Geometric deep learning on graphs and manifolds

Michael Bronstein[1,2,3] Joan Bruna[6] Arthur Szlam[5]

Xavier Bresson[4] Yann LeCun[5,6]

[1]USI
Switzerland

[2]Imperial College
UK

[3]Intel
Israel

[4]NTU
Singapore

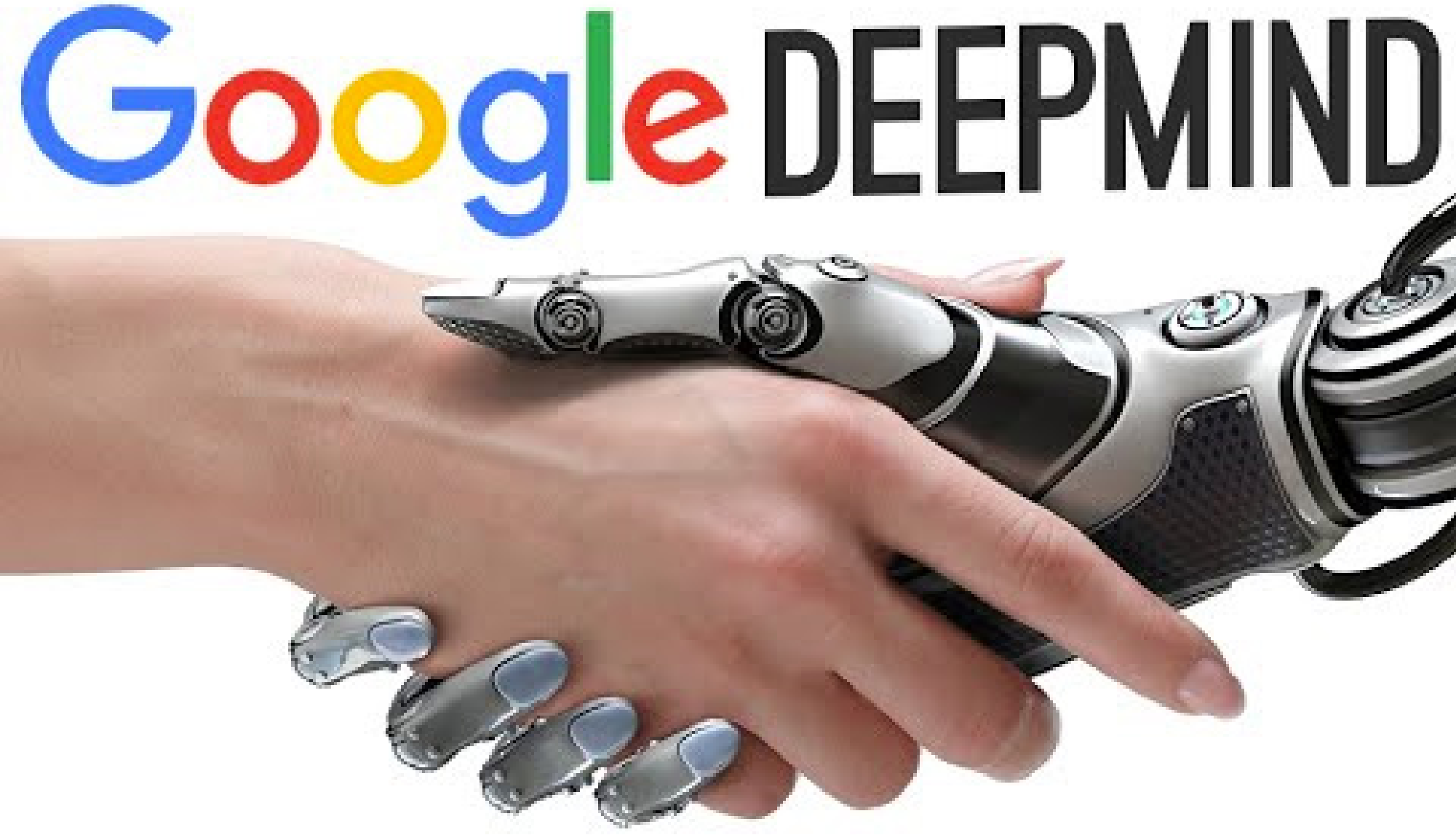
[5]FAIR
USA

[6]NYU
USA

SIAM Tutorial 12 July 2018, Portland



Facebook opens AI Research in 2014



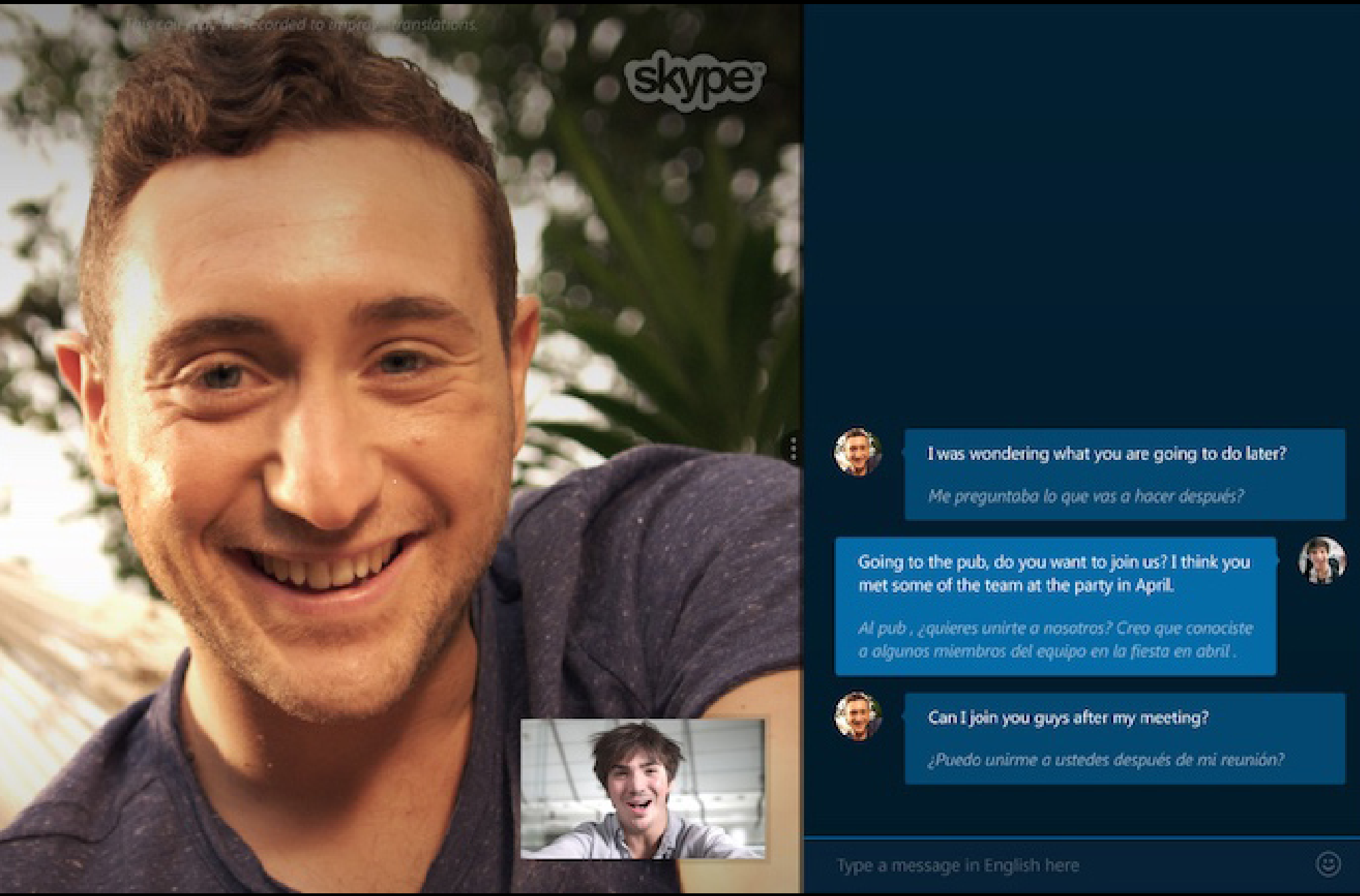
Google acquires DeepMind in 2014



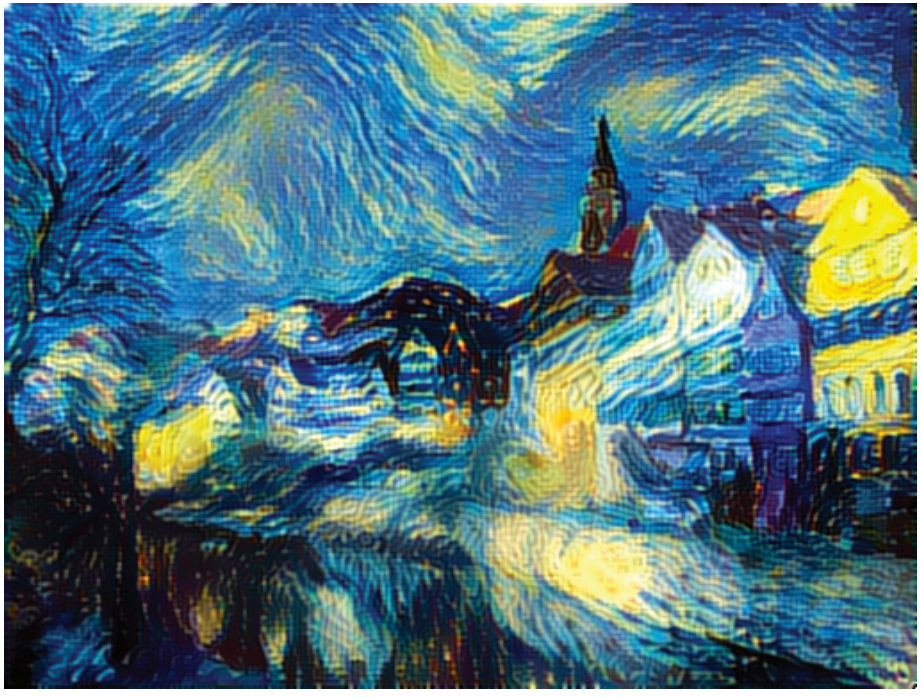
Google DeepMind AlphaGo beats human player



Tesla autonomous driving car

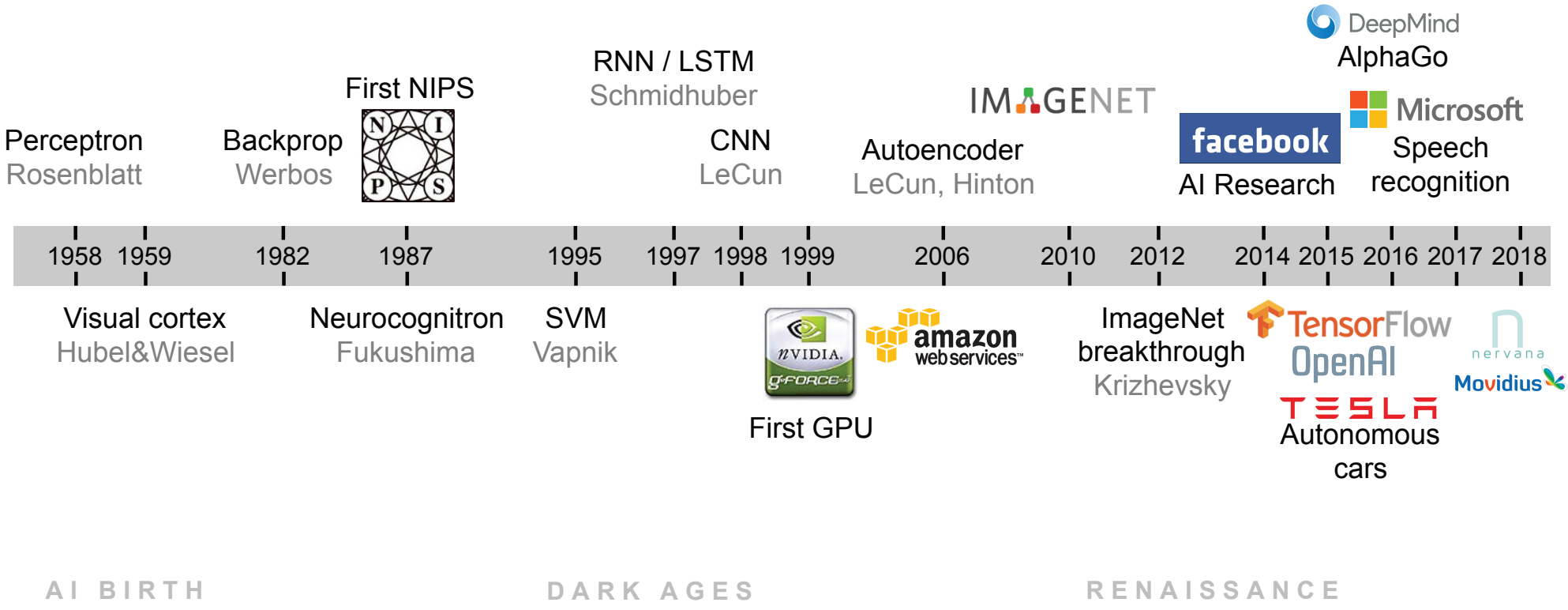


Microsoft Skype real-time voice translation 2016

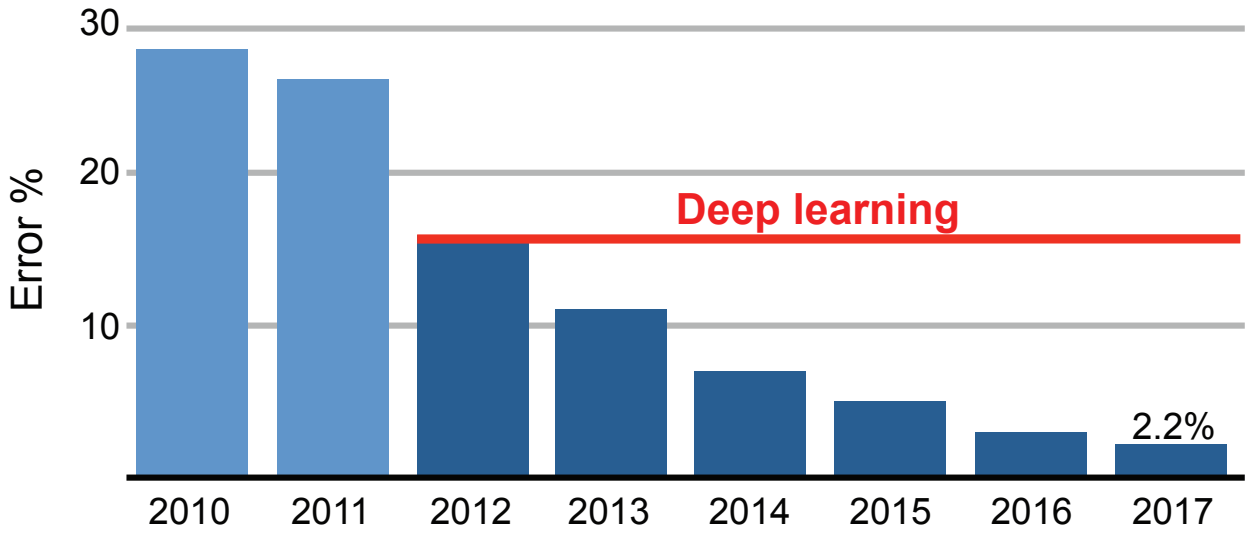




Gatys et al. 2016; Karras et al. 2018

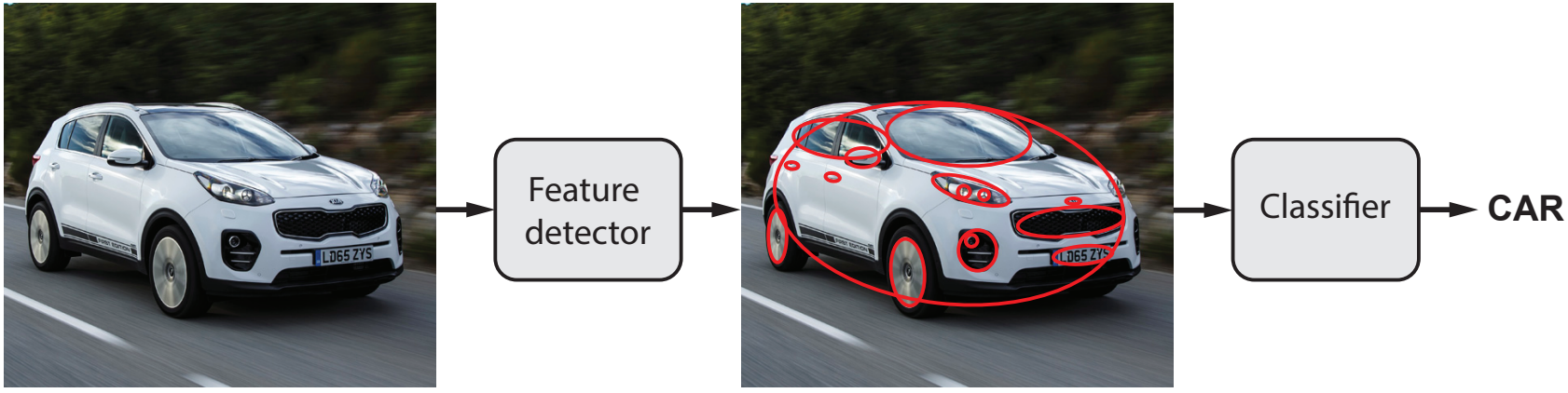

DeepMind
AlphaGo
RNN / LSTM
Schmidhuber
First NIPS
IMAGENET
Microsoft
Speech recognition
Perceptron
Rosenblatt
Backprop
Werbos
CNN
LeCun
Autoencoder
LeCun, Hinton
facebook
AI Research
1958
1959
1982
1987
1995
1997
1998
1999
2006
2010
2012
2014
2015
2016
2017
2018
Visual cortex
Hubel&Wiesel
Neurocognitron
Fukushima
SVM
Vapnik
First GPU
amazon web services
ImageNet breakthrough
Krizhevsky
TensorFlow
OpenAI
nervana
Movidius
Autonomous cars
AI BIRTH
DARK AGES
RENAISSANCE

Breakthrough in image recognition



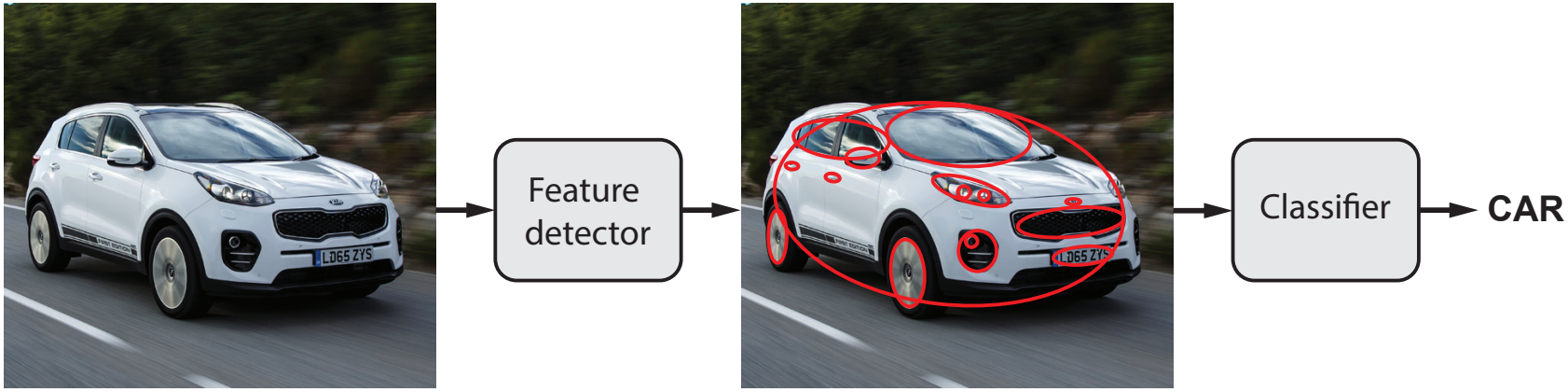
ImageNet ILSVRC Challenge

Handcrafted vs Learned features

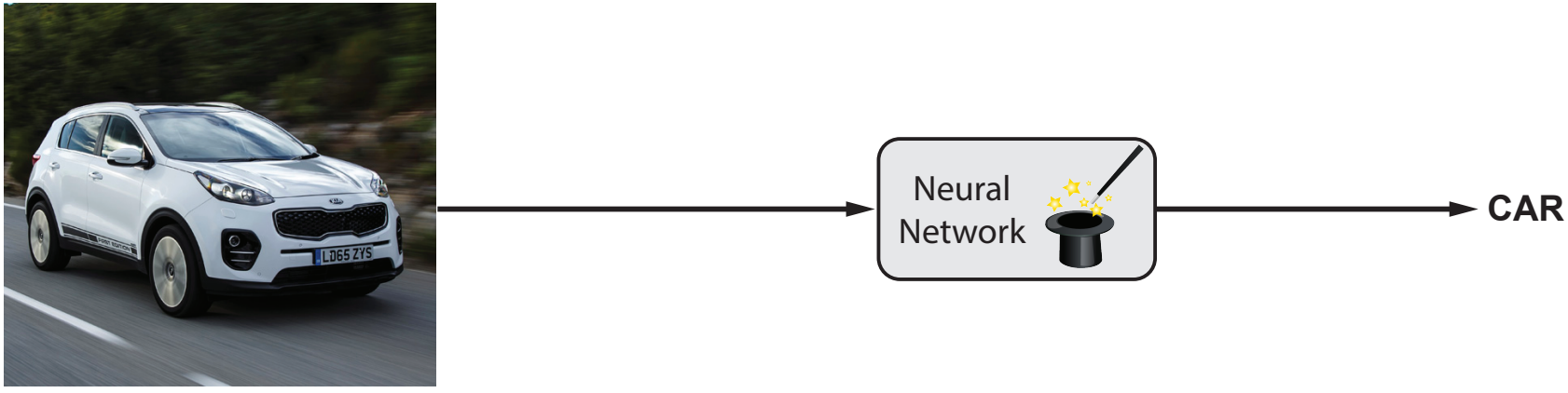


Classical computer vision: hand-crafted features (e.g. SIFT)
+ simple classifier (e.g. SVM)

Handcrafted vs Learned features



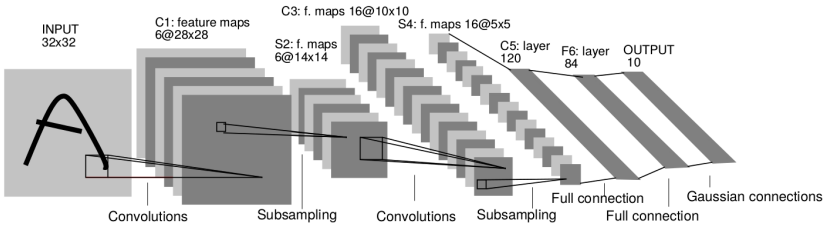
Classical computer vision: hand-crafted features (e.g. SIFT)
+ simple classifier (e.g. SVM)



Modern computer vision: data-driven end-to-end systems

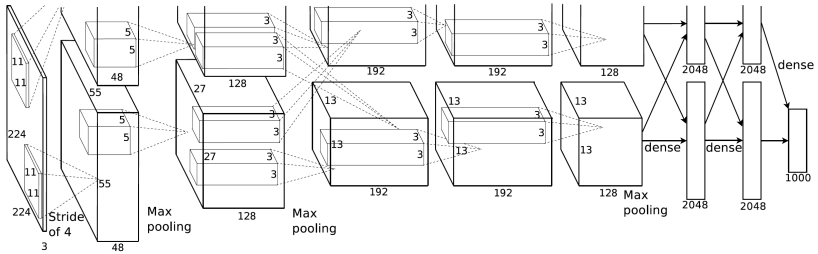
Convolutional Neural Networks (CNN)

1989



- 3 convolutional + 1 fully connected layer
- 1M parameters
- Trained on MNIST 70K
- CPU-based
- $\tanh$ non-linearity

2012



- 5 convolutional + 3 fully connected layers
- 60M parameters
- Trained on ImageNet 1.5M
- GPU-based
- ReLU, Dropout

LeCun et al. 1998; Krizhevsky, Sutskever, Hinton 2012

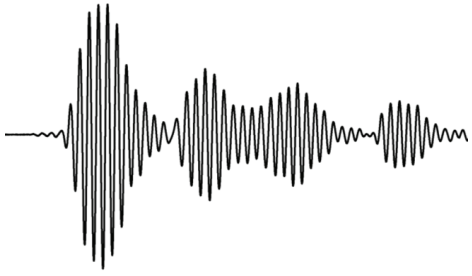
Datasets over algorithms

Year	Breakthroughs in AI	Datasets (First Available)	Algorithms (First Proposed)
1994	Human-level spontaneous speech recognition	Spoken Wall Street Journal articles and other texts (1991)	Hidden Markov Model (1984)
1997	IBM Deep Blue defeated Garry Kasparov	700,000 Grandmaster chess games, aka "The Extended Book" (1991)	Negascout planning algorithm (1983)
2005	Google's Arabic- and Chinese-to-English translation	1.8 trillion tokens from Google Web and News pages (collected in 2005)	Statistical machine translation algorithm (1988)
2011	IBM Watson became the world Jeopardy! champion	8.6 million documents from Wikipedia, Wiktionary, Wikiquote, and Project Gutenberg (updated in 2010)	Mixture-of-Experts algorithm (1991)
2014	Google's GoogLeNet object classification at near-human performance	ImageNet corpus of 1.5 million labeled images and 1,000 object categories (2010)	Convolution neural network algorithm (1989)
2015	Google's Deepmind achieved human parity in playing 29 Atari games by learning general control from video	Arcade Learning Environment dataset of over 50 Atari games (2013)	Q-learning algorithm (1992)
Average No. of Years to Breakthrough:		**3 years**	**18 years**

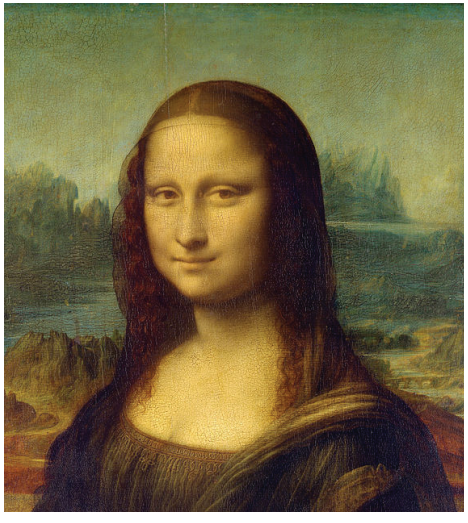
Wissner-Gross 2016

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

Text



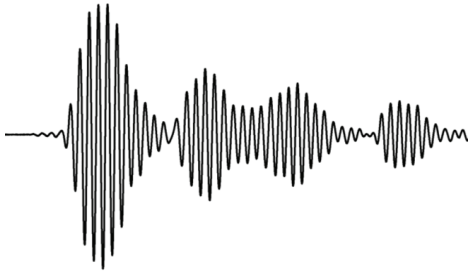
Audio signals



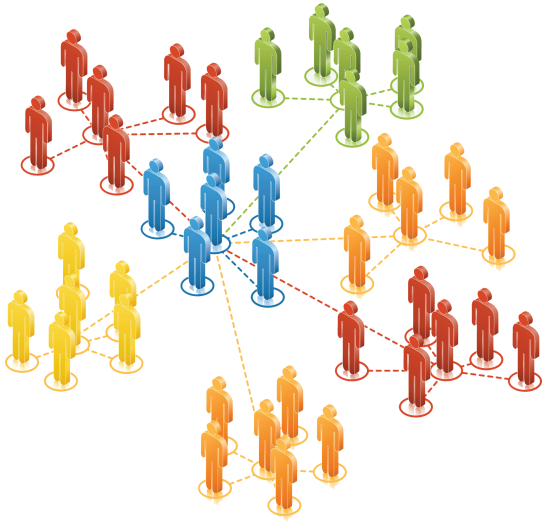
Images

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

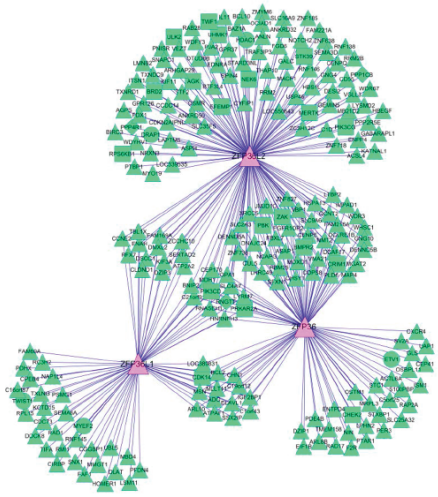
Text



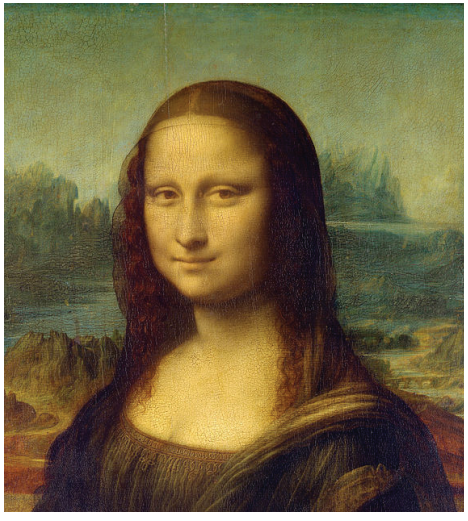
Audio signals



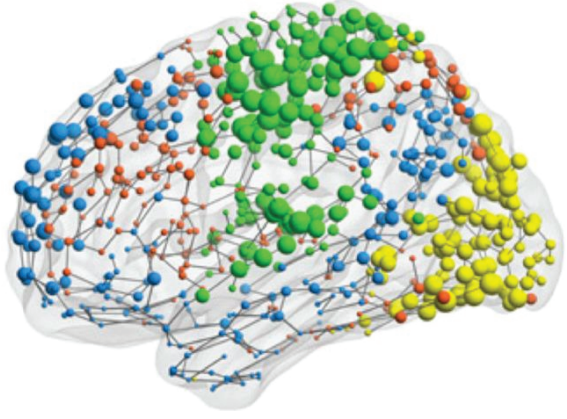
Social networks



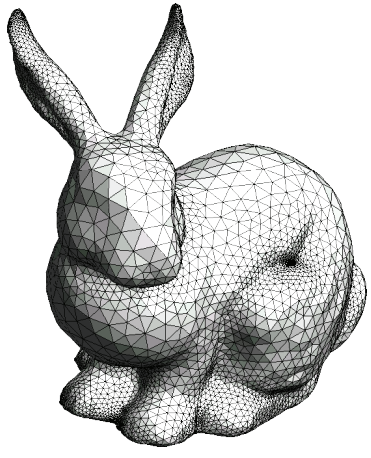
Regulatory networks



Images



Functional networks



3D shapes

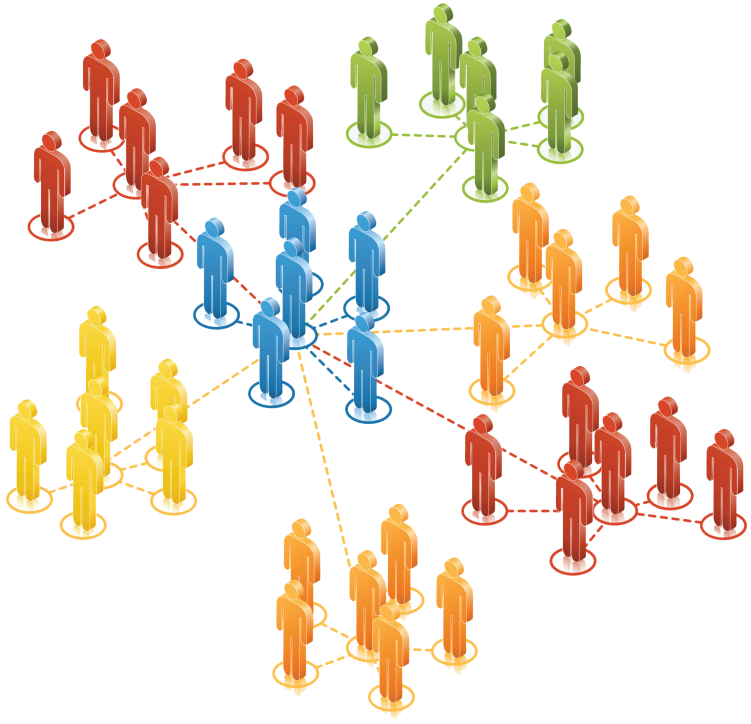
How to design deep nets on non-Euclidean domains and what to do with them?

How to design deep nets on non-Euclidean domains and what to do with them? a.k.a. Geometric deep learning

Prototypical non-Euclidean objects

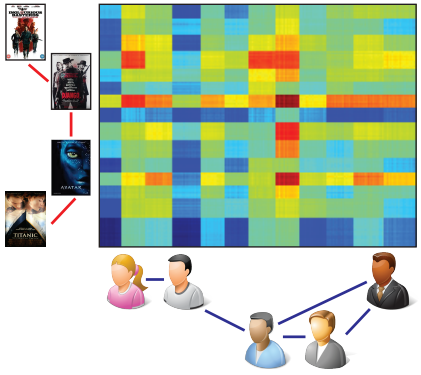


Manifolds

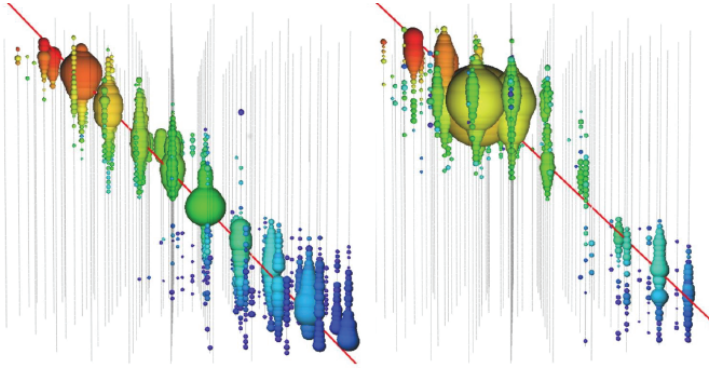


Graphs

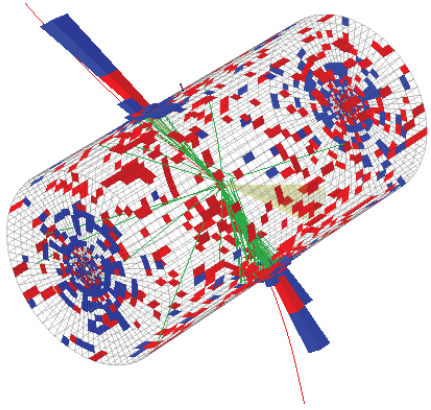
Application of geometric deep learning on graphs



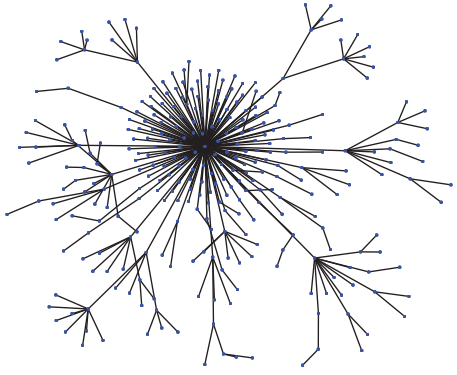
Recommender system



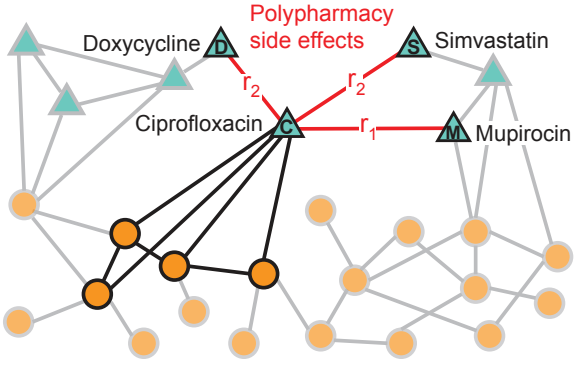
Neutrino detection



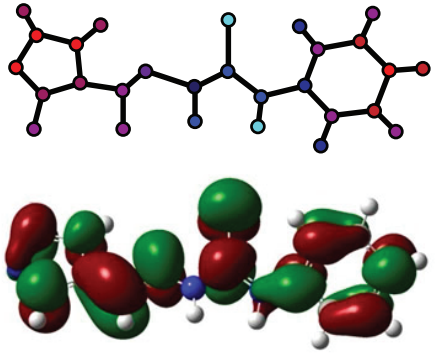
LHC



Fake news detection



Drug repurposing

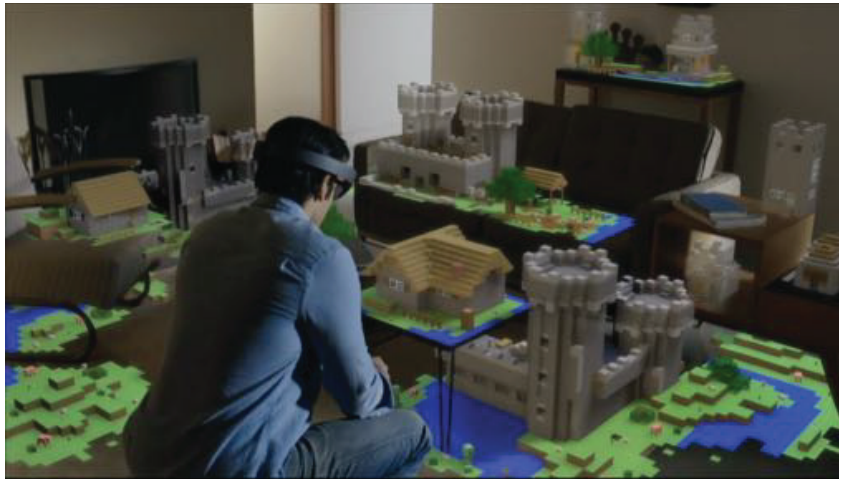


Chemistry

Application of geometric deep learning on manifolds



Computer graphics



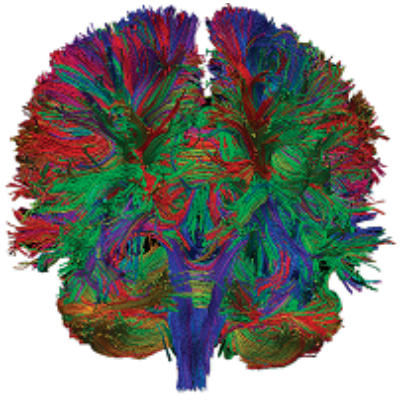
Virtual/augmented reality



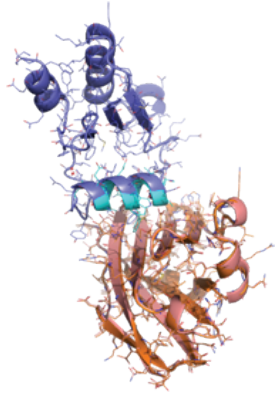
Robotics



Autonomous driving



Medicine

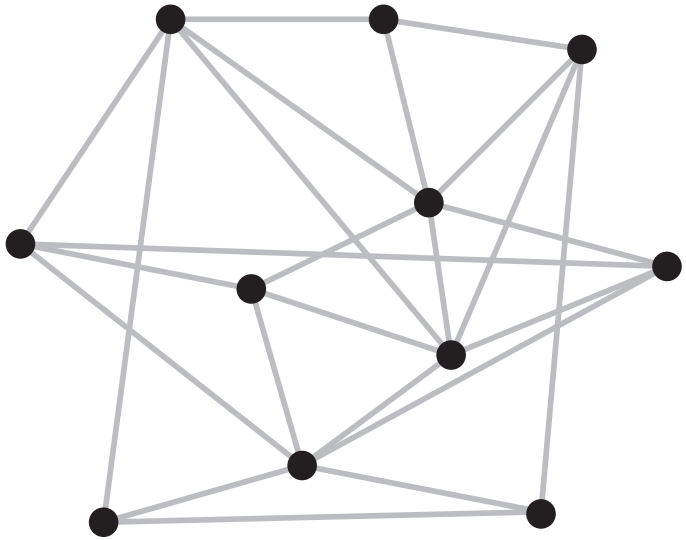


Drug design

Domain structure vs Data on a domain

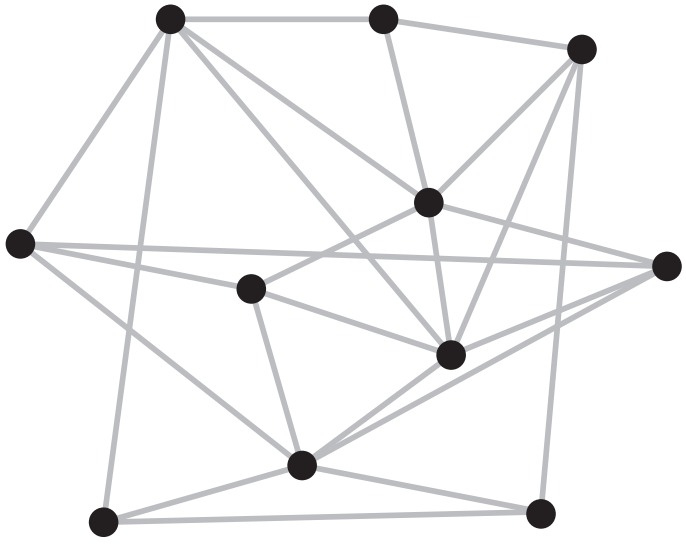


Domain structure vs Data on a domain

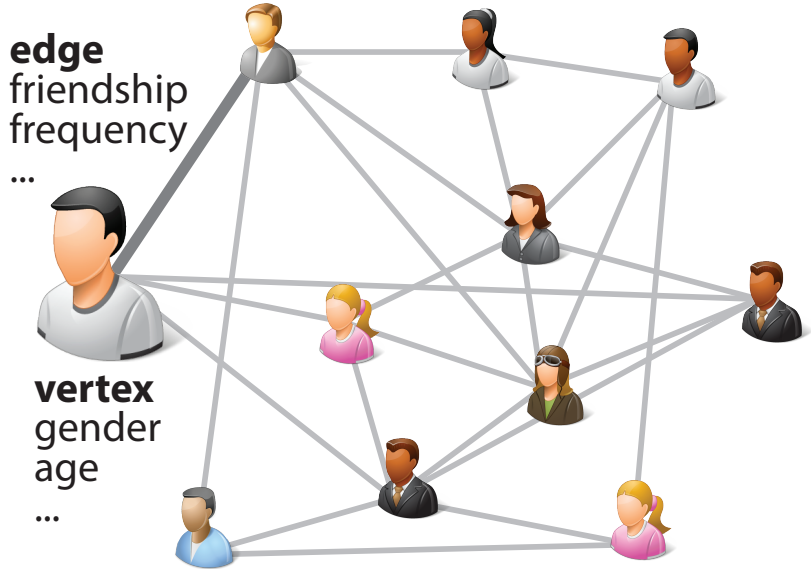


Domain structure

Domain structure vs Data on a domain



Domain structure



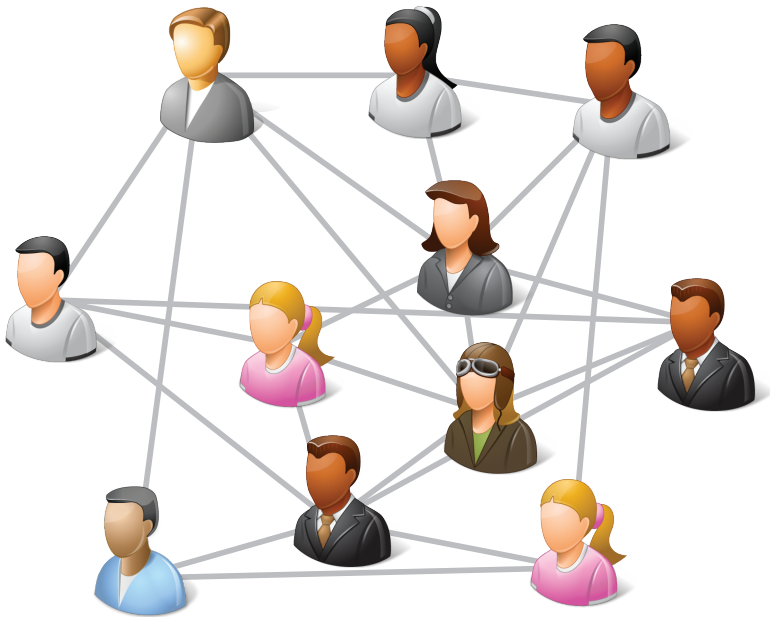
Data on a domain

Fixed vs different domain

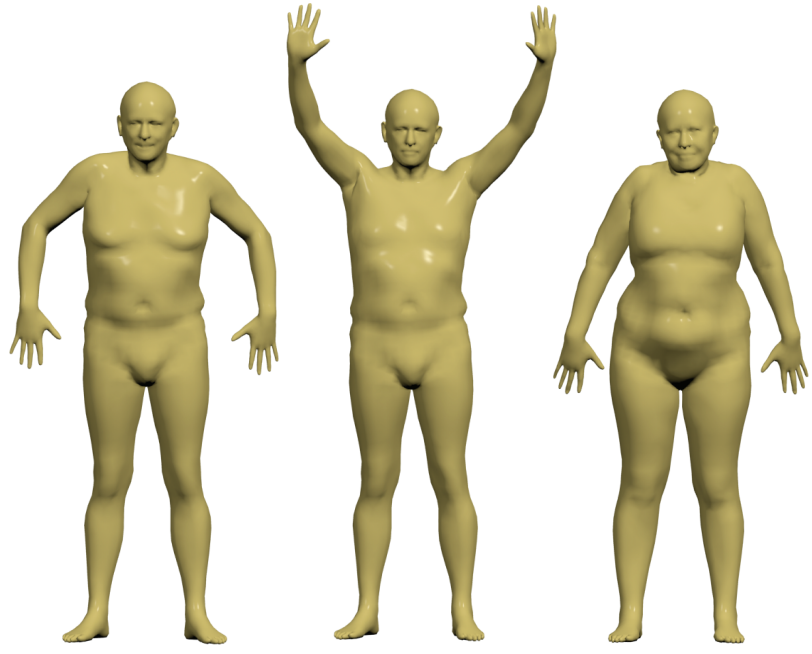


Social network
(fixed graph)

Fixed vs different domain



Social network
(fixed graph)

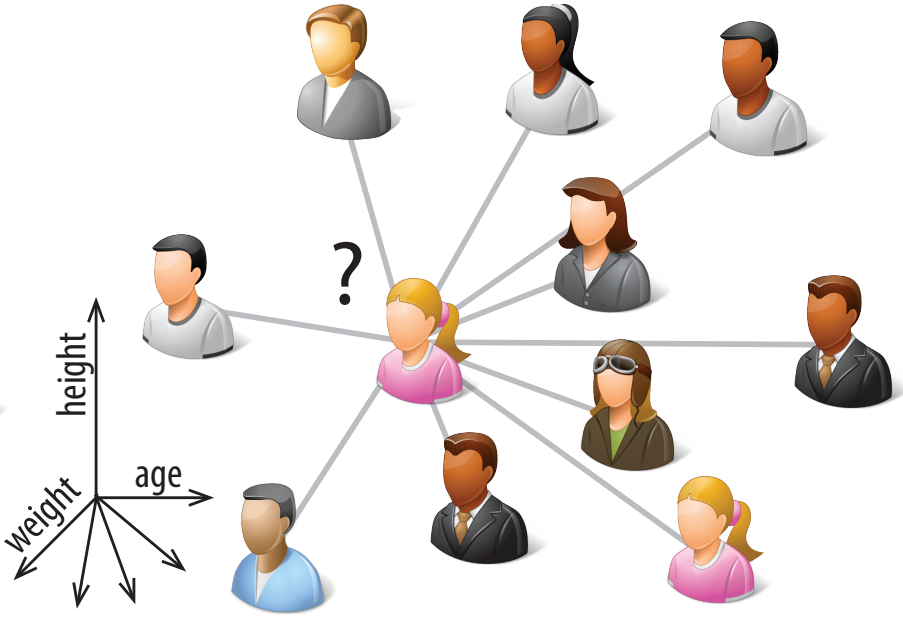


3D shapes
(different manifolds)

Known vs unknown domain

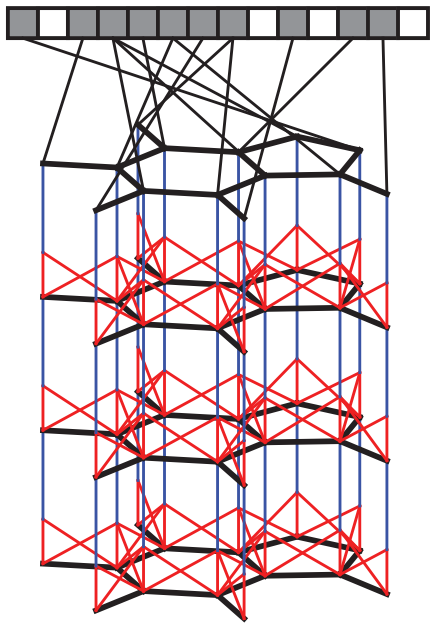


Given graph



Learned graph

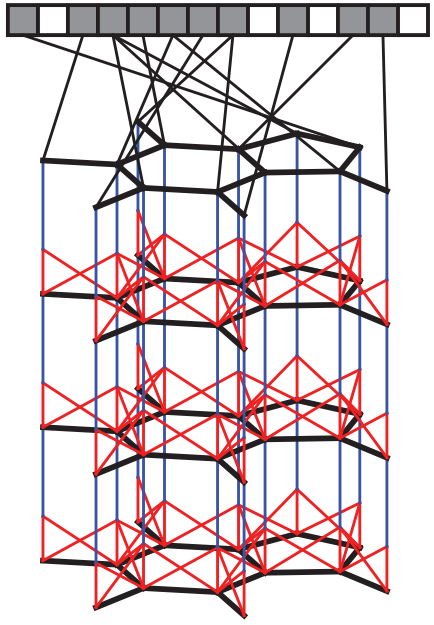
Graph-wise classification



Molecule graph

Duvenaud et al. 2015

Graph-wise classification



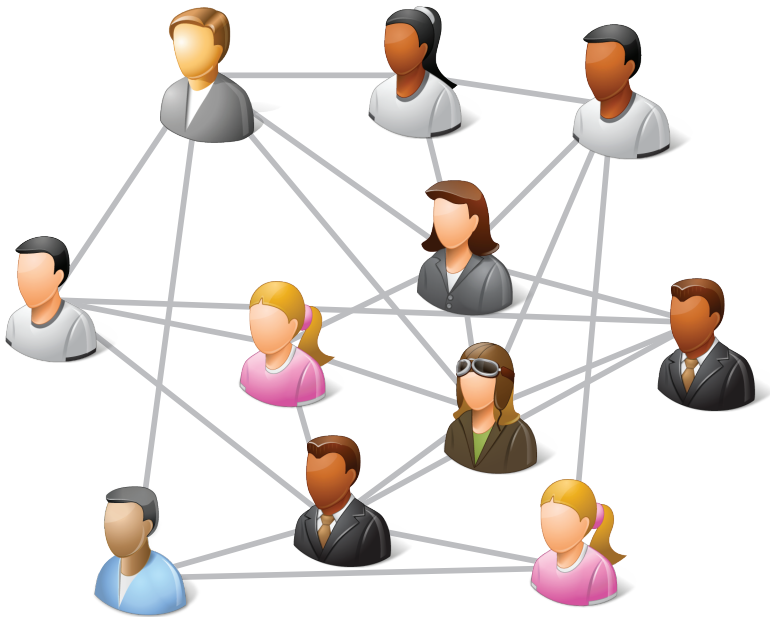
Molecule graph



Image recognition

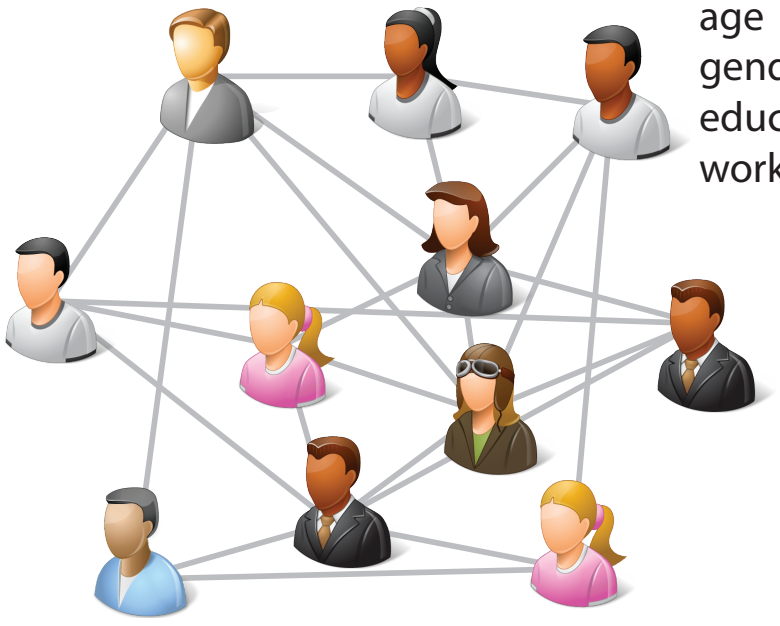
Duvenaud et al. 2015

Vertex-wise classification



Social network

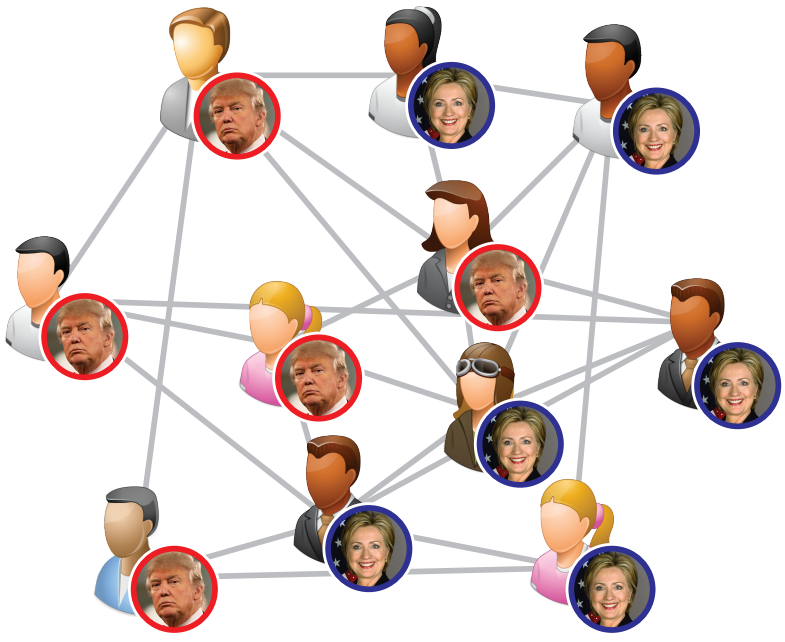
Vertex-wise classification



age
gender
education
work

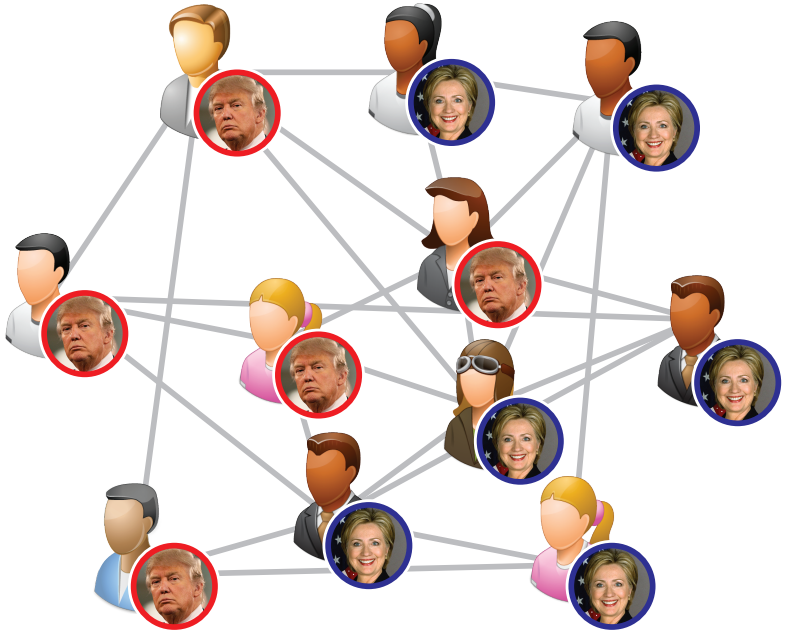
Social network

Vertex-wise classification



Social network

Vertex-wise classification

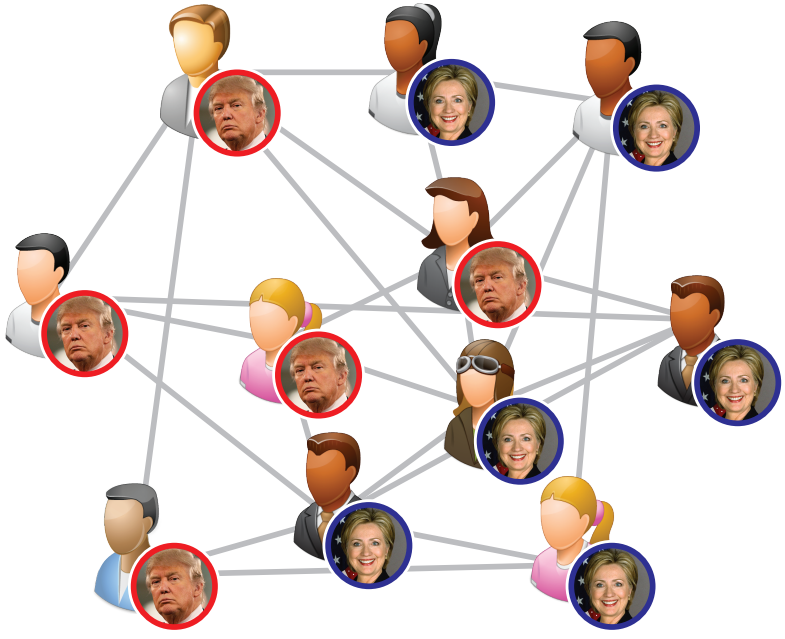


Social network

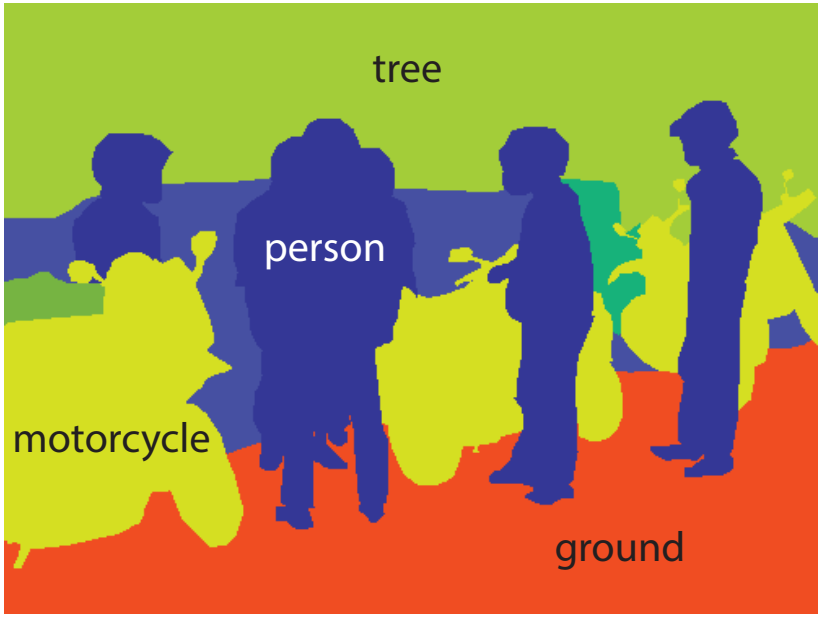


Semantic image segmentation

Vertex-wise classification



Social network



Semantic image segmentation

Different formulations of non-Euclidean CNNs

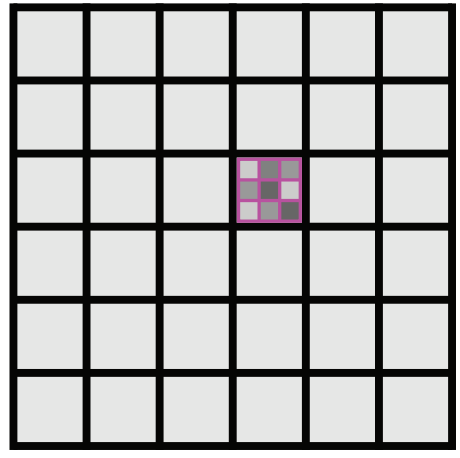


Spectral domain



Spatial domain



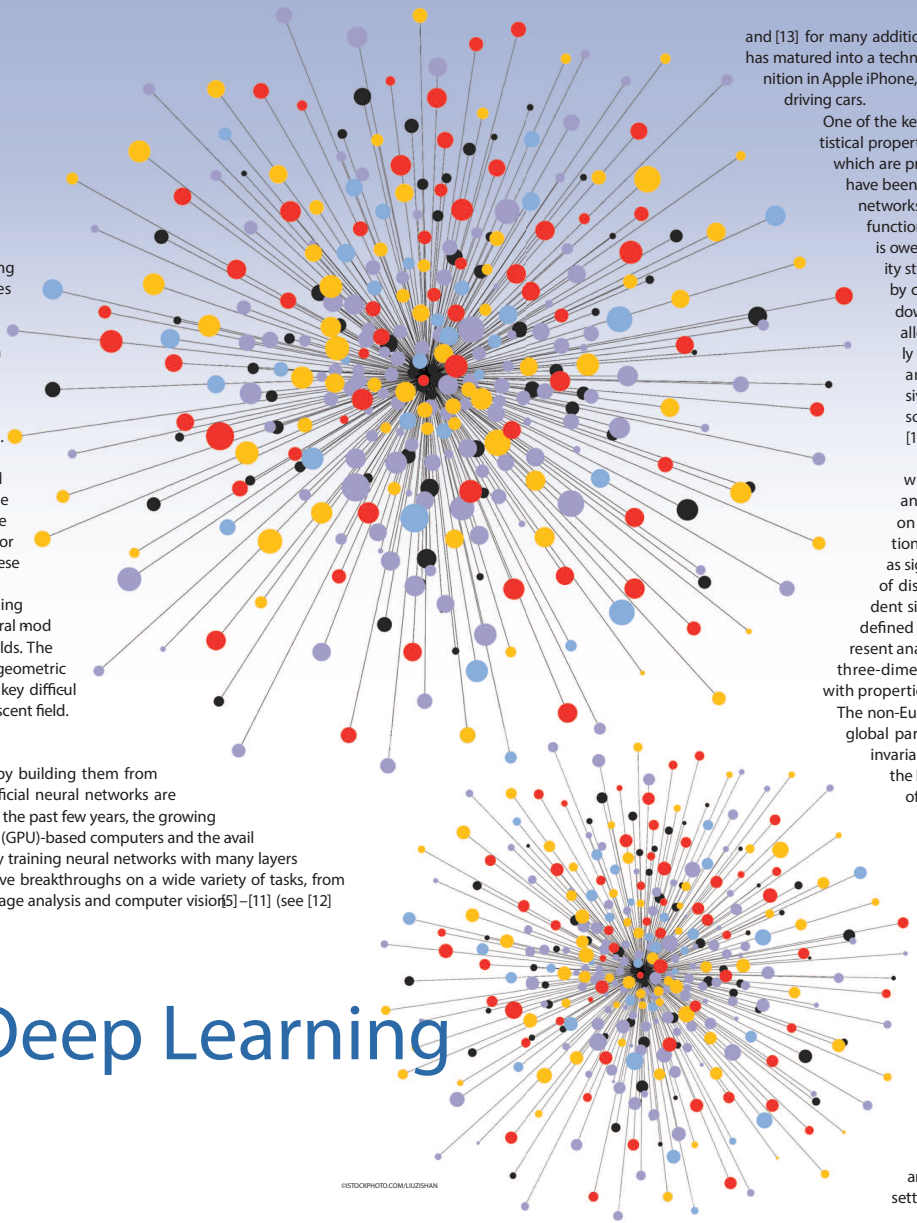


Parametric domain

What this tutorial is about?

- Basics of deep learning
- Basics of graph theory and differential geometry
- Spectral analysis on graphs and manifolds
- Spectral-domain geometric deep learning methods
- Spatial-domain geometric deep learning methods
- Applications: network analysis, recommender systems, computer graphics and vision, chemistry, high-energy physics, drug design, etc.

Michael M. Bronstein, Joan Bruna, Yann LeCun,
Arthur Szlam, and Pierre Vandergheynst



Geometric Deep Learning

Going beyond Euclidean data

Digital Object Identifier 10.1109/MSP.2017.2693418
Date of publication: 11 July 2017

©ISTOCKPHOTO.COM/LIUZISHAN

Many scientific fields study data with an underlying structure that is non-Euclidean. Some examples include social networks in computational social sciences, sensor networks in communications, functional networks in brain imaging, regulatory networks in genetics, and meshed surfaces in computer graphics. In many applications, such geometric data are large and complex (in the case of social networks, on the scale of billions) and are natural targets for machine-learning techniques. In particular, we would like to use deep neural networks, which have recently proven to be powerful tools for a broad range of problems from computer vision, natural-language processing, and audio analysis. However, these tools have been most successful on data with an underlying Euclidean or grid-like structure and in cases where the invariances of these structures are built into networks used to model them.

Geometric deep learning is an umbrella term for emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains, such as graphs and manifolds. The purpose of this article is to overview different examples of geometric deep-learning problems and present available solutions, key difficulties, applications, and future research directions in this nascent field.

Overview of deep learning

Deep learning refers to learning complicated concepts by building them from simpler ones in a hierarchical or multilayer manner. Artificial neural networks are popular realizations of such deep multilayer hierarchies. In the past few years, the growing computational power of modern graphics processing unit (GPU)-based computers and the availability of large training data sets have allowed successfully training neural networks with many layers and degrees of freedom (DoF)[1]. This has led to qualitative breakthroughs on a wide variety of tasks, from speech recognition[2], [3] and machine translation[4] to image analysis and computer vision[5]–[11] (see [12] and [13] for many additional examples of successful applications of deep learning). Today, deep learning has matured into a technology that is widely used in commercial applications, including Siri speech recognition in Apple iPhone, Google text translation, and Mobileye vision-based technology for autonomously driving cars.

One of the key reasons for the success of deep neural networks is their ability to leverage statistical properties of the data, such as stationarity and compositionality through local statistics, which are present in natural images, video, and speech[14], [15]. These statistical properties have been related to physics[16] and formalized in specific classes of convolutional neural networks (CNNs) [17]–[19]. In image analysis applications, one can consider images as functions on the Euclidean space (plane), sampled on a grid. In this setting, stationarity is owed to shift invariance, locality is due to the local connectivity, and compositionality stems from the multiresolution structure of the grid. These properties are exploited by convolutional architectures[20], which are built of alternating convolutional and downsampling (pooling) layers. The use of convolutions has a twofold effect. First, it allows extracting local features that are shared across the image domain and greatly reduces the number of parameters in the network with respect to generic deep architectures (and thus also the risk of overfitting), without sacrificing the expressive capacity of the network. Second, the convolutional architecture itself imposes some priors about the data, which appear very suitable especially for natural images [17]–[19], [21].

While deep-learning models have been particularly successful when dealing with speech, image, and video signals, in which there are an underlying Euclidean structure, recently there has been a growing interest in trying to apply learning on non-Euclidean geometric data. Such kinds of data arise in numerous applications. For instance, in social networks, the characteristics of users can be modeled as signals on the vertices of the social graph[22]. Sensor networks are graph models of distributed interconnected sensors, whose readings are modeled as time-dependent signals on the vertices. In genetics, gene expression data are modeled as signals defined on the regulatory network[23]. In neuroscience, graph models are used to represent anatomical and functional structures of the brain. In computer graphics and vision, three-dimensional (3-D) objects are modeled as Riemannian manifolds (surfaces) endowed with properties such as color texture.

The non-Euclidean nature of such data implies that there are no such familiar properties as global parameterization, common system of coordinates, vector space structure, or shift invariance. Consequently, basic operations like convolution that are taken for granted in the Euclidean case are even not well defined on non-Euclidean domains. The purpose of this article is to show different methods of translating the key ingredients of successful deep-learning methods, such as CNNs, to non-Euclidean data.

Geometric learning problems

Broadly speaking, we can distinguish between two classes of geometric learning problems. In the first class of problems, the goal is to characterize the structure of the data. The second class of problems deals with analyzing functions defined on a given non-Euclidean domain. These two classes are related, because understanding the properties of functions defined on a domain conveys certain information about the domain, and vice versa, the structure of the domain imposes certain properties on the functions on it.

Structure of the domain

As an example of the first class of problems, assume to be given a set of data points with some underlying low-dimensional structure embedded into a high-dimensional Euclidean space. Recovering that low-dimensional structure is often referred to as *manifold learning* or nonlinear dimensionality reduction and is an instance of unsupervised learning (note that the notion of manifold in this setting can be considerably more general than a classical smooth manifold; see, e.g.,

1053-5888/17©2017IEEE

geometricdeeplearning.com

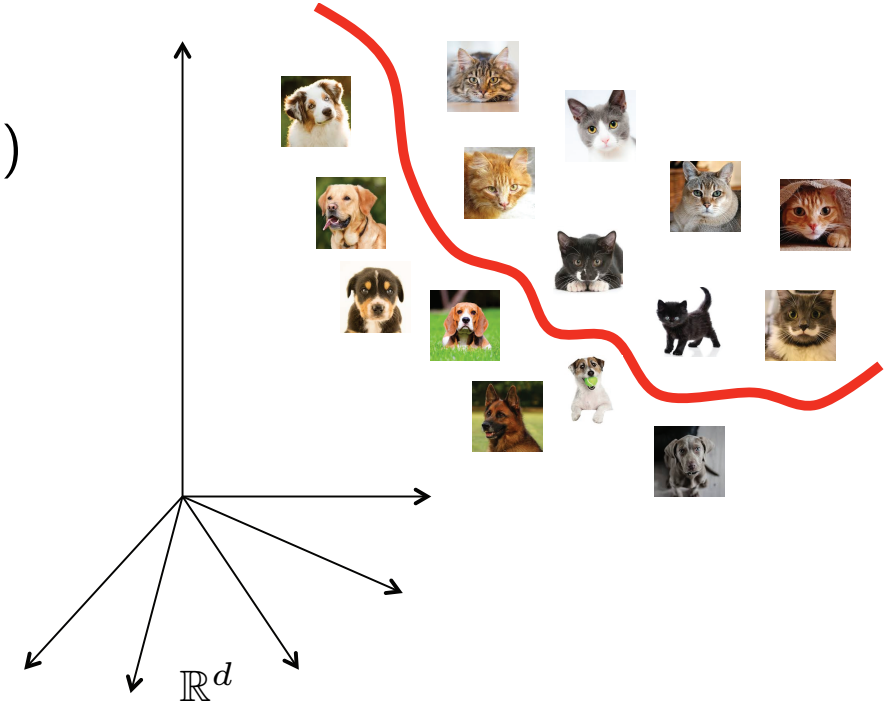
Machine Learning Basics

Supervised learning: classification example

- Data vectors $\mathbf{x} \in \mathbb{R}^d$
  (e.g. for 512×512 images $d \approx 10^5$)
- Unknown classification functional
  $f : \mathbb{R}^d \to \{1, \ldots, L\}$ in L classes
- Training set

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

- Parametric model $f_{\boldsymbol{\Theta}}$ of f

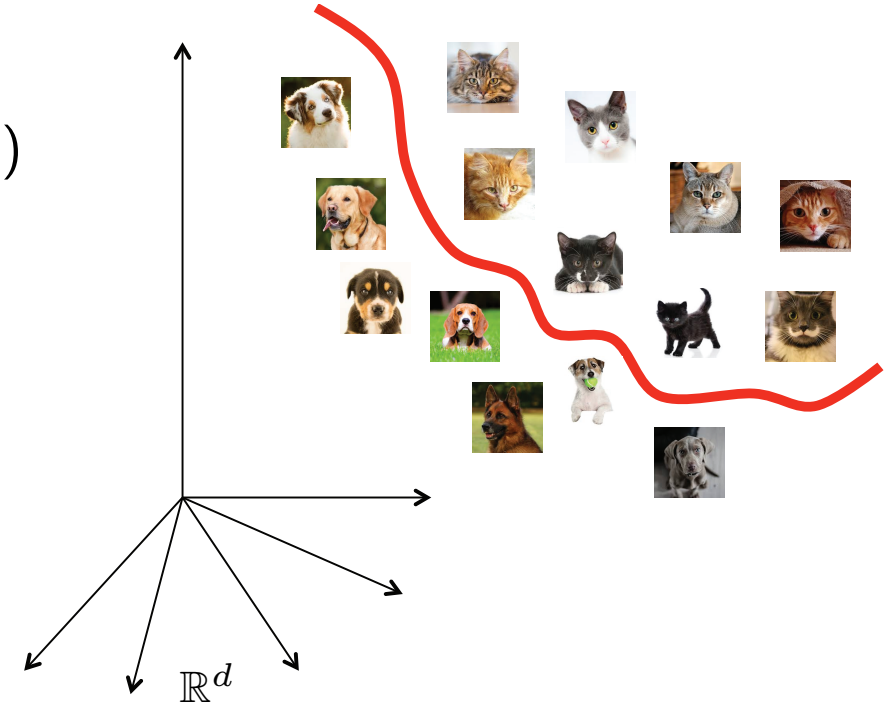


Supervised learning: classification example

- Data vectors $\mathbf{x} \in \mathbb{R}^d$
  (e.g. for 512×512 images $d \approx 10^5$)
- Unknown classification functional
  $f : \mathbb{R}^d \to \{1, \ldots, L\}$ in L classes
- Training set

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

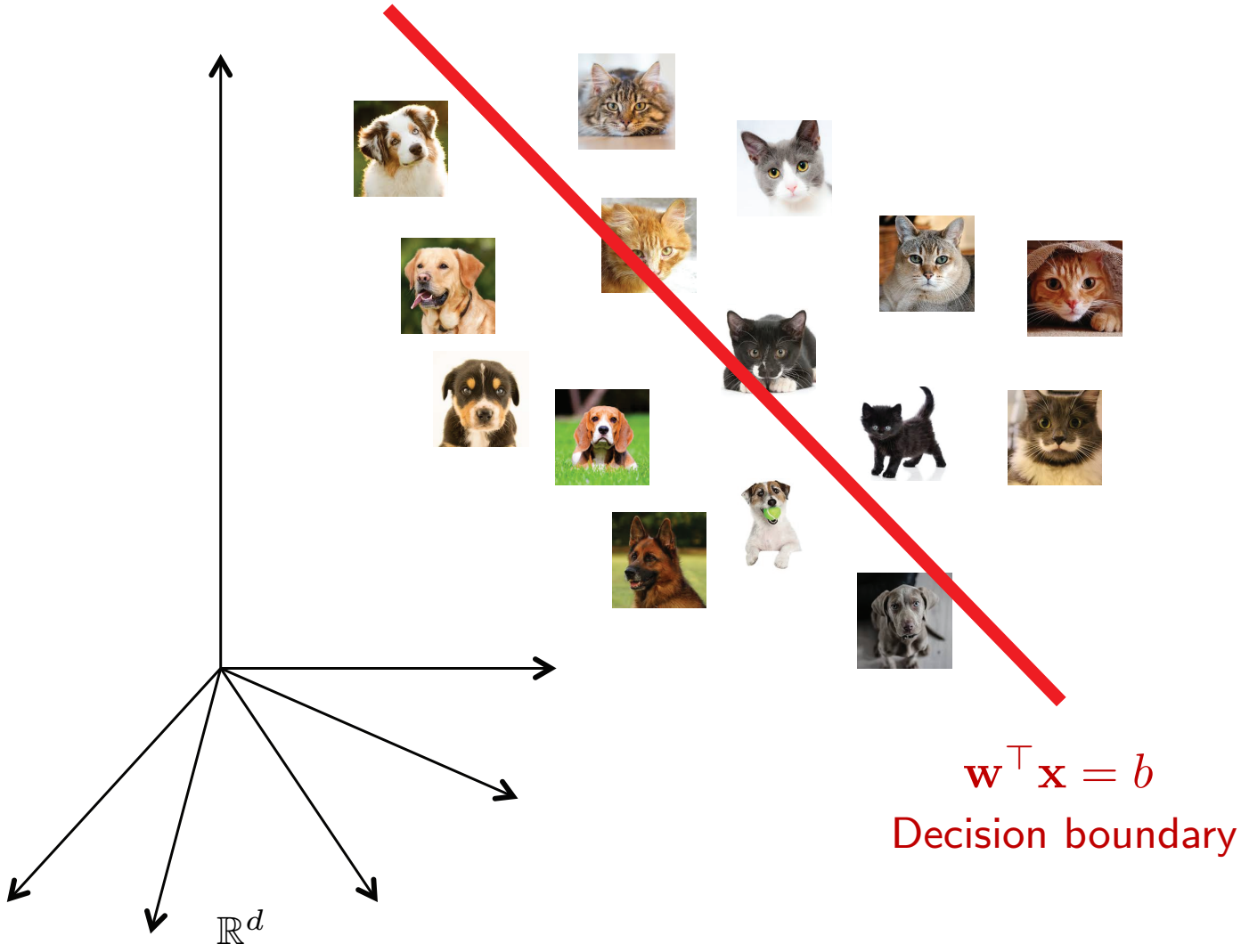
- Parametric model $f_{\mathbf{\Theta}}$ of f



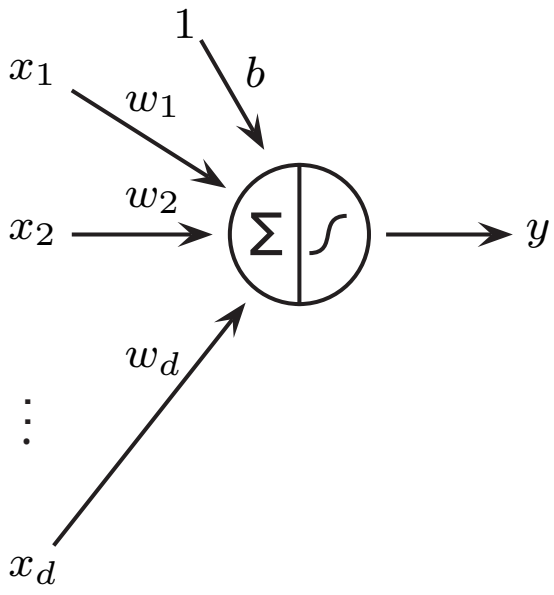
Supervised learning: find optimal model parameters by minimizing the loss ℓ on the training set

$$\mathbf{\Theta}^* = \underset{\mathbf{\Theta}}{\operatorname{argmin}} \sum_{i=1}^{T} \ell(f_{\mathbf{\Theta}}(\mathbf{x}_i), y_i)$$

Linear classifier



Simplest neural network: Perceptron



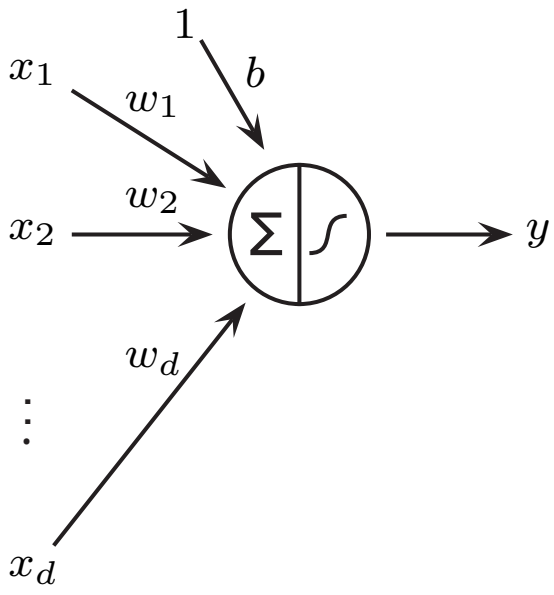
Linear layer $$y = \xi\left(\sum_{\ell=1}^{d} w_\ell x_\ell + b\right)$$

Activation, e.g. $\xi(x) = \tanh(x)$

Parameters layer weights $\mathbf{w} = (b, w_1, \ldots, w_d)$, including bias

Rosenblatt 1957

Simplest neural network: Perceptron



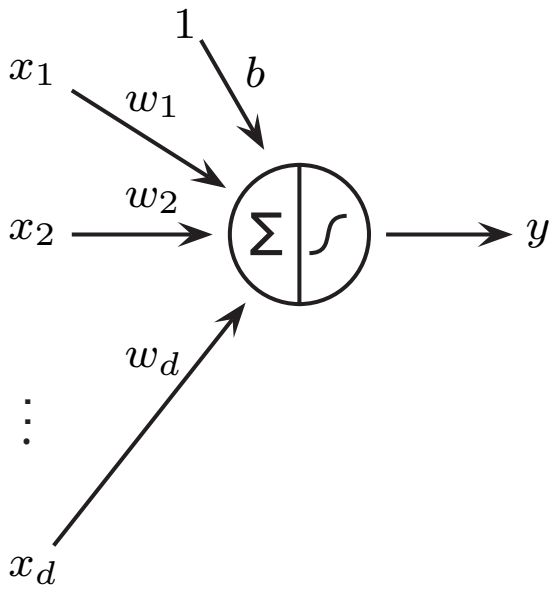
Linear layer $$y = \xi\left(\sum_{\ell=0}^{d} w_\ell x_\ell\right)$$

Activation, e.g. $\xi(x) = \tanh(x)$

Parameters layer weights $\mathbf{w} = (b, w_1, \ldots, w_d)$, including bias

Rosenblatt 1957

Simplest neural network: Perceptron



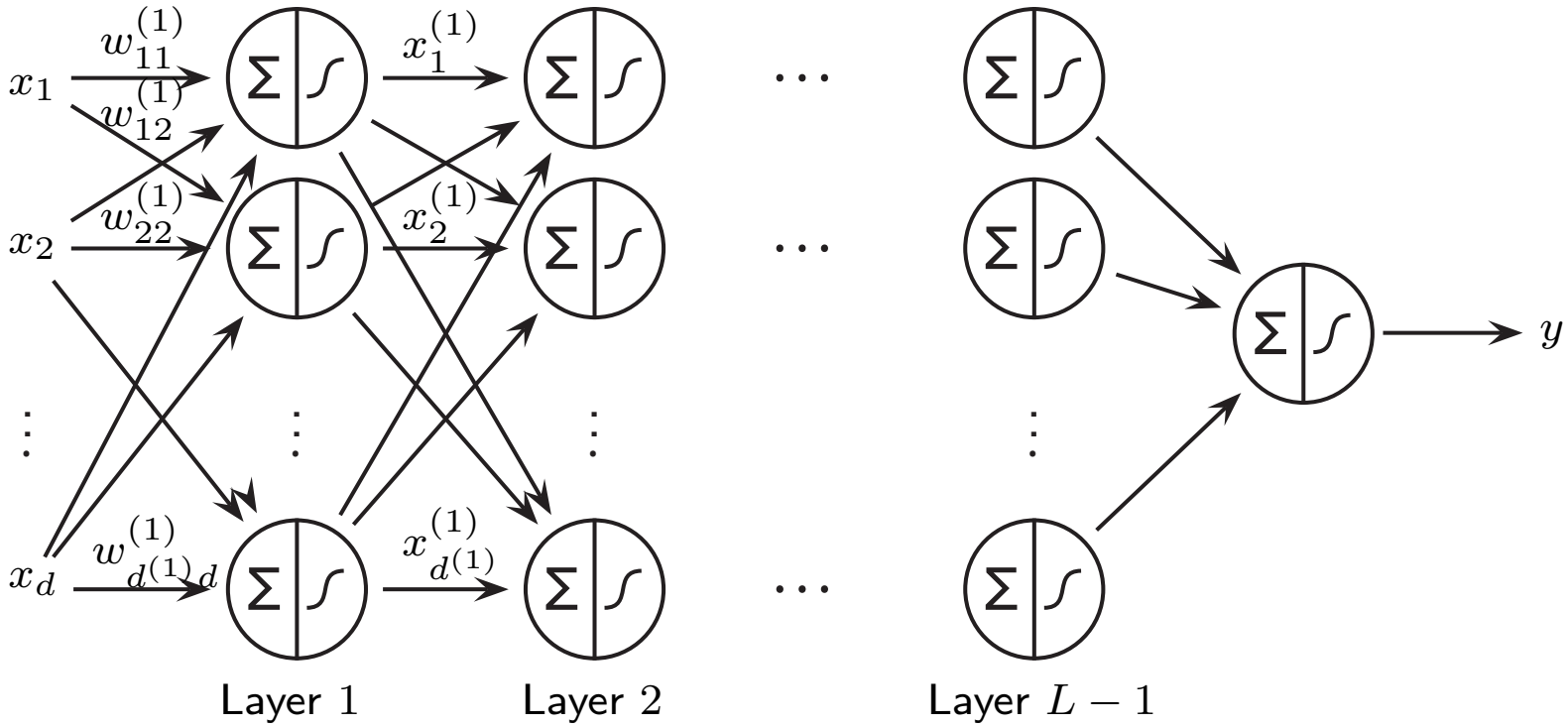
Linear layer $y = \xi\left(\sum_{\ell=0}^{d} w_\ell x_\ell\right) = \xi\left(\mathbf{w}^\top \mathbf{x}\right)$

Activation, e.g. $\xi(x) = \tanh(x)$

Parameters layer weights $\mathbf{w} = (b, w_1, \ldots, w_d)$, including bias

Rosenblatt 1957

Multi-layer neural network

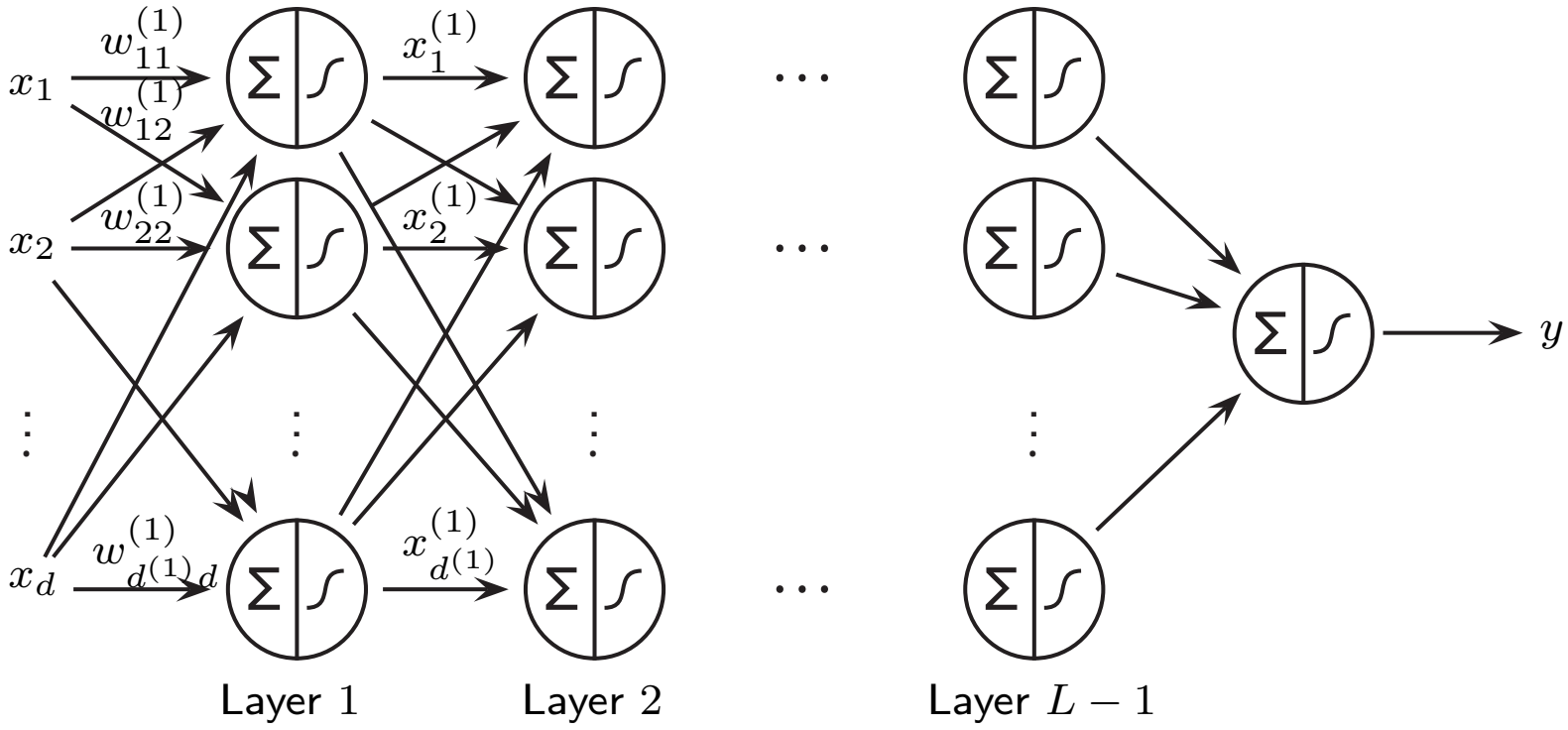


Linear layer $$x_{\ell'}^{(l+1)} = \xi\left(\sum_{\ell=0}^{d^{(l)}} w_{\ell'\ell}^{(l+1)} x_\ell^{(l)}\right)$$

Activation, e.g. $\xi(x) = \tanh(x)$

Parameters layer weights $\mathbf{W}^{(l+1)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$

Multi-layer neural network



Linear layer $\mathbf{x}^{(l+1)} = \xi\left(\mathbf{W}^{(l+1)}\mathbf{x}^{(l)}\right)$

Activation, e.g. $\xi(x) = \tanh(x)$

Parameters layer weights $\mathbf{W}^{(l+1)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$

Neural nets as universal approximators

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

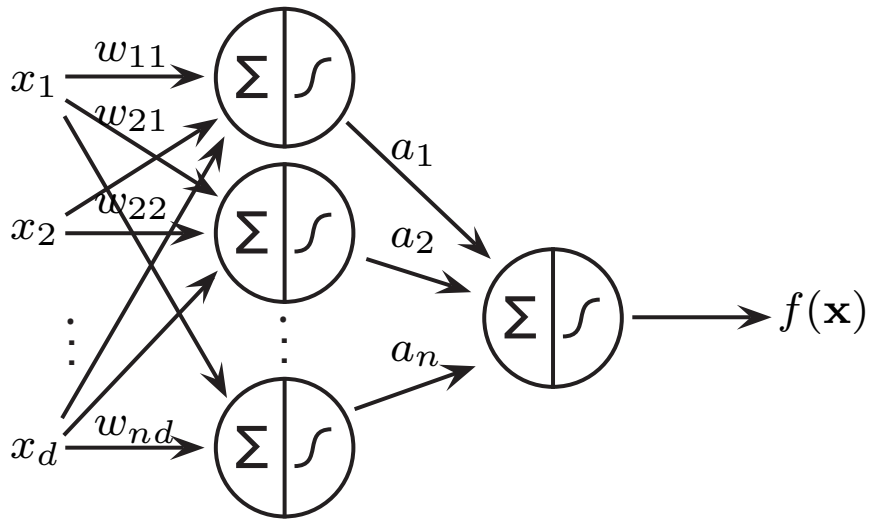
$$\left| \sum_{i=1}^{n} a_i \xi(\mathbf{w}_i^\top \mathbf{x} + b_i) - f(\mathbf{x}) \right| < \epsilon \qquad \forall \mathbf{x} \in [0,1]^d$$

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, $f : [0,1]^d \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times d}$ s.t.

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

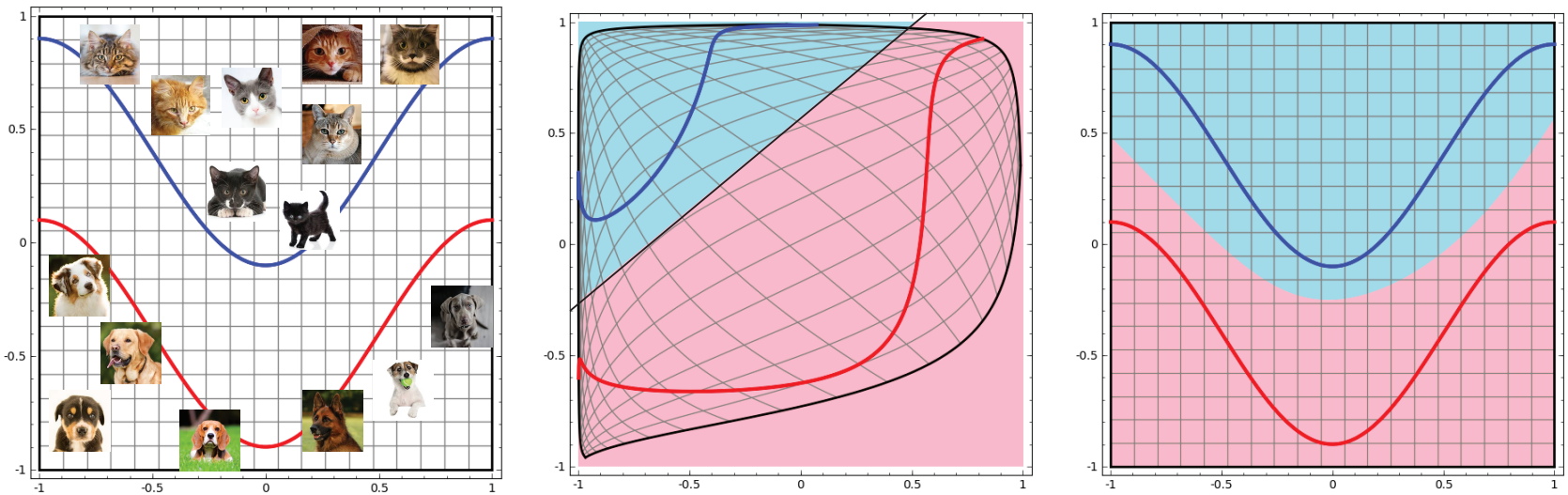


Cybenko 1989; Hornik 1991

Neural nets as universal approximators

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

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



Cybenko 1989; Hornik 1991 (Image: Christopher Olah)

Neural nets as universal approximators

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

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

☺ 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, $f : [0,1]^d \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times d}$ s.t.

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

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

☹ How many neurons?

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, $f : [0,1]^d \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times d}$ s.t.

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

- ☺ Any continuous function can be approximated arbitrarily well by a neural network with a single hidden layer
- ☹ How many neurons?
- ☹ 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, $f : [0,1]^d \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times d}$ s.t.

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

- ☺ Any continuous function can be approximated arbitrarily well by a neural network with a single hidden layer
- ☹ How many neurons?
- ☹ 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, $f : [0,1]^d \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times d}$ s.t.

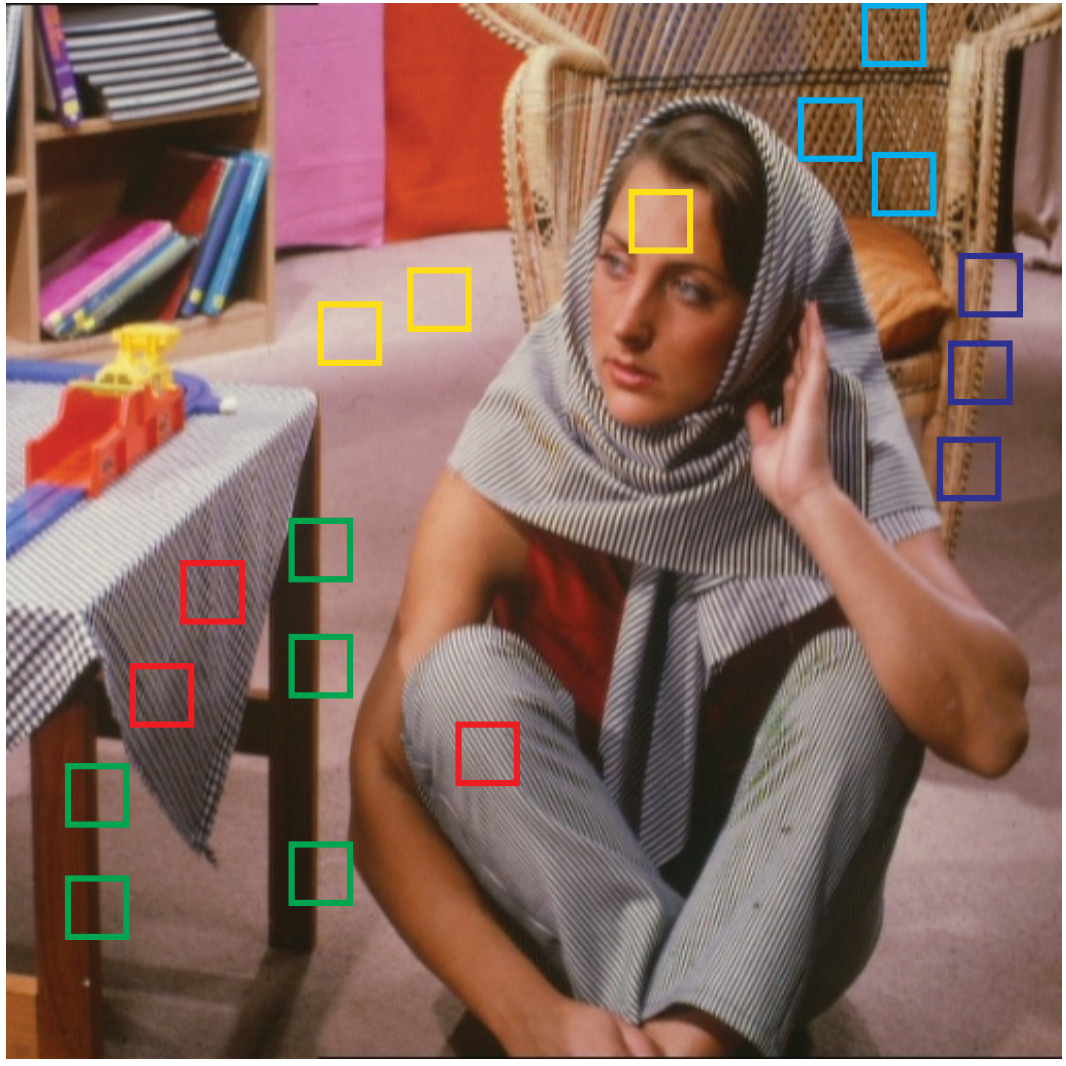
$$\left| \sum_{i=1}^{n} a_i \xi(\mathbf{w}_i^\top \mathbf{x} + b_i) - f(\mathbf{x}) \right| < \epsilon \qquad \forall \mathbf{x} \in [0,1]^d$$

- ☺ Any continuous function can be approximated arbitrarily well by a neural network with a single hidden layer
- ☹ How many neurons?
- ☹ 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

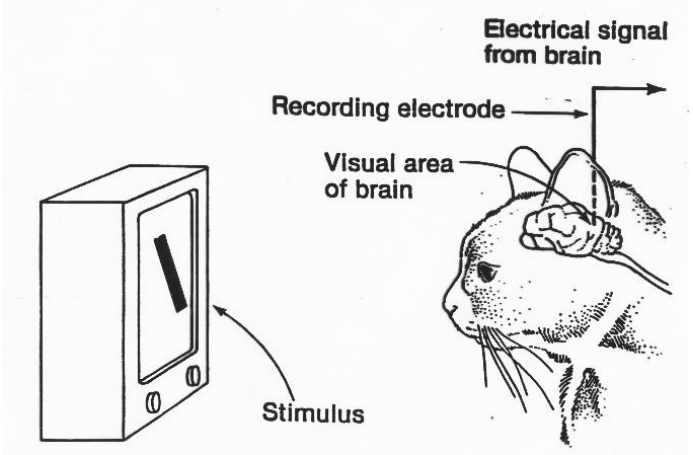
Take advantage of the structure of the data!

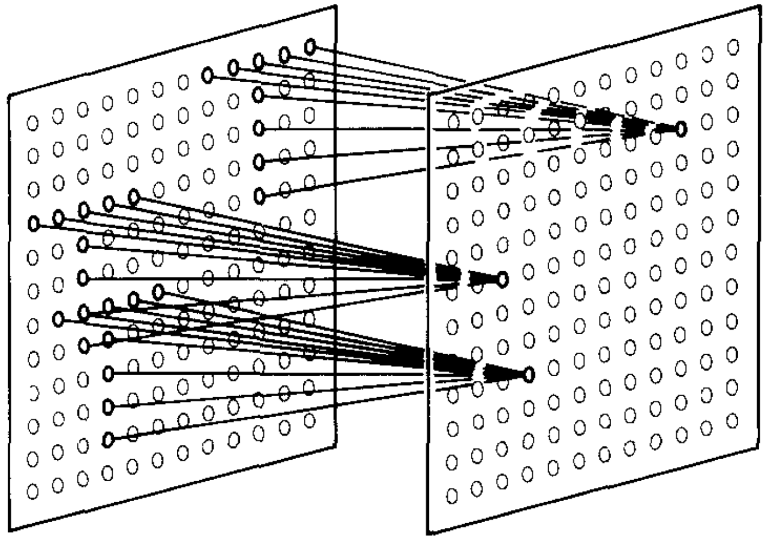
Stationarity and Self-similarity

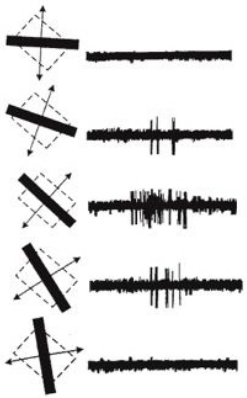


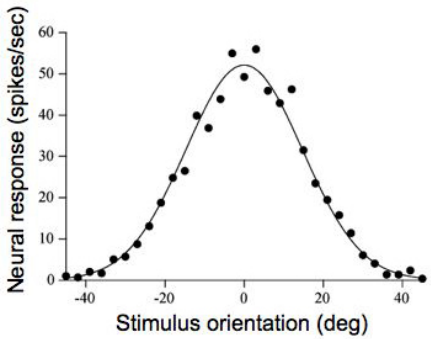
Data is self-similar across the domain

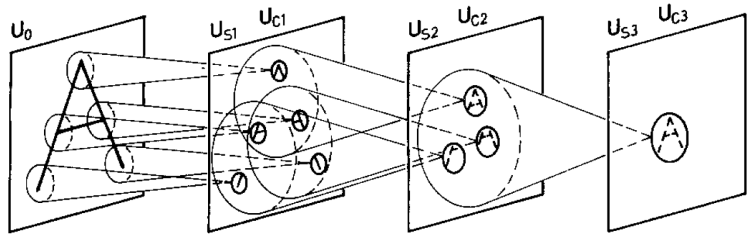
Locality









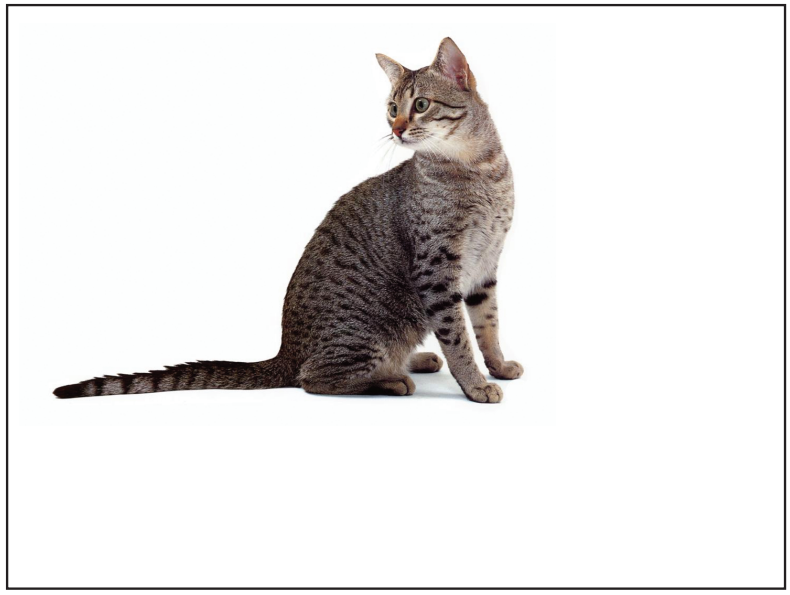


Visual cortex cell structure (local receptive fields) emulated in Neocognitron neural network model

Hubel, Wiesel 1962; Fukushima 1980

Translation invariance (image classification tasks)





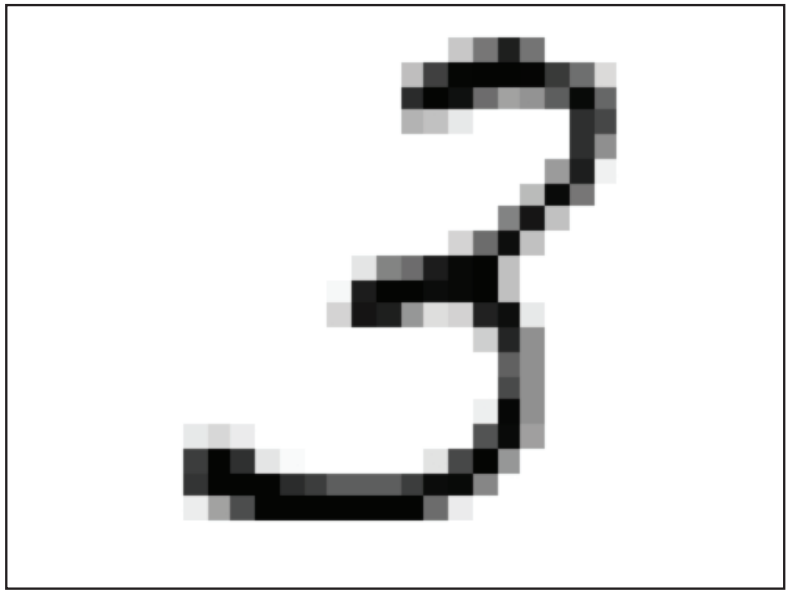
$$f(\mathcal{T}_{\mathbf{v}}x) = f(x) \quad \forall x, \mathbf{v}$$

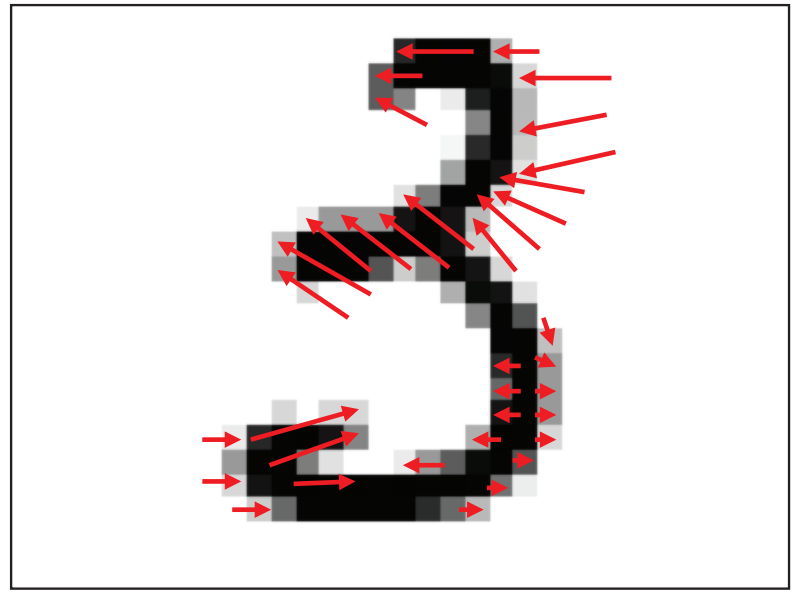
where

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

Bruna, Mallat 2012

Deformation invariance (image classification tasks)





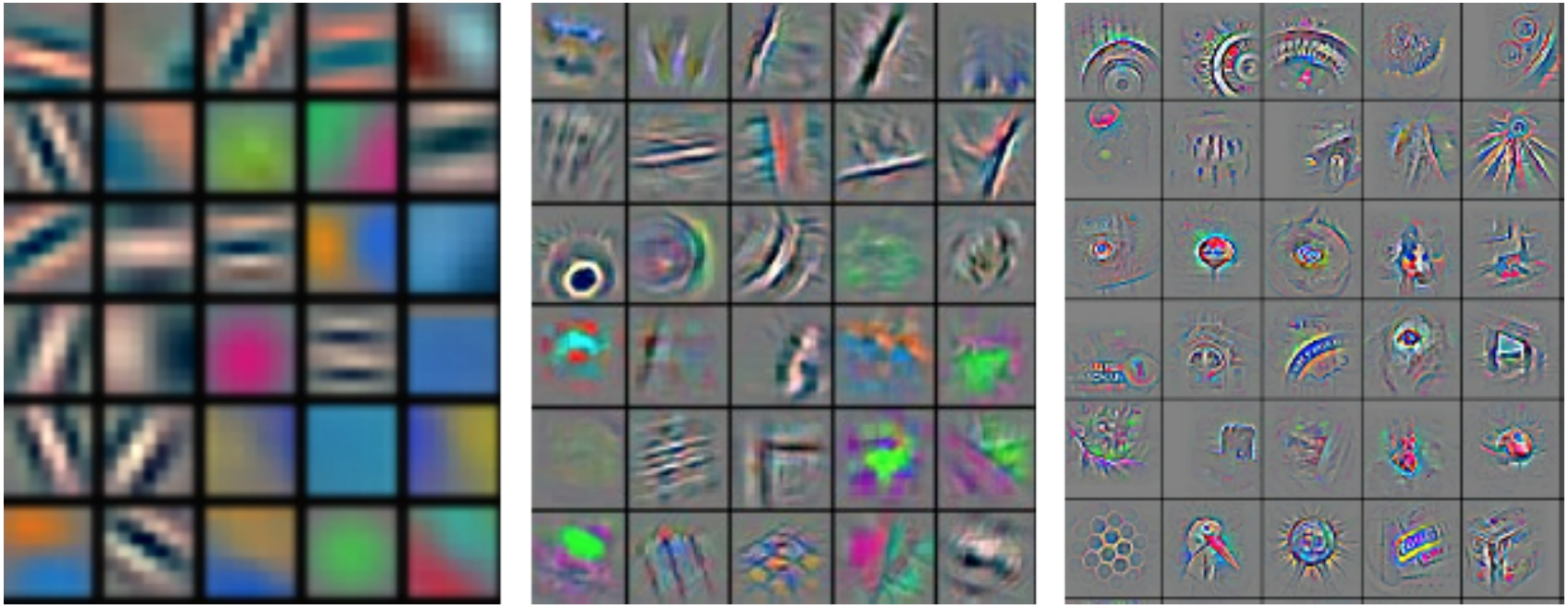
$$|f(\mathcal{L}_\tau x) - f(x)| \approx \|\nabla \tau\| \quad \forall f, \tau$$

where

- image is modeled as a function $x \in L^2([0,1]^2)$
- $\mathcal{L}_\tau x(\mathbf{u}) = x(\mathbf{u} - \tau(\mathbf{u}))$ is a warping operator
- $\tau : [0,1]^2 \to [0,1]^2$ is a smooth deformation field
- $f : L^2([0,1]^2) \to \{1, \ldots, L\}$ is a classification functional

Bruna, Mallat 2012

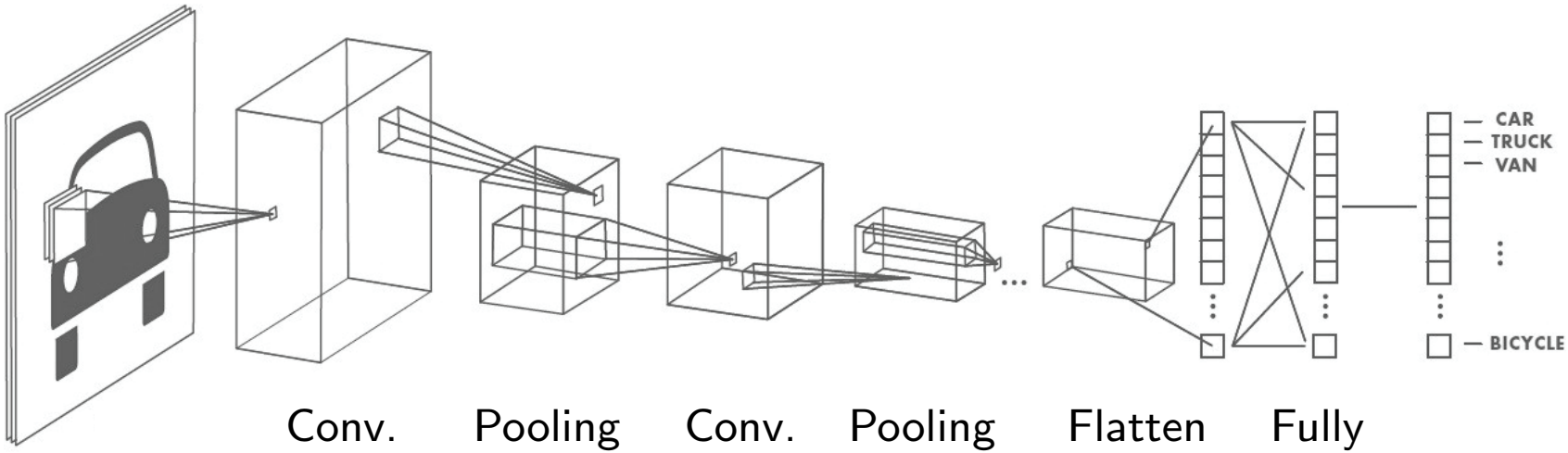
Hierarchy and Compositionality



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

Zeiler, Fergus 2013

Convolutional Neural Networks (CNN)



Conv. Pooling Conv. Pooling Flatten Fully connected

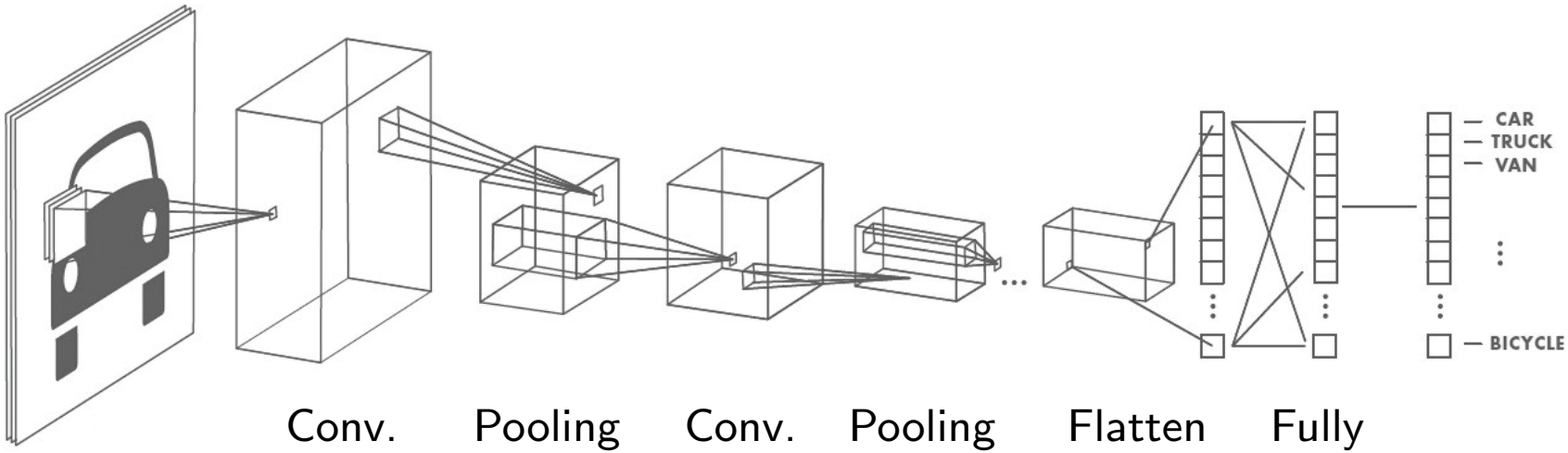
Conv. layer $$x_{\ell'}^{(l+1)}(\mathbf{u}) = \xi\left(\sum_{\ell=1}^{d^{(l)}}(w_{\ell'\ell}^{(l+1)} \star x_{\ell}^{(l)})(\mathbf{u})\right)$$

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

Parameters filters $W^{(1)}, \ldots, W^{(L)}$

LeCun et al. 1989 (Image: Debarko De)

Convolutional Neural Networks (CNN)



Conv. Pooling Conv. Pooling Flatten Fully connected

Conv. layer $$x_{\ell'}^{(l+1)}(\mathbf{u}) = \xi\left(\sum_{\ell=1}^{d^{(l)}}(w_{\ell'\ell}^{(l+1)} \star x_{\ell}^{(l)})(\mathbf{u})\right)$$

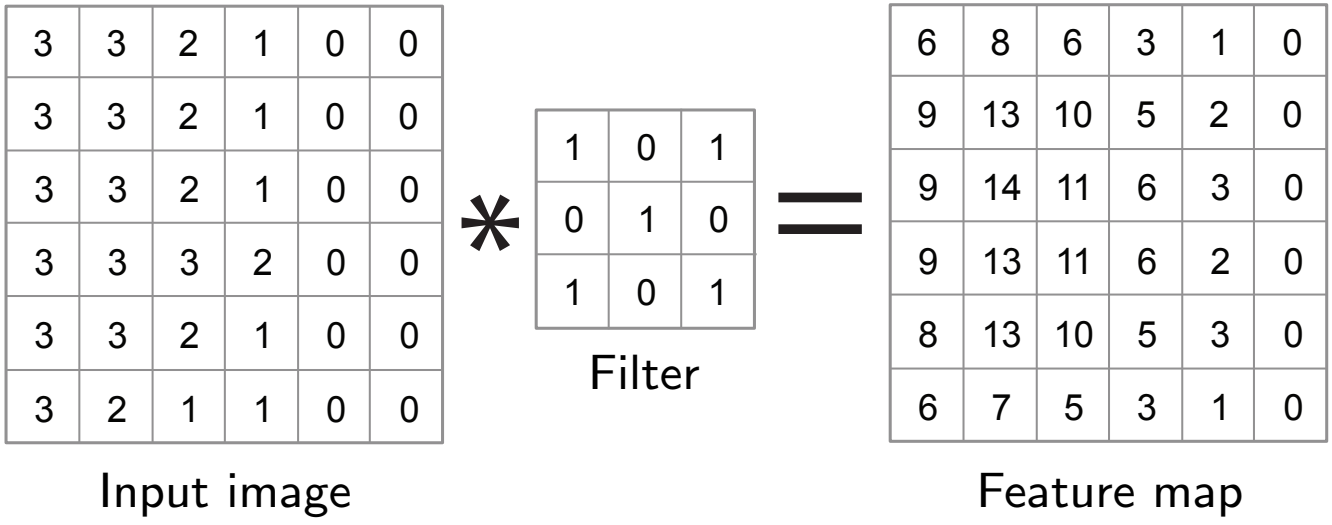
Activation, e.g. $\xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Parameters filters $W^{(1)}, \ldots, W^{(L)}$

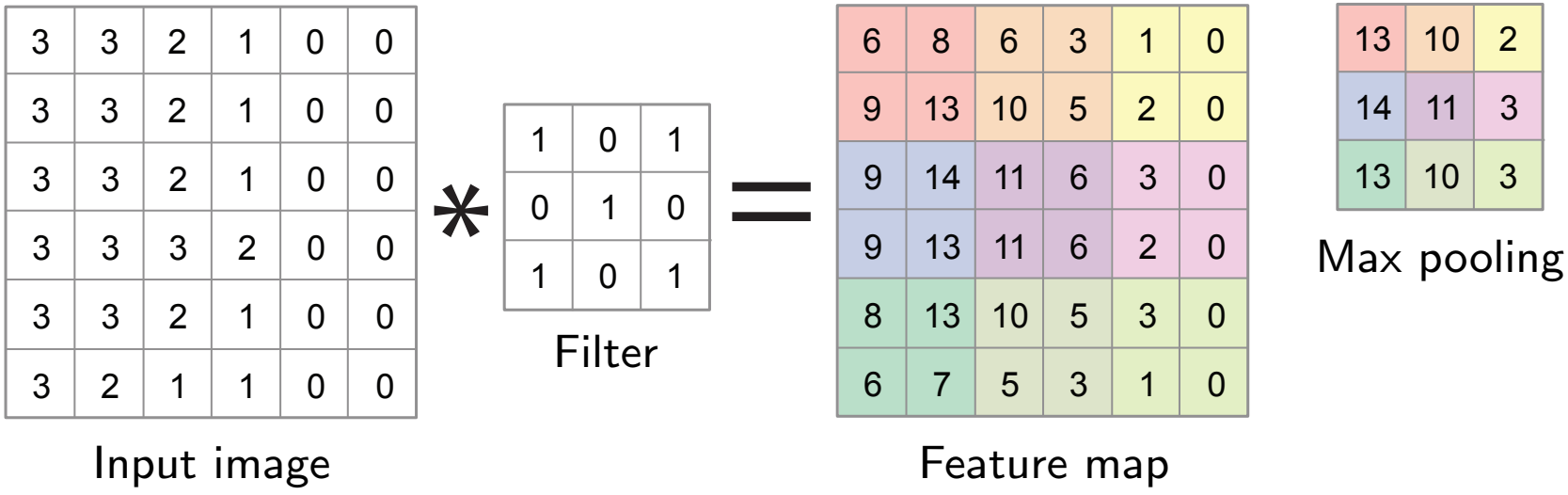
Pooling $x_{\ell}^{(l+1)}(\mathbf{u}) = \|x_{\ell}^{(l)}(\mathbf{u}') : \mathbf{u}' \in \mathcal{N}(\mathbf{u})\|_p \quad p = 1, 2, \text{ or } \infty$

LeCun et al. 1989 (Image: Debarko De)

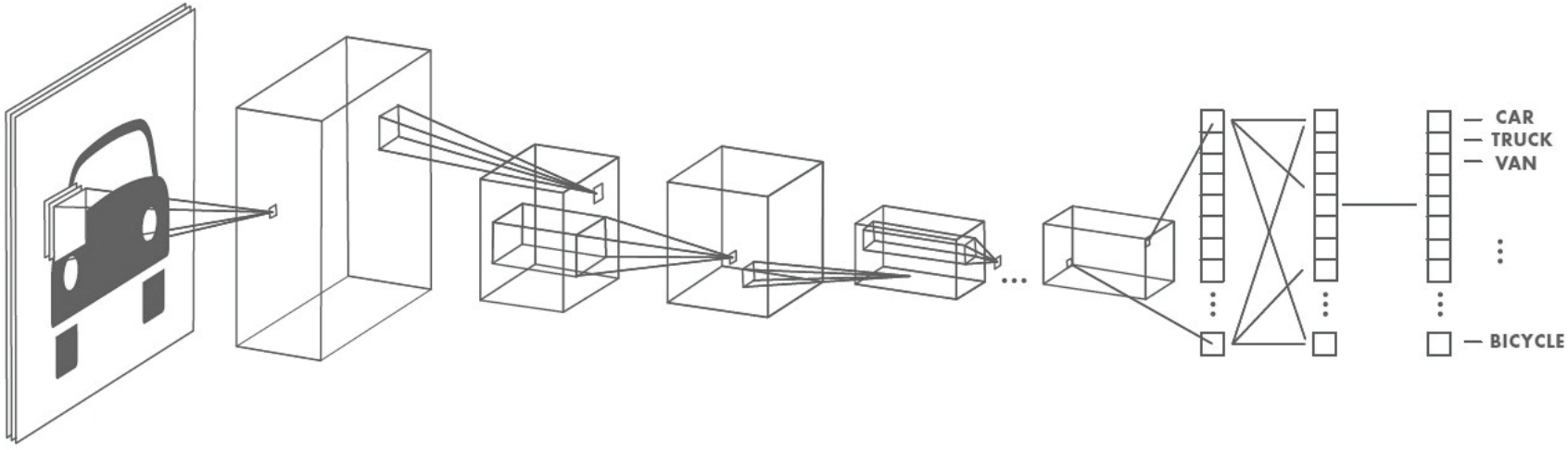
Pooling


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

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

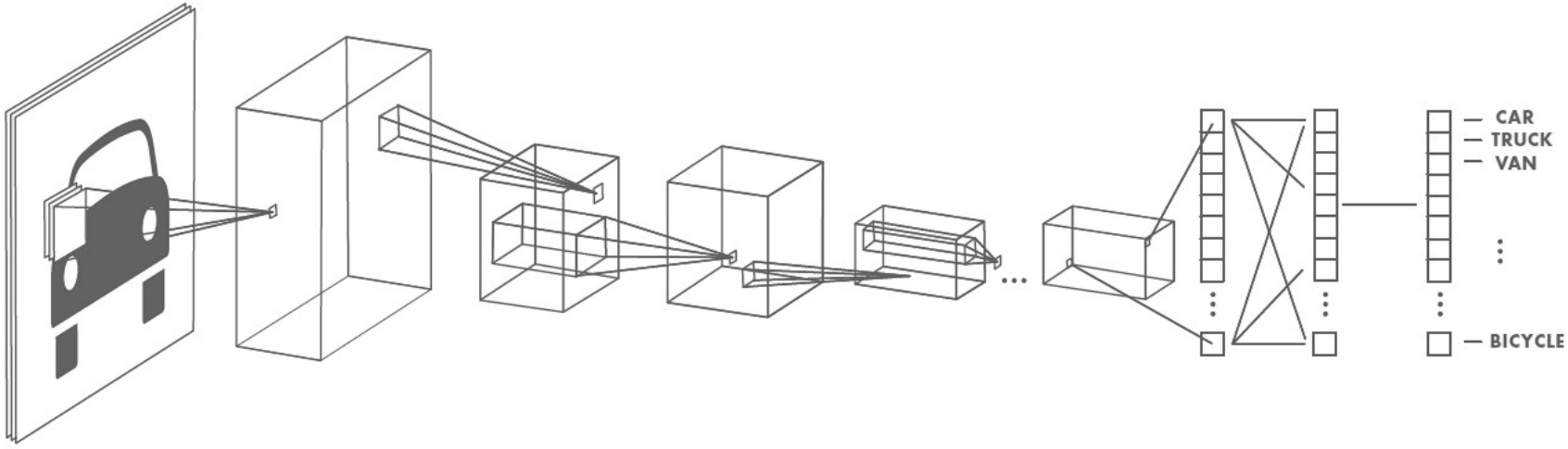
Key properties of CNNs



☺ Convolutional filters **(Translation invariance+Self-similarity)**

LeCun et al. 1989

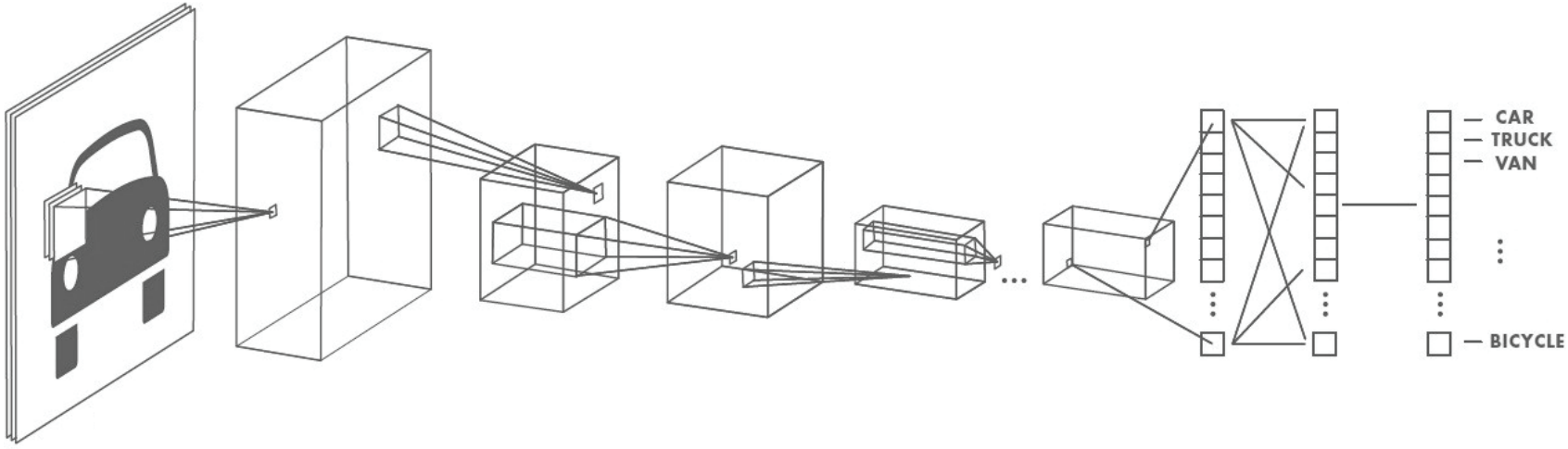
Key properties of CNNs



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

LeCun et al. 1989

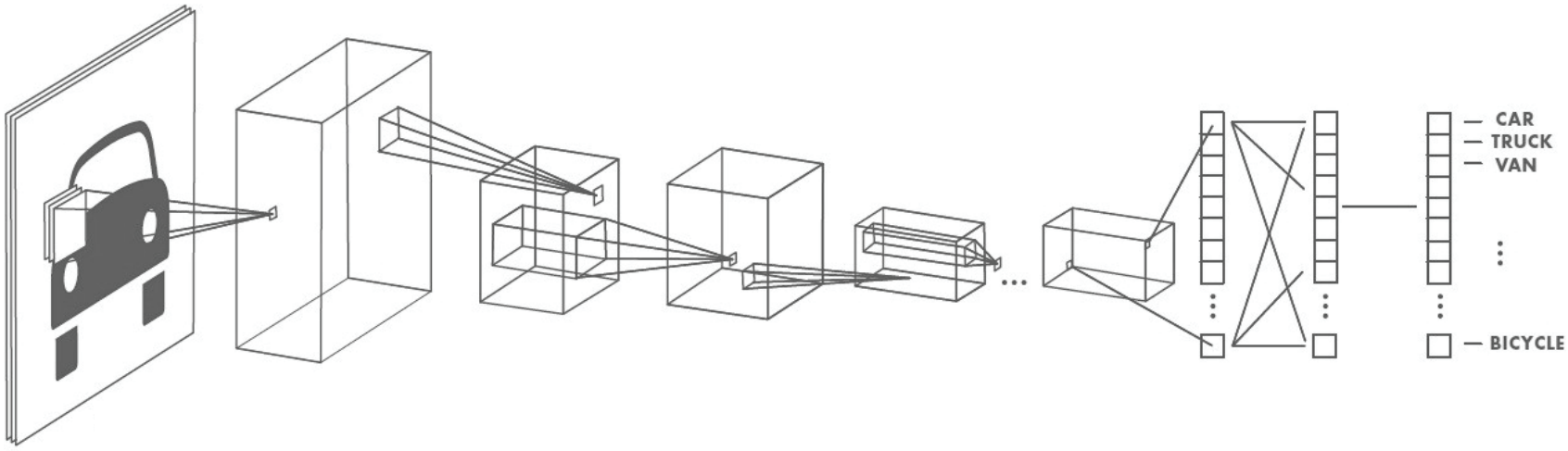
Key properties of CNNs



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

LeCun et al. 1989

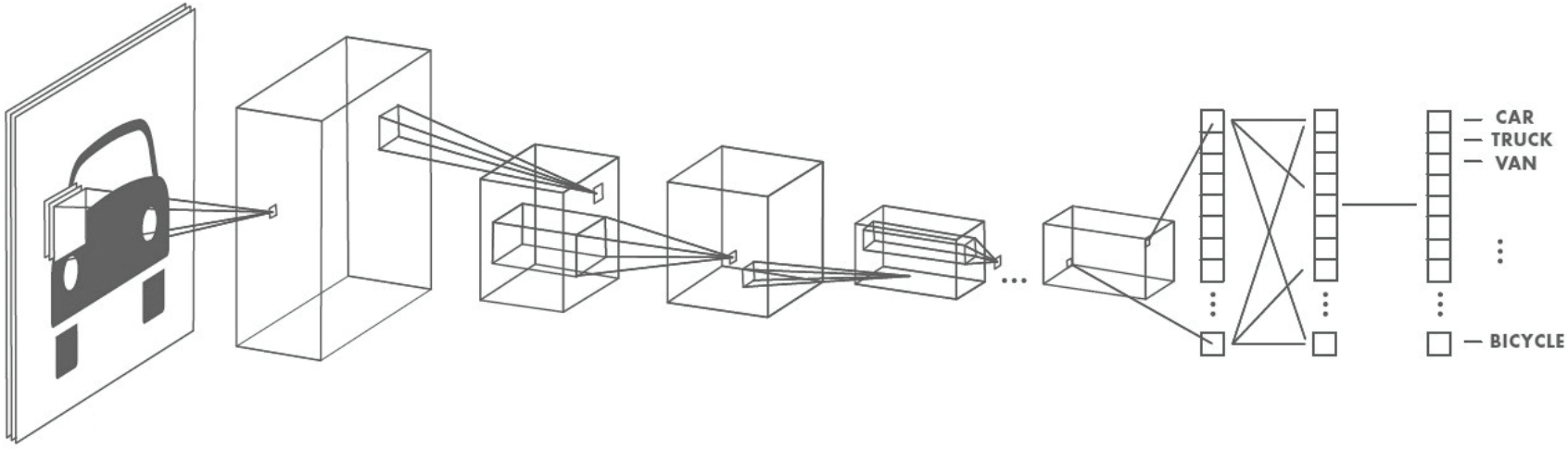
Key properties of CNNs



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

LeCun et al. 1989

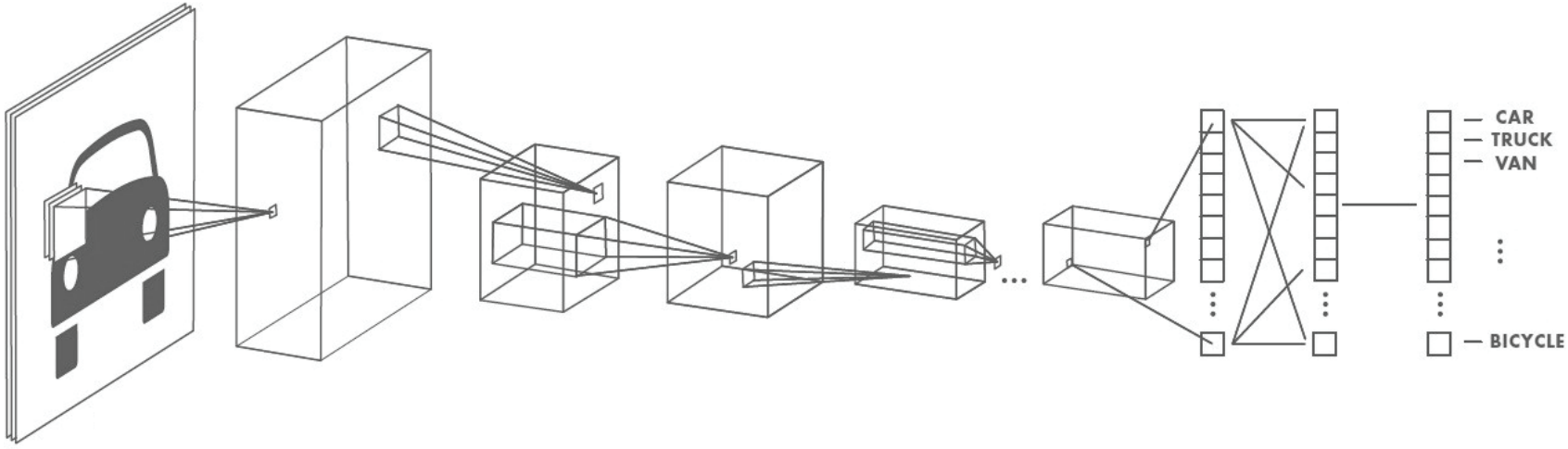
Key properties of CNNs



- ☺ Convolutional filters **(Translation invariance+Self-similarity)**
- ☺ Multiple layers **(Compositionality)**
- ☺ Filters localized in space **(Locality)**
- ☺ $\mathcal{O}(1)$ 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+Self-similarity)**
- ☺ Multiple layers **(Compositionality)**
- ☺ Filters localized in space **(Locality)**
- ☺ $\mathcal{O}(1)$ 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

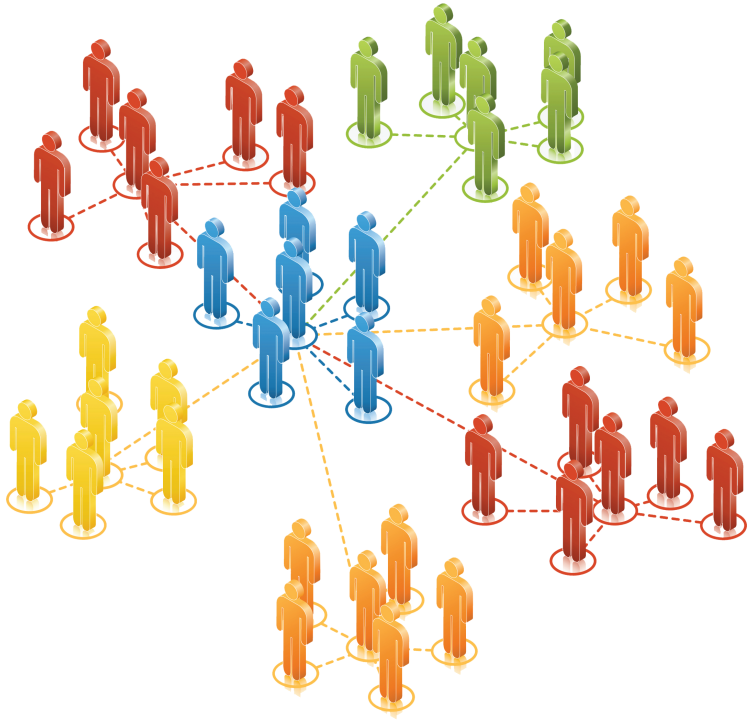
LeCun et al. 1989

Going non-Euclidean

Prototypical non-Euclidean objects



Manifolds



Graphs

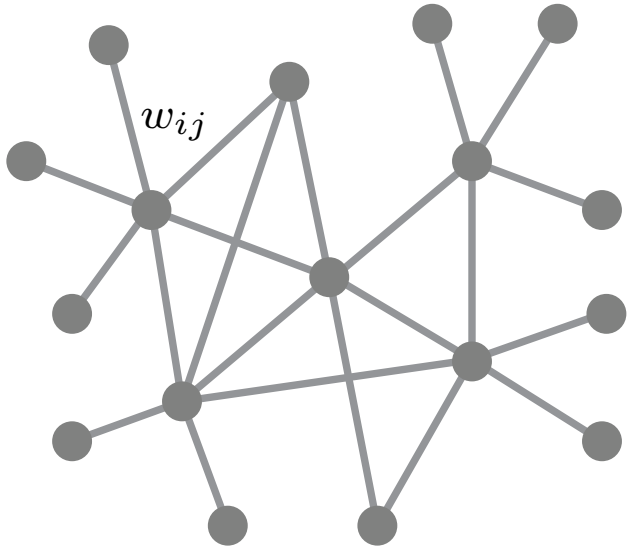
Challenges of geometric deep learning

- Extend neural network techniques to graph- or manifold-structured data
- Assumption: Non-Euclidean data are locally stationary and manifest hierarchical structures
- How to define compositionality? (convolution and pooling on graphs/manifolds)
- How to make them fast? (linear complexity)

Spectral analysis on graphs and manifolds

Graph theory in one minute

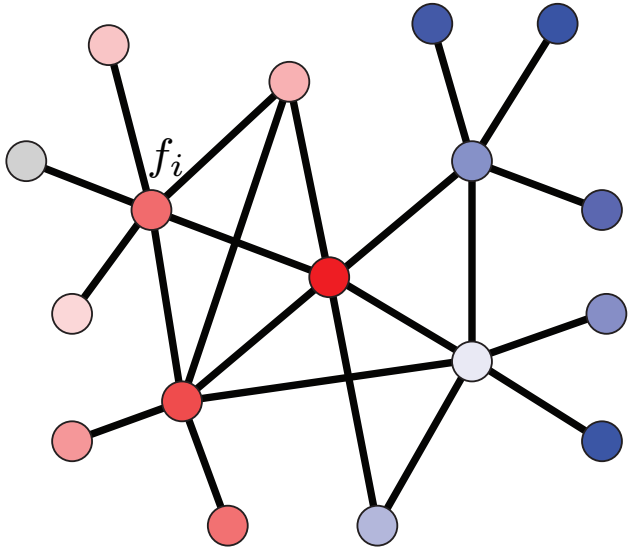
- Weighted undirected graph $\mathcal{G}$ with vertices $\mathcal{V} = \{1, \ldots, n\}$, edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and edge weights $w_{ij} \geq 0$ for $(i, j) \in \mathcal{E}$



Graph theory in one minute

- Weighted undirected graph $\mathcal{G}$ with vertices $\mathcal{V} = \{1, \ldots, n\}$, edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and edge weights $w_{ij} \geq 0$ for $(i, j) \in \mathcal{E}$

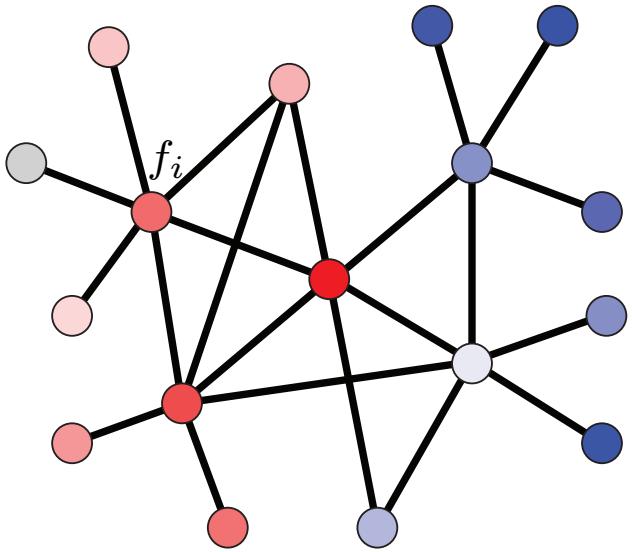
- Functions over the vertices $L^2(\mathcal{V}) = \{f : \mathcal{V} \to \mathbb{R}\}$



Graph theory in one minute

- Weighted undirected graph $\mathcal{G}$ with vertices $\mathcal{V} = \{1, \ldots, n\}$, edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and edge weights $w_{ij} \geq 0$ for $(i, j) \in \mathcal{E}$

- Functions over the vertices $L^2(\mathcal{V}) = \{f : \mathcal{V} \to \mathbb{R}\}$ represented as vectors $\mathbf{f} = (f_1, \ldots, f_n)$



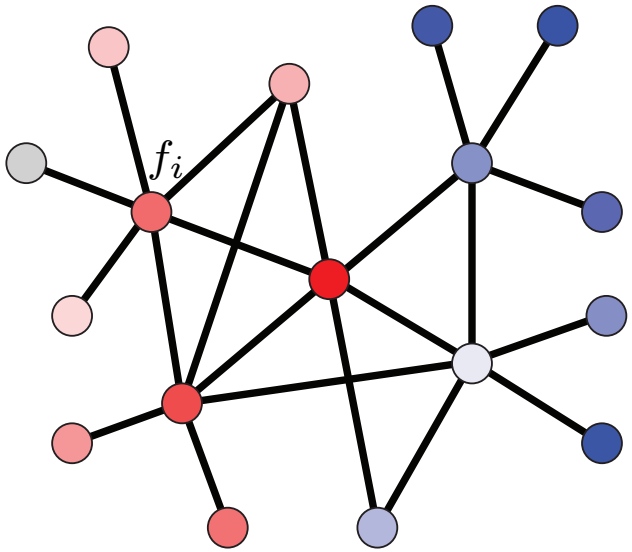
Graph theory in one minute

- Weighted undirected graph $\mathcal{G}$ with vertices $\mathcal{V} = \{1, \ldots, n\}$, edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and edge weights $w_{ij} \geq 0$ for $(i, j) \in \mathcal{E}$

- Functions over the vertices $L^2(\mathcal{V}) = \{f : \mathcal{V} \to \mathbb{R}\}$ represented as vectors $\mathbf{f} = (f_1, \ldots, f_n)$

- Hilbert space with inner product

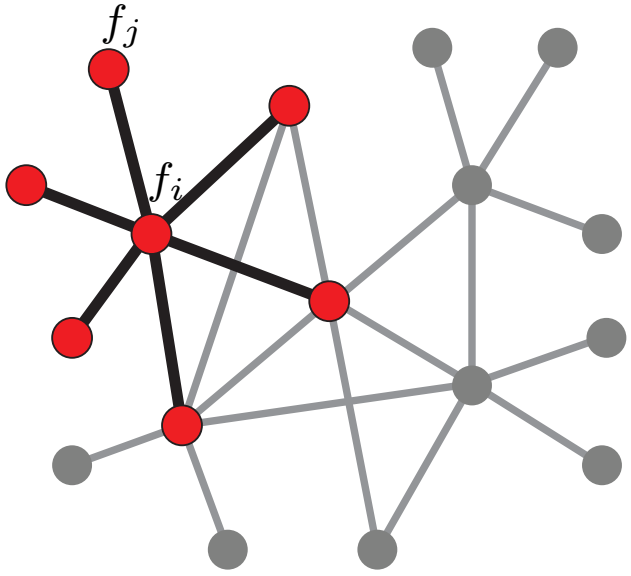
$$\langle f, g \rangle_{L^2(\mathcal{V})} = \sum_{i \in \mathcal{V}} f_i \, g_i = \mathbf{f}^\top \mathbf{g}$$



Graph Laplacian

- Unnormalized Laplacian $\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$

$$(\Delta f)_i \quad = \quad \sum_{j:(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j)$$

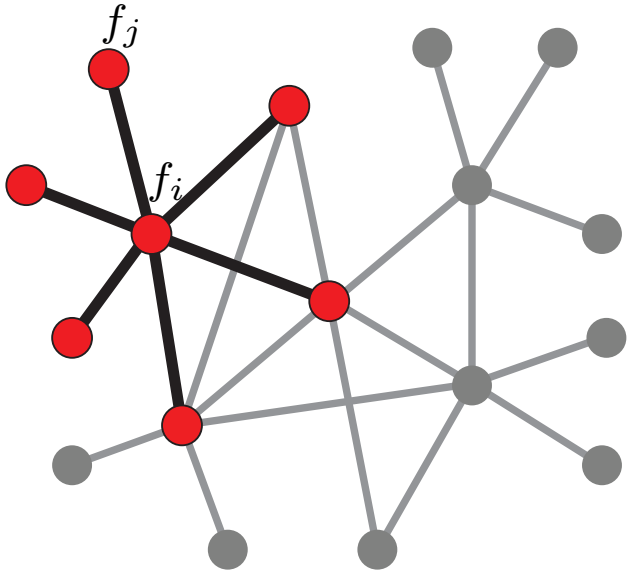


Graph Laplacian

- Unnormalized Laplacian $\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$

$$
\begin{aligned}
(\Delta f)_i &= \sum_{j:(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j) \\
&= f_i \sum_{j:(i,j)\in\mathcal{E}} w_{ij} - \sum_{j:(i,j)\in\mathcal{E}} w_{ij} f_j
\end{aligned}
$$

(up to scale) difference between f and its local average



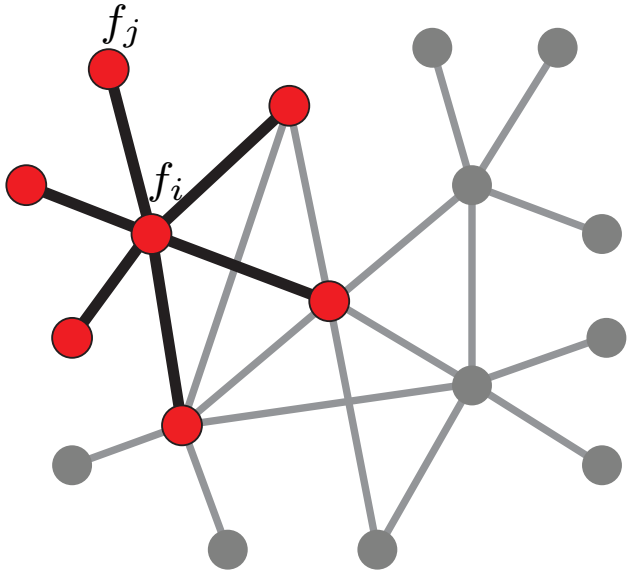
Graph Laplacian

- Unnormalized Laplacian $\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$

$$(\Delta f)_i \quad = \quad \sum_{j:(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j)$$

(up to scale) difference between f and its local average

- Represented as a positive semi-definite $n \times n$ matrix $\mathbf{\Delta} = \mathbf{D} - \mathbf{W}$ where $\mathbf{W} = (w_{ij})$ and $\mathbf{D} = \mathrm{diag}(\sum_{j\neq i} w_{ij})$

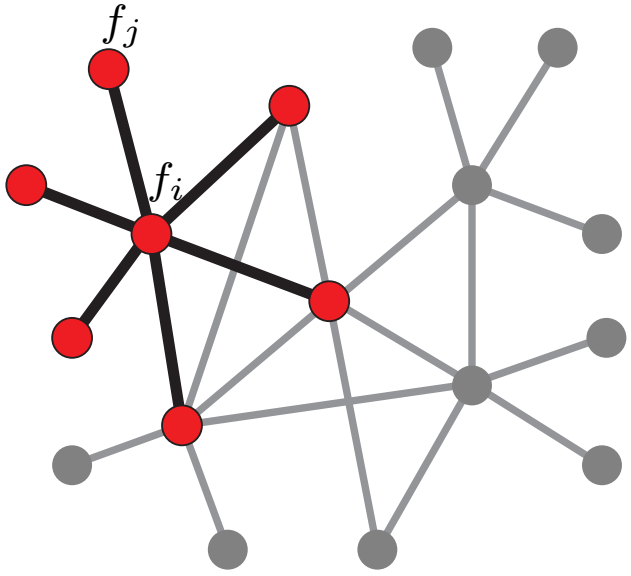


Graph Laplacian

- Unnormalized Laplacian $\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$

$$(\Delta f)_i \quad = \quad \sum_{j:(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j)$$

(up to scale) difference between f and its local average



- Represented as a positive semi-definite $n \times n$ matrix $\mathbf{\Delta} = \mathbf{D} - \mathbf{W}$ where $\mathbf{W} = (w_{ij})$ and $\mathbf{D} = \mathrm{diag}(\sum_{j\neq i} w_{ij})$

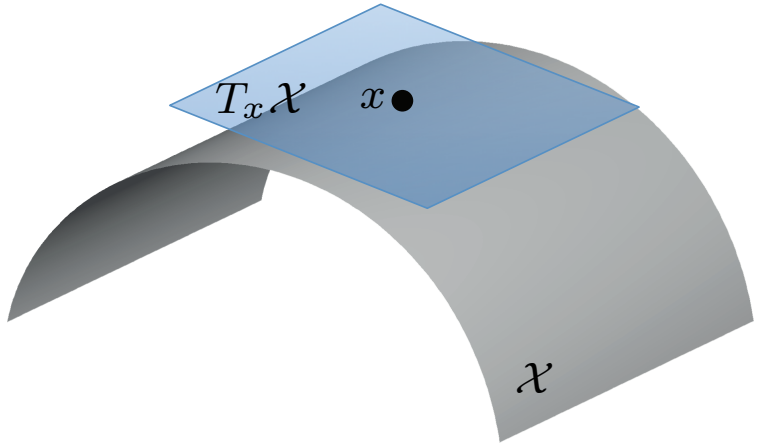
- Dirichlet energy of f

$$\|f\|^2_{\mathcal{G}} = \frac{1}{2}\sum_{ij=1}^{n} w_{ij}(f_i - f_j)^2 = \mathbf{f}^\top \mathbf{\Delta f}$$

measures the smoothness of f (how fast it changes locally)

Riemannian manifolds in one minute

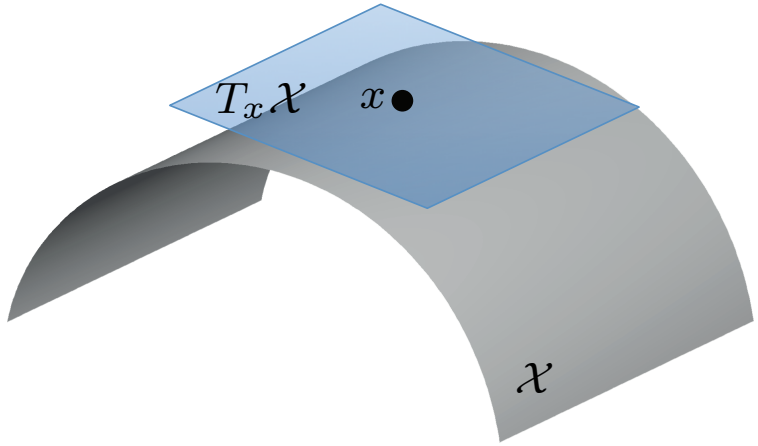
- Manifold $\mathcal{X}$ = topological space
- Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x



Riemannian manifolds in one minute

- Manifold $\mathcal{X}$ = topological space
- Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x
- Riemannian metric describes the local intrinsic structure at x

$$\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$$

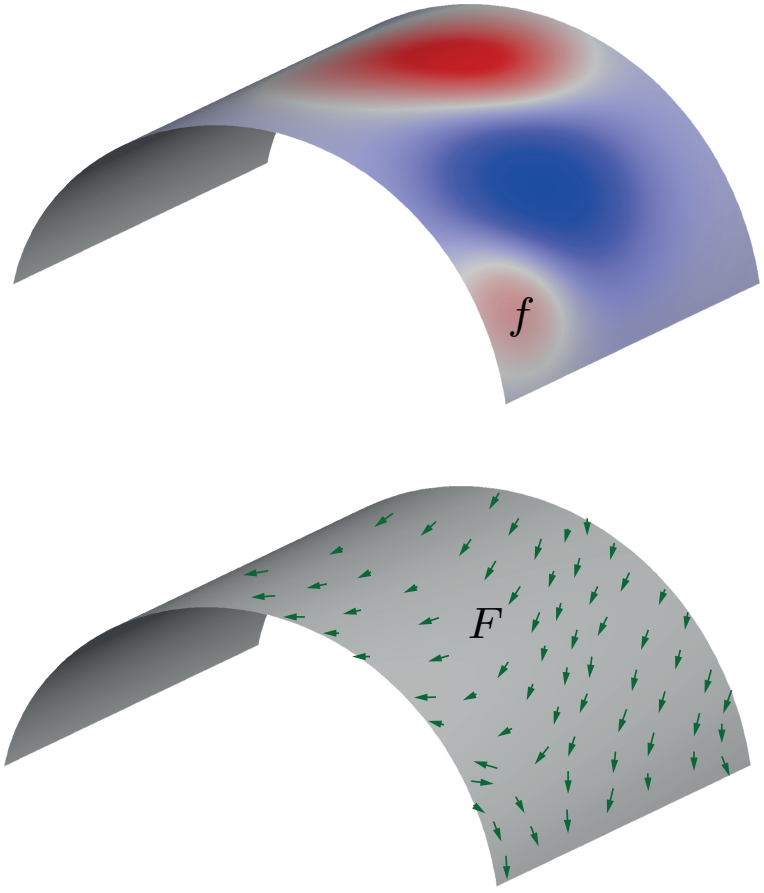


Riemannian manifolds in one minute

- Manifold $\mathcal{X}$ = topological space
- Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x
- Riemannian metric describes the local intrinsic structure at x

$$\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$$

- Scalar fields $f : \mathcal{X} \to \mathbb{R}$ and vector fields $F : \mathcal{X} \to T\mathcal{X}$



Riemannian manifolds in one minute

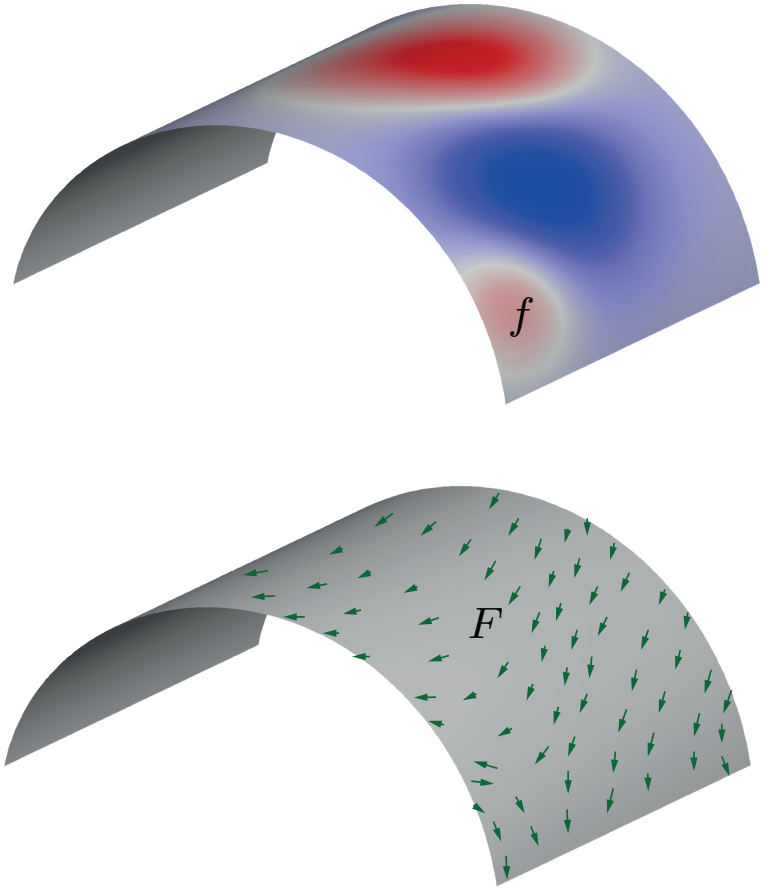
- Manifold $\mathcal{X}$ = topological space
- Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x
- Riemannian metric describes the local intrinsic structure at x

$$\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$$

- Scalar fields $f : \mathcal{X} \to \mathbb{R}$ and vector fields $F : \mathcal{X} \to T\mathcal{X}$
- Hilbert spaces with inner products

$$\langle f, g \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x) g(x) dx$$

$$\langle F, G \rangle_{L^2(T\mathcal{X})} = \int_{\mathcal{X}} \langle F(x), G(x) \rangle_{T_x\mathcal{X}} dx$$



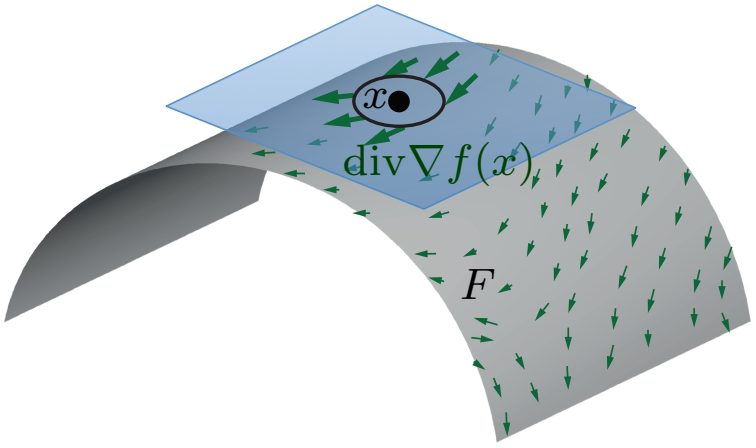
Manifold Laplacian

- Laplacian $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

$$\Delta f(x) = -\operatorname{div} \nabla f(x)$$

where gradient $\nabla : L^2(\mathcal{X}) \to L^2(T\mathcal{X})$
and divergence $\operatorname{div} : L^2(T\mathcal{X}) \to L^2(\mathcal{X})$
are adjoint operators

$\langle F, \nabla f \rangle_{L^2(T\mathcal{X})} = \langle -\operatorname{div} F, f \rangle_{L^2(\mathcal{X})}$



Manifold Laplacian

- Laplacian $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

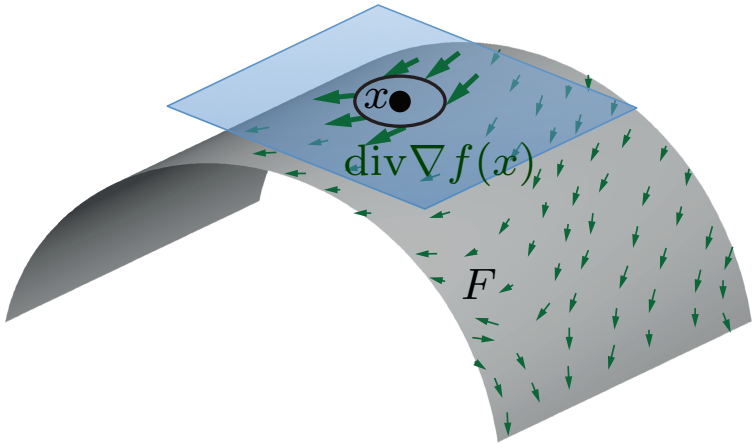
  $$\Delta f(x) = -\operatorname{div} \nabla f(x)$$

  where gradient $\nabla : L^2(\mathcal{X}) \to L^2(T\mathcal{X})$
  and divergence $\operatorname{div} : L^2(T\mathcal{X}) \to L^2(\mathcal{X})$
  are adjoint operators

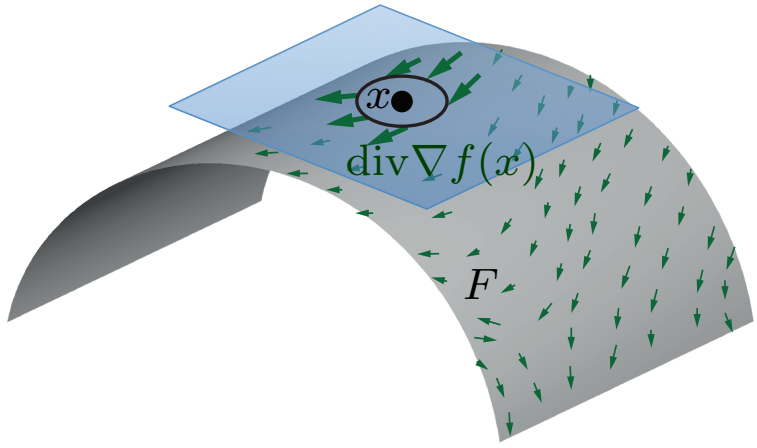
  $\langle F, \nabla f \rangle_{L^2(T\mathcal{X})} = \langle -\operatorname{div} F, f \rangle_{L^2(\mathcal{X})}$

- Laplacian is self-adjoint

  $$\langle \Delta f, f \rangle_{L^2(\mathcal{X})} = \langle f, \Delta f \rangle_{L^2(\mathcal{X})}$$



Manifold Laplacian



- Laplacian $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

$$\Delta f(x) = -\operatorname{div} \nabla f(x)$$

where gradient $\nabla : L^2(\mathcal{X}) \to L^2(T\mathcal{X})$
and divergence $\operatorname{div} : L^2(T\mathcal{X}) \to L^2(\mathcal{X})$
are adjoint operators

$$\langle F, \nabla f \rangle_{L^2(T\mathcal{X})} = \langle -\operatorname{div} F, f \rangle_{L^2(\mathcal{X})}$$

- Laplacian is self-adjoint

$$\langle \Delta f, f \rangle_{L^2(\mathcal{X})} = \langle f, \Delta f \rangle_{L^2(\mathcal{X})}$$

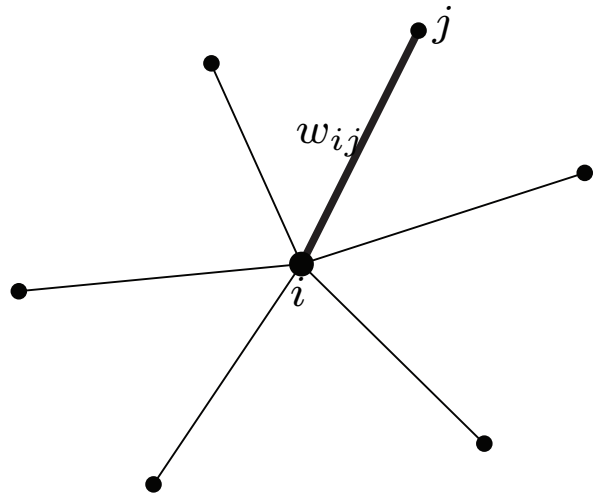
- Continuous limit of graph
  Laplacian under some conditions

- Dirichlet energy of f

$$\langle \nabla f, \nabla f \rangle_{L^2(T\mathcal{X})} = \int_{\mathcal{X}} f(x) \Delta f(x) dx$$

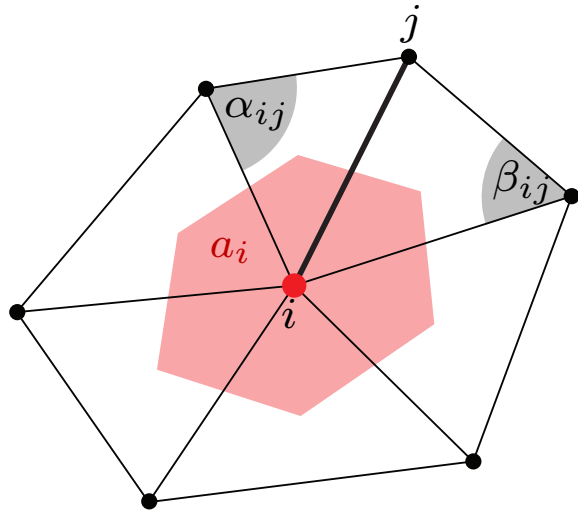
measures the smoothness of f (how fast it changes locally)

Discrete Laplacian



Undirected graph $(\mathcal{V}, \mathcal{E})$

$$(\Delta f)_i \approx \sum_{(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j)$$



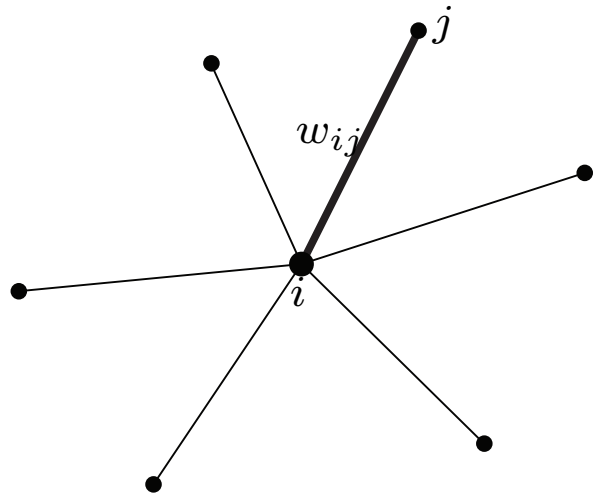
Triangular mesh $(\mathcal{V}, \mathcal{E}, \mathcal{F})$

$$(\Delta f)_i \approx \frac{1}{a_i} \sum_{(i,j)\in\mathcal{E}} \frac{\cot\alpha_{ij} + \cot\beta_{ij}}{2}(f_i - f_j)$$

$a_i =$ local area element

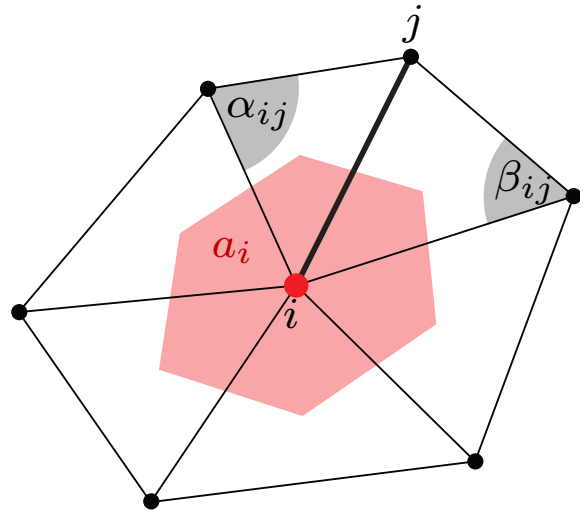
Tutte 1963; MacNeal 1949; Duffin 1959; Pinkall, Polthier 1993

Discrete Laplacian



Undirected graph $(\mathcal{V}, \mathcal{E})$

$$(\Delta f)_i \approx \sum_{(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j)$$



Triangular mesh $(\mathcal{V}, \mathcal{E}, \mathcal{F})$

$$(\Delta f)_i \approx \frac{1}{a_i} \sum_{(i,j)\in\mathcal{E}} \frac{\cot\alpha_{ij} + \cot\beta_{ij}}{2}(f_i - f_j)$$

a_i = local area element

In matrix-vector notation

$$\mathbf{\Delta f} = \mathbf{A}^{-1}(\mathbf{D} - \mathbf{W})\mathbf{f}$$

where $\mathbf{f} = (f_1, \ldots, f_n)^\top$, $\mathbf{W}$ is the stiffness matrix, $\mathbf{A} = \mathrm{diag}(a_1, \ldots, a_n)$ is the mass matrix, and $\mathbf{D} = \mathrm{diag}(\sum_{j\neq 1} w_{1j}, \ldots, \sum_{j\neq n} w_{nj})$

Tutte 1963; MacNeal 1949; Duffin 1959; Pinkall, Polthier 1993

Orthogonal bases on graphs and manifolds

Find the smoothest orthogonal basis $\{\phi_1, \ldots, \phi_n\} \subseteq L^2(\mathcal{V})$

$$\min_{\phi_1} \; E_{\text{Dir}}(\psi_1) \quad \text{s.t.} \quad \|\phi_1\| = 1$$

$$\min_{\phi_k} \; E_{\text{Dir}}(\psi_k) \quad \text{s.t.} \quad \|\phi_k\| = 1, \quad k = 2, 3, \ldots n$$

$$\phi_k \perp \text{span}\{\phi_1, \ldots, \phi_{k-1}\}$$

Orthogonal bases on graphs and manifolds

Find the smoothest orthogonal basis $\{\phi_1, \ldots, \phi_n\} \subseteq L^2(\mathcal{V})$

$$\min_{\mathbf{\Phi} \in \mathbb{R}^{n \times n}} \operatorname{trace}(\mathbf{\Phi}^\top \mathbf{\Delta} \mathbf{\Phi}) \quad \text{s.t.} \quad \mathbf{\Phi}^\top \mathbf{\Phi} = \mathbf{I}$$

Orthogonal bases on graphs and manifolds

Find the smoothest orthogonal basis $\{\phi_1, \ldots, \phi_n\} \subseteq L^2(\mathcal{V})$

$$\min_{\mathbf{\Phi} \in \mathbb{R}^{n \times n}} \operatorname{trace}(\mathbf{\Phi}^\top \mathbf{\Delta} \mathbf{\Phi}) \quad \text{s.t.} \quad \mathbf{\Phi}^\top \mathbf{\Phi} = \mathbf{I}$$

Solution: $\mathbf{\Phi} =$ Laplacian eigenvectors

Laplacian eigenvectors and eigenvalues

Eigendecomposition of a graph Laplacian

$$\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top$$

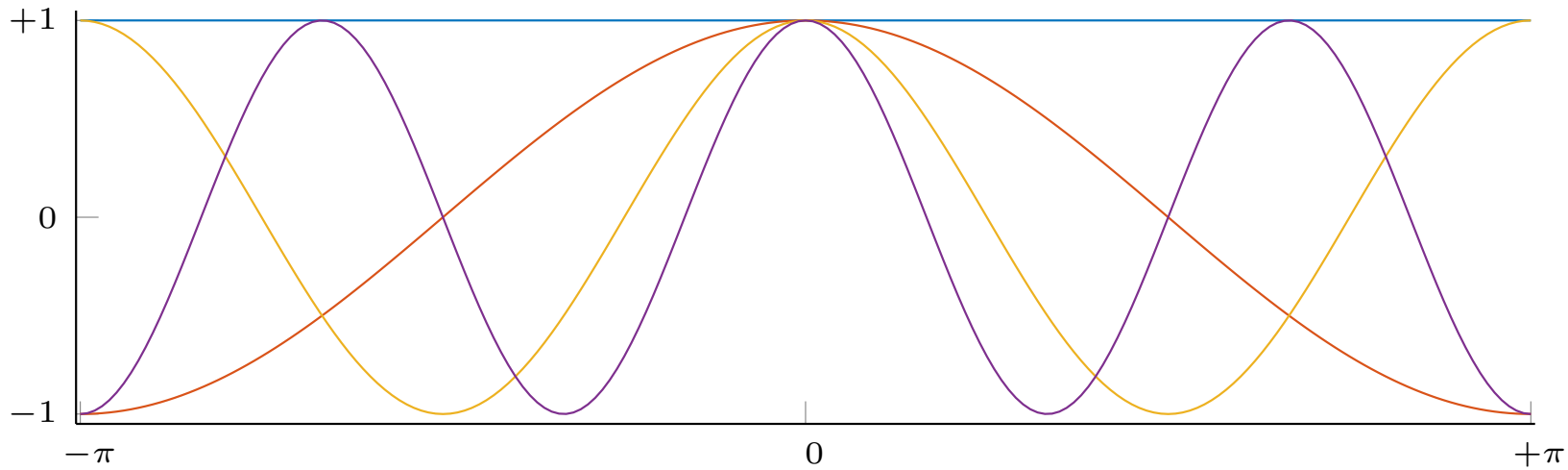
where $\mathbf{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$ are orthogonal eigenvectors ($\mathbf{\Phi}^\top\mathbf{\Phi} = \mathbf{I}$) and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ the corresponding non-negative eigenvalues

Laplacian eigenvectors and eigenvalues

Eigendecomposition of a graph Laplacian

$$\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top$$

where $\mathbf{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$ are orthogonal eigenvectors ($\mathbf{\Phi}^\top\mathbf{\Phi} = \mathbf{I}$) and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ the corresponding non-negative eigenvalues



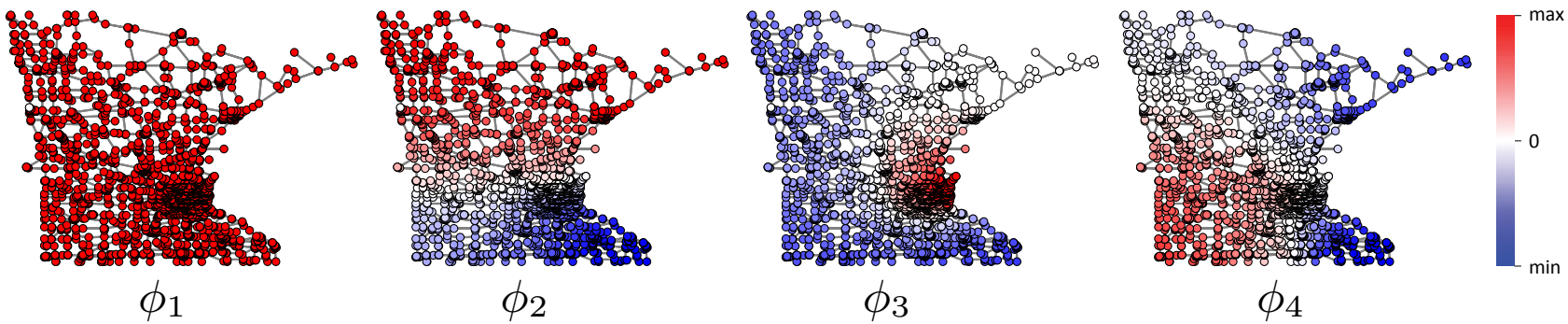
First eigenfunctions of 1D Euclidean Laplacian

Laplacian eigenvectors and eigenvalues

Eigendecomposition of a graph Laplacian

$$\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top$$

where $\mathbf{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$ are orthogonal eigenvectors ($\mathbf{\Phi}^\top\mathbf{\Phi} = \mathbf{I}$) and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ the corresponding non-negative eigenvalues



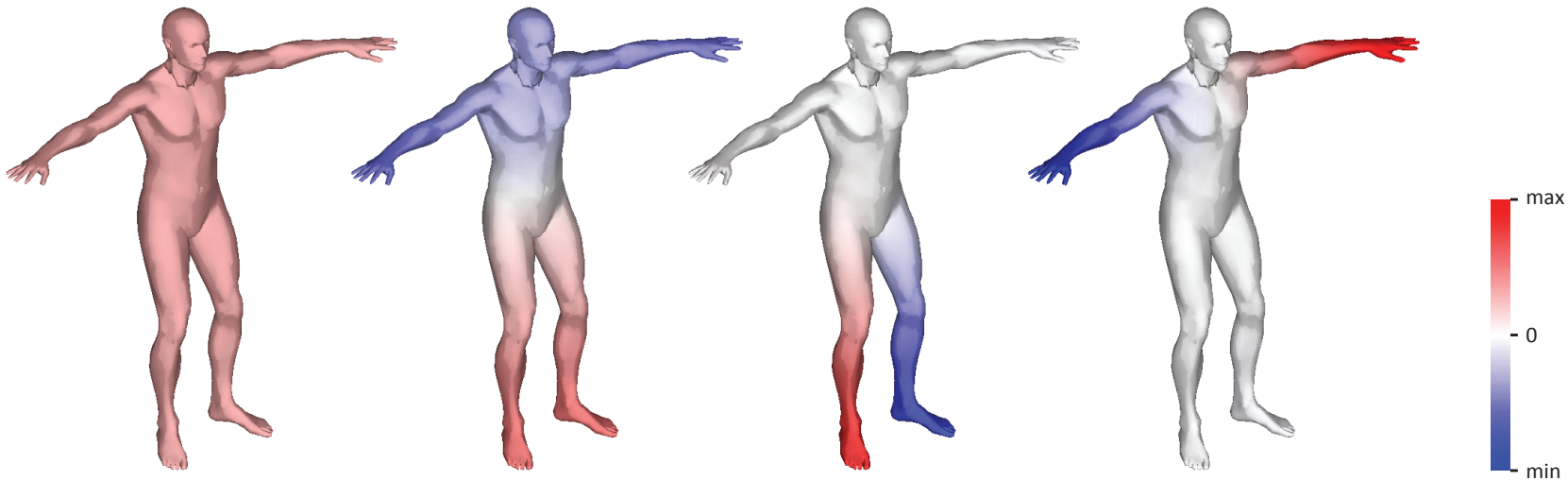
First eigenfunctions of a graph Laplacian

Laplacian eigenvectors and eigenvalues

Eigendecomposition of a graph Laplacian

$$\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top$$

where $\mathbf{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$ are orthogonal eigenvectors ($\mathbf{\Phi}^\top\mathbf{\Phi} = \mathbf{I}$) and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ the corresponding non-negative eigenvalues



$\boldsymbol{\phi}_1$ $\boldsymbol{\phi}_2$ $\boldsymbol{\phi}_3$ $\boldsymbol{\phi}_4$

First eigenfunctions of a manifold Laplacian

Fourier analysis on Euclidean spaces

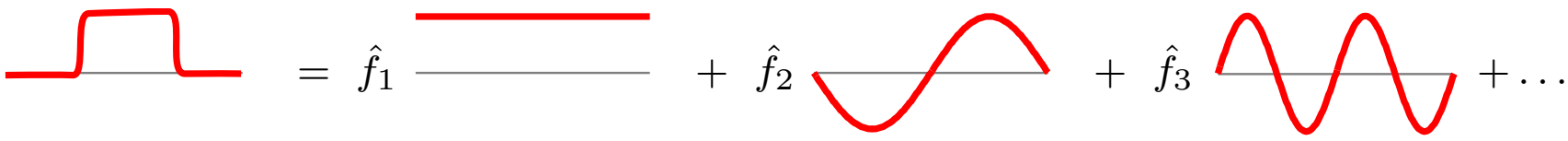
A function $f : [-\pi, \pi] \to \mathbb{R}$ can be written as a Fourier series

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

Fourier analysis on Euclidean spaces

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

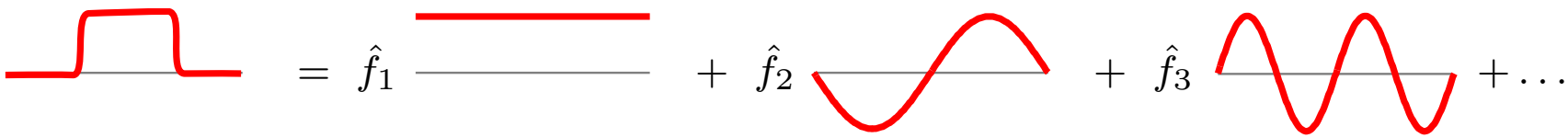
$$f(x) = \sum_{k \geq 0} \langle f, e^{ikx} \rangle_{L^2([-\pi,\pi])} \, e^{ikx}$$



Fourier analysis on Euclidean spaces

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

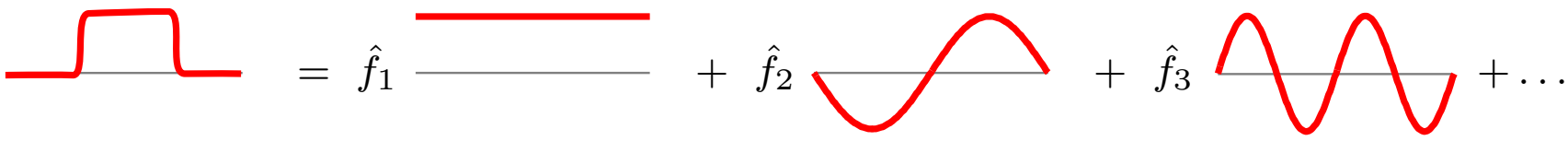
$$f(x) = \sum_{k \geq 0} \underbrace{\langle f, e^{ikx} \rangle_{L^2([-\pi,\pi])}}_{\hat{f}_k \text{ Fourier coefficient}} e^{ikx}$$



Fourier analysis on Euclidean spaces

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

$$f(x) = \sum_{k \geq 0} \underbrace{\langle f, e^{ikx} \rangle_{L^2([-\pi,\pi])}}_{\hat{f}_k \text{ Fourier coefficient}} e^{ikx}$$

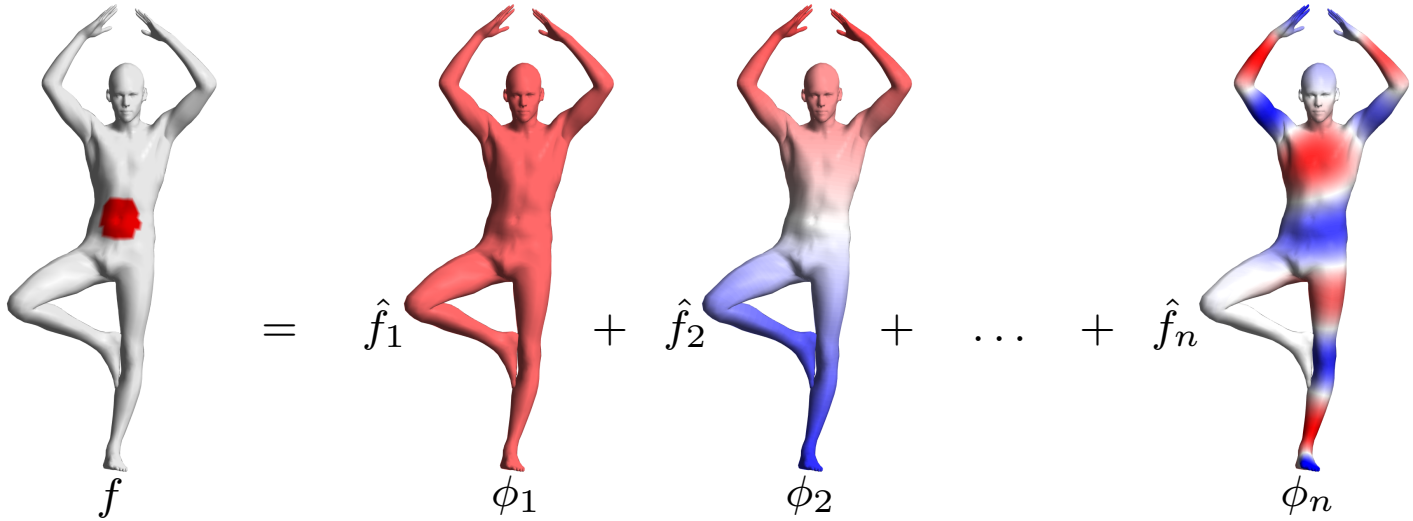


Fourier basis = Laplacian eigenfunctions: $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

Fourier analysis on graphs and manifolds

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

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



Fourier basis = Laplacian eigenfunctions: $\Delta\phi_k = \lambda_k \phi_k$

Physical application: heat equation

$$f_t = -c\Delta f$$

Newton's law of cooling: rate of change of the temperature of an object is proportional to the difference between its own temperature and the temperature of the surrounding

c [m^2/sec] = thermal diffusivity constant

Newton 1701

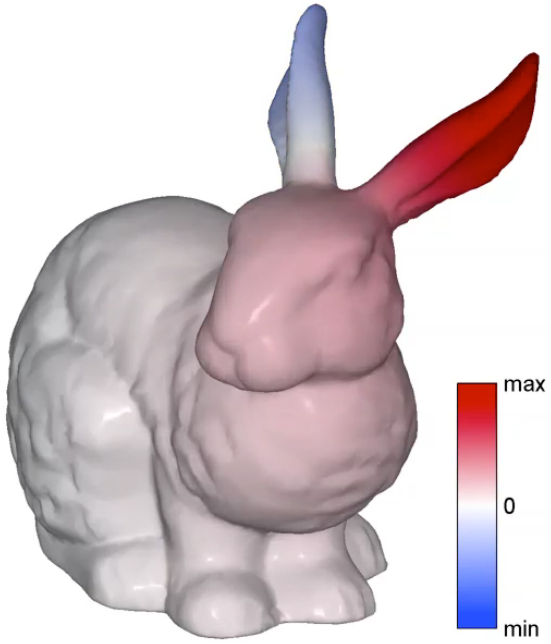
Heat diffusion on manifolds

$$\begin{cases} f_t(x,t) = -\Delta f(x,t) \\ f(x,0) = f_0(x) \end{cases}$$

- $f(x,t) =$ amount of heat at point x at time t
- $f_0(x) =$ initial heat distribution

Heat diffusion on manifolds

$$\begin{cases} f_t(x,t) = -\Delta f(x,t) \\ f(x,0) = f_0(x) \end{cases}$$



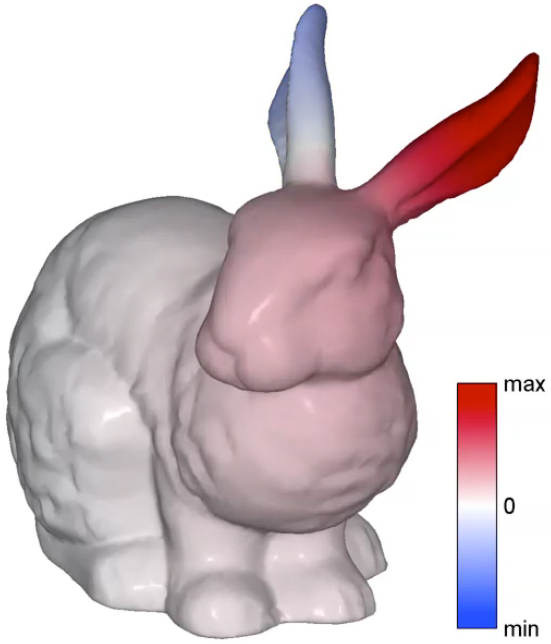
- $f(x,t) =$ amount of heat at point x at time t
- $f_0(x) =$ initial heat distribution

Solution of the heat equation expressed through the heat operator

$$f(x,t) \quad = \quad e^{-t\Delta} f_0(x)$$

Heat diffusion on manifolds

$$\begin{cases} f_t(x,t) = -\Delta f(x,t) \\ f(x,0) = f_0(x) \end{cases}$$



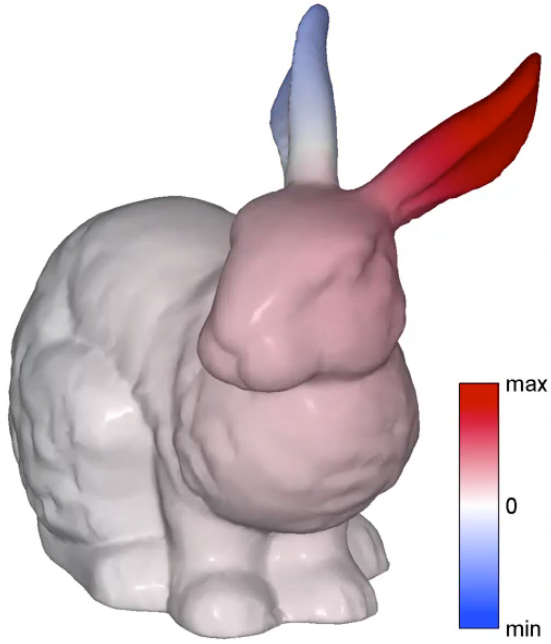
- $f(x,t) =$ amount of heat at point x at time t
- $f_0(x) =$ initial heat distribution

Solution of the heat equation expressed through the heat operator

$$f(x,t) \quad = \quad e^{-t\Delta} f_0(x) = \sum_{k \geq 1} \langle f_0, \phi_k \rangle_{L^2(\mathcal{X})} e^{-t\lambda_k} \phi_k(x)$$

Heat diffusion on manifolds

$$\begin{cases} f_t(x,t) = -\Delta f(x,t) \\ f(x,0) = f_0(x) \end{cases}$$



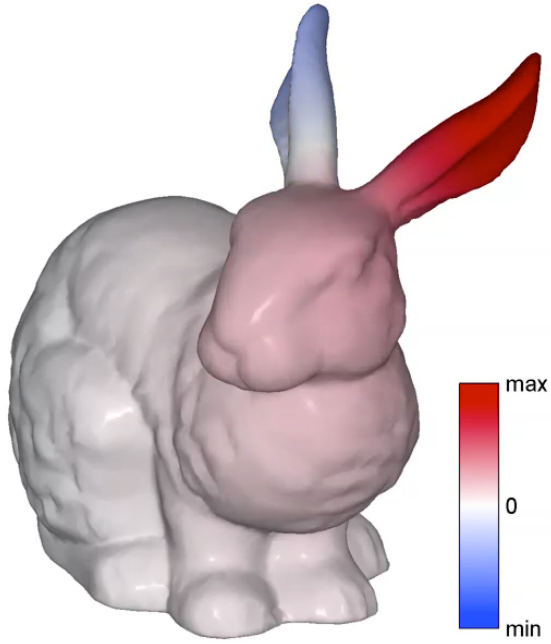
- $f(x,t) =$ amount of heat at point x at time t
- $f_0(x) =$ initial heat distribution

Solution of the heat equation expressed through the heat operator

$$\begin{aligned} f(x,t) &= e^{-t\Delta} f_0(x) = \sum_{k\geq 1} \langle f_0, \phi_k \rangle_{L^2(\mathcal{X})} e^{-t\lambda_k} \phi_k(x) \\ &= \int_{\mathcal{X}} f_0(x') \sum_{k\geq 1} e^{-t\lambda_k} \phi_k(x) \phi_k(x')\, dx' \end{aligned}$$

Heat diffusion on manifolds

$$\begin{cases} f_t(x,t) = -\Delta f(x,t) \\ f(x,0) = f_0(x) \end{cases}$$

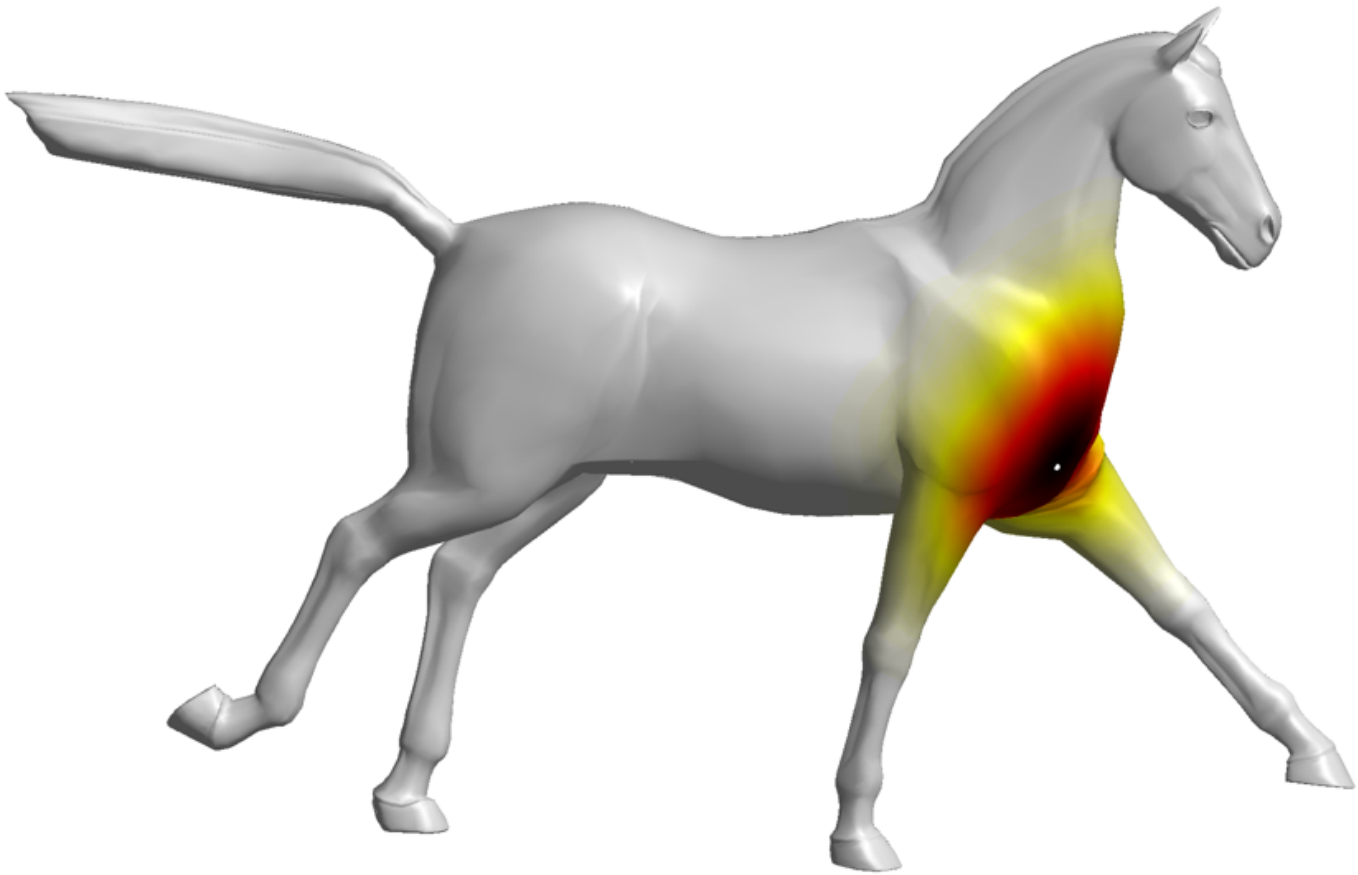


- $f(x,t) =$ amount of heat at point x at time t
- $f_0(x) =$ initial heat distribution

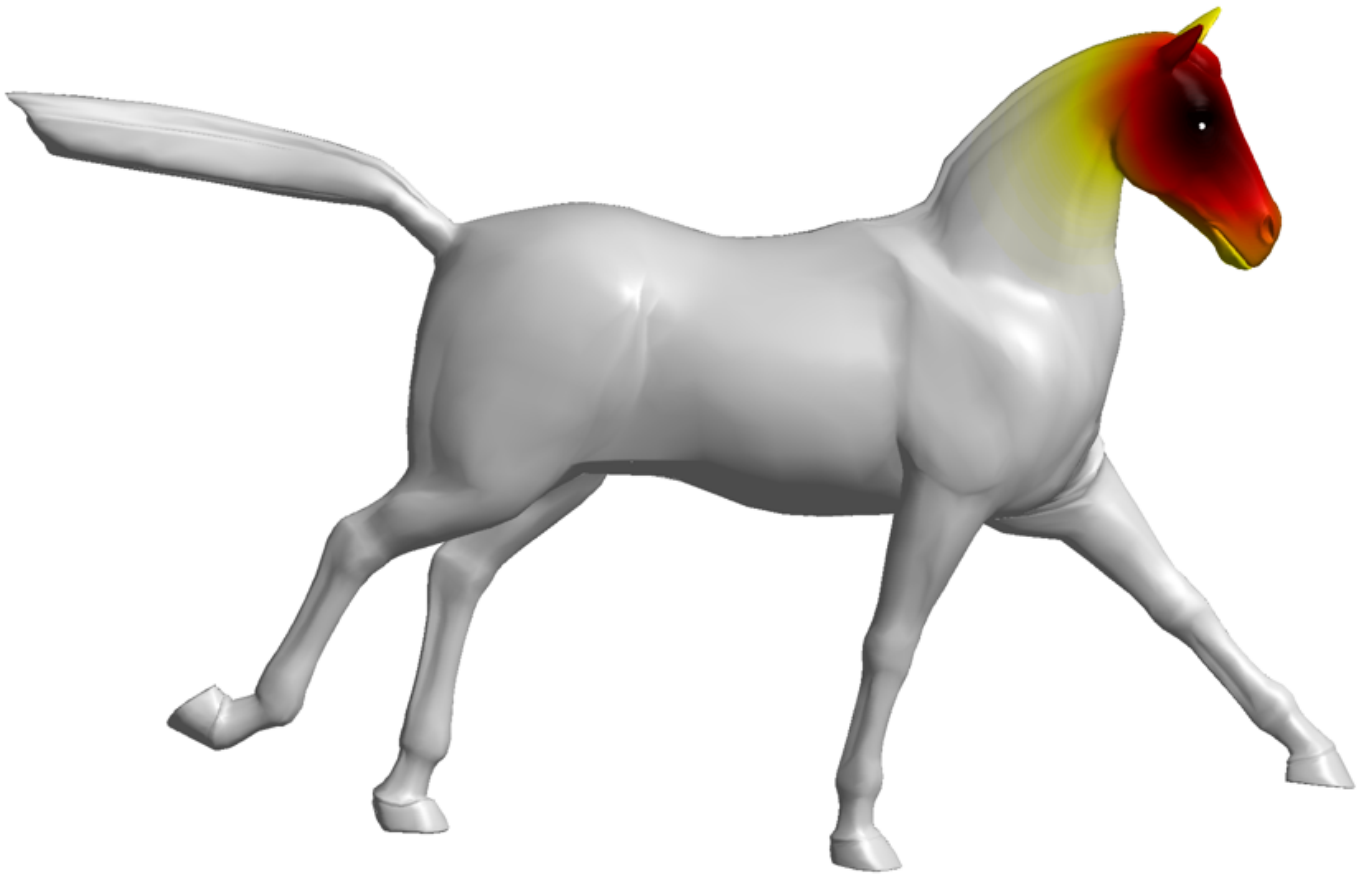
Solution of the heat equation expressed through the heat operator

$$\begin{aligