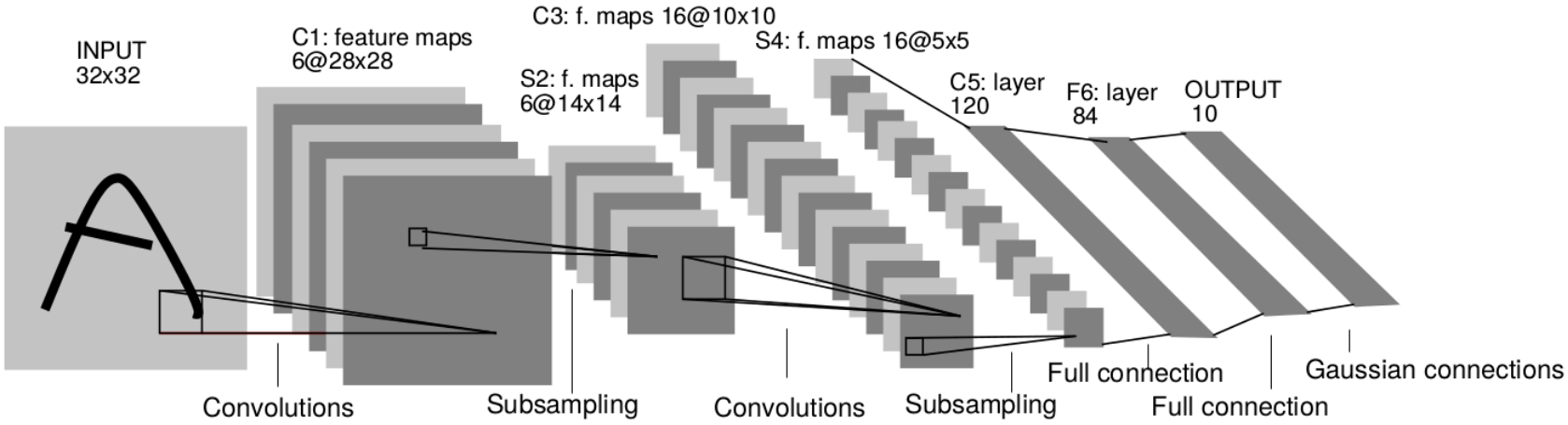
13	10	2
14	11	3
13	10	3

Max pooling



2x2 Max pooling

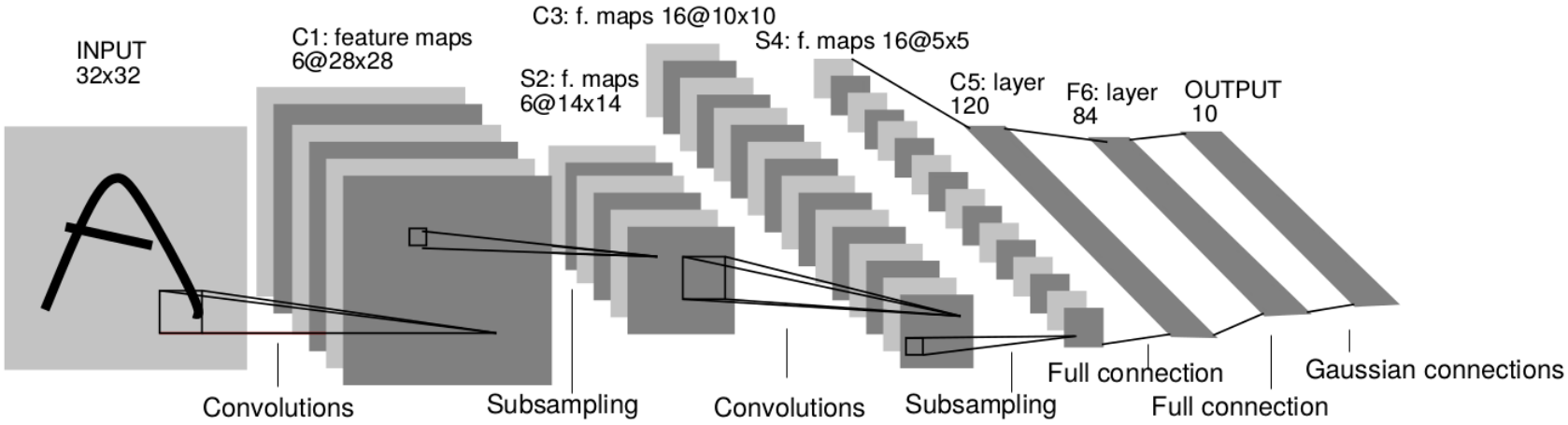
Key properties of CNNs



☺ Convolutional filters **(Translation invariance)**

LeCun et al. 1989

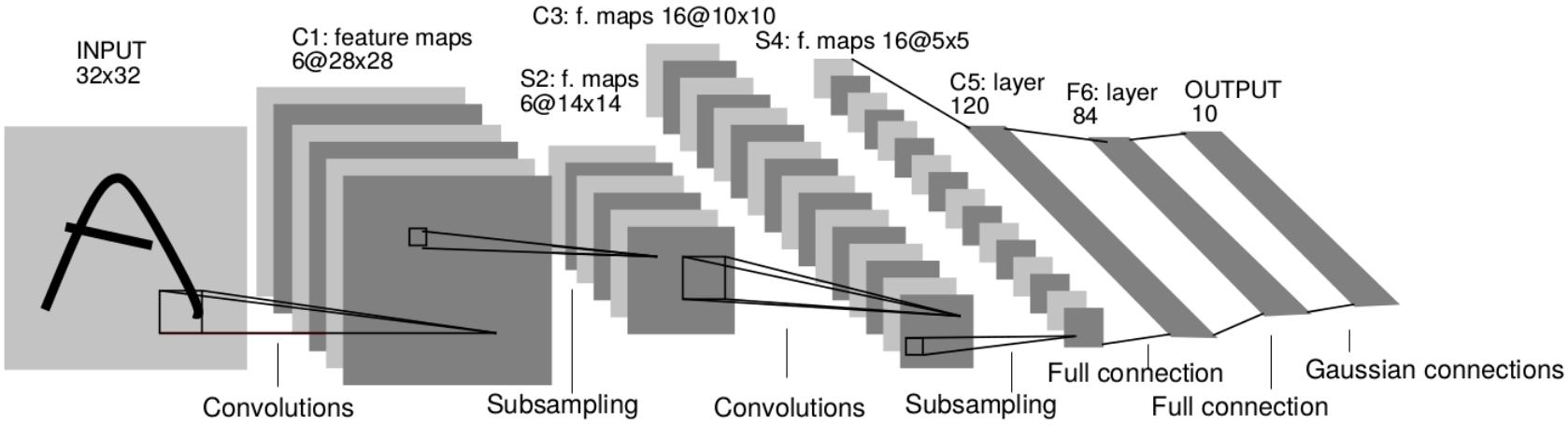
Key properties of CNNs



- ☺ Convolutional filters **(Translation invariance)**
- ☺ Multiple layers **(Compositionality)**

LeCun et al. 1989

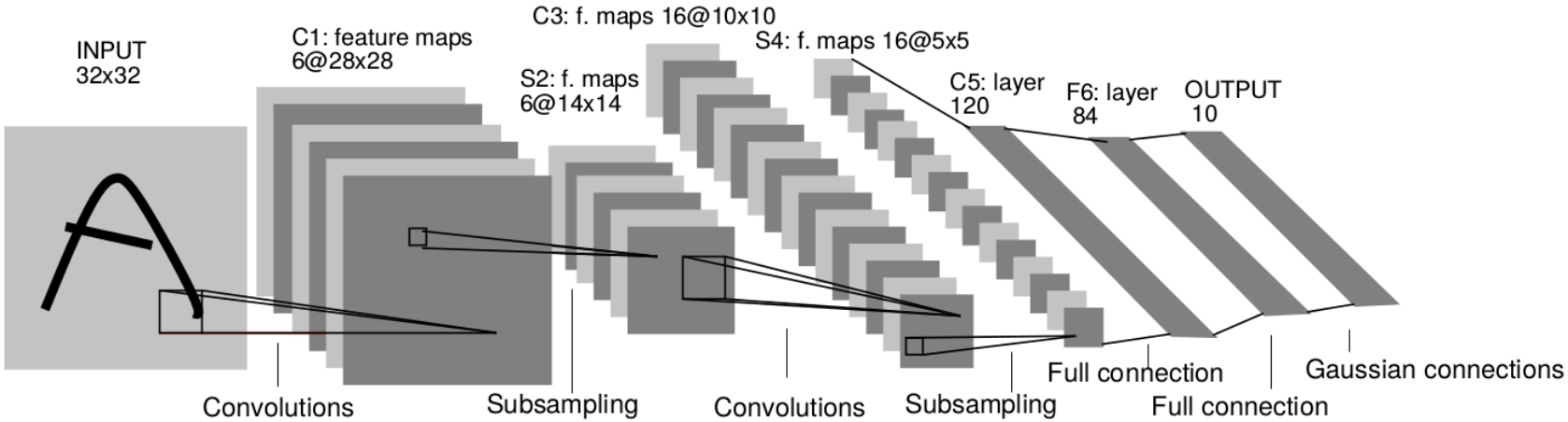
Key properties of CNNs



- ☺ Convolutional filters **(Translation invariance)**
- ☺ Multiple layers **(Compositionality)**
- ☺ Filters localized in space **(Locality)**

LeCun et al. 1989

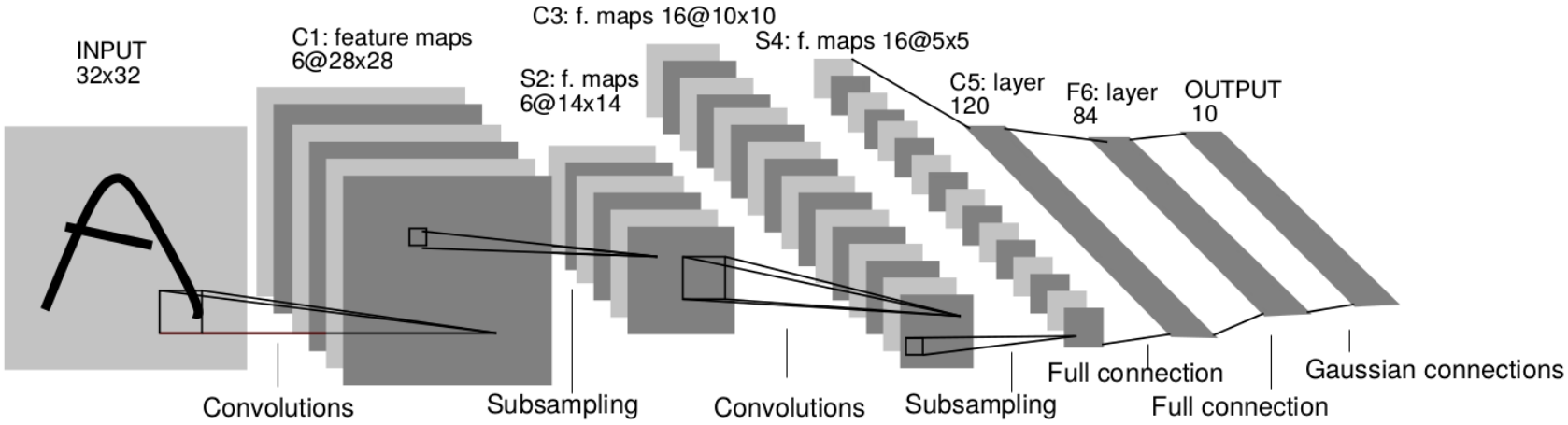
Key properties of CNNs



- ☺ Convolutional filters **(Translation invariance)**
- ☺ Multiple layers **(Compositionality)**
- ☺ Filters localized in space **(Locality)**
- ☺ Weight sharing **(Self-similarity)**

LeCun et al. 1989

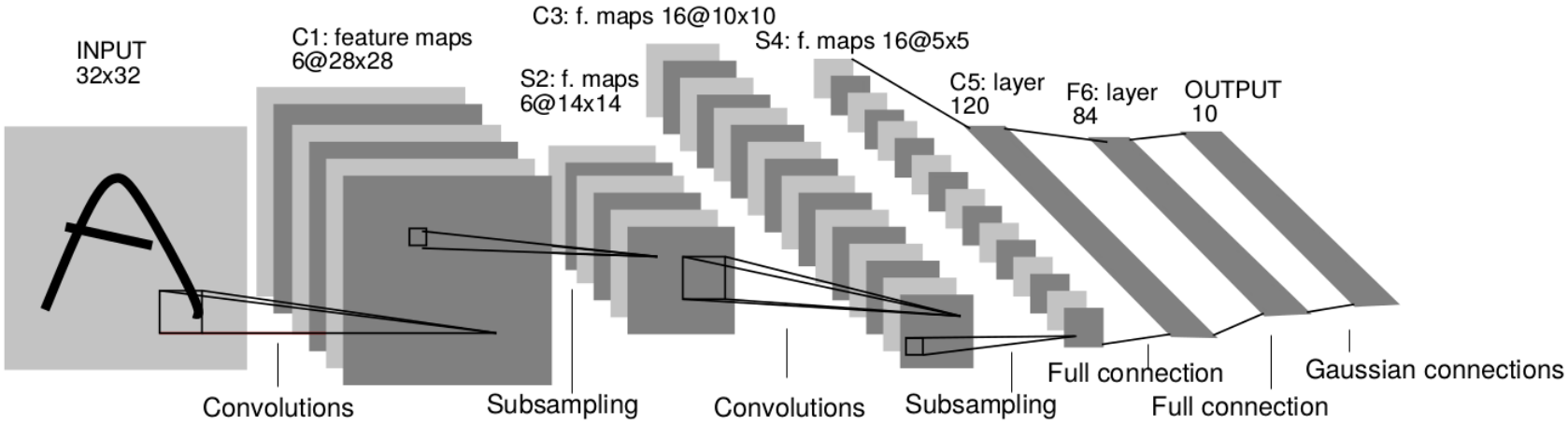
Key properties of CNNs



- ☺ Convolutional filters **(Translation invariance)**
- ☺ Multiple layers **(Compositionality)**
- ☺ Filters localized in space **(Locality)**
- ☺ Weight sharing **(Self-similarity)**
- ☺ $\mathcal{O}(1)$ parameters per filter (independent of input image size n)

LeCun et al. 1989

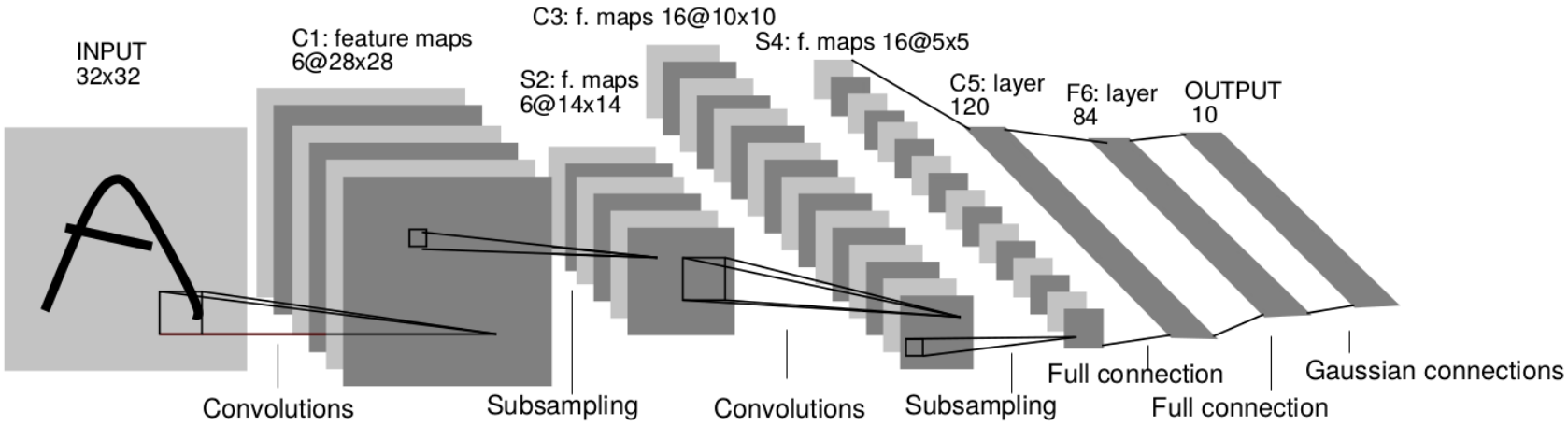
Key properties of CNNs



- ☺ Convolutional filters **(Translation invariance)**
- ☺ Multiple layers **(Compositionality)**
- ☺ Filters localized in space **(Locality)**
- ☺ Weight sharing **(Self-similarity)**
- ☺ $\mathcal{O}(1)$ parameters per filter (independent of input image size n)
- ☺ $\mathcal{O}(n)$ complexity per layer (filtering done in the spatial domain)

LeCun et al. 1989

Key properties of CNNs

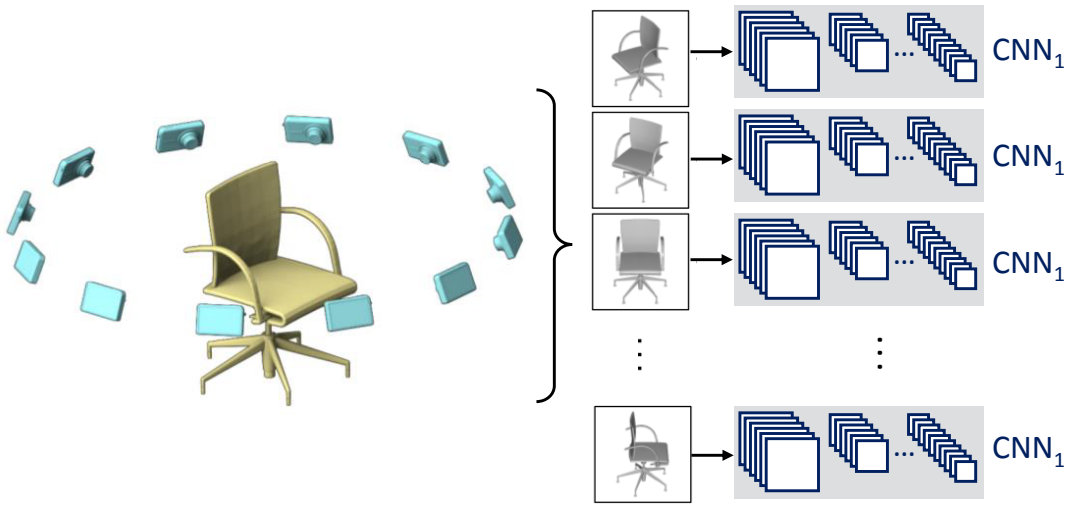


- ☺ Convolutional filters **(Translation invariance)**
- ☺ Multiple layers **(Compositionality)**
- ☺ Filters localized in space **(Locality)**
- ☺ Weight sharing **(Self-similarity)**
- ☺ $\mathcal{O}(1)$ parameters per filter (independent of input image size n)
- ☺ $\mathcal{O}(n)$ complexity per layer (filtering done in the spatial domain)
- ☺ $\mathcal{O}(\log n)$ layers in classification tasks

LeCun et al. 1989

Extrinsic Methods

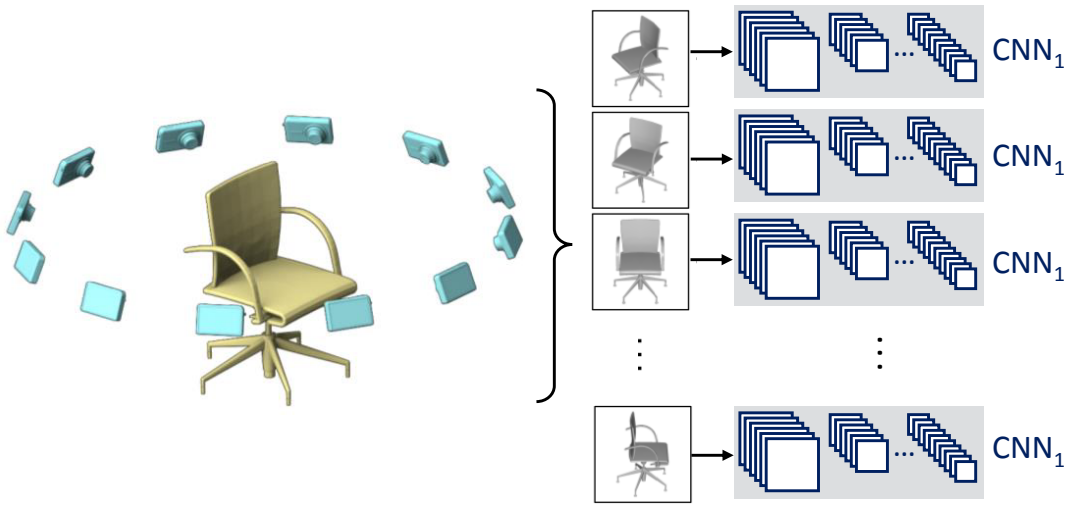
Multi-view CNNs



- Represent 3D object as a collection of range images from different views

Su et al. 2015

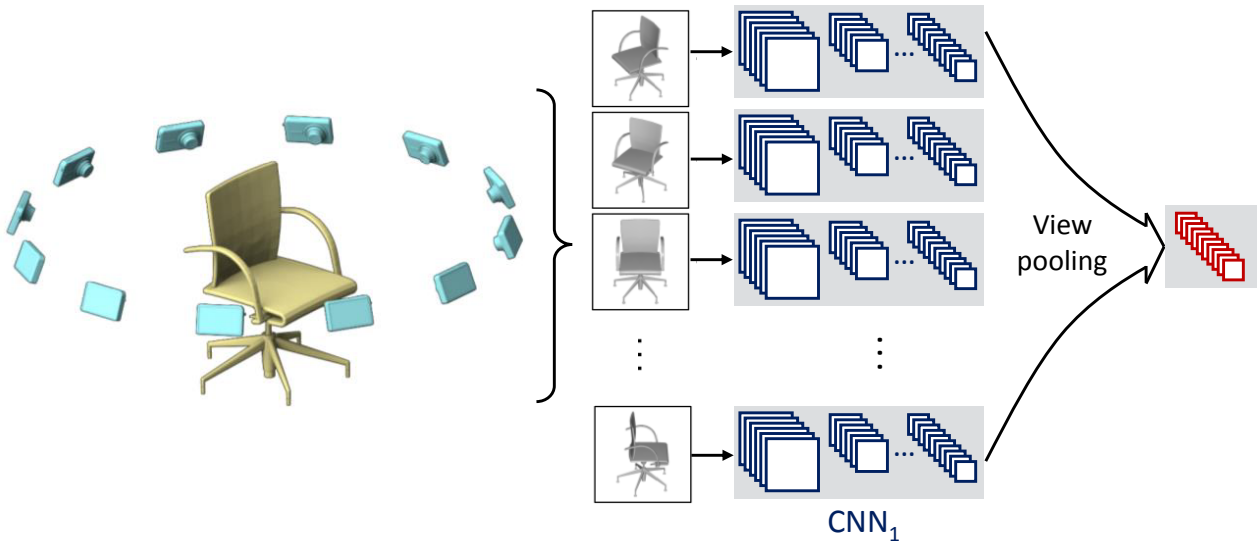
Multi-view CNNs



- Represent 3D object as a collection of range images from different views
- CNN_1: Extract image features (parameters are shared across views)

Su et al. 2015

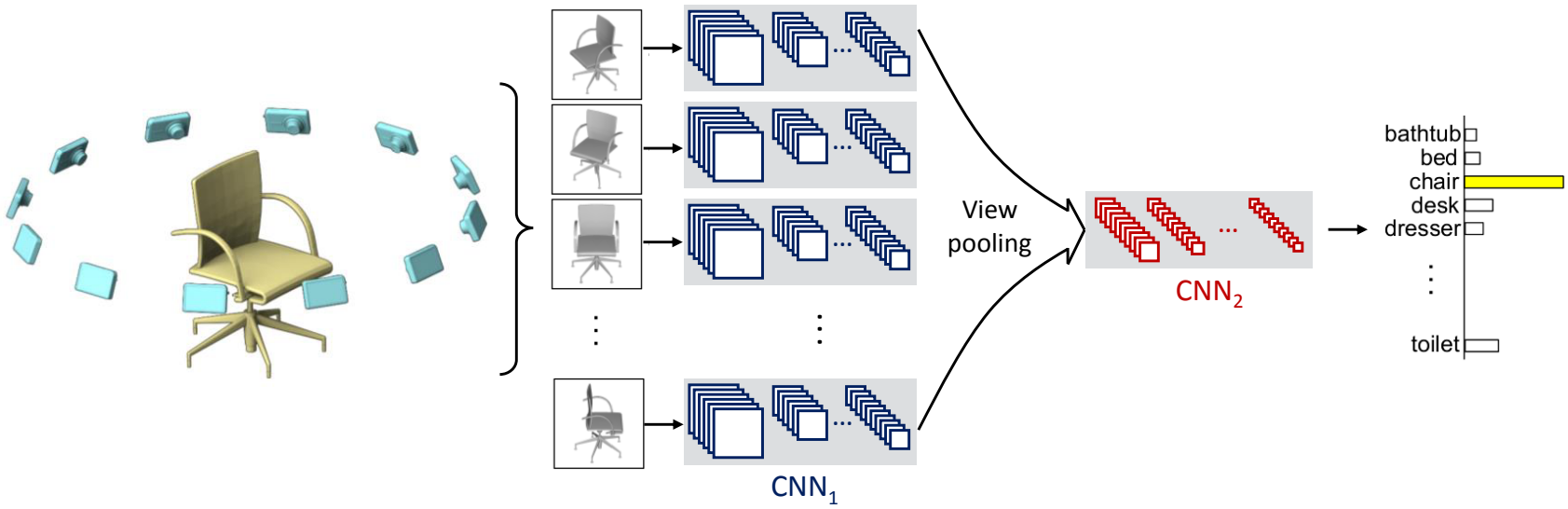
Multi-view CNNs



- Represent 3D object as a collection of range images from different views
- CNN_1: Extract image features (parameters are shared across views)
- Element-wise max pooling across all views

Su et al. 2015

Multi-view CNNs



- Represent 3D object as a collection of range images from different views
- CNN_1: Extract image features (parameters are shared across views)
- Element-wise max pooling across all views
- CNN_2: Produce shape descriptors + final prediction

Su et al. 2015

Applications of Multi-view CNNs

- 3D shape classification and retrieval
  - Pre-trained on ImageNet
  - Fine-tuned on 2D views

Su et al. 2015

Applications of Multi-view CNNs

- 3D shape classification and retrieval
  - Pre-trained on ImageNet
  - Fine-tuned on 2D views

classify $\Longrightarrow$ **"chair"**

- Sketch classification
  - Mimic views by jittering

classify $\Longrightarrow$ **"chair"**

Su et al. 2015

Applications of Multi-view CNNs

- 3D shape classification and retrieval
  - Pre-trained on ImageNet
  - Fine-tuned on 2D views


classify ⟹ **"chair"**

- Sketch classification
  - Mimic views by jittering


classify ⟹ **"chair"**

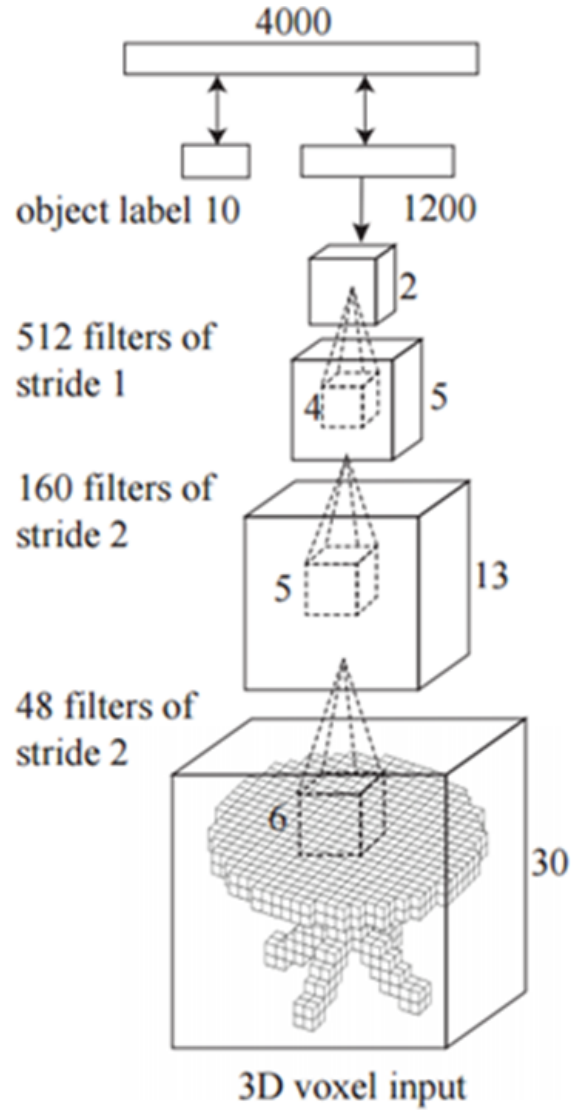
- Sketch-based shape retrieval
  - Render views with hand-drawn style (edge maps)


retrieve ⟹

Su et al. 2015

3D ShapeNets

- Volumetric representation (shape = binary voxels on 3D grid)

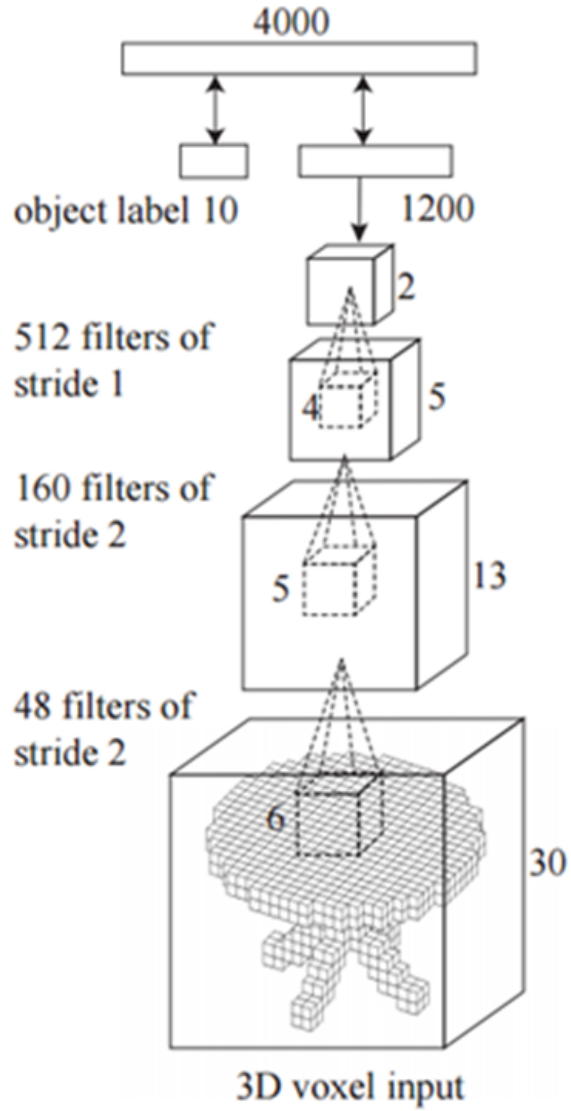


Convolutional deep belief network

Wu et al. 2015

3D ShapeNets

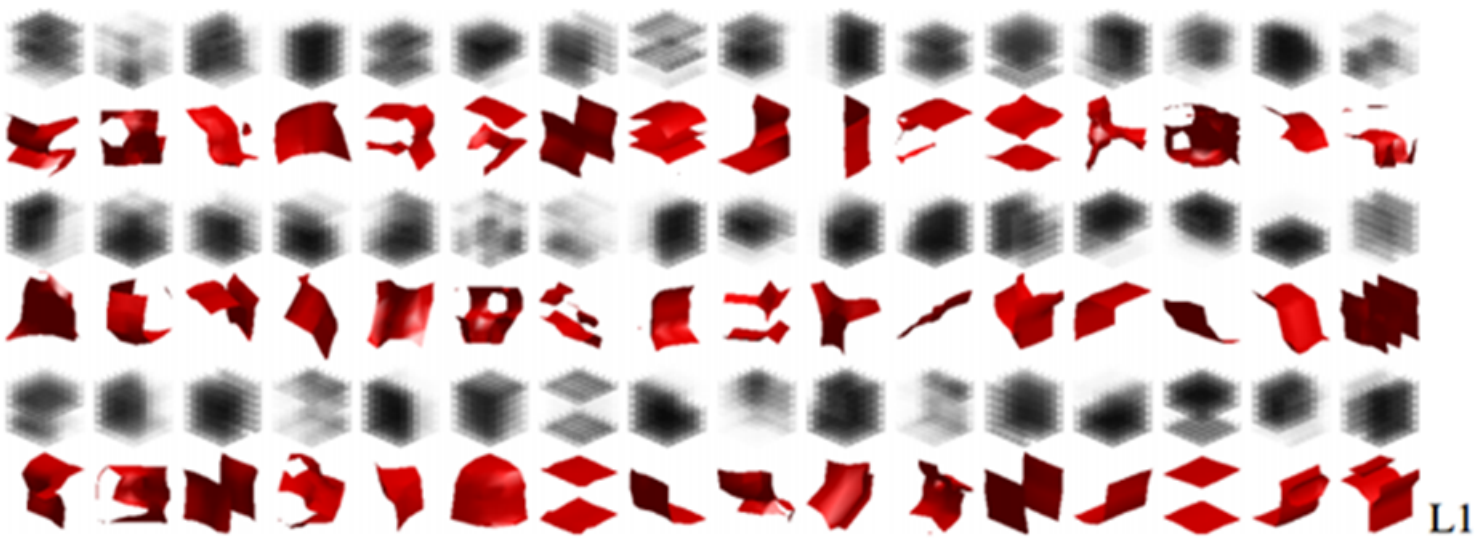
- Volumetric representation (shape = binary voxels on 3D grid)
- 3D convolutional network



Convolutional deep belief network

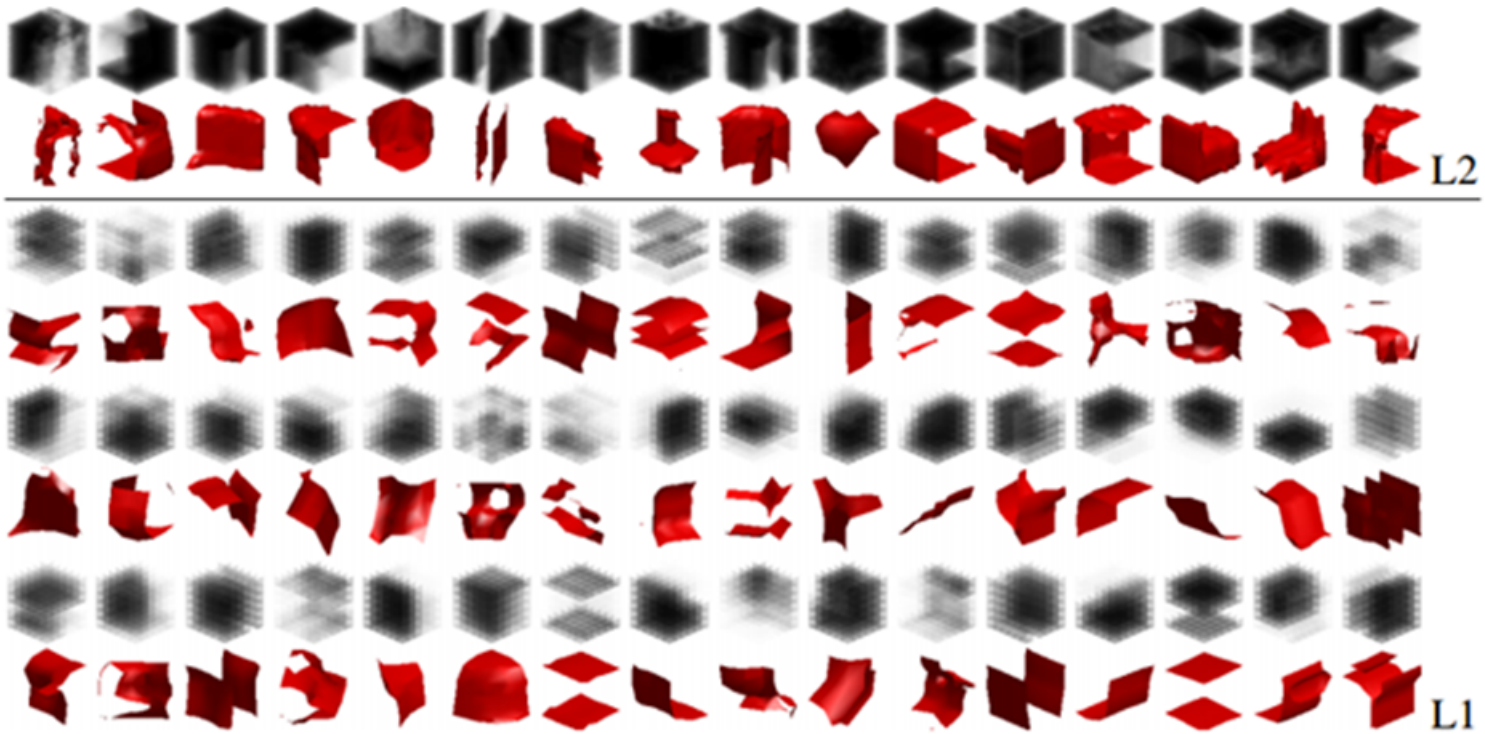
Wu et al. 2015

Learned features: 3D primitives



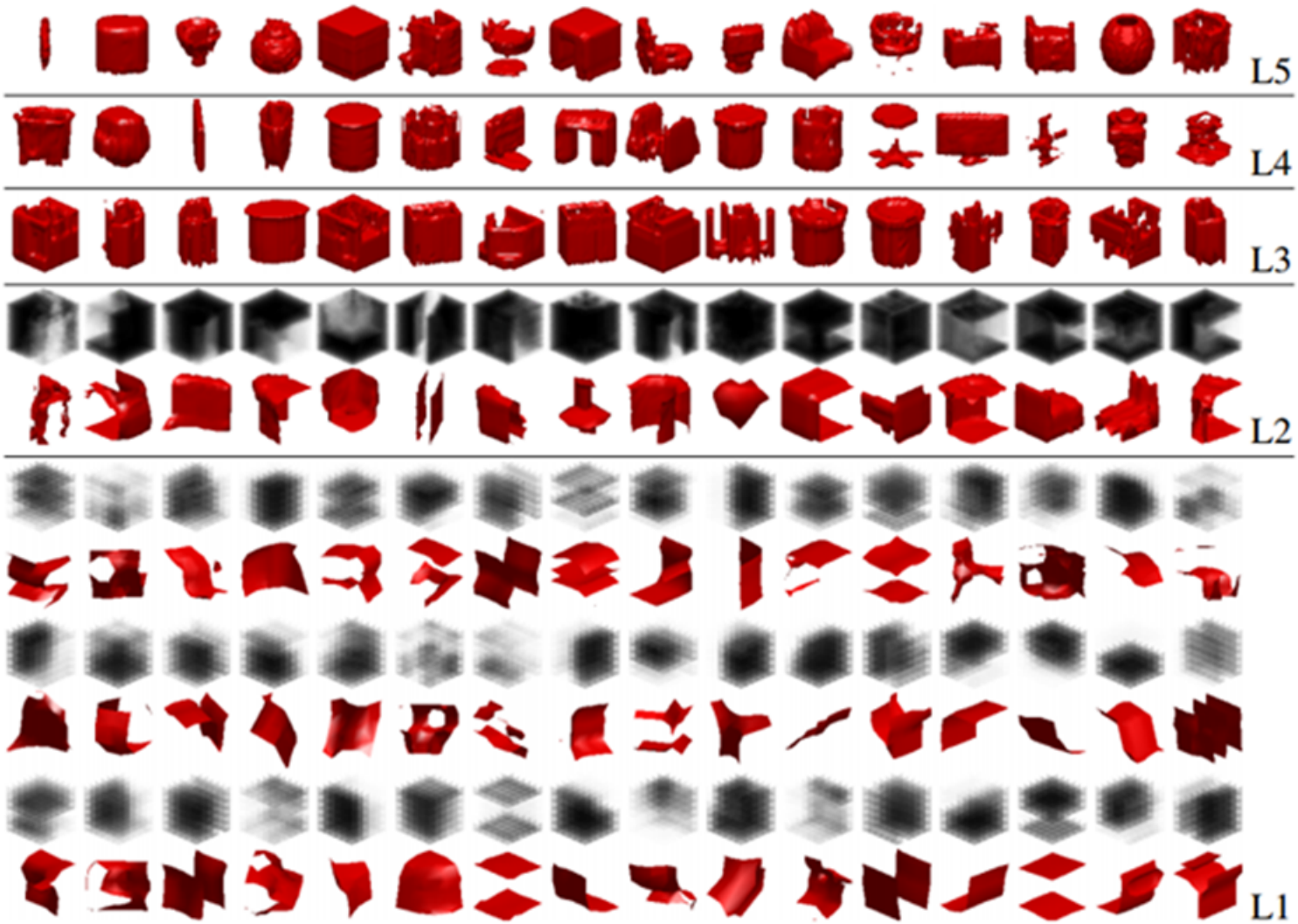
Wu et al. 2015

Learned features: 3D primitives



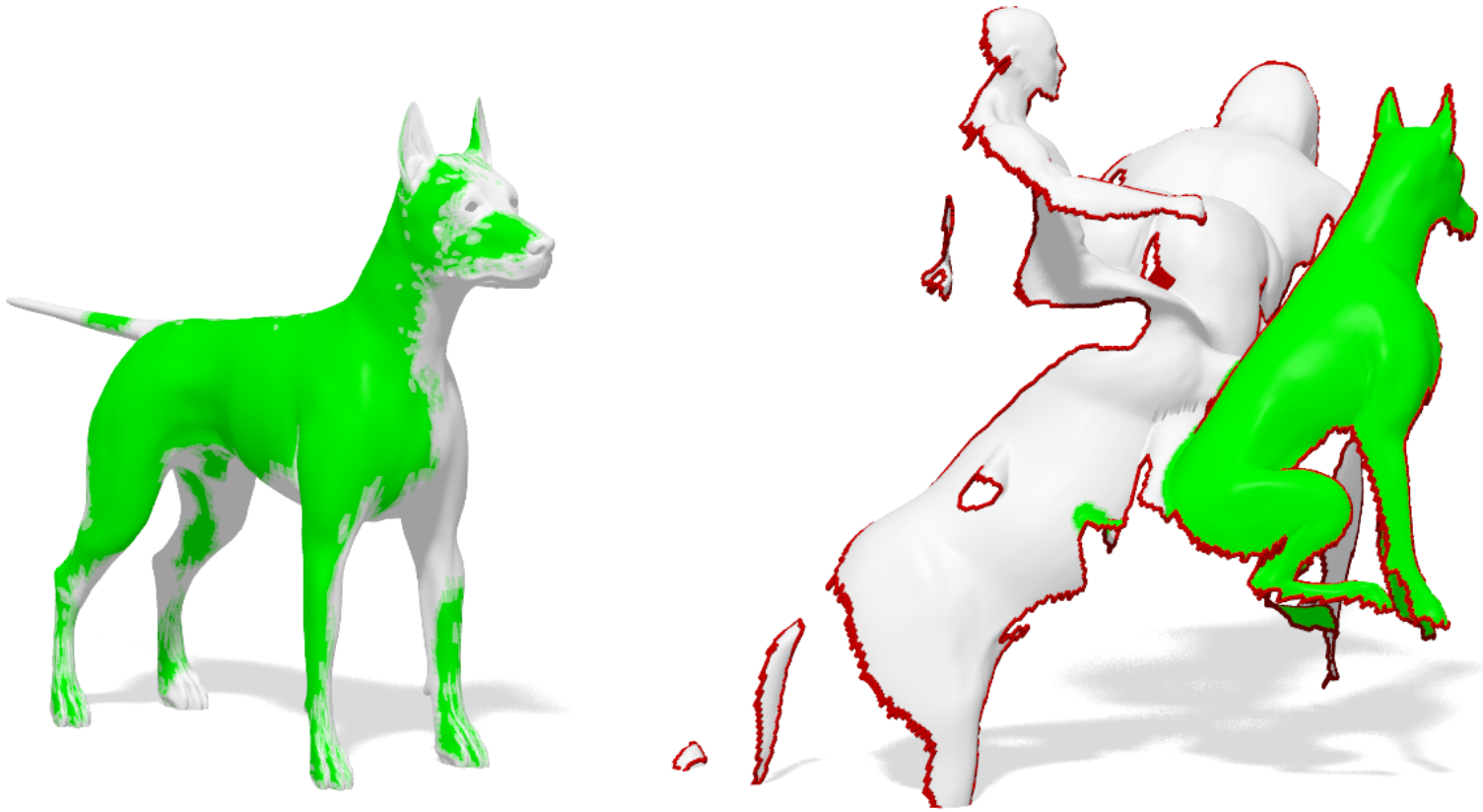
Wu et al. 2015

Learned features: 3D primitives



Wu et al. 2015

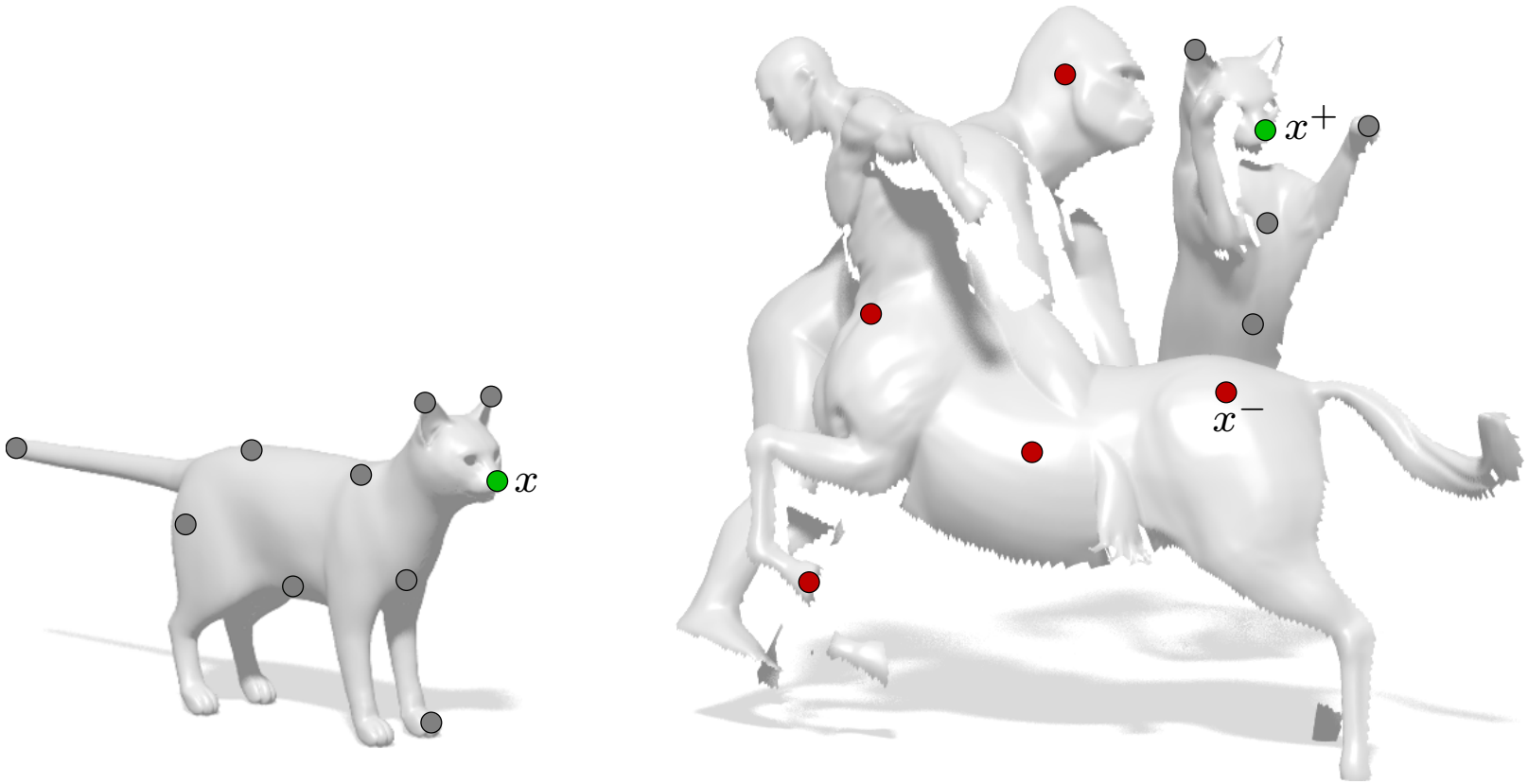
Non-rigid clutter



Clutter, missing parts, and non-rigid deformations

Cosmo, Rodolà, Masci, Torsello, Bronstein 2016

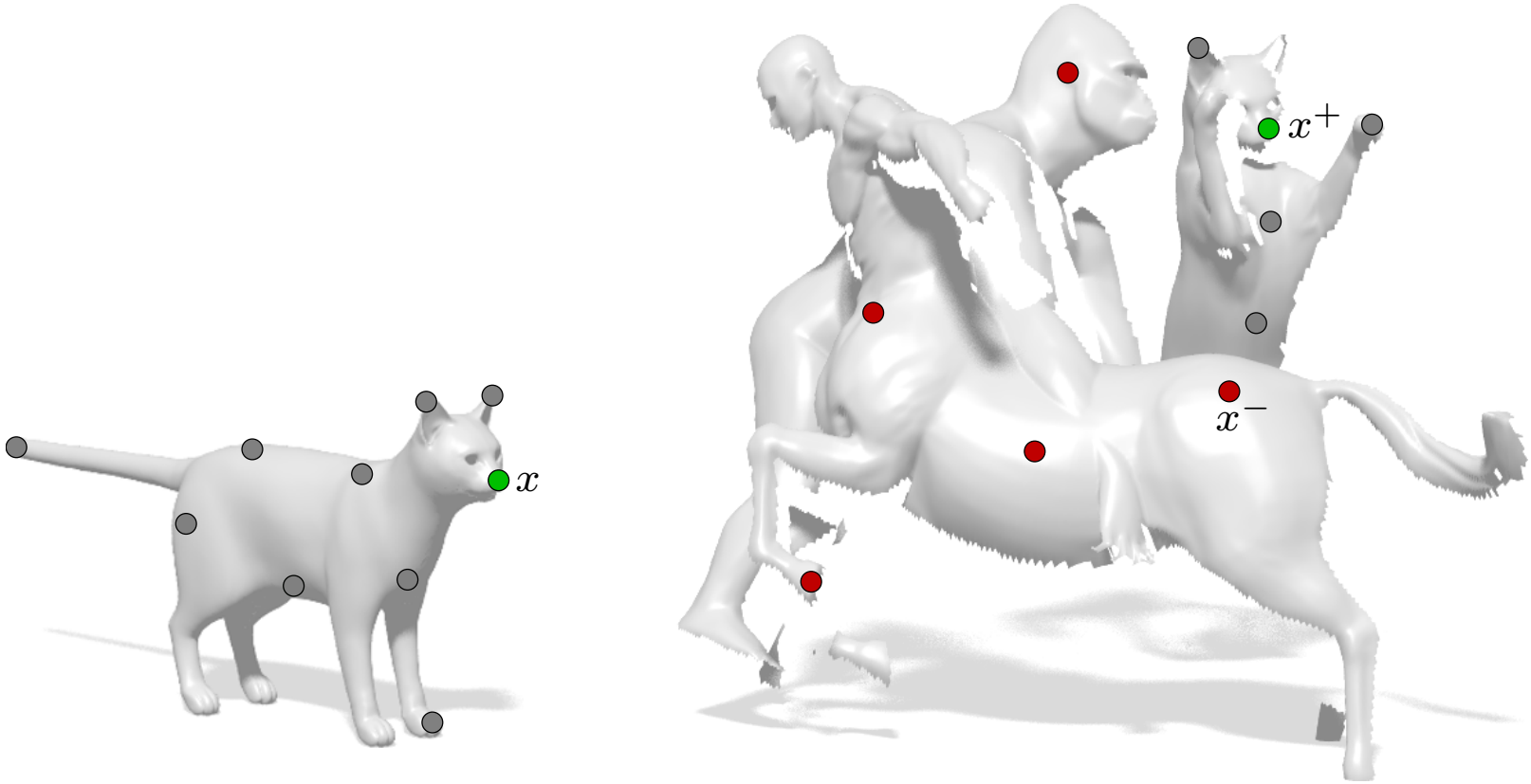
Descriptor metric learning



Training set consisting of pairs of positives (x, x^+) and negatives (x, x^-)

Cosmo, Rodolà, Masci, Torsello, Bronstein 2016

Descriptor metric learning

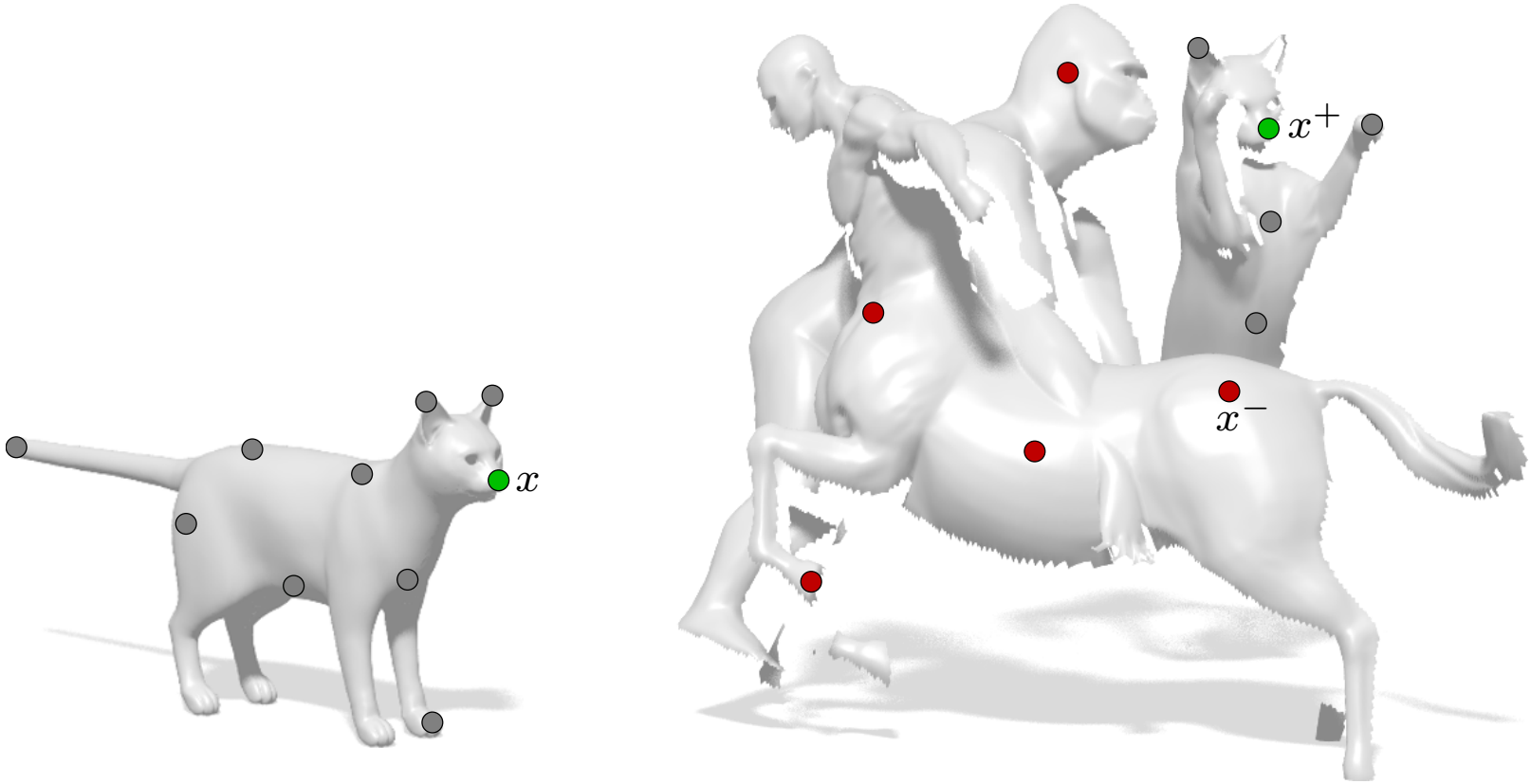


Training set consisting of pairs of positives (x, x^+) and negatives (x, x^-)

Each point x has an associated local descriptor $\mathbf{g}(x)$

Cosmo, Rodolà, Masci, Torsello, Bronstein 2016

Descriptor metric learning



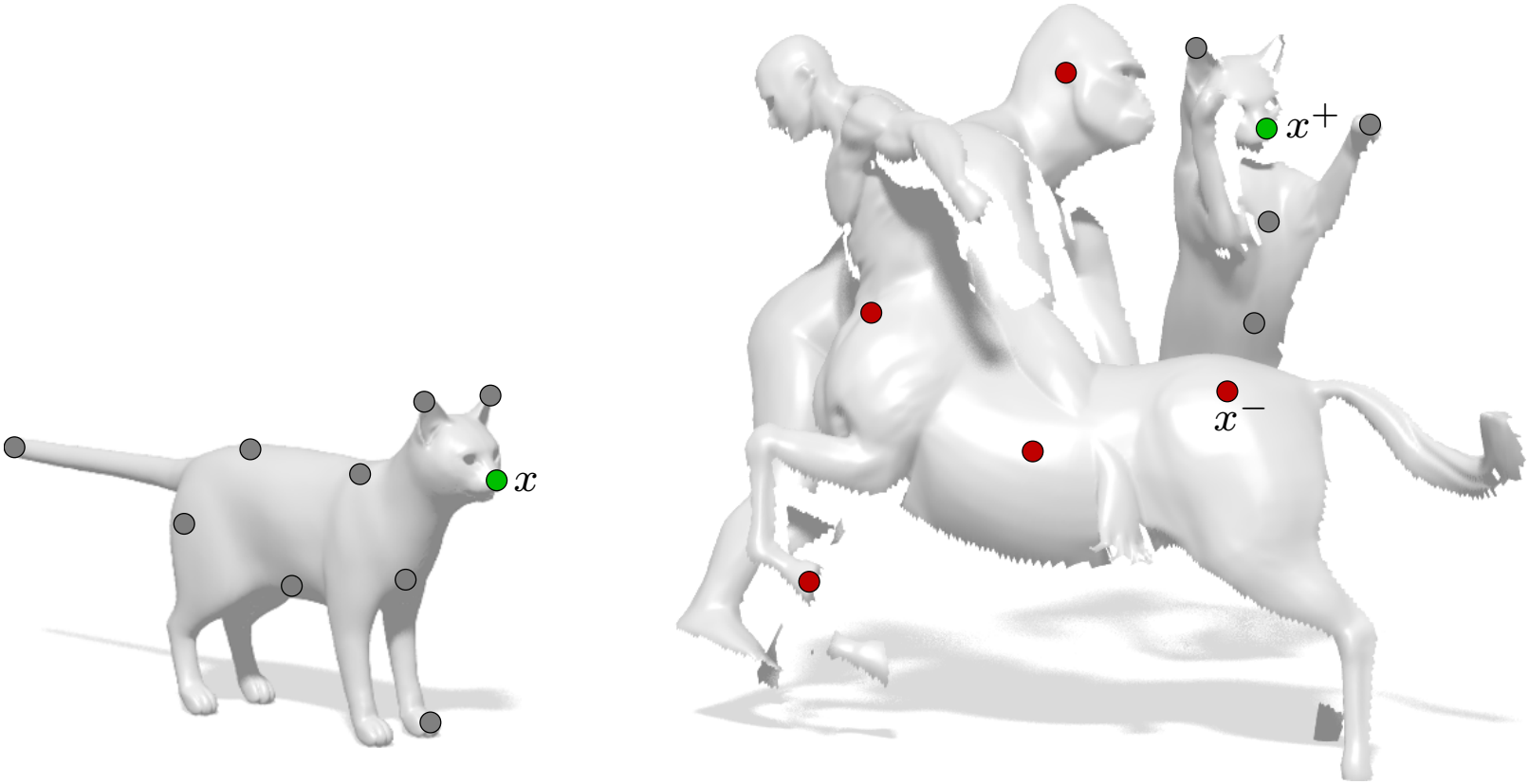
Training set consisting of pairs of positives (x, x^+) and negatives (x, x^-)

Each point x has an associated local descriptor $\mathbf{g}(x)$

Goal: learn a metric $d(\mathbf{g}(x), \mathbf{g}(x'))$ between the descriptors representing as reliably as possible the similarity/dissimilarity of positives/negatives

Cosmo, Rodolà, Masci, Torsello, Bronstein 2016

Descriptor metric learning



Training set consisting of pairs of positives (x, x^+) and negatives (x, x^-)

Each point x has an associated local descriptor $\mathbf{g}(x)$

Goal: learn a metric s.t. $d(\mathbf{g}(x), \mathbf{g}(x^+)) \approx 0$ and $d(\mathbf{g}(x), \mathbf{g}(x^-)) \gg 0$

Cosmo, Rodolà, Masci, Torsello, Bronstein 2016

Descriptor metric learning

Parametrize the metric by applying a deep net $\mathbf{f}_{\boldsymbol{\Theta}}$ to the descriptor

$$d(\mathbf{g}(x), \mathbf{g}(x')) = \|\mathbf{f}_{\boldsymbol{\Theta}}(\mathbf{g}(x)) - \mathbf{f}_{\boldsymbol{\Theta}}(\mathbf{g}(x'))\|_2$$

Cosmo, Rodolà, Masci, Torsello, Bronstein 2016

Descriptor metric learning

Parametrize the metric by applying a deep net $\mathbf{f}_{\boldsymbol{\Theta}}$ to the descriptor

$$d(\mathbf{g}(x), \mathbf{g}(x')) = \|\mathbf{f}_{\boldsymbol{\Theta}}(\mathbf{g}(x)) - \mathbf{f}_{\boldsymbol{\Theta}}(\mathbf{g}(x'))\|_2$$

Learn network parameters $\boldsymbol{\Theta}$ by minimizing the siamese loss

$$\begin{aligned}\mathcal{L}_{\mathsf{s}}(\boldsymbol{\Theta}) = &\sum_{x, x^+ \in \mathcal{T}^+} \gamma \|\mathbf{f}_{\boldsymbol{\Theta}}(x) - \mathbf{f}_{\boldsymbol{\Theta}}(x^+)\|_2^2 \\ &+ \sum_{x, x^- \in \mathcal{T}^-} (1-\gamma)(m_s - \|\mathbf{f}_{\boldsymbol{\Theta}}(x) - \mathbf{f}_{\boldsymbol{\Theta}}(x^-)\|_2)_+^2\end{aligned}$$

Cosmo, Rodolà, Masci, Torsello, Bronstein 2016; Bromley et al. 1994; Hadsell et al. 2006

Descriptor metric learning

Parametrize the metric by applying a deep net $\mathbf{f}_{\boldsymbol{\Theta}}$ to the descriptor

$$d(\mathbf{g}(x), \mathbf{g}(x')) = \|\mathbf{f}_{\boldsymbol{\Theta}}(\mathbf{g}(x)) - \mathbf{f}_{\boldsymbol{\Theta}}(\mathbf{g}(x'))\|_2$$

Learn network parameters $\boldsymbol{\Theta}$ by minimizing the siamese loss

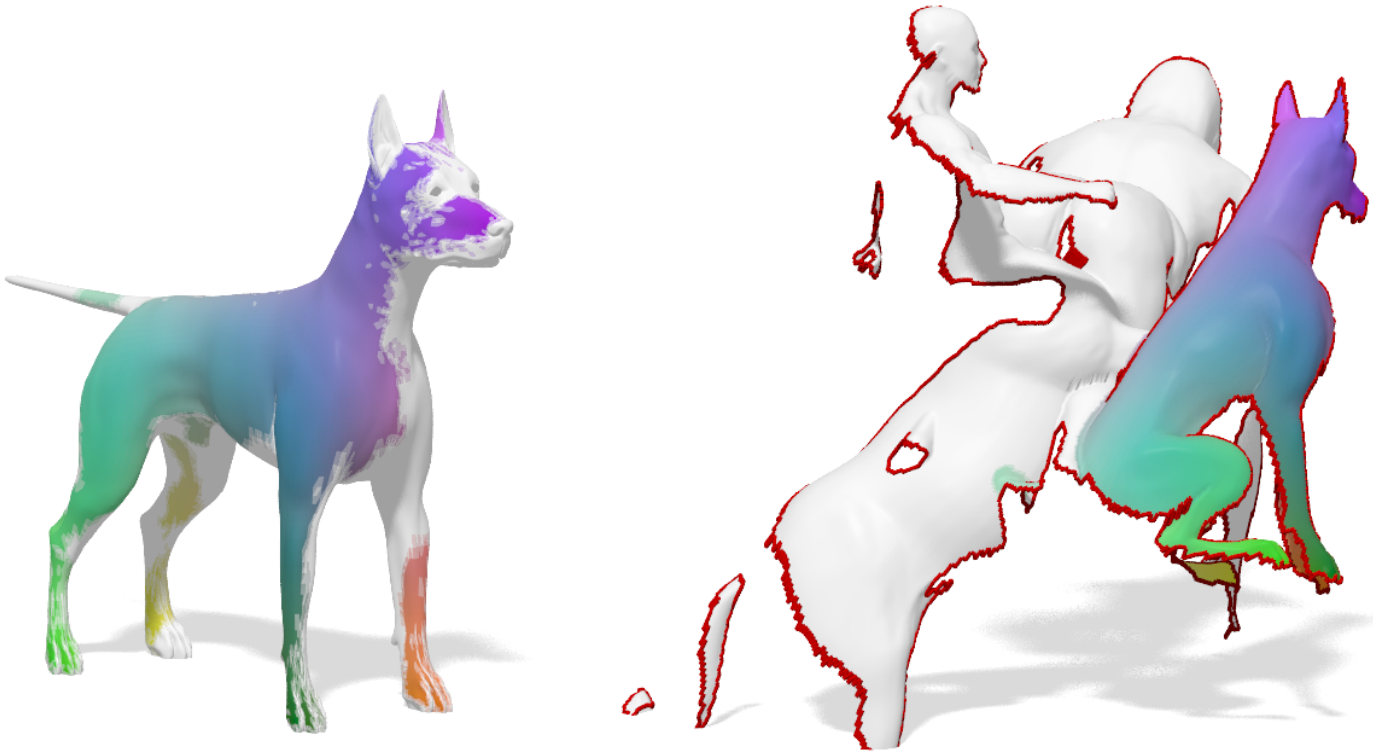
$$\begin{aligned}\mathcal{L}_{\mathrm{s}}(\boldsymbol{\Theta}) = &\sum_{x, x^+ \in \mathcal{T}^+} \gamma \|\mathbf{f}_{\boldsymbol{\Theta}}(x) - \mathbf{f}_{\boldsymbol{\Theta}}(x^+)\|_2^2 \\ &+ \sum_{x, x^- \in \mathcal{T}^-} (1-\gamma)(m_s - \|\mathbf{f}_{\boldsymbol{\Theta}}(x) - \mathbf{f}_{\boldsymbol{\Theta}}(x^-)\|_2)_+^2\end{aligned}$$

Regularize with global distribution penalty

$$\mathcal{L}_{\mathrm{g}}(\boldsymbol{\Theta}) = \sigma^+_{\boldsymbol{\Theta}} + \sigma^-_{\boldsymbol{\Theta}} + (m_g + \mu^+_{\boldsymbol{\Theta}} - \mu^-_{\boldsymbol{\Theta}})_+$$

Cosmo, Rodolà, Masci, Torsello, Bronstein 2016; Bromley et al. 1994; Hadsell et al. 2006; Kumar et al. 2015

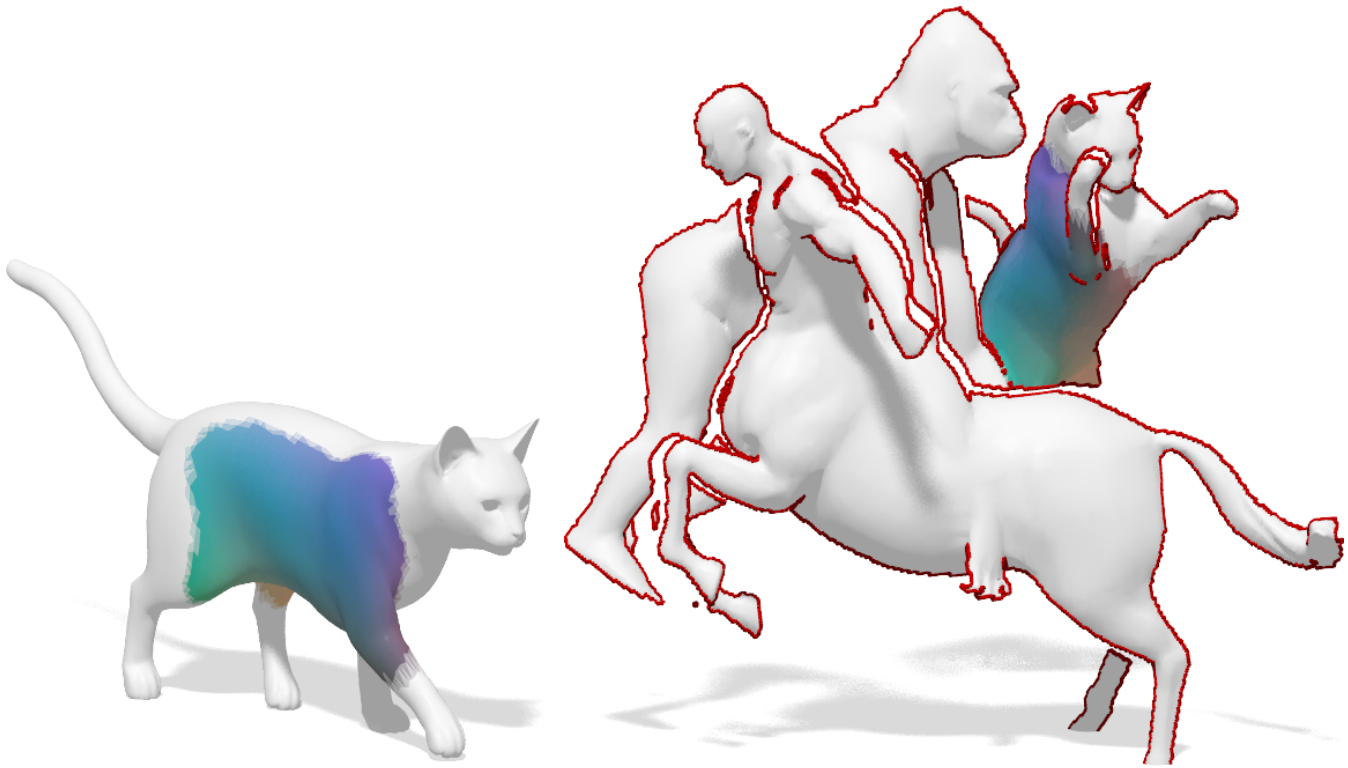
Cluttered matching examples



Correspondence examples (corresponding points are marked with same color)

Cosmo, Rodolà, Masci, Torsello, Bronstein 2016

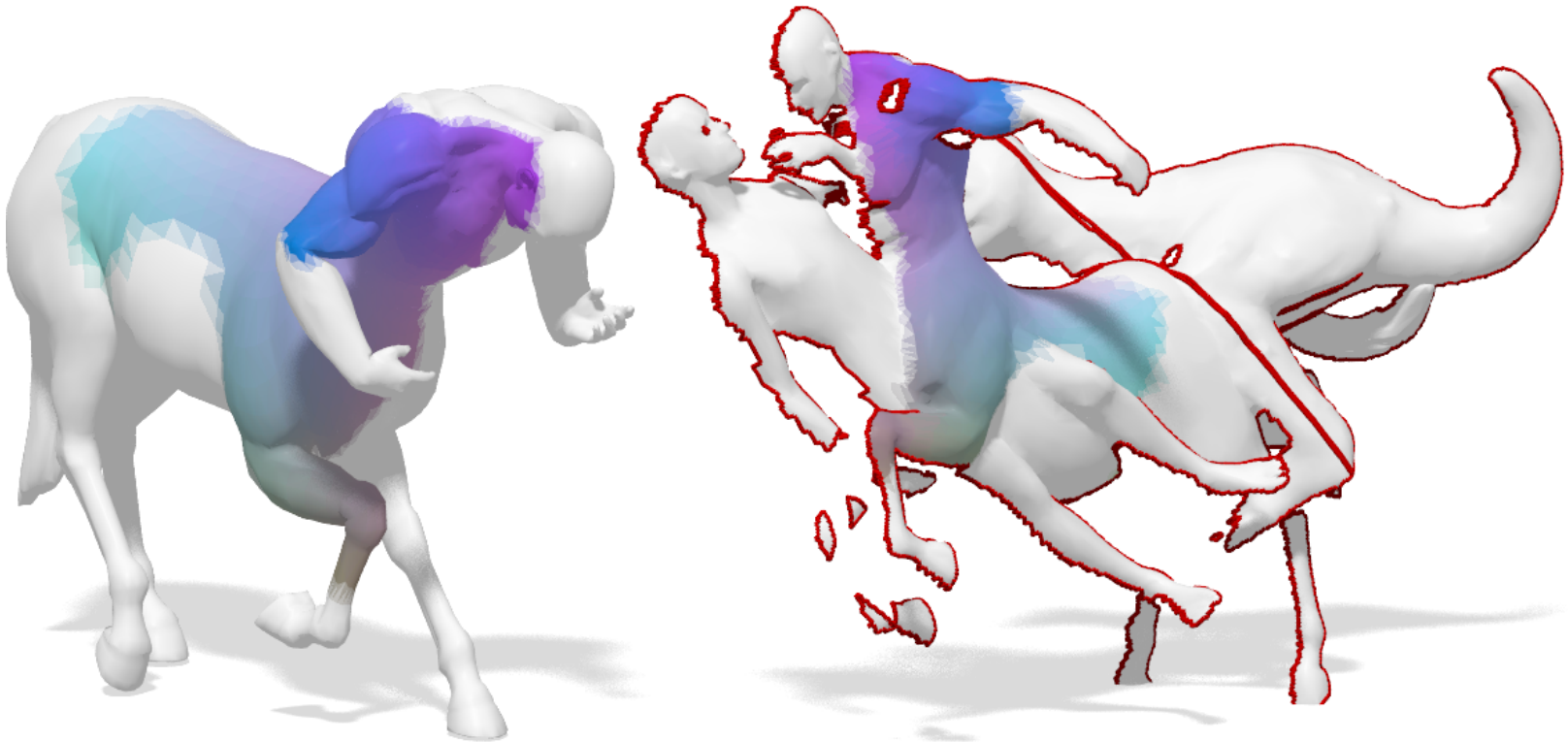
Cluttered matching examples



Correspondence examples (corresponding points are marked with same color)

Cosmo, Rodolà, Masci, Torsello, Bronstein 2016

Cluttered matching examples



Correspondence examples (corresponding points are marked with same color)

Cosmo, Rodolà, Masci, Torsello, Bronstein 2016

Going non-Euclidean

Challenges of geometric deep learning

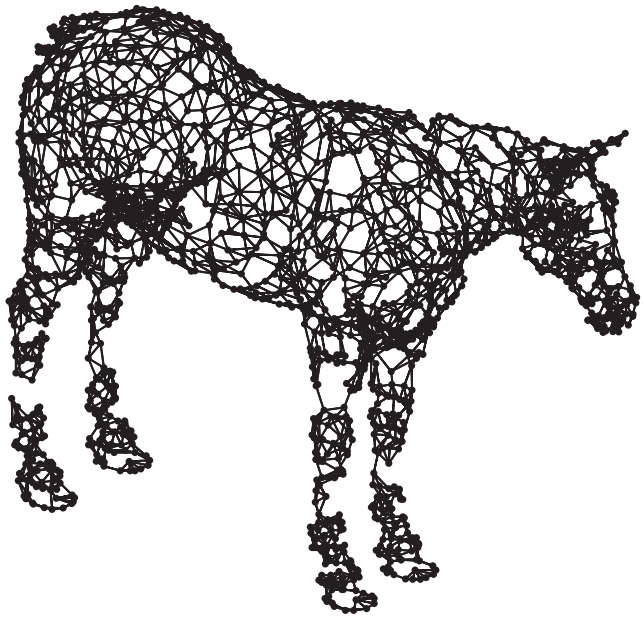
- Non-Euclidean data often have local, self-similar, and hierarchical structure
- How to define convolution?
- How to do pooling?
- How to work fast?

Prototypical non-Euclidean objects

Manifolds

Graphs

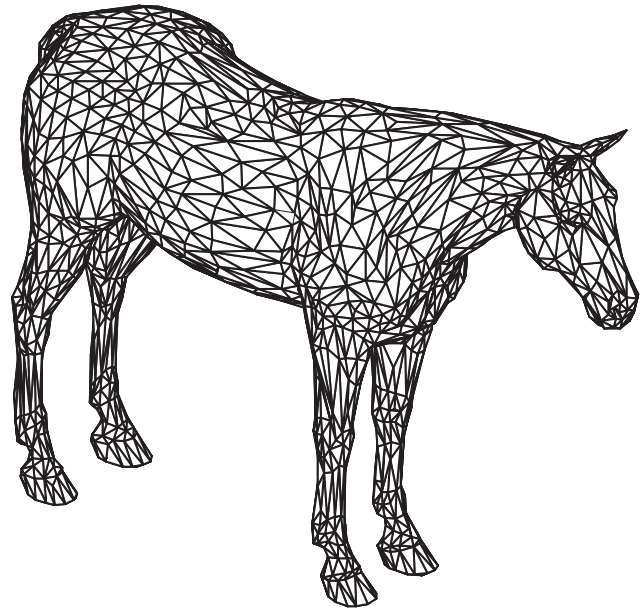
Discrete manifolds



Nearest neighbor graph

Vertices $\mathcal{V} = \{1, \ldots, n\}$

Edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$



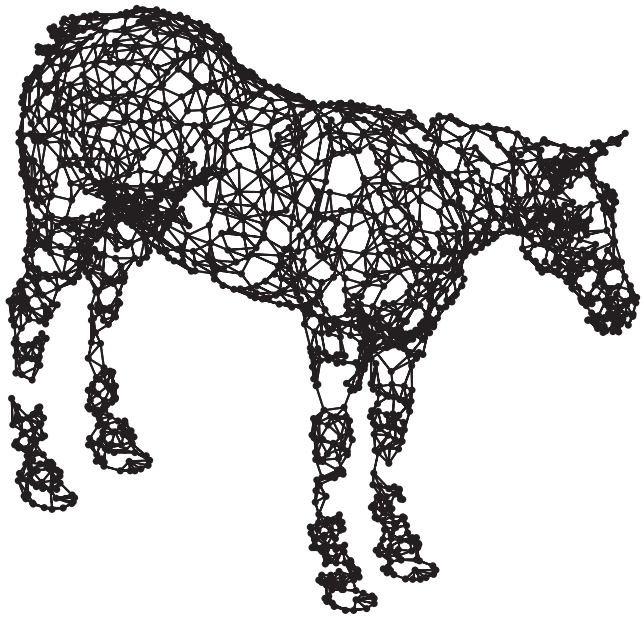
Triangular mesh

Vertices $\mathcal{V} = \{1, \ldots, n\}$

Edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$

Faces $\mathcal{F} = \{(i,j,k) \in \mathcal{V} \times \mathcal{V} \times \mathcal{V} : (i,j), (j,k), (k,i) \in \mathcal{E}\}$

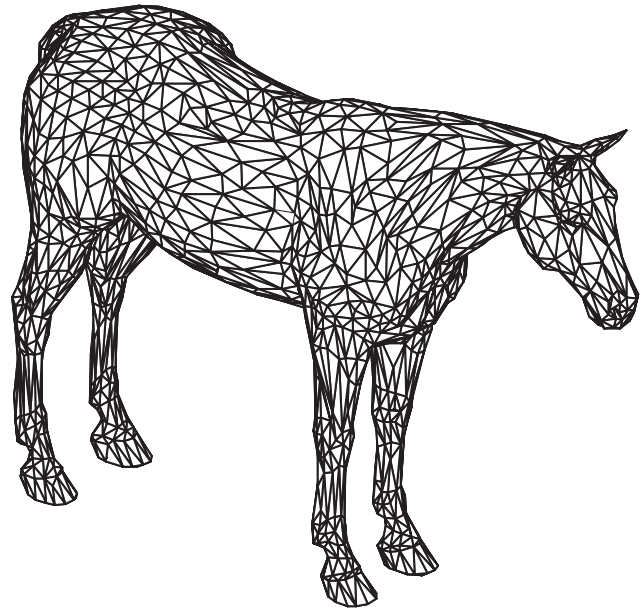
Discrete manifolds



Nearest neighbor graph

Vertices $\mathcal{V} = \{1, \ldots, n\}$

Edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$



Triangular mesh

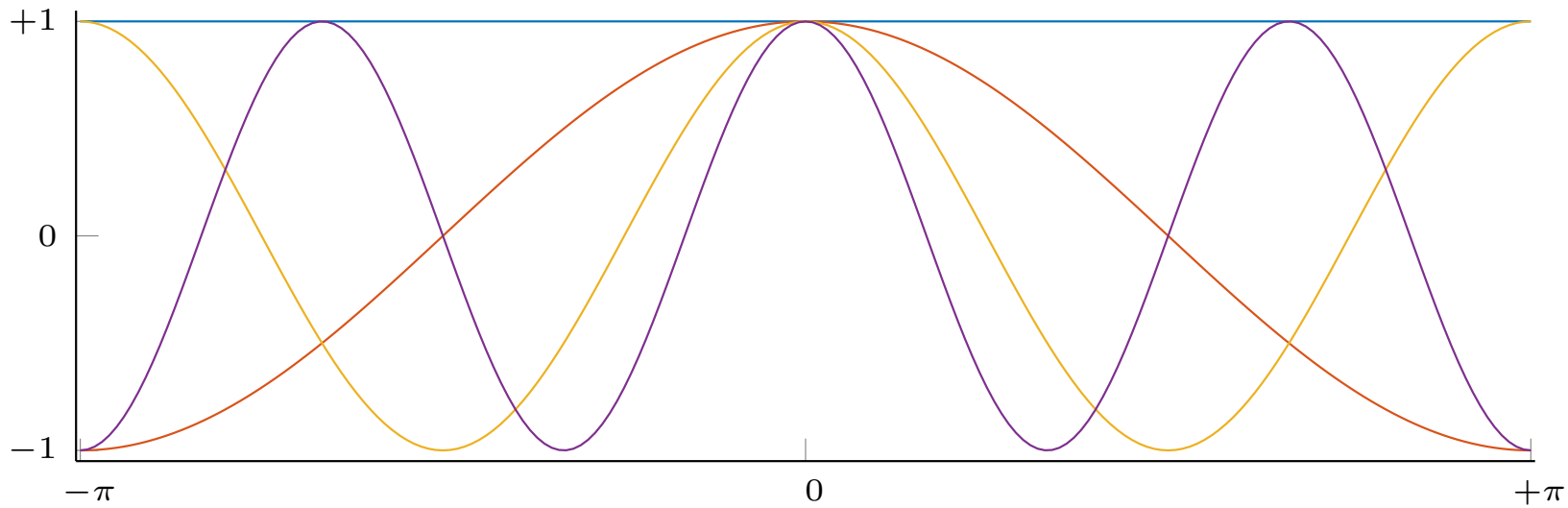
Vertices $\mathcal{V} = \{1, \ldots, n\}$

Edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$

Faces $\mathcal{F} = \{(i,j,k) \in \mathcal{V} \times \mathcal{V} \times \mathcal{V} : (i,j), (j,k), (k,i) \in \mathcal{E}\}$

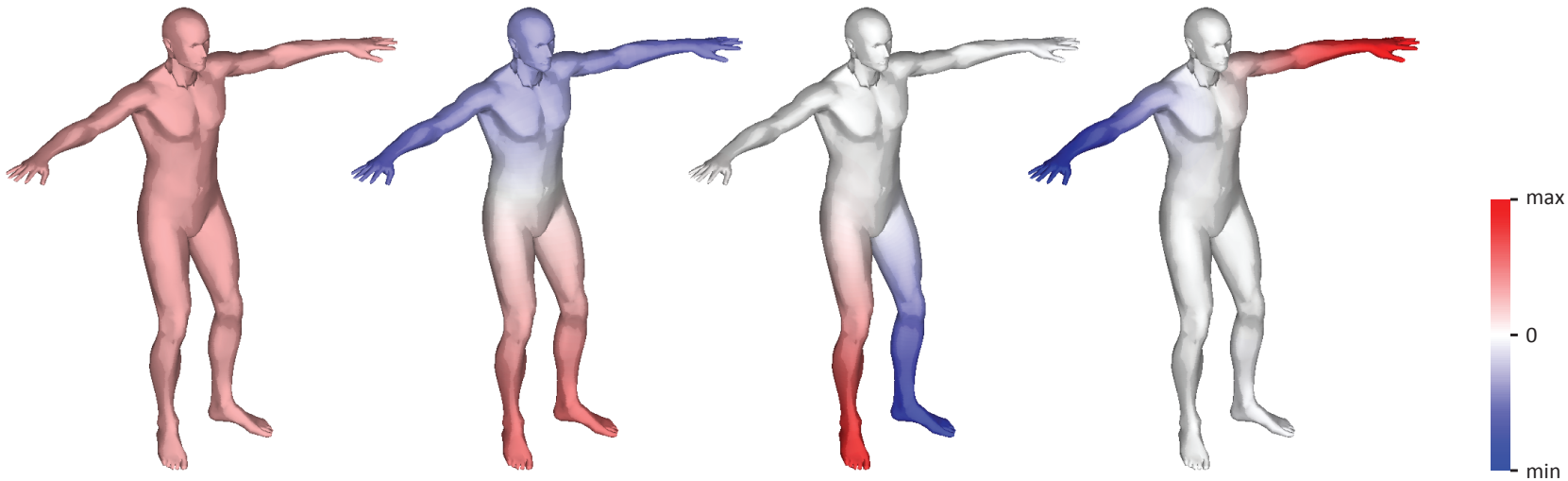
Manifold mesh = each edge is shared by 2 faces + each vertex has 1 loop

Laplacian eigenfunctions: Euclidean



First eigenfunctions of 1D Euclidean Laplacian = standard Fourier basis

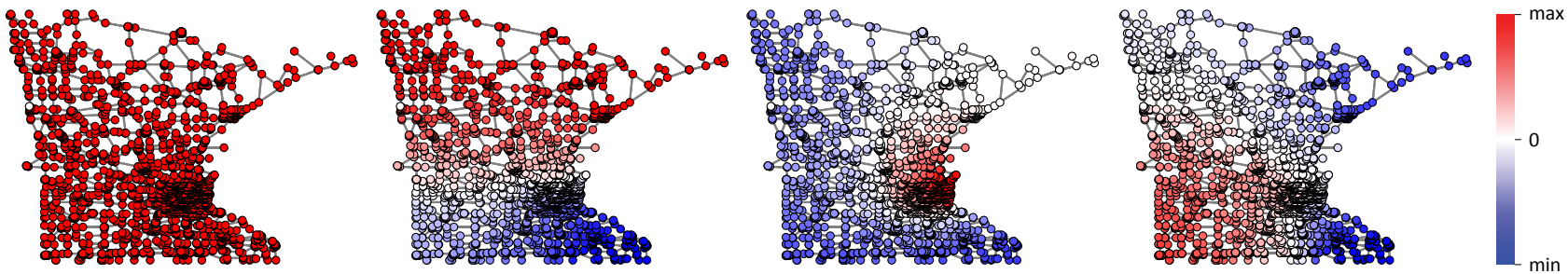
Laplacian eigenfunctions: manifold



ϕ_1 ϕ_2 ϕ_3 ϕ_4

First eigenfunctions of a manifold Laplacian

Laplacian eigenfunctions: graph



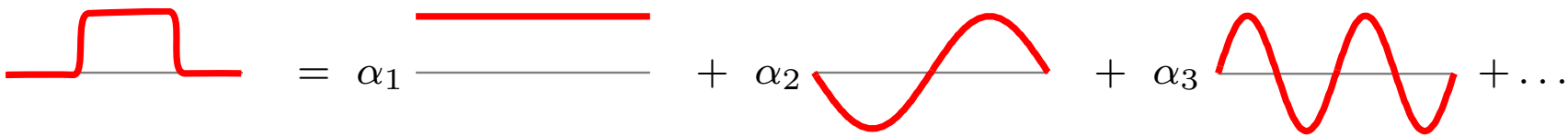
ϕ_1 ϕ_2 ϕ_3 ϕ_4

First eigenfunctions of a graph Laplacian

Fourier analysis: Euclidean space

A function $f : [-\pi, \pi] \to \mathbb{R}$ can be written as Fourier series

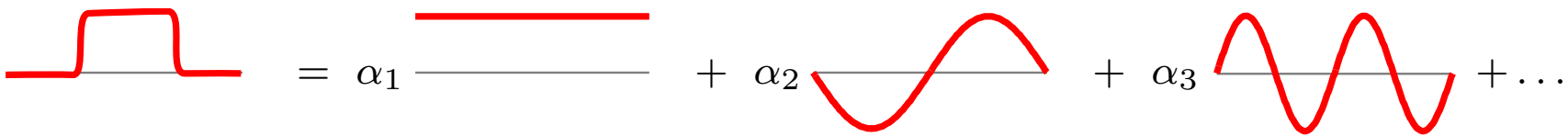
$$f(x) = \sum_{k \geq 0} \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} \, dx' \, e^{ikx}$$



Fourier analysis: Euclidean space

A function $f : [-\pi, \pi] \to \mathbb{R}$ can be written as Fourier series

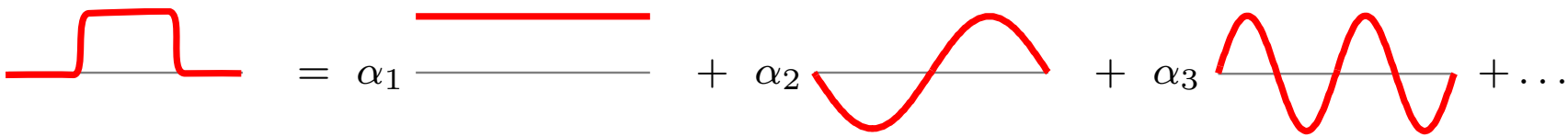
$$f(x) = \sum_{k \geq 0} \underbrace{\frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} \, dx'}_{\hat{f}_k = \langle f, e^{ikx} \rangle_{L^2([-\pi,\pi])}} e^{ikx}$$



Fourier analysis: Euclidean space

A function $f : [-\pi, \pi] \to \mathbb{R}$ can be written as Fourier series

$$f(x) = \sum_{k \geq 0} \underbrace{\frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} dx'}_{\hat{f}_k = \langle f, e^{ikx} \rangle_{L^2([-\pi,\pi])}} e^{ikx}$$

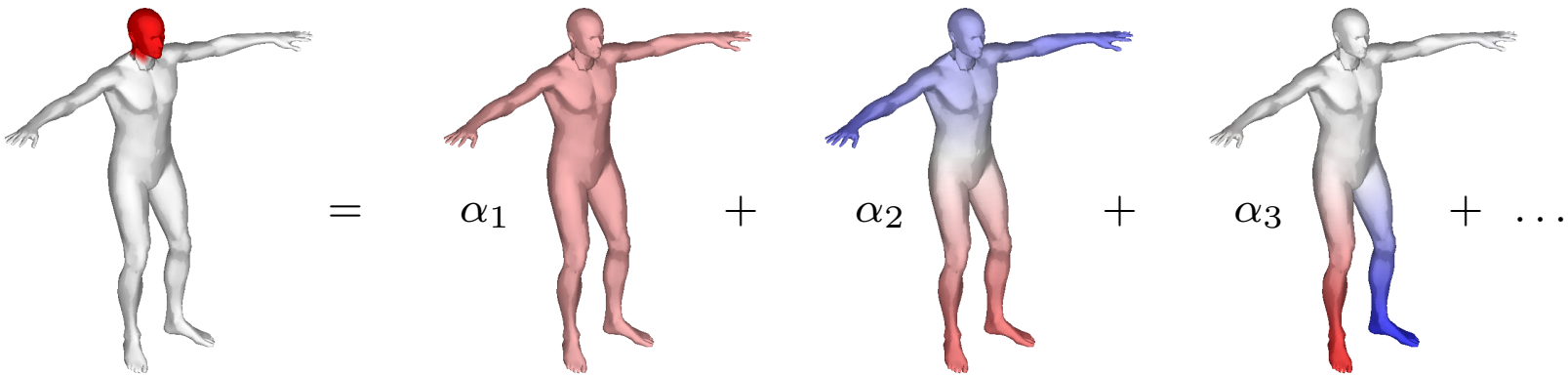


Fourier basis = Laplacian eigenfunctions: $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

Fourier analysis: non-Euclidean space

A function $f : \mathcal{X} \to \mathbb{R}$ can be written as Fourier series

$$f(x) = \sum_{k \geq 1} \underbrace{\int_{\mathcal{X}} f(x')\phi_k(x')dx'}_{\hat{f}_k = \langle f, \phi_k \rangle_{L^2(\mathcal{X})}} \phi_k(x)$$



Fourier basis = Laplacian eigenfunctions: $\Delta\phi_k(x) = \lambda_k\phi_k(x)$

Convolution: Euclidean space

Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

d'Alembert 1754

Convolution: Euclidean space

Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

- Shift-invariance: $f(x - x_0) \star g(x) = (f \star g)(x - x_0)$

d'Alembert 1754

Convolution: Euclidean space

Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

- Shift-invariance: $f(x - x_0) \star g(x) = (f \star g)(x - x_0)$
- Convolution operator commutes with Laplacian: $(\Delta f) \star g = \Delta(f \star g)$

d'Alembert 1754

Convolution: Euclidean space

Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

- Shift-invariance: $f(x - x_0) \star g(x) = (f \star g)(x - x_0)$
- Convolution operator commutes with Laplacian: $(\Delta f) \star g \ = \Delta(f \star g)$
- Convolution theorem: Fourier transform diagonalizes the convolution operator

d'Alembert 1754; Borel 1899

Convolution: Euclidean space

Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

- Shift-invariance: $f(x - x_0) \star g(x) = (f \star g)(x - x_0)$
- Convolution operator commutes with Laplacian: $(\Delta f) \star g \ = \Delta(f \star g)$
- Convolution theorem: Fourier transform diagonalizes the convolution operator $\Rightarrow$ convolution can be computed in the Fourier domain as

$$\widehat{(f \star g)} = \hat{f} \cdot \hat{g}$$

d'Alembert 1754; Borel 1899

Convolution: Euclidean space

Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

- Shift-invariance: $f(x - x_0) \star g(x) = (f \star g)(x - x_0)$
- Convolution operator commutes with Laplacian: $(\Delta f) \star g = \Delta(f \star g)$
- Convolution theorem: Fourier transform diagonalizes the convolution operator $\Rightarrow$ convolution can be computed in the Fourier domain as

$$\widehat{(f \star g)} = \hat{f} \cdot \hat{g}$$

- Efficient computation using FFT

d'Alembert 1754; Borel 1899

Convolution Theorem: discrete case

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} = \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

Convolution Theorem: discrete case

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} = \underbrace{\begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix}}_{\text{circulant matrix}} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

Convolution Theorem: discrete case

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} = \underbrace{\begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix}}_{\text{diagonalized by Fourier basis}} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

Convolution Theorem: discrete case

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} = \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

$$= \mathbf{\Phi} \begin{bmatrix} \hat{g}_1 & & & \\ & \ddots & & \\ & & & \hat{g}_n \end{bmatrix} \mathbf{\Phi}^\top \mathbf{f}$$

Convolution Theorem: discrete case

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} = \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

$$= \mathbf{\Phi} \begin{bmatrix} \hat{g}_1 & & \\ & \ddots & \\ & & \hat{g}_n \end{bmatrix} \begin{bmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_n \end{bmatrix}$$

Convolution Theorem: discrete case

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} = \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

$$= \mathbf{\Phi} \begin{bmatrix} \hat{f}_1 \cdot \hat{g}_1 \\ \vdots \\ \hat{f}_n \cdot \hat{g}_n \end{bmatrix}$$

Spectral convolution

Generalized convolution of $f, g \in L^2(\mathcal{X})$ can be defined by analogy

$$f \star g \quad = \quad \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \, \phi_k$$

Spectral convolution

Generalized convolution of $f, g \in L^2(\mathcal{X})$ can be defined by analogy

$$f \star g \quad = \quad \sum_{k \geq 1} \underbrace{\langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})}}_{\text{product in the Fourier domain}} \phi_k$$

Spectral convolution

Generalized convolution of $f, g \in L^2(\mathcal{X})$ can be defined by analogy

$$f \star g = \underbrace{\sum_{k \geq 1} \underbrace{\langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})}}_{\text{product in the Fourier domain}} \phi_k}_{\text{inverse Fourier transform}}$$

Spectral convolution

Generalized convolution of $f, g \in L^2(\mathcal{X})$ can be defined by analogy

$$f \star g \quad = \quad \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \, \phi_k$$

In matrix-vector notation

$$\mathbf{f} \star \mathbf{g} = \mathbf{\Phi} \left(\mathbf{\Phi}^\top \mathbf{g} \right) \circ \left(\mathbf{\Phi}^\top \mathbf{f} \right)$$

Spectral convolution

Generalized convolution of $f, g \in L^2(\mathcal{X})$ can be defined by analogy

$$f \star g \quad = \quad \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \, \phi_k$$

In matrix-vector notation

$$\mathbf{f} \star \mathbf{g} = \mathbf{\Phi} \, \mathrm{diag}(\hat{g}_1, \ldots, \hat{g}_n) \mathbf{\Phi}^\top \mathbf{f}$$

Spectral convolution

Generalized convolution of $f, g \in L^2(\mathcal{X})$ can be defined by analogy

$$f \star g \quad = \quad \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \, \phi_k$$

In matrix-vector notation

$$\mathbf{f} \star \mathbf{g} = \underbrace{\mathbf{\Phi} \, \mathrm{diag}(\hat{g}_1, \ldots, \hat{g}_n) \mathbf{\Phi}^\top}_{\mathbf{G}} \mathbf{f}$$

Spectral convolution

Generalized convolution of $f, g \in L^2(\mathcal{X})$ can be defined by analogy

$$f \star g \quad = \quad \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \, \phi_k$$

In matrix-vector notation

$$\mathbf{f} \star \mathbf{g} = \underbrace{\mathbf{\Phi} \, \mathrm{diag}(\hat{g}_1, \ldots, \hat{g}_n) \mathbf{\Phi}^\top}_{\mathbf{G}} \mathbf{f}$$

- Not shift-invariant! ($\mathbf{G}$ has no circulant structure)

Spectral convolution

Generalized convolution of $f, g \in L^2(\mathcal{X})$ can be defined by analogy

$$f \star g \quad = \quad \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \, \phi_k$$

In matrix-vector notation

$$\mathbf{f} \star \mathbf{g} = \underbrace{\mathbf{\Phi} \operatorname{diag}(\hat{g}_1, \ldots, \hat{g}_n) \mathbf{\Phi}^\top}_{\mathbf{G}} \mathbf{f}$$

- Not shift-invariant! ($\mathbf{G}$ has no circulant structure)
- Commutes with Laplacian: $\mathbf{G}\mathbf{\Delta}\mathbf{f} = \mathbf{\Delta}\mathbf{G}\mathbf{f}$

Spectral convolution

Generalized convolution of $f, g \in L^2(\mathcal{X})$ can be defined by analogy

$$f \star g \quad = \quad \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \, \phi_k$$

In matrix-vector notation

$$\mathbf{f} \star \mathbf{g} = \underbrace{\mathbf{\Phi} \operatorname{diag}(\hat{g}_1, \ldots, \hat{g}_n) \mathbf{\Phi}^\top}_{\mathbf{G}} \mathbf{f}$$

- Not shift-invariant! ($\mathbf{G}$ has no circulant structure)
- Commutes with Laplacian: $\mathbf{G}\mathbf{\Delta}\mathbf{f} = \mathbf{\Delta}\mathbf{G}\mathbf{f}$
- Filter coefficients depend on basis $\phi_1, \ldots, \phi_n$

Spectral domain deep learning methods

Spectral graph CNN

Convolutional layer expressed in the spectral domain

$$\mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{\Phi}\mathbf{W}_{l,l'}\mathbf{\Phi}^\top \mathbf{f}_{l'}\right) \quad \begin{matrix} l = 1, \ldots, q \\ l' = 1, \ldots, p \end{matrix}$$

where $\mathbf{W}_{l,l} = n \times n$ diagonal matrix of filter coefficients

Bruna et al. 2014

Spectral graph CNN

Convolutional layer expressed in the spectral domain

$$\mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{\Phi}\mathbf{W}_{l,l'}\mathbf{\Phi}^\top \mathbf{f}_{l'}\right) \quad \begin{matrix} l = 1, \ldots, q \\ l' = 1, \ldots, p \end{matrix}$$

where $\mathbf{W}_{l,l} = n \times n$ diagonal matrix of filter coefficients

☹ Filters are basis-dependent $\Rightarrow$ does not generalize across graphs!

Bruna et al. 2014

Spectral graph CNN

Convolutional layer expressed in the spectral domain

$$\mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{\Phi}\mathbf{W}_{l,l'}\mathbf{\Phi}^\top \mathbf{f}_{l'}\right) \quad \begin{matrix} l = 1, \ldots, q \\ l' = 1, \ldots, p \end{matrix}$$

where $\mathbf{W}_{l,l} = n \times n$ diagonal matrix of filter coefficients

☹ Filters are basis-dependent $\Rightarrow$ does not generalize across graphs!

☹ $\mathcal{O}(n)$ parameters per layer

Bruna et al. 2014

Spectral graph CNN

Convolutional layer expressed in the spectral domain

$$\mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{\Phi}\mathbf{W}_{l,l'}\mathbf{\Phi}^\top \mathbf{f}_{l'}\right) \quad \begin{matrix} l = 1,\dots,q \\ l' = 1,\dots,p \end{matrix}$$

where $\mathbf{W}_{l,l} = n \times n$ diagonal matrix of filter coefficients

- ☹ Filters are basis-dependent $\Rightarrow$ does not generalize across graphs!
- ☹ $\mathcal{O}(n)$ parameters per layer
- ☹ $\mathcal{O}(n^2)$ computation of forward and inverse Fourier transforms $\mathbf{\Phi}^\top, \mathbf{\Phi}$ (no FFT on graphs)

Bruna et al. 2014

Spectral graph CNN

Convolutional layer expressed in the spectral domain

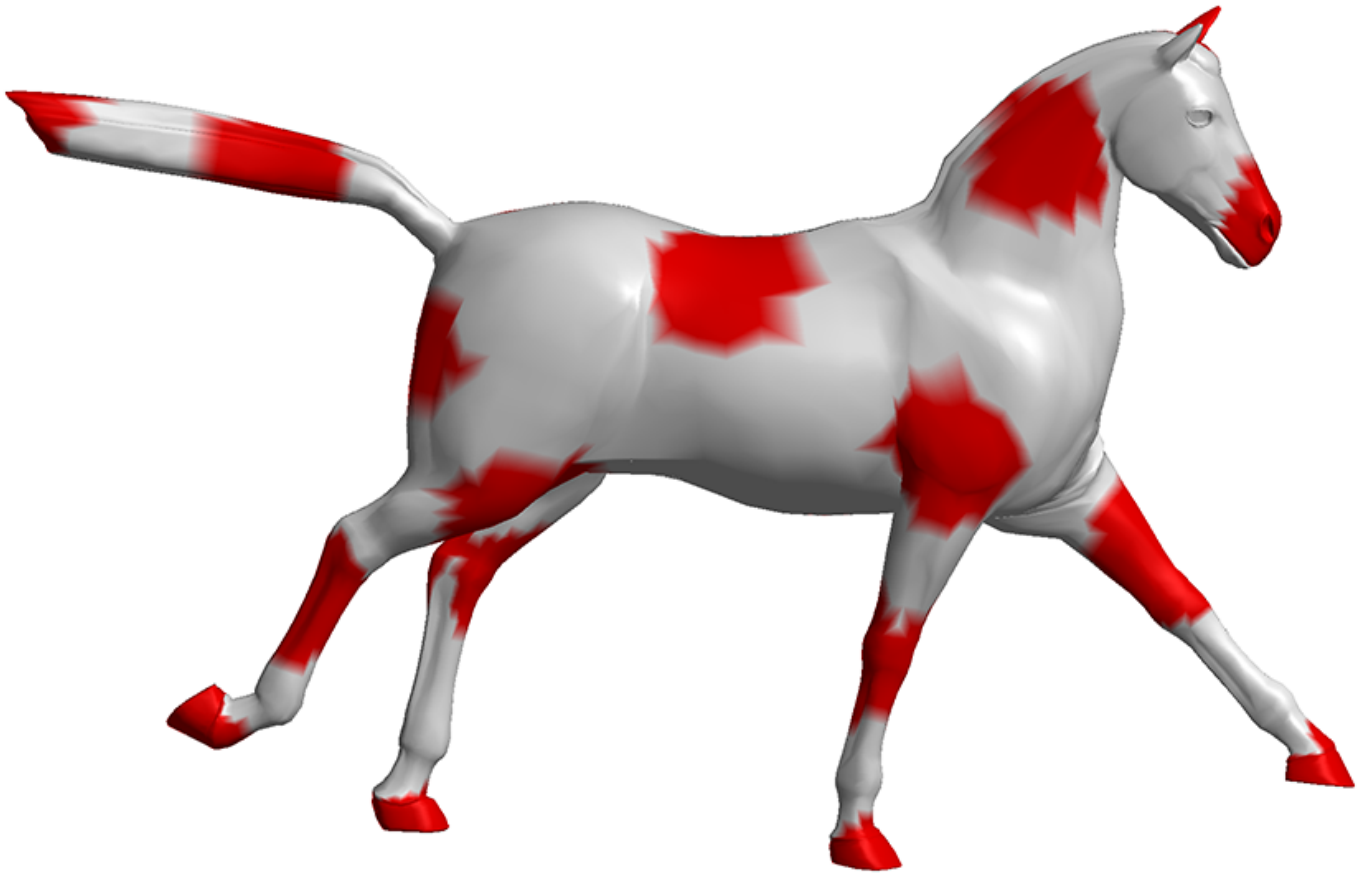
$$\mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{\Phi}\mathbf{W}_{l,l'}\mathbf{\Phi}^\top \mathbf{f}_{l'}\right) \quad \begin{matrix} l = 1, \ldots, q \\ l' = 1, \ldots, p \end{matrix}$$

where $\mathbf{W}_{l,l} = n \times n$ diagonal matrix of filter coefficients

- ☹ Filters are basis-dependent $\Rightarrow$ does not generalize across graphs!
- ☹ $\mathcal{O}(n)$ parameters per layer
- ☹ $\mathcal{O}(n^2)$ computation of forward and inverse Fourier transforms $\mathbf{\Phi}^\top, \mathbf{\Phi}$ (no FFT on graphs)
- ☹ No guarantee of spatial localization of filters

Bruna et al. 2014

Basis dependence



Function $\mathbf{f}$

Basis dependence

'Edge detecting' spectral filter $\mathbf{\Phi W \Phi}^\top \mathbf{f}$

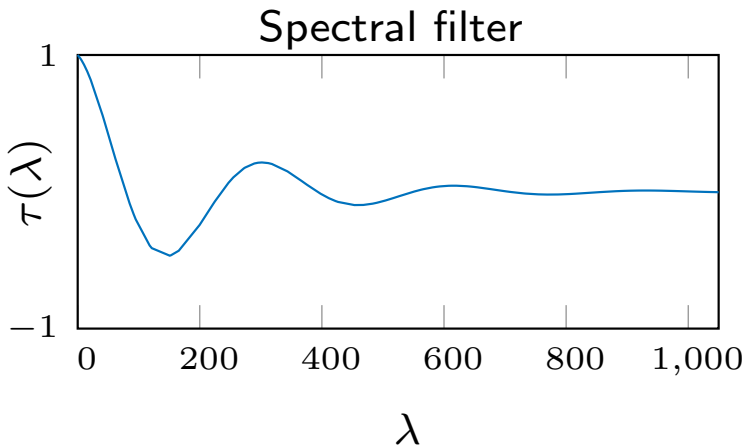
Basis dependence

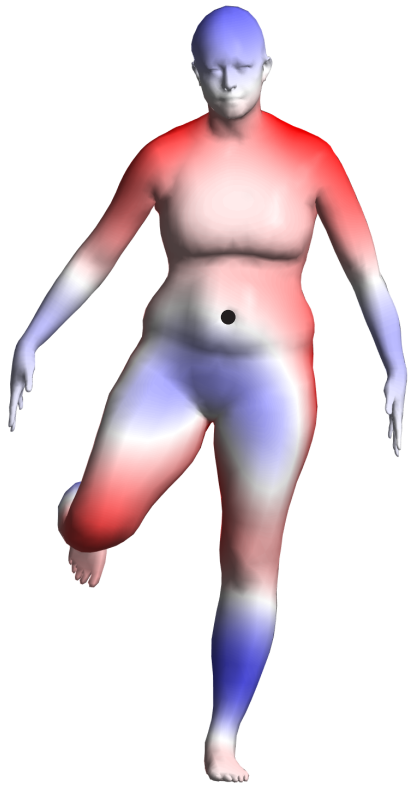
Same spectral filter, different basis $\mathbf{\Psi W \Psi}^\top \mathbf{f}$

Basis dependence

High-frequency Laplacian eigenvector ϕ_{50}

Filtering in different bases





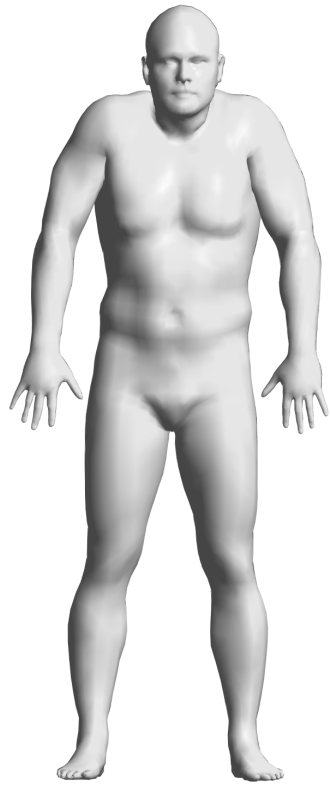
$\boldsymbol{\Phi}_1 \tau(\boldsymbol{\Lambda}_1) \boldsymbol{\Phi}_1^\top \boldsymbol{\delta}_0$

$\boldsymbol{\Phi}_2 \tau(\boldsymbol{\Lambda}_2) \boldsymbol{\Phi}_2^\top \boldsymbol{\delta}_0$

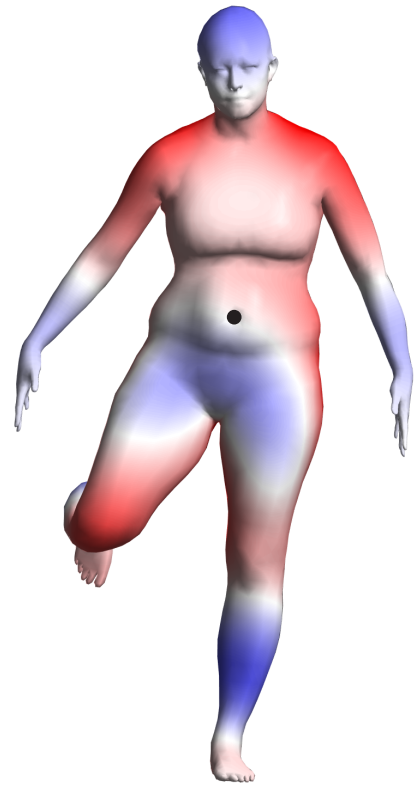
Apply spectral filter $\tau(\lambda)$ in different bases $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$
$\Rightarrow$ different results!

Filtering in different bases

$\boldsymbol{\Phi}_1 \tau(\boldsymbol{\Lambda}_1) \boldsymbol{\Phi}_1^\top \boldsymbol{\delta}_0$



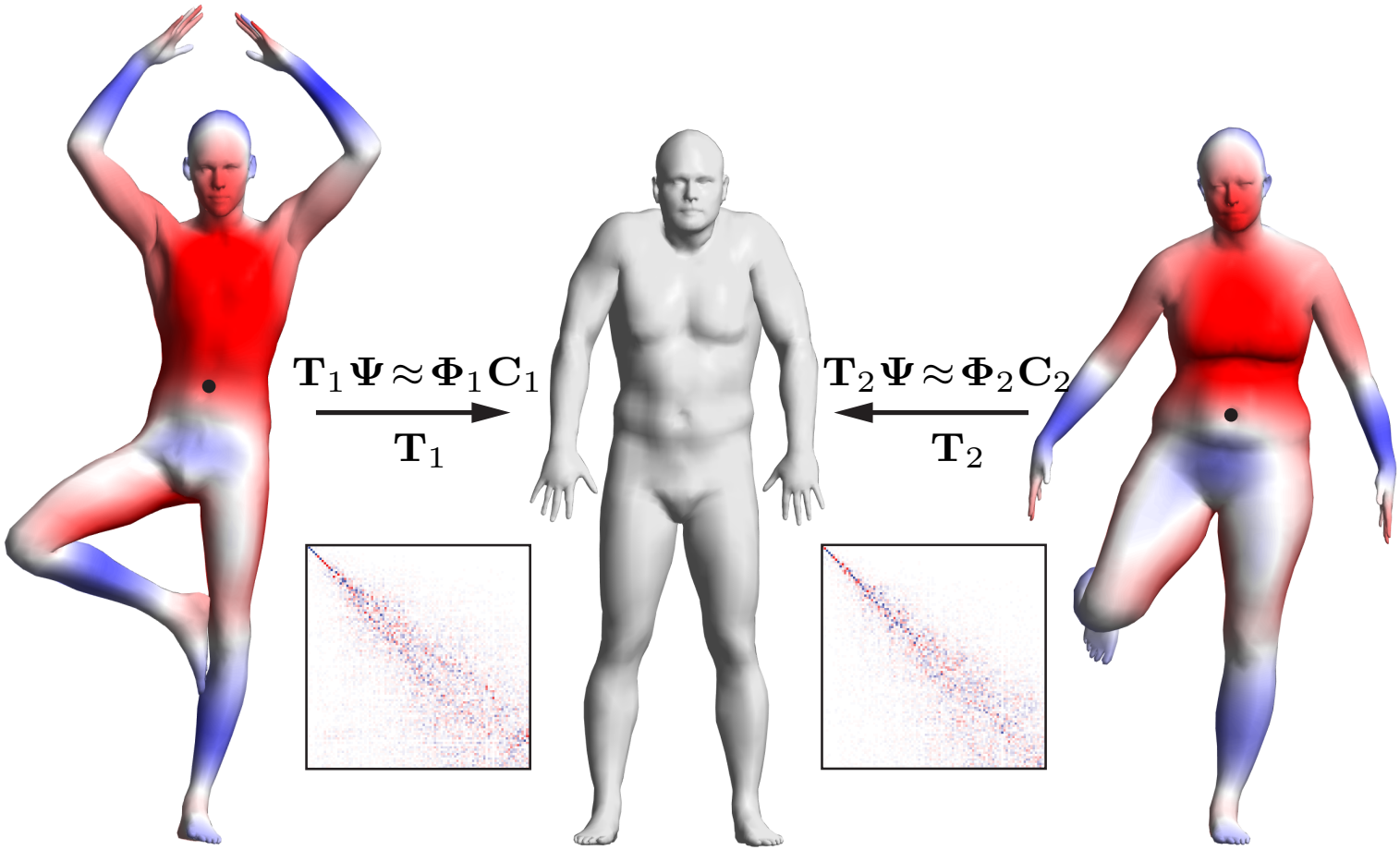
Canonical shape
with basis $\boldsymbol{\Psi}$



$\boldsymbol{\Phi}_2 \tau(\boldsymbol{\Lambda}_2) \boldsymbol{\Phi}_2^\top \boldsymbol{\delta}_0$

Apply spectral filter $\tau(\lambda)$ in different bases $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$
$\Rightarrow$ different results!

Filtering in synchronized bases



Apply spectral filter $\tau(\lambda)$ in synchronized bases $\mathbf{\Phi}_1\mathbf{C}_1$ and $\mathbf{\Phi}_2\mathbf{C}_2$
$\Rightarrow$ similar results!

Yi et al. 2017

Localization and Smoothness

In the Euclidean setting (by Parseval's identity)

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

Bruna et al. 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

In the Euclidean setting (by Parseval's identity)

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

$\Rightarrow$ Localization in space = smoothness in frequency domain

Bruna et al. 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

In the Euclidean setting (by Parseval's identity)

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

$\Rightarrow$ Localization in space $=$ smoothness in frequency domain

Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$

Bruna et al. 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

In the Euclidean setting (by Parseval's identity)

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

$\Rightarrow$ Localization in space $=$ smoothness in frequency domain

Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$

Application of the filter

$$\tau(\mathbf{\Delta})\mathbf{f} = \mathbf{\Phi}\tau(\mathbf{\Lambda})\mathbf{\Phi}^\top \mathbf{f}$$

Bruna et al. 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

In the Euclidean setting (by Parseval's identity)

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

$\Rightarrow$ Localization in space = smoothness in frequency domain

Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$

Application of the filter

$$\tau(\mathbf{\Delta})\mathbf{f} = \mathbf{\Phi} \begin{pmatrix} \tau(\lambda_1) & & \\ & \ddots & \\ & & \tau(\lambda_n) \end{pmatrix} \mathbf{\Phi}^\top \mathbf{f}$$

Bruna et al. 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

In the Euclidean setting (by Parseval's identity)

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

$\Rightarrow$ Localization in space = smoothness in frequency domain

Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$

Application of the parametric filter with learnable parameters $\boldsymbol{\alpha}$

$$\tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \boldsymbol{\Phi} \begin{pmatrix} \tau_{\boldsymbol{\alpha}}(\lambda_1) & & \\ & \ddots & \\ & & \tau_{\boldsymbol{\alpha}}(\lambda_n) \end{pmatrix} \boldsymbol{\Phi}^\top \mathbf{f}$$

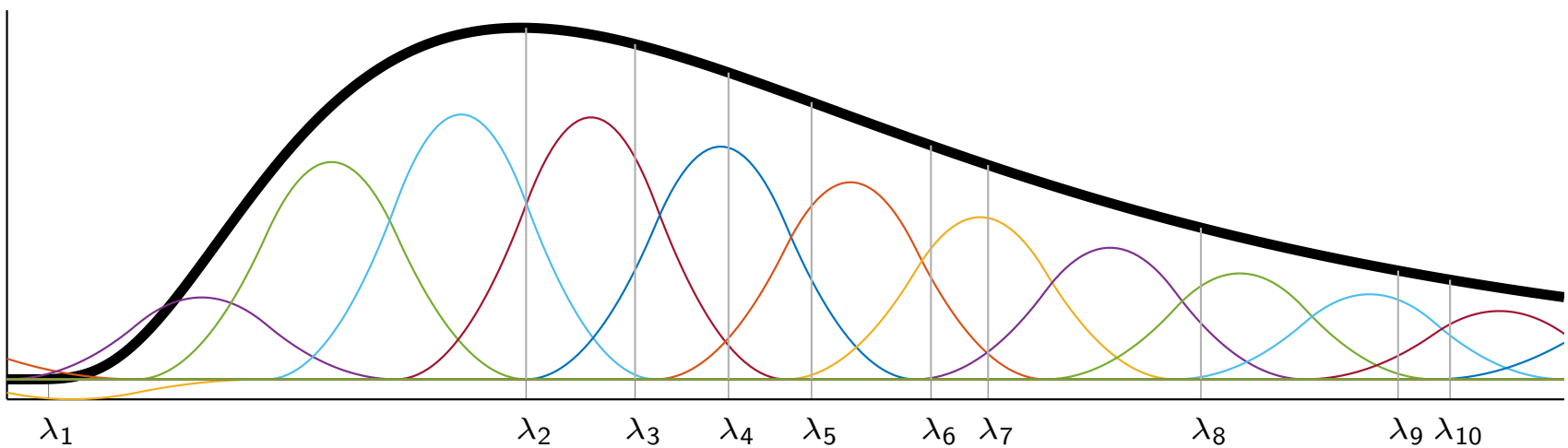
Bruna et al. 2014; Henaff, Bruna, LeCun 2015

Spectral graph CNN with smooth spectral filters

Represent spectral transfer function as a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=1}^{r} \alpha_j \beta_j(\lambda)$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)^\top$ is the vector of filter parameters



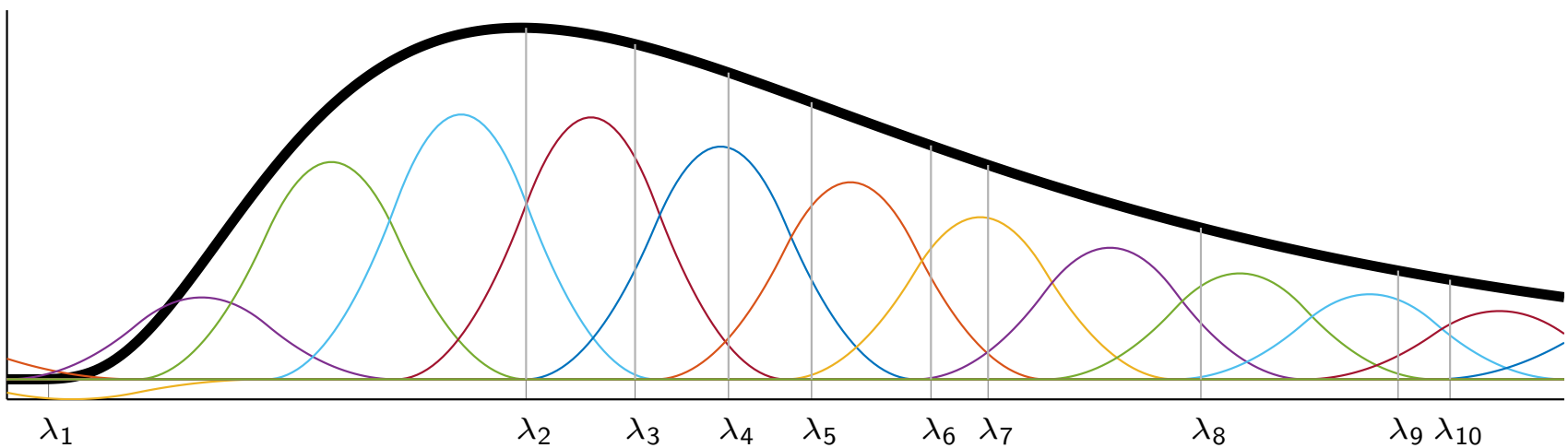
(Litman, Bronstein, 2014); Henaff, Bruna, LeCun 2015

Spectral graph CNN with smooth spectral filters

Represent spectral transfer function as a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$

$$\tau_{\boldsymbol{\alpha}}(\lambda_k) = \sum_{j=1}^{r} \alpha_j \beta_j(\lambda_k)$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)^\top$ is the vector of filter parameters



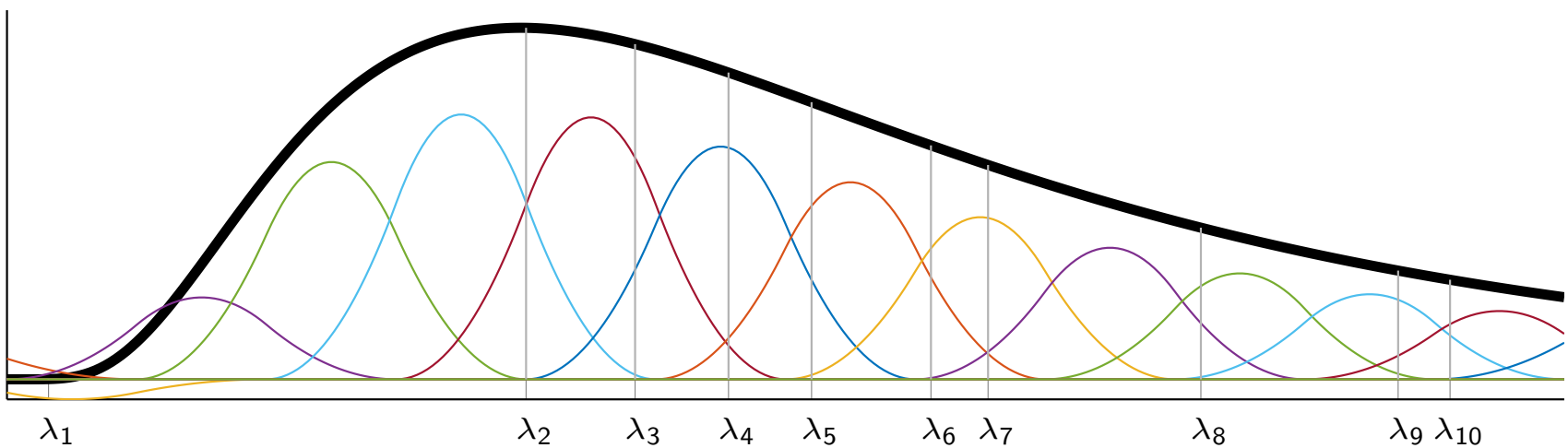
(Litman, Bronstein, 2014); Henaff, Bruna, LeCun 2015

Spectral graph CNN with smooth spectral filters

Represent spectral transfer function as a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$

$$\tau_{\boldsymbol{\alpha}}(\lambda_k) = \sum_{j=1}^{r} \alpha_j \beta_j(\lambda_k) = (\mathbf{B}\boldsymbol{\alpha})_k$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)^\top$ is the vector of filter parameters



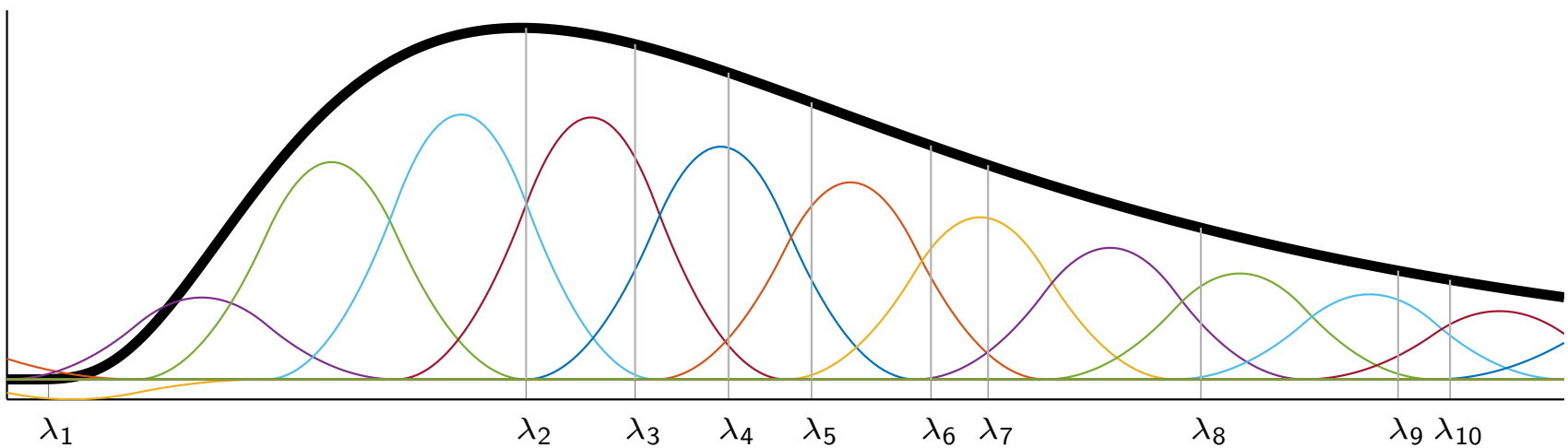
(Litman, Bronstein, 2014); Henaff, Bruna, LeCun 2015

Spectral graph CNN with smooth spectral filters

Represent spectral transfer function as a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$

$$\mathbf{W} = \mathrm{Diag}(\mathbf{B}\boldsymbol{\alpha})$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)^\top$ is the vector of filter parameters



(Litman, Bronstein, 2014); Henaff, Bruna, LeCun 2015

Spectral graph CNN with smooth spectral filters

Represent spectral transfer function as a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$

$$\mathbf{W} = \mathrm{Diag}(\mathbf{B}\boldsymbol{\alpha})$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)^\top$ is the vector of filter parameters

☺ $\mathcal{O}(1)$ parameters per layer

(Litman, Bronstein, 2014); Henaff, Bruna, LeCun 2015

Spectral graph CNN with smooth spectral filters

Represent spectral transfer function as a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$

$$\mathbf{W} = \mathrm{Diag}(\mathbf{B}\boldsymbol{\alpha})$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)^\top$ is the vector of filter parameters

☺ $\mathcal{O}(1)$ parameters per layer

☺ Fast-decaying filters in space

(Litman, Bronstein, 2014); Henaff, Bruna, LeCun 2015

Spectral graph CNN with smooth spectral filters

Represent spectral transfer function as a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$

$$\mathbf{W} = \mathrm{Diag}(\mathbf{B}\boldsymbol{\alpha})$$

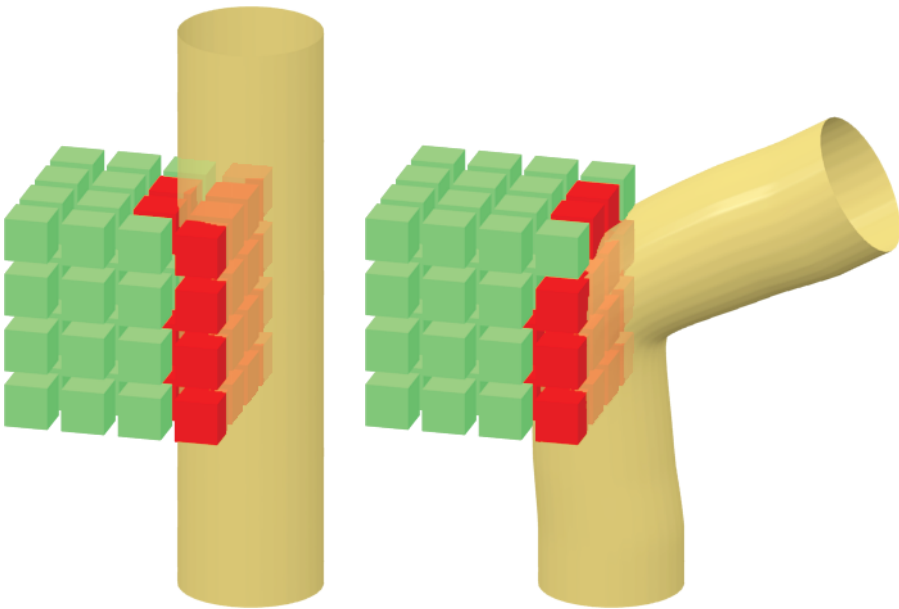
where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)^\top$ is the vector of filter parameters

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Fast-decaying filters in space
- ☹ $\mathcal{O}(n^2)$ computation of forward and inverse Fourier transforms $\boldsymbol{\Phi}^\top, \boldsymbol{\Phi}$ (no FFT on graphs)

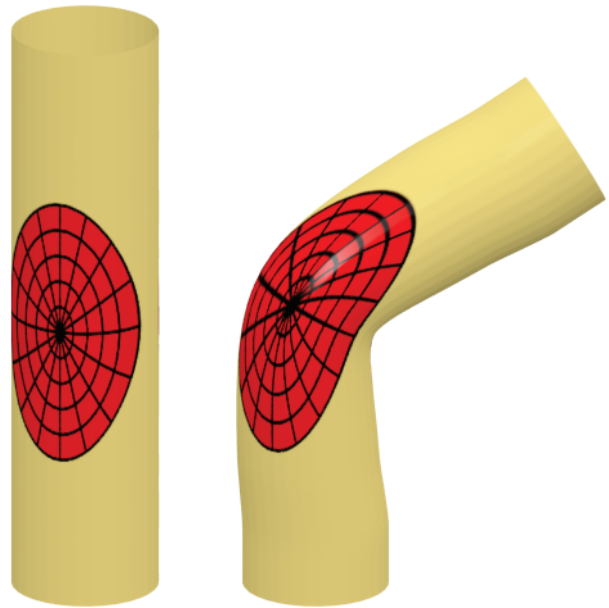
(Litman, Bronstein, 2014); Henaff, Bruna, LeCun 2015

Spatial domain deep learning methods

Extrinsic vs Intrinsic



Extrinsic



Intrinsic

Extrinsic vs Intrinsic

Extrinsic **Intrinsic**

What is convolution on manifolds?

Convolution

Euclidean

Spatial domain

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(\xi) g(x - \xi) d\xi$$

Spectral domain

$$\widehat{(f \star g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

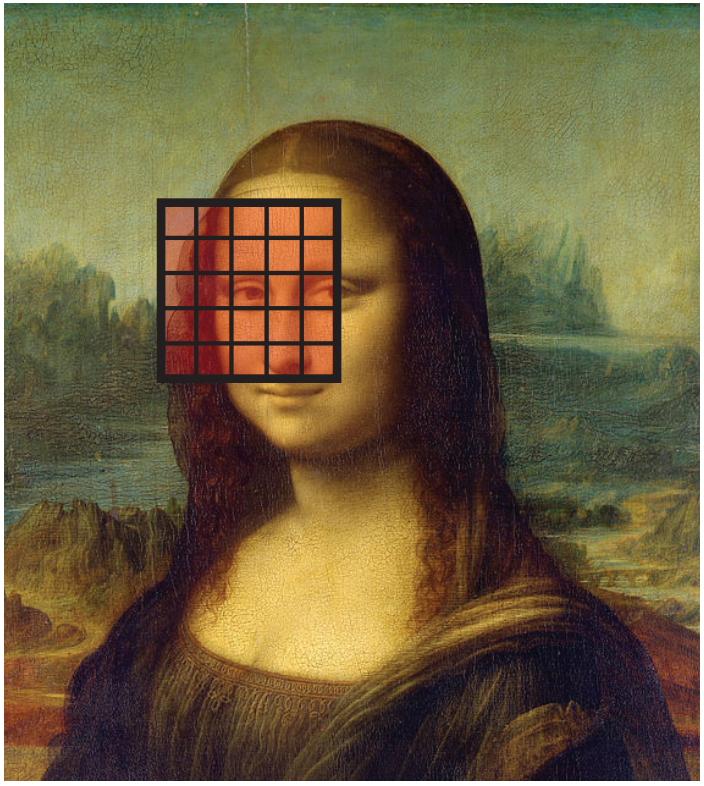
'Convolution Theorem'

Non-Euclidean

?

$$\widehat{(f \star g)}_k = \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})}$$

Spatial convolution



Euclidean

Non-Euclidean

Boscaini, Masci, Bronstein, Vandergheynst 2015

Convolution in the spatial domain

A	B	C	D	E	A	B	C	D	E
F	G	H	I	J	F	G	H	I	J
K	L	M	N	O	K	L	M	N	O
P	R	S	T	U	P	R	S	T	U
V	W	X	Y	Z	V	W	X	Y	Z
A	B	C	D	E	A	B	C	D	E
F	G	H	I	J	F	G	H	I	J
K	L	M	N	O	K	L	M	N	O
P	R	S	T	U	P	R	S	T	U
V	W	X	Y	Z	V	W	X	Y	Z

Euclidean

Non-Euclidean

- No canonical global system of coordinates

Convolution in the spatial domain

A	B	C	D	E	A	B	C	D	E
F	G	H	I	J	F	G	H	I	J
K	L	M	N	O	K	L	M	N	O
P	R	S	T	U	P	R	S	T	U
V	W	X	Y	Z	V	W	X	Y	Z
A	B	C	D	E	A	B	C	D	E
F	G	H	I	J	F	G	H	I	J
K	L	M	N	O	K	L	M	N	O
P	R	S	T	U	P	R	S	T	U
V	W	X	Y	Z	V	W	X	Y	Z

Euclidean

Non-Euclidean

- No canonical global system of coordinates
- No grid structure (no regular memory access)

Convolution in the spatial domain

A	B	C	D	E	A	B	C	D	E
F	G	H	I	J	F	G	H	I	J
K	L	M	N	O	K	L	M	N	O
P	R	S	T	U	P	R	S	T	U
V	W	X	Y	Z	V	W	X	Y	Z
A	B	C	D	E	A	B	C	D	E
F	G	H	I	J	F	G	H	I	J
K	L	M	N	O	K	L	M	N	O
P	R	S	T	U	P	R	S	T	U
V	W	X	Y	Z	V	W	X	Y	Z

Euclidean

Non-Euclidean

- No canonical global system of coordinates
- No grid structure (no regular memory access)
- No shift-invariance (patch is position-dependent)

Patch operator

$$(f \star g)(x) = \int (D(x)f)(\mathbf{u}) \times g(\mathbf{u})\, d\mathbf{u}$$

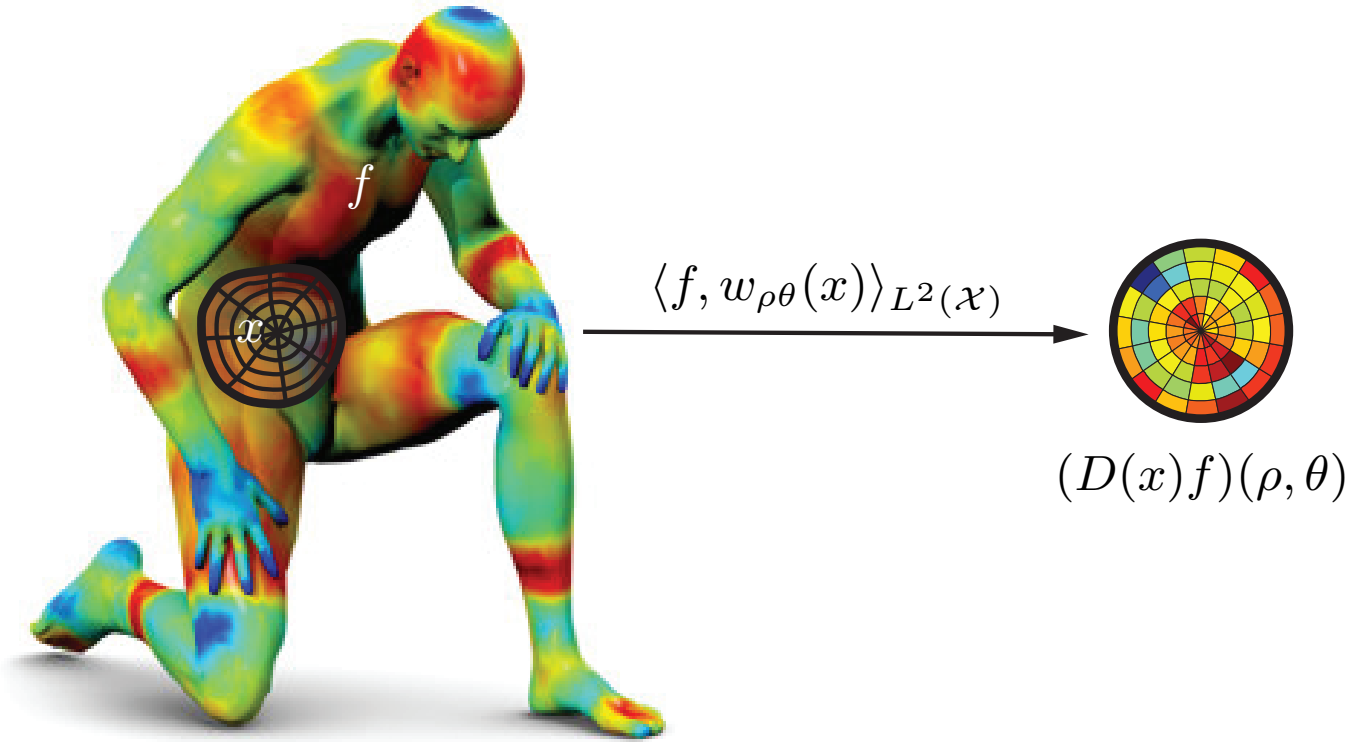
Masci et al. 2015; Boscaini, Masci, Rodolà, Bronstein 2016

Convolution

Euclidean	**Non-Euclidean**
Spatial domain	
$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x-x')dx'$	$(f \star g)(x) = \iint (D(x)f)(\rho,\theta)g(\rho,\theta)d\rho d\theta$
Spectral domain	
$\widehat{(f \star g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$	$\widehat{(f \star g)}_k = \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})}$
'Convolution Theorem'	

Geodesic CNNs

Patch operator



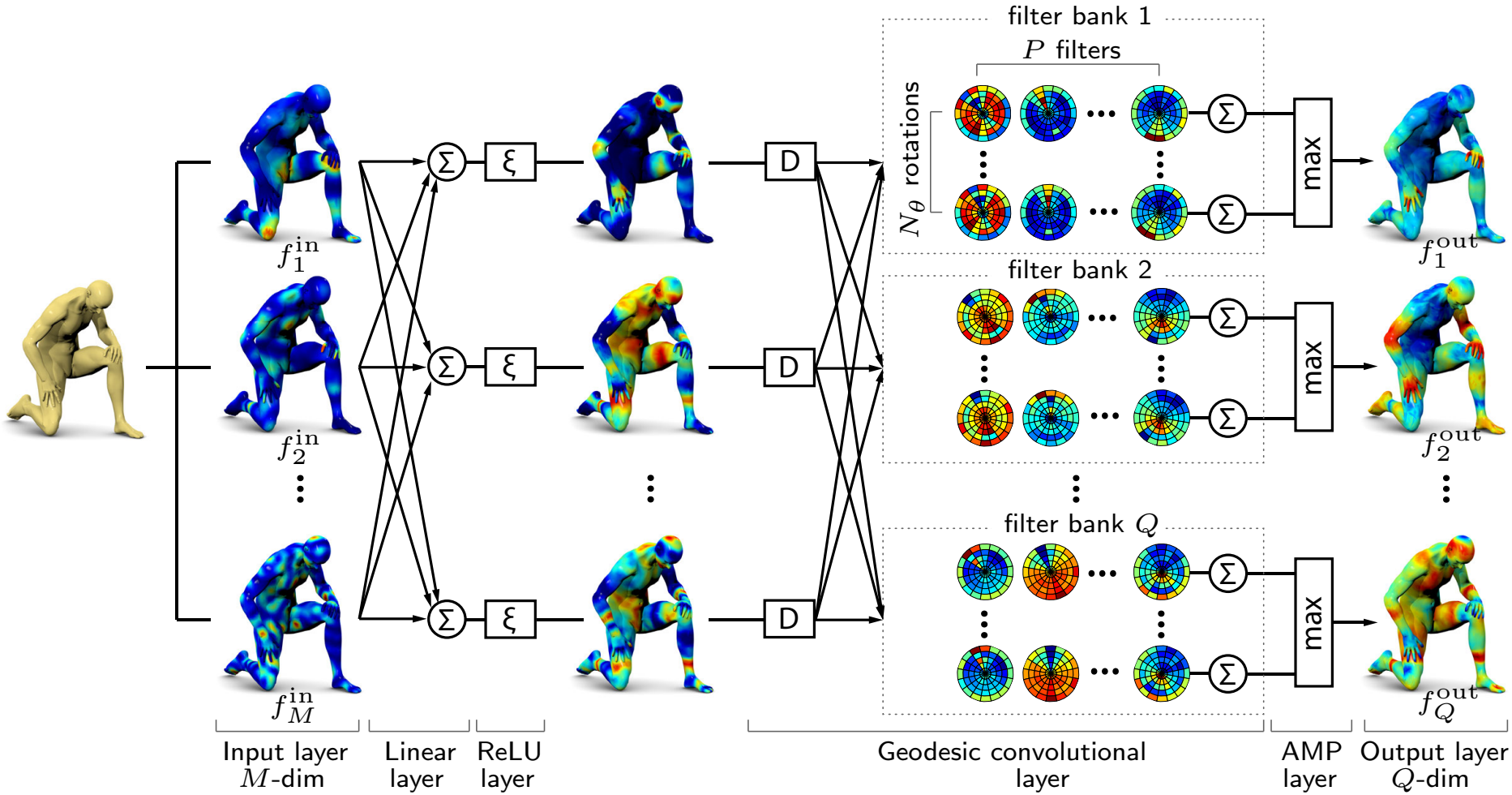
Boscaini, Masci, Bronstein, Vandergheynst 2015

Patch operator

$$(f \star g)(x) = \iint \underbrace{(D(x)f)(\rho,\theta)}_{\text{image}} \cdot \underbrace{g(\rho,\theta)}_{\text{image}} \, d\rho \, d\theta$$

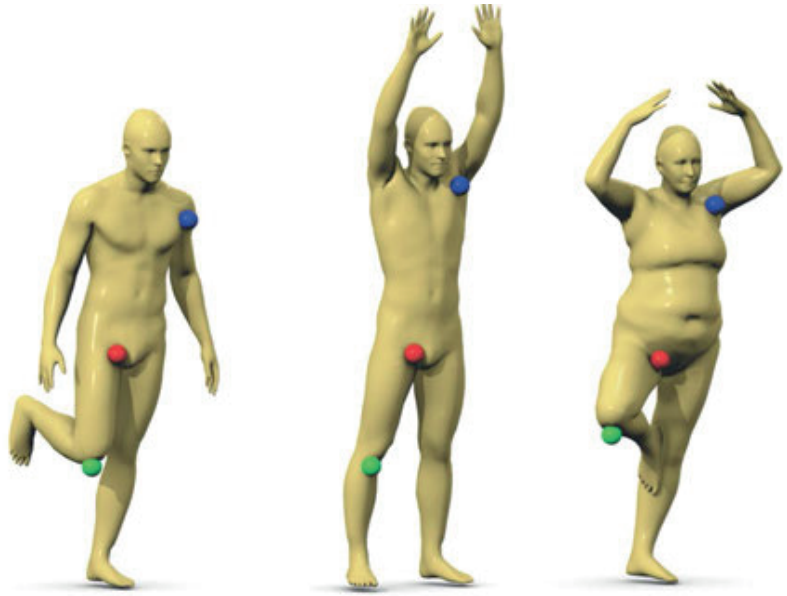
Boscaini, Masci, Bronstein, Vandergheynst 2015

Toy Geodesic CNN (GCNN) architecture



Masci, Boscaini, Bronstein, Vandergheynst 2015

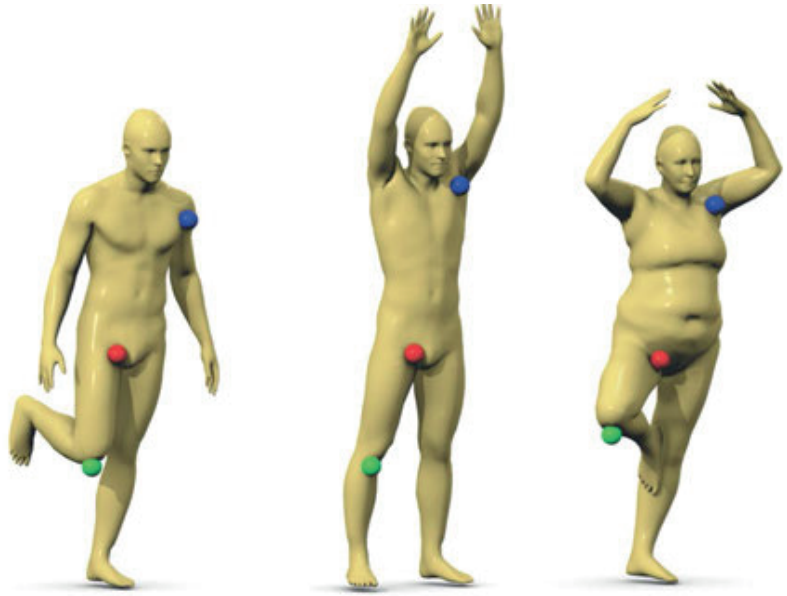
Learning local descriptors with GCNN



- As similar as possible on positives $\mathcal{T}^+$
- As dissimilar as possible on negatives $\mathcal{T}^-$

Masci, Boscaini, Bronstein, Vandergheynst 2015

Learning local descriptors with GCNN



- As similar as possible on positives $\mathcal{T}^+$
- As dissimilar as possible on negatives $\mathcal{T}^-$
- Minimize siamese loss w.r.t. GCNN parameters $\boldsymbol{\Theta}$

$$\begin{aligned}\ell(\boldsymbol{\Theta}) &= (1-\gamma) \sum_{(x,x^+)\in\mathcal{T}^+} \|\mathbf{f}_{\boldsymbol{\Theta}}(x) - \mathbf{f}_{\boldsymbol{\Theta}}(x^+)\| \\ &+ \gamma \sum_{(x,x^-)\in\mathcal{T}^-} (\mu - \|\mathbf{f}_{\boldsymbol{\Theta}}(x) - \mathbf{f}_{\boldsymbol{\Theta}}(x^-)\|)_+\end{aligned}$$

Masci, Boscaini, Bronstein, Vandergheynst 2015

Descriptor robustness

HKS descriptor distance

Descriptors: Sun, Ovsjanikov, Guibas 2009 (HKS); Aubry, Schlickewei, Cremers 2011 (WKS); Masci, Boscaini, Bronstein, Vandergheynst 2015 (GCNN); data: Bronstein et al. 2008 (TOSCA); Anguelov et al. 2005 (SCAPE); Bogo et al. 2014 (FAUST)

Descriptor robustness

WKS descriptor distance

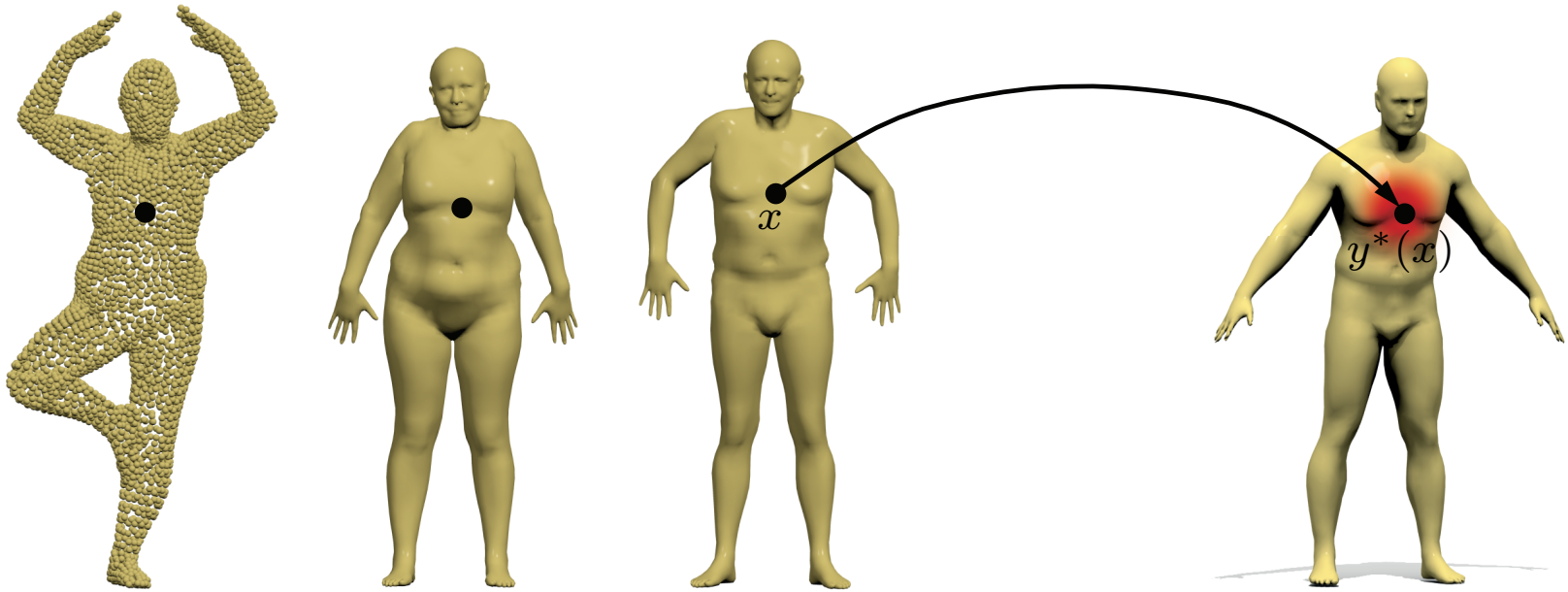
Descriptors: Sun, Ovsjanikov, Guibas 2009 (HKS); Aubry, Schlickewei, Cremers 2011 (WKS); Masci, Boscaini, Bronstein, Vandergheynst 2015 (GCNN); data: Bronstein et al. 2008 (TOSCA); Anguelov et al. 2005 (SCAPE); Bogo et al. 2014 (FAUST)

Descriptor robustness

GCNN descriptor distance

Descriptors: Sun, Ovsjanikov, Guibas 2009 (HKS); Aubry, Schlickewei, Cremers 2011 (WKS); Masci, Boscaini, Bronstein, Vandergheynst 2015 (GCNN); data: Bronstein et al. 2008 (TOSCA); Anguelov et al. 2005 (SCAPE); Bogo et al. 2014 (FAUST)

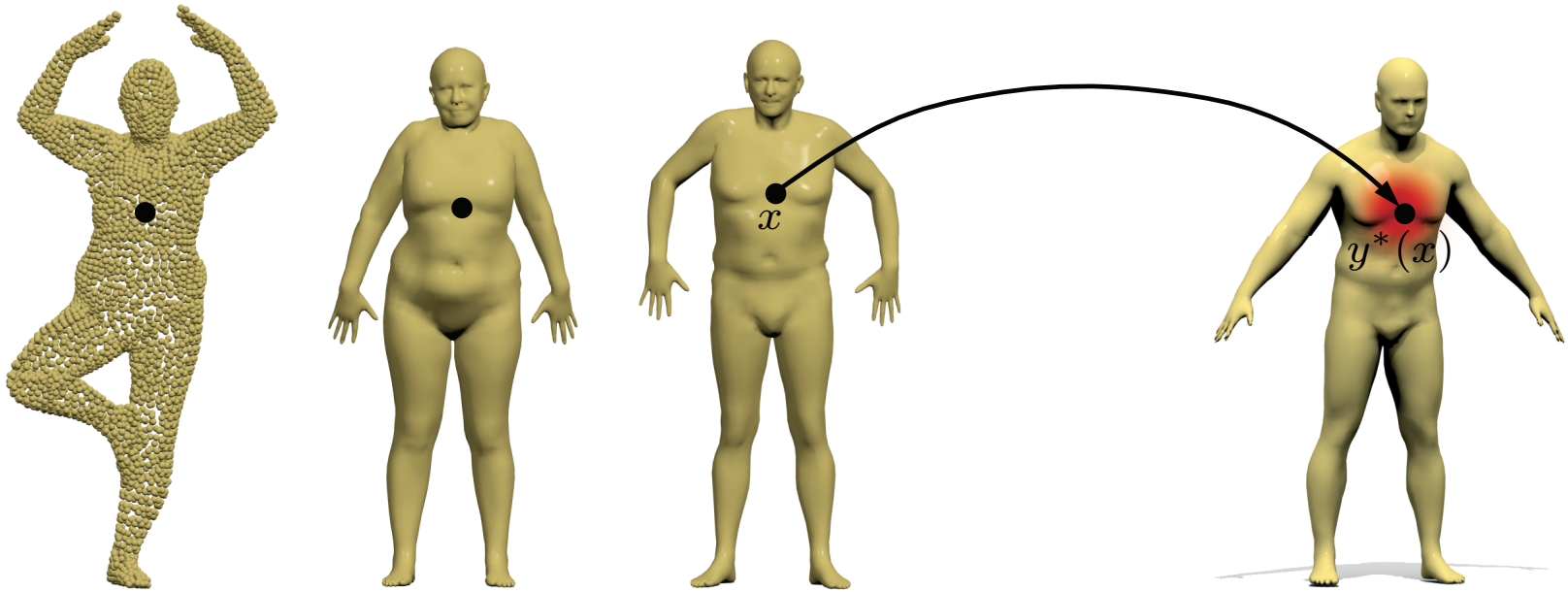
Learning shape correspondence with GCNN



- Correspondence = labeling problem

Rodolà et al. 2014; Masci, Boscaini, Bronstein, Vandergheynst 2015

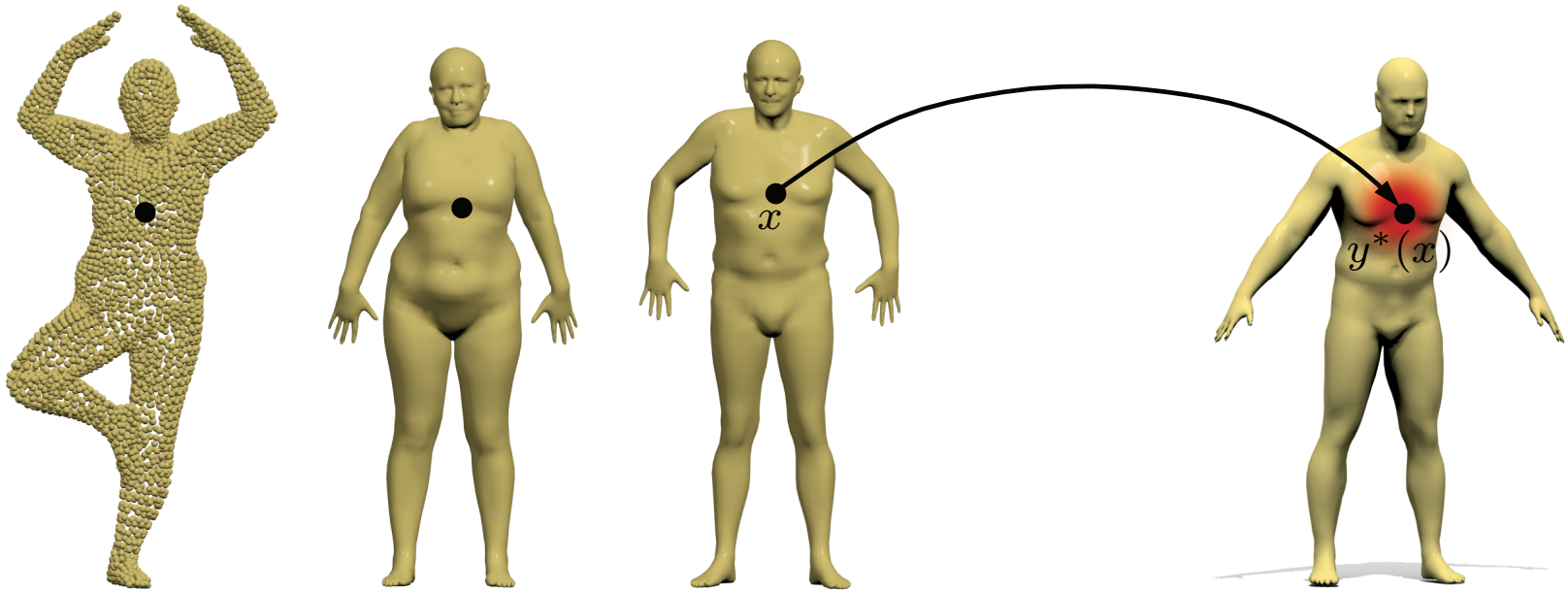
Learning shape correspondence with GCNN



- Correspondence = labeling problem
- GCNN output $\mathbf{f}_{\boldsymbol{\Theta}}(x)$ = probability distribution on reference $\mathcal{Y}$

Rodolà et al. 2014; Masci, Boscaini, Bronstein, Vandergheynst 2015

Learning shape correspondence with GCNN

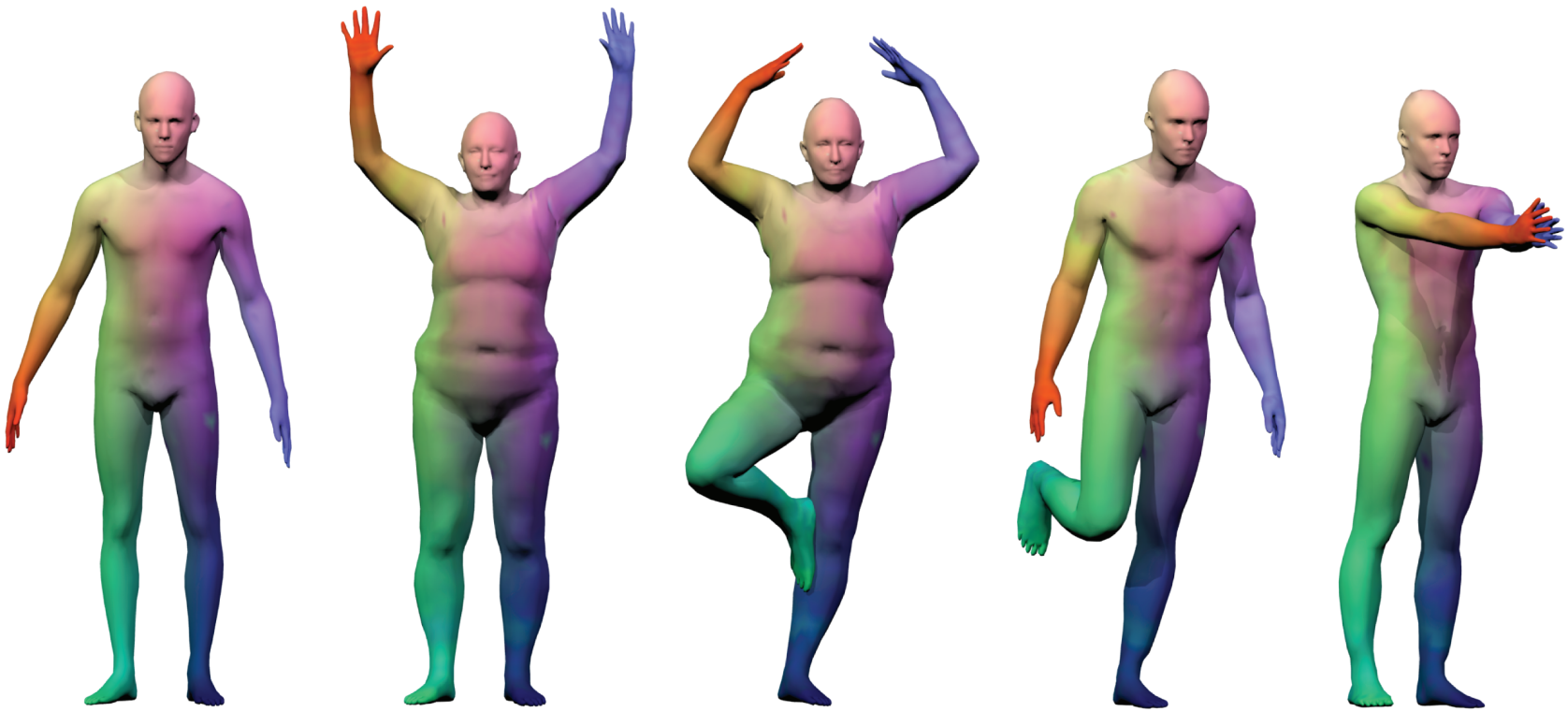


- Correspondence = labeling problem
- GCNN output $\mathbf{f}_{\boldsymbol{\Theta}}(x)$ = probability distribution on reference $\mathcal{Y}$
- Minimize logistic regression cost w.r.t. GCNN parameters $\boldsymbol{\Theta}$

$$\ell(\boldsymbol{\Theta}) = - \sum_{(x,y^*(x))\in\mathcal{T}} \langle \delta_{y^*(x)}, \log \mathbf{f}_{\boldsymbol{\Theta}}(x) \rangle_{L^2(\mathcal{Y})}$$

Rodolà et al. 2014; Masci, Boscaini, Bronstein, Vandergheynst 2015

Correspondence example



Correspondence found using GCNN
(similar colors encode corresponding points)

Masci, Boscaini, Bronstein, Vandergheynst 2015

Anisotropic CNNs

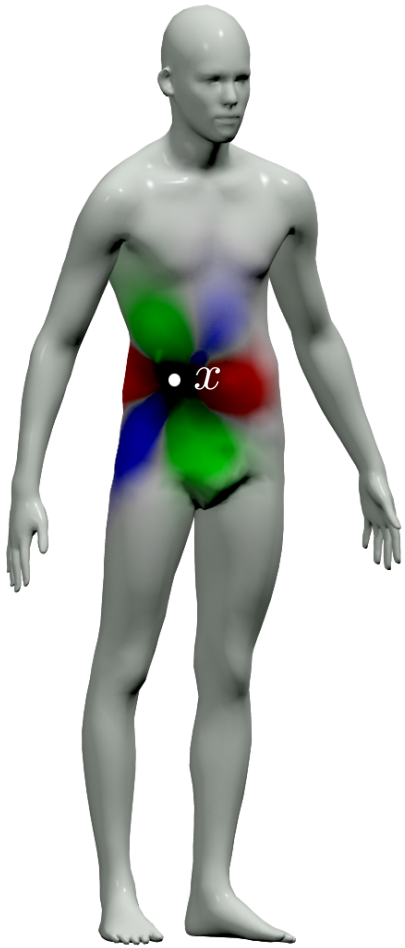
Anisotropic heat kernels

$$h_{\alpha\theta t}(x, x') = \sum_{k \geq 0} e^{-t\lambda_{\alpha\theta k}} \phi_{\alpha\theta k}(x)\phi_{\alpha\theta k}(x')$$

Scale t Orientation θ Elongation α

Boscaini, Masci, Rodolà, Bronstein, Cremers 2016

Anisotropic CNN (ACNN)



Given a function $f \in L^2(\mathcal{X})$, the patch operator

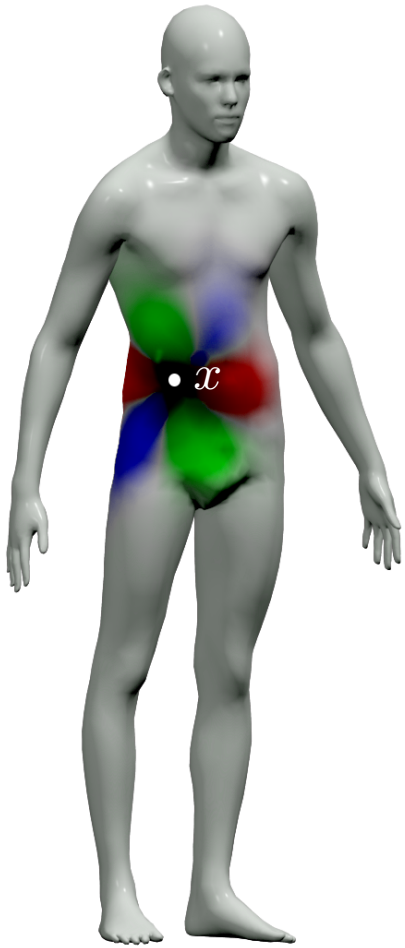
$$(D(x)f)(\theta, t) = \langle f, h_{\alpha\theta t}(x, \cdot)\rangle_{L^2(\mathcal{X})}$$

produces a local representation of f around point x

- $\theta =$ 'angular coordinate'
- $t =$ 'radial coordinate'

Boscaini, Masci, Rodolà, Bronstein 2016

Anisotropic CNN (ACNN)



Given a function $f \in L^2(\mathcal{X})$, the patch operator

$$(D(x)f)(\theta, t) = \langle f, h_{\alpha\theta t}(x, \cdot)\rangle_{L^2(\mathcal{X})}$$

produces a local representation of f around point x

- $\theta =$ 'angular coordinate'
- $t =$ 'radial coordinate'

Anisotropic convolution

$$(f \star a)(x) = \sum_{\theta,t} (D(x)f)(\theta, t)a(\theta, t)$$

Boscaini, Masci, Rodolà, Bronstein 2016

Learned correspondence visualization

Texture transferred from reference to query shapes
using correspondence learned by ACNN

Boscaini, Masci, Rodolà, B 2016

Learned correspondence on point clouds

Colors transferred from reference to query shapes using correspondence learned with ACNN (similar colors encode corresponding points)

Boscaini, Masci, Rodolà, B, Cremers 2015

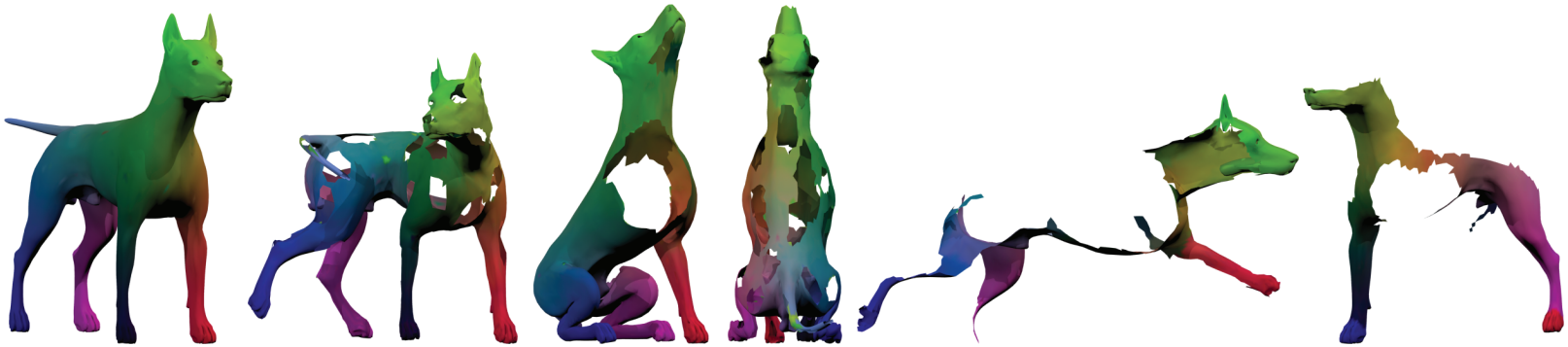
Partial correspondence with ACNN

Correspondence

Correspondence error

Boscaini, Masci, Rodolà, Bronstein 2016

Partial correspondence with ACNN



Correspondence

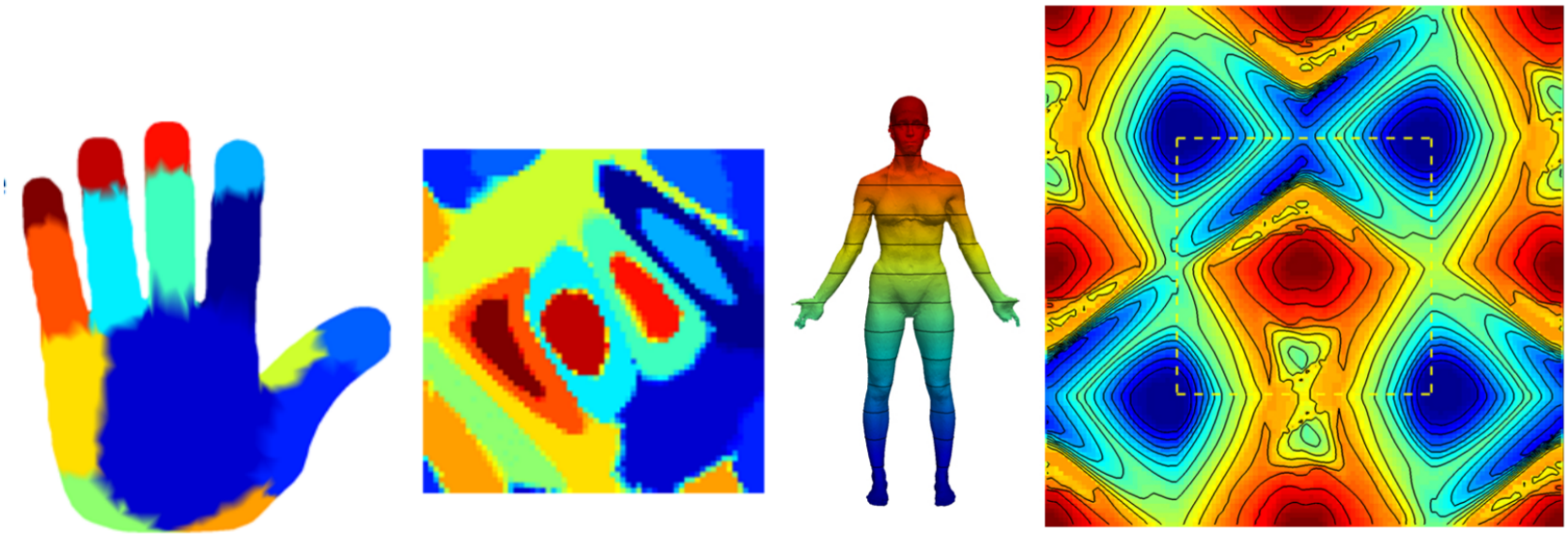
Correspondence error

Boscaini, Masci, Rodolà, Bronstein 2016

Embedding domain deep learning methods

Global parametrization

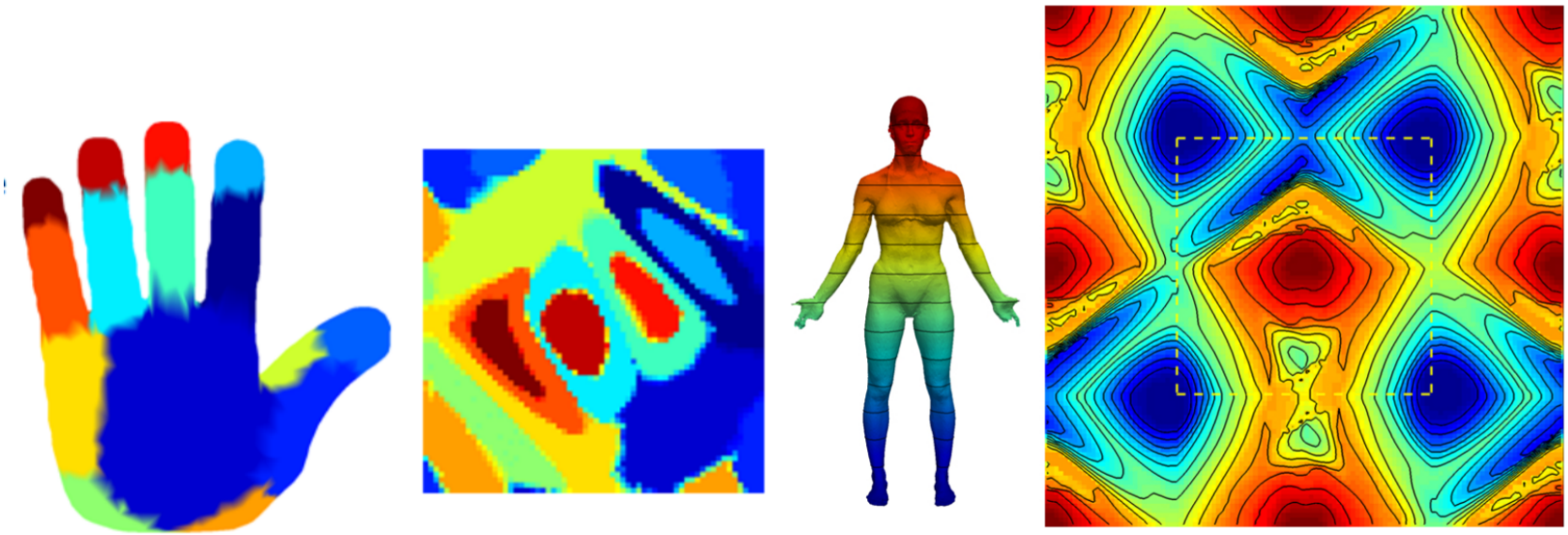
Key idea: map the input surface to some parametric domain (e.g. 2D plane) where operations can be defined more easily.



Sinha et al. 2016; Maron et al. 2017

Global parametrization

Key idea: map the input surface to some parametric domain (e.g. 2D plane) where operations can be defined more easily.

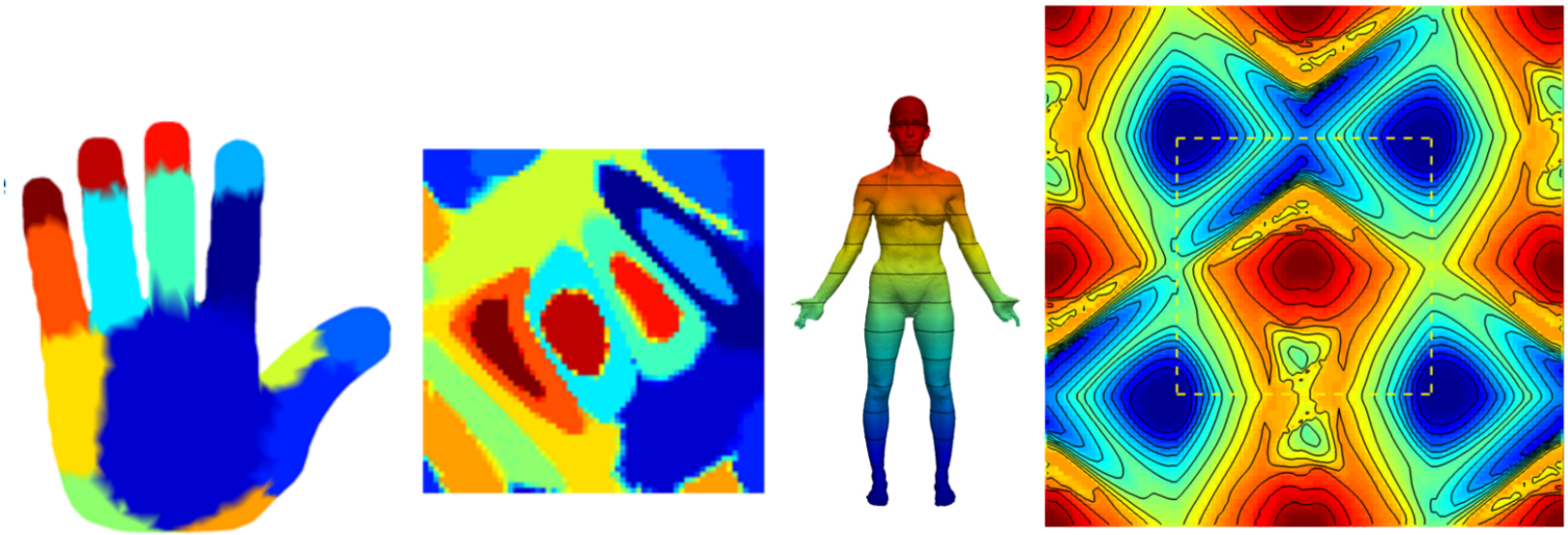


- Enables adoption of Euclidean techniques in the embedding space

Sinha et al. 2016; Maron et al. 2017

Global parametrization

Key idea: map the input surface to some parametric domain (e.g. 2D plane) where operations can be defined more easily.

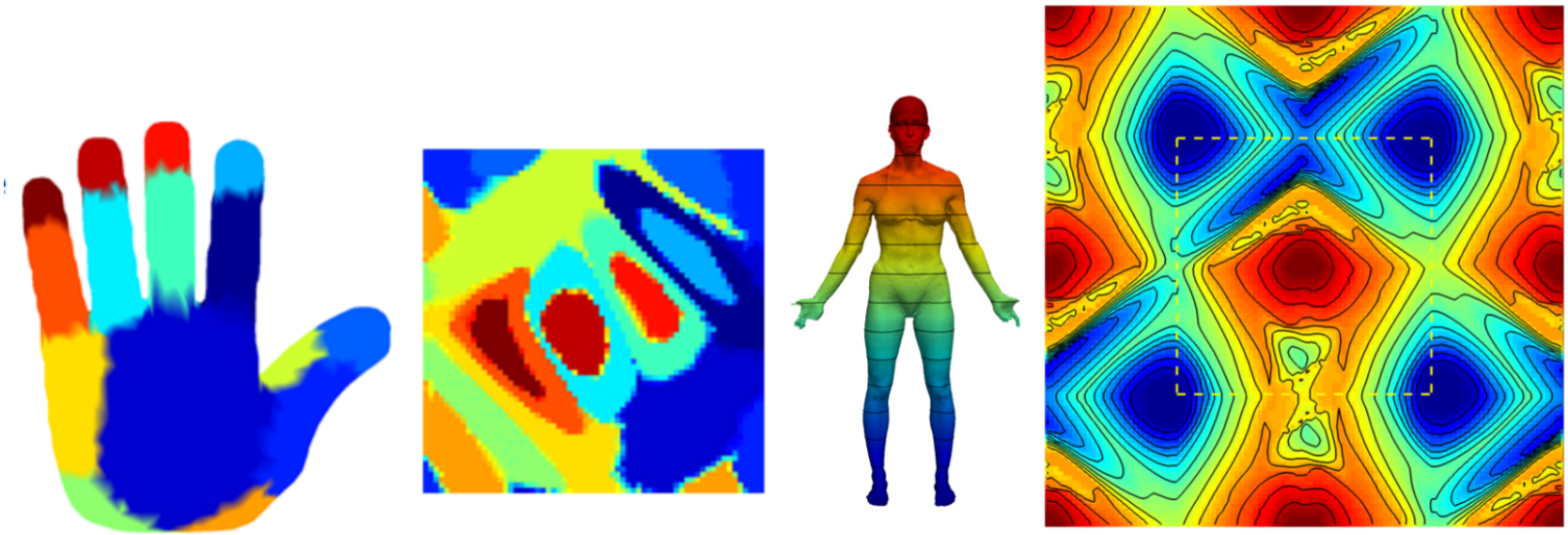


- Enables adoption of Euclidean techniques in the embedding space
- Provides invariance to certain operations

Sinha et al. 2016; Maron et al. 2017

Global parametrization

Key idea: map the input surface to some parametric domain (e.g. 2D plane) where operations can be defined more easily.

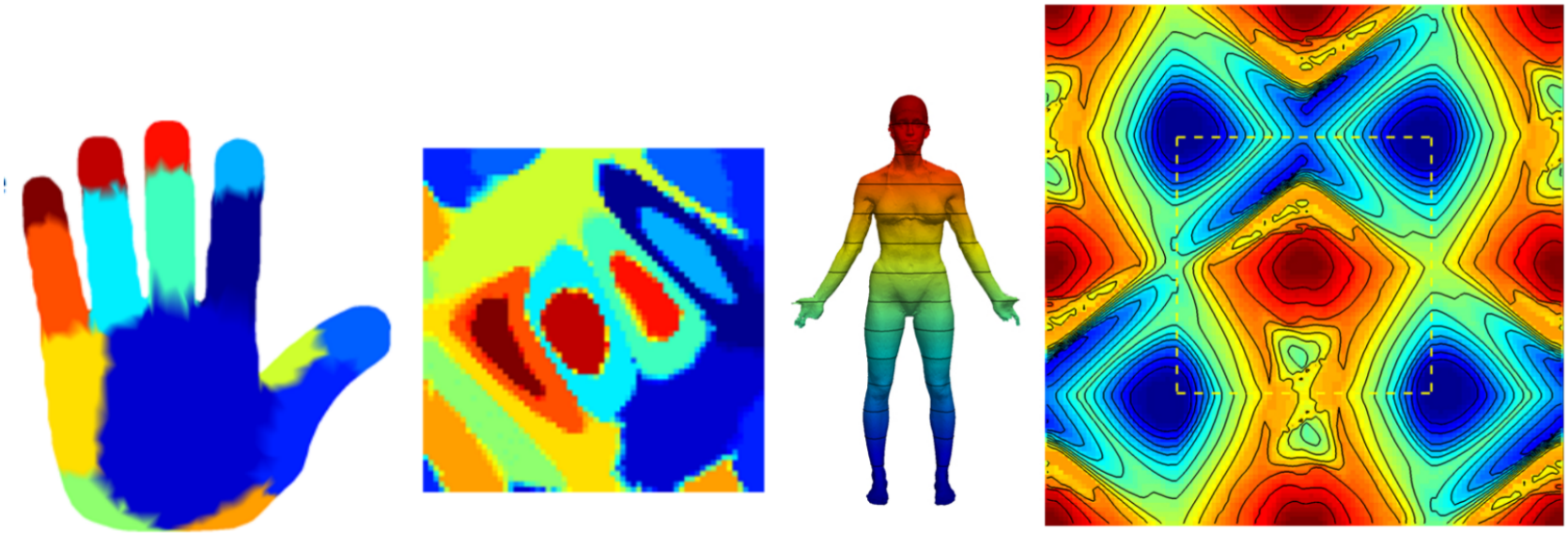


- Enables adoption of Euclidean techniques in the embedding space
- Provides invariance to certain operations
- Parametrization may be non-unique

Sinha et al. 2016; Maron et al. 2017

Global parametrization

Key idea: map the input surface to some parametric domain (e.g. 2D plane) where operations can be defined more easily.



- Enables adoption of Euclidean techniques in the embedding space
- Provides invariance to certain operations
- Parametrization may be non-unique
- The map can introduce distortions

Sinha et al. 2016; Maron et al. 2017

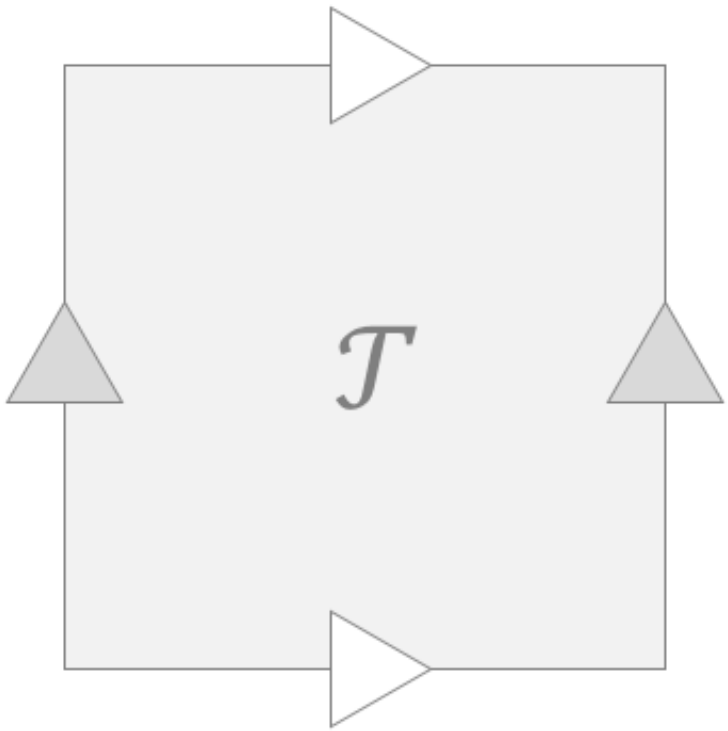
Convolution on surfaces

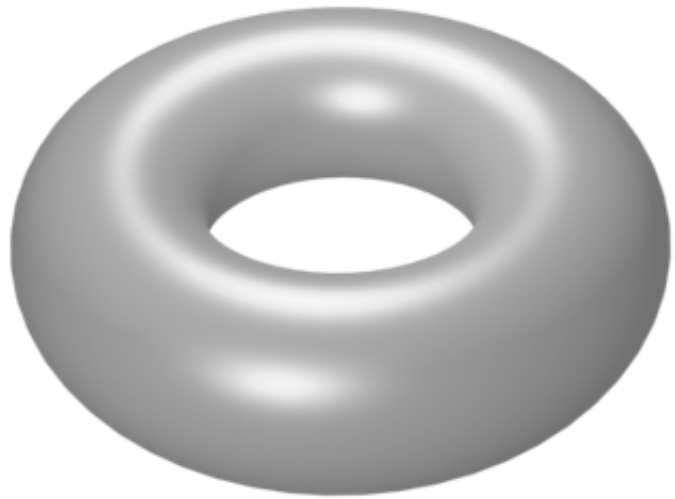
Is translation-invariant convolution on surfaces possible?

Convolution on surfaces

Is translation-invariant convolution on surfaces possible?

Yes! The torus is the only closed orientable surface admitting a translational group.

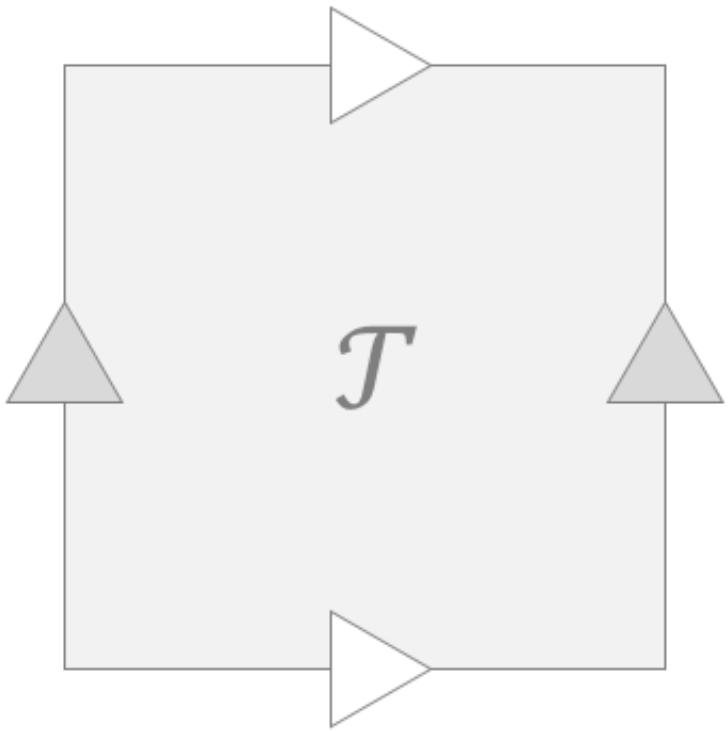


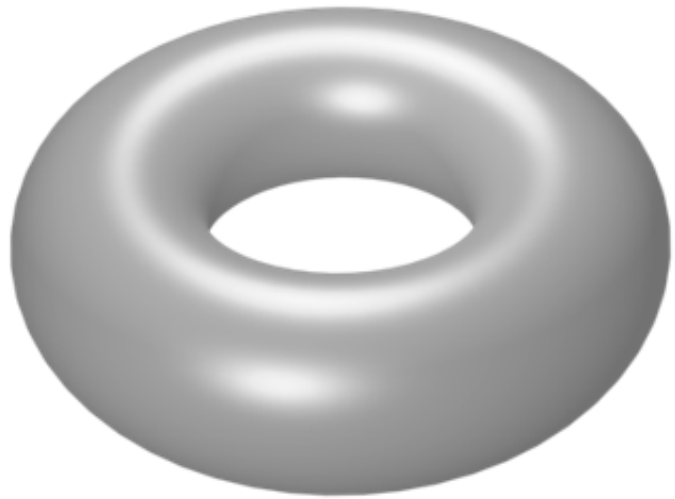


Convolution on surfaces

Is translation-invariant convolution on surfaces possible?

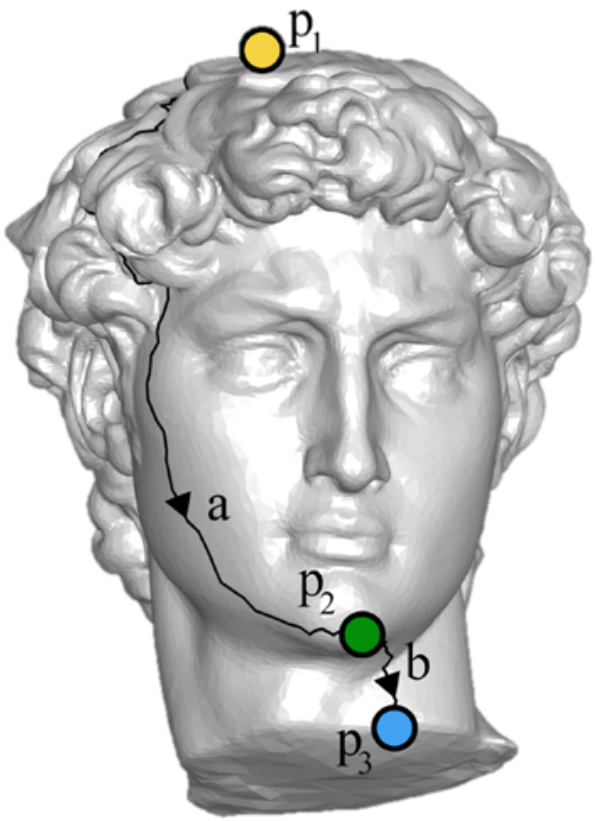
Yes! The torus is the only closed orientable surface admitting a translational group.





CNNs can be well-defined over the flat-torus!

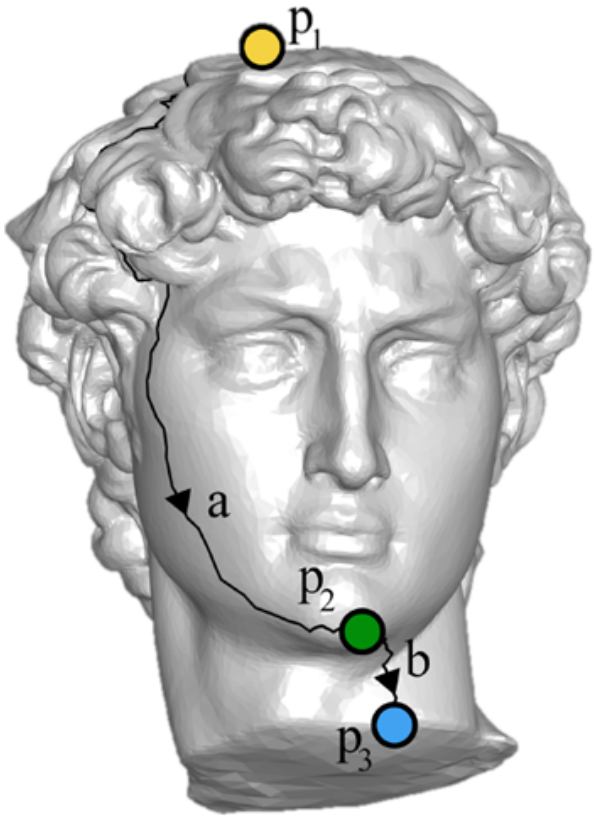
Torus 4-cover



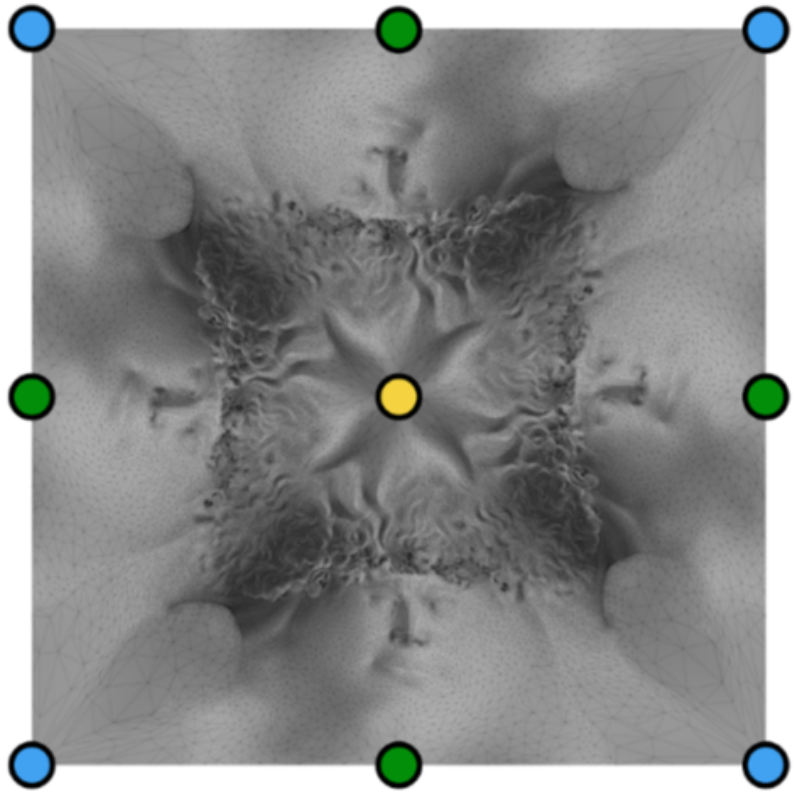
Surface $\mathcal{S}$ with sphere topology

Maron et al. 2017

Torus 4-cover



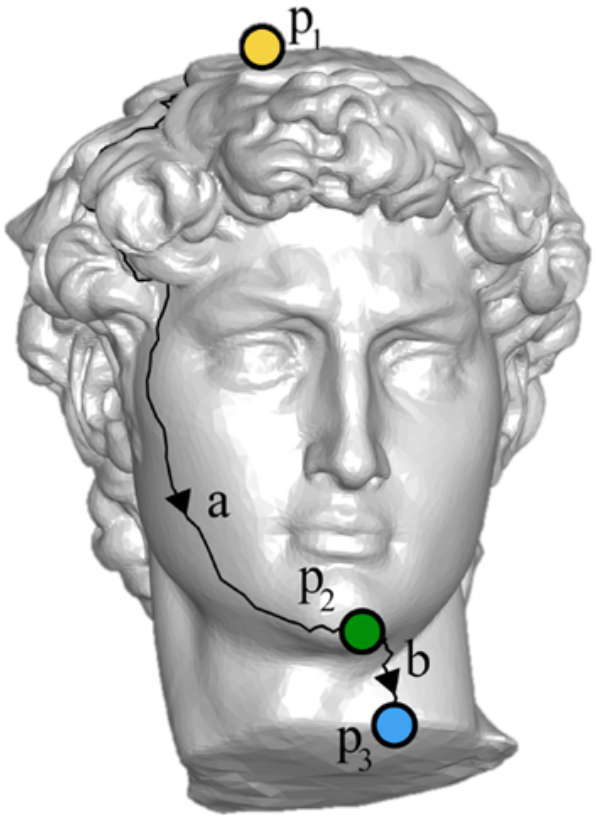
Surface $\mathcal{S}$ with sphere topology



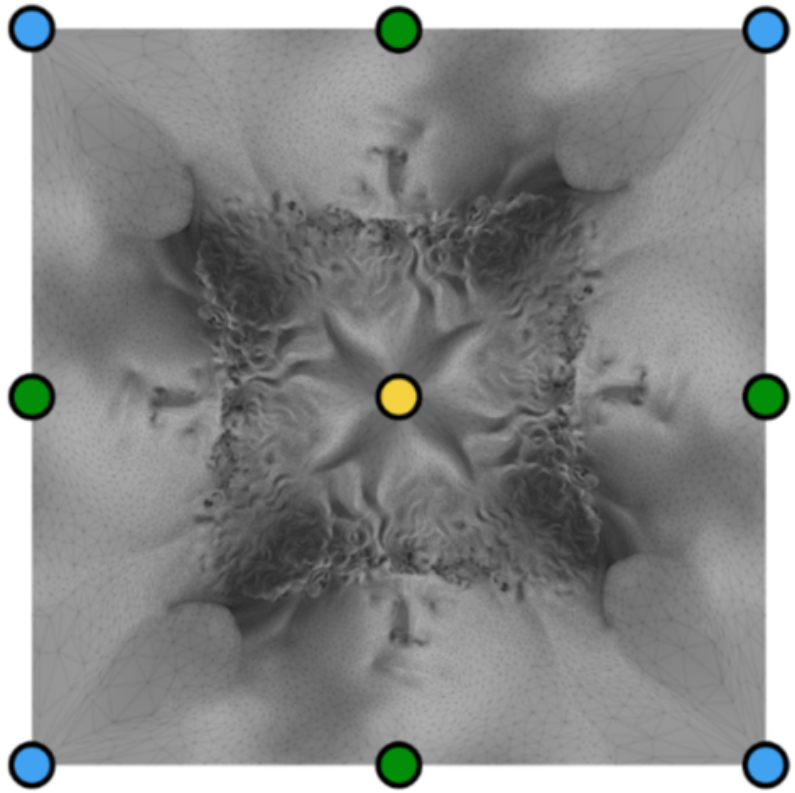
Flat-torus $\mathcal{T}$ with 4 replicas of $\mathcal{S}$

Maron et al. 2017

Torus 4-cover



Surface $\mathcal{S}$ with sphere topology



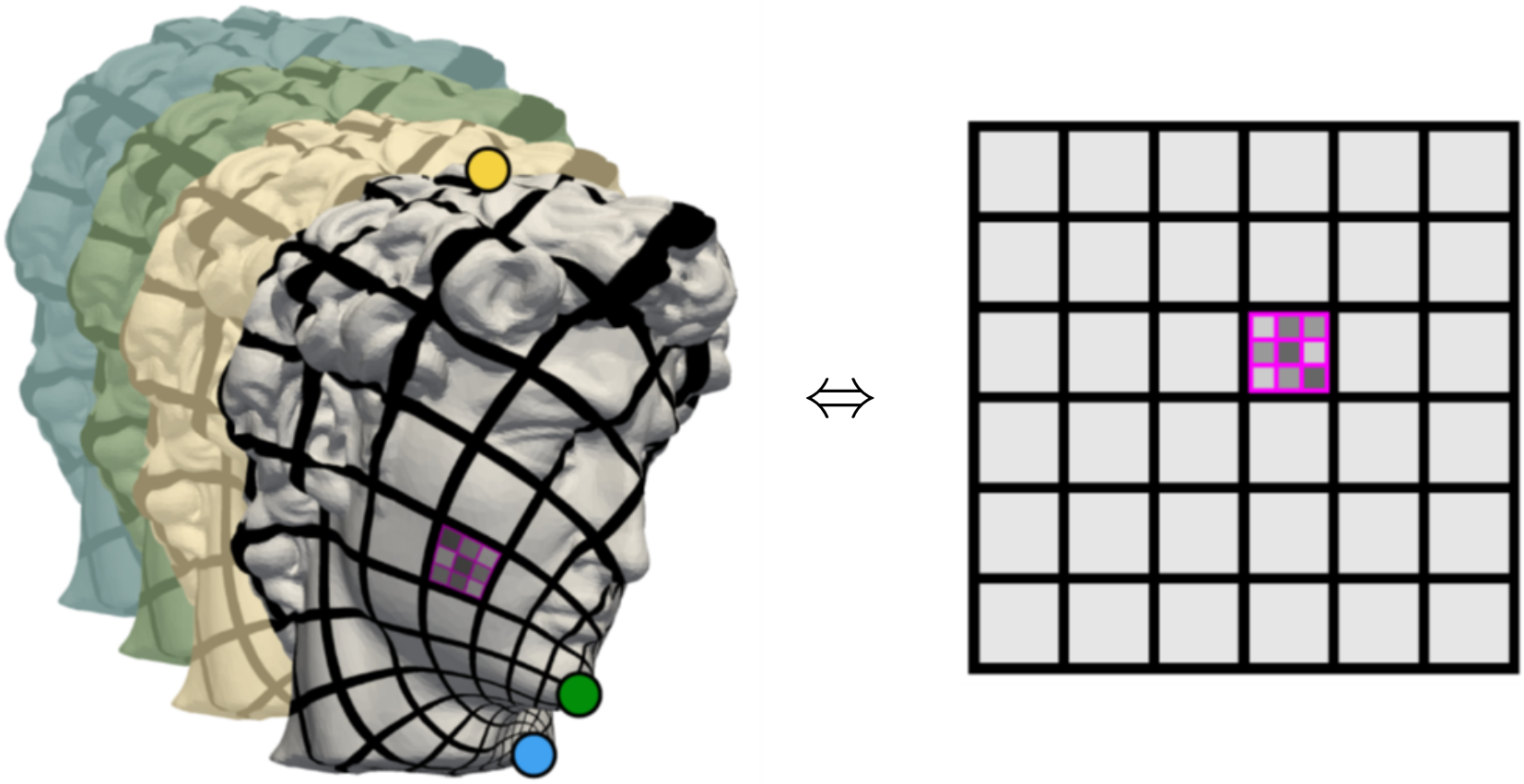
Flat-torus $\mathcal{T}$ with 4 replicas of $\mathcal{S}$

Standard Euclidean 2D CNN architectures can now be used on $\mathcal{T}$.

Maron et al. 2017

Torus 4-cover

For each triplet $\{p_1, p_2, p_3\} \in \mathcal{S}$, use orbifold-Tutte to map $\mathcal{S}^4$ to $\mathcal{T}$.



Tha mapping from $\mathcal{S}^4$ to $\mathcal{T}$ is a conformal homeomorphism.

Maron et al. 2017

Conformal zoom

Conformality introduces scale changes.

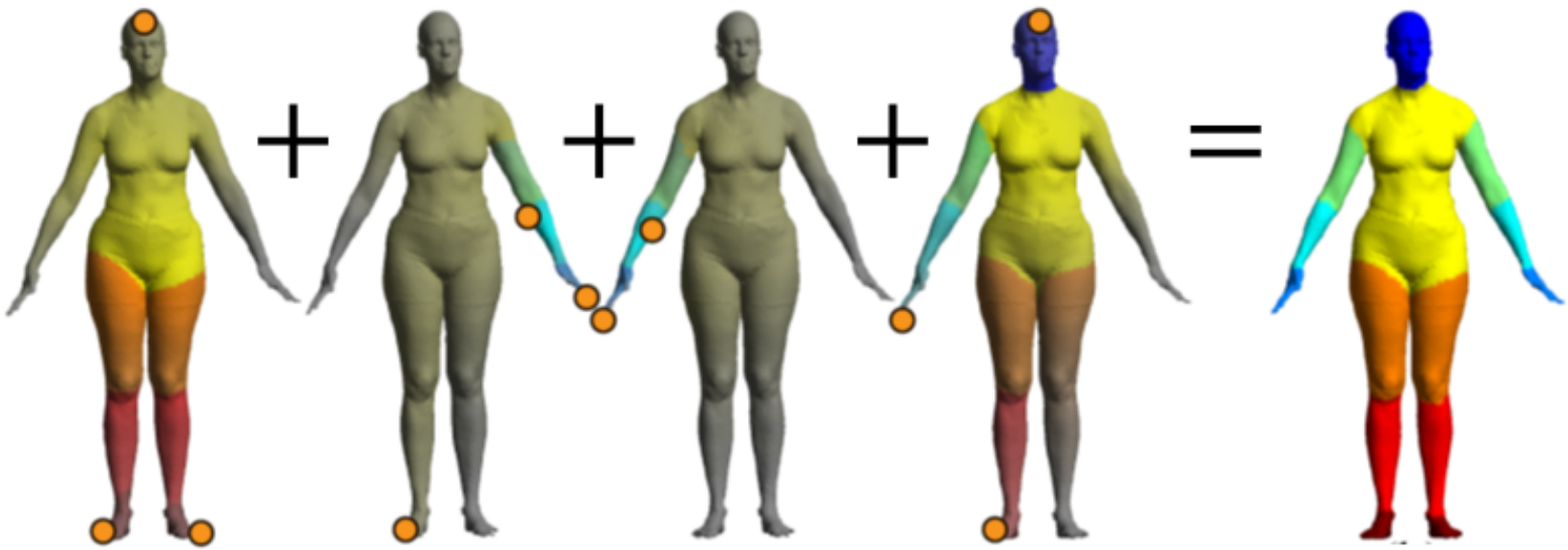
- "Magnifying glass" effect

Maron et al. 2017

Conformal zoom

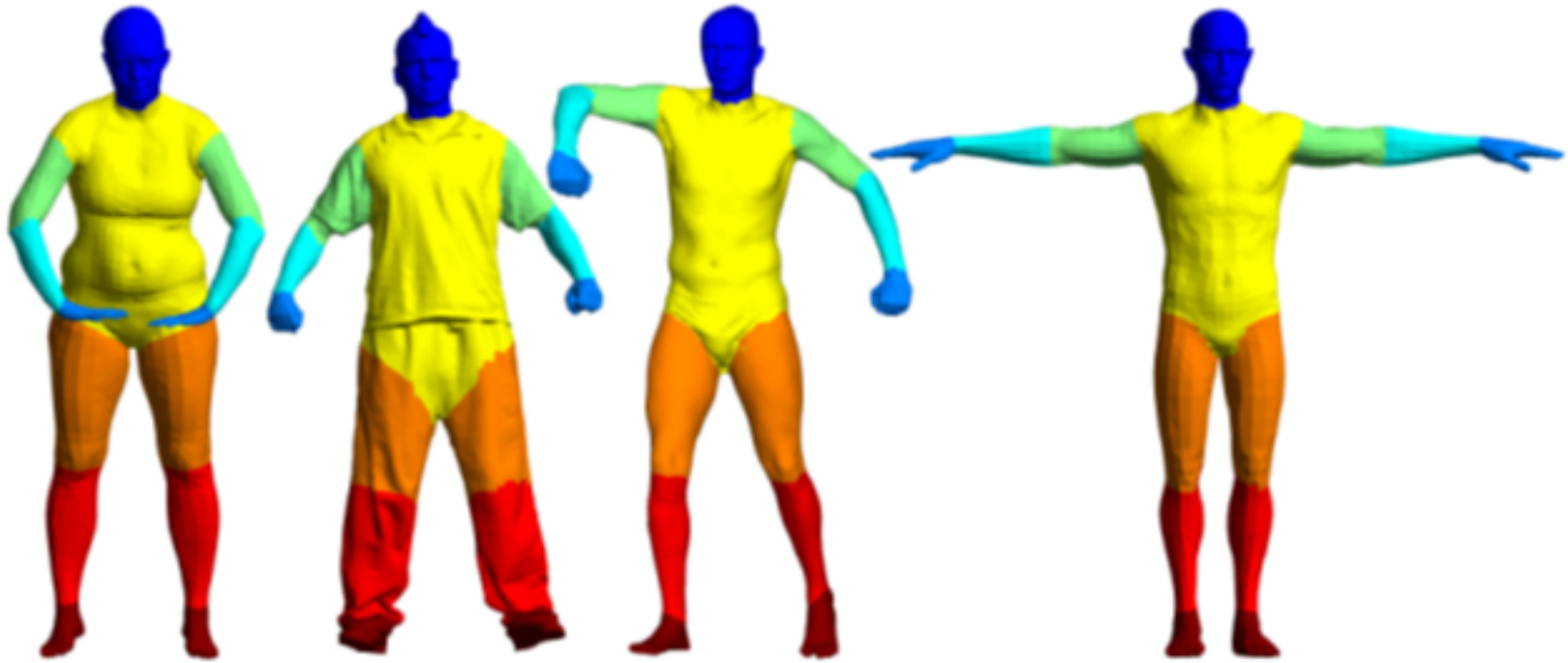
Conformality introduces scale changes.

- "Magnifying glass" effect
- Prediction aggregation from different triplets at test time



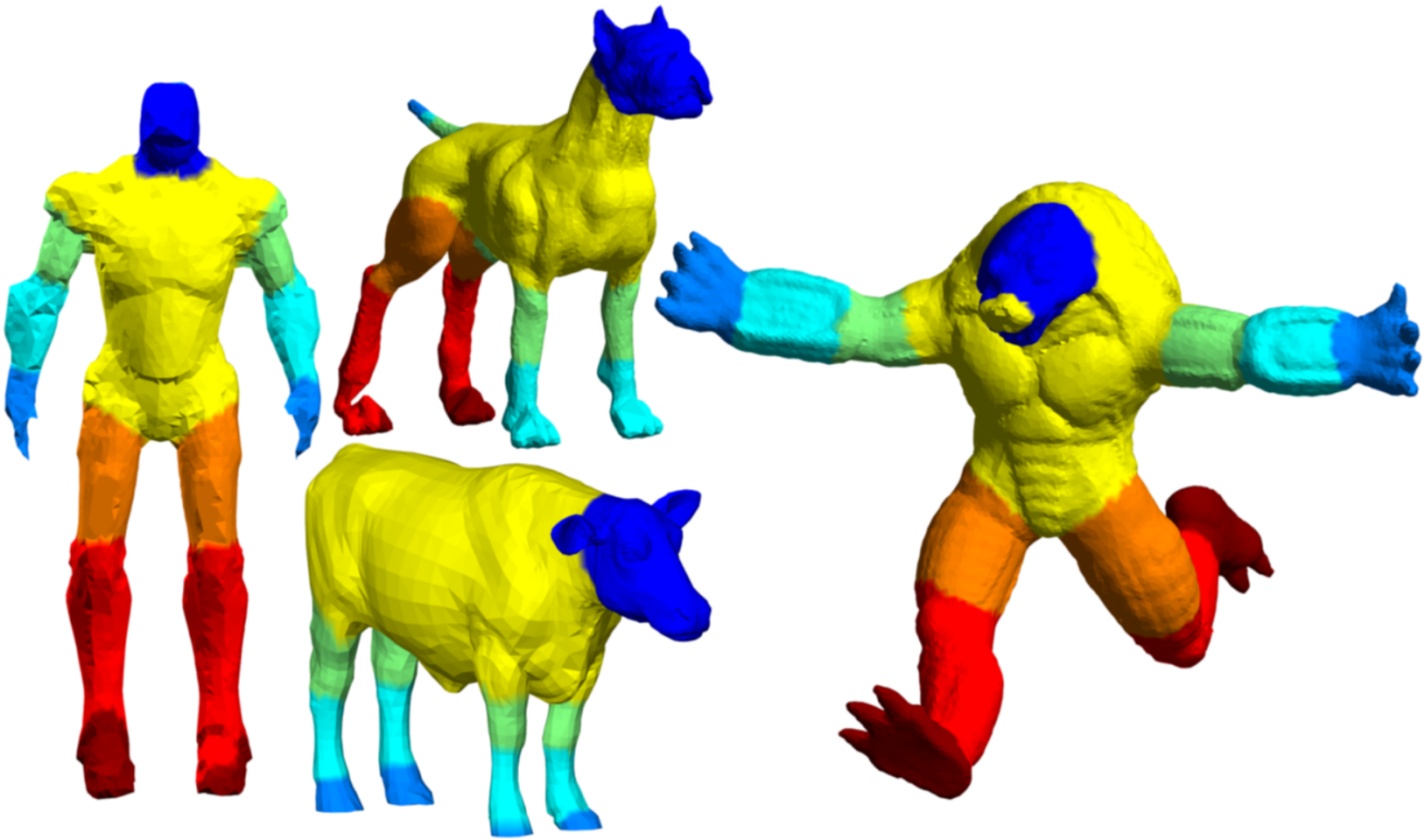
Maron et al. 2017

Segmentation results



Maron et al. 2017

Segmentation results



Maron et al. 2017

Conclusions

- Application / extension of deep learning to geometric and non-Euclidean structured data
- Deep learning allows designing task-specific features and learning from data invariance that is impossible to model axiomatically
- In graphics, an almost virgin field that is very rapidly developing
- Many synergies with other fields
- State-of-the-art results in many 'old' applications
- New applications

Challenges

- Theoretical questions
- Generalization across domains
- Time-varying data / domains
- Directed graphs
- Other shape representations
- Synthesis
- Computation

Shameless advertisement

Shameless advertisement

Geometric and Visual Computing
at Sapienza University of Rome

Topics: shape analysis, 3D vision, reconstruction, geometry processing, machine learning, interdisciplinary applications

Thank you!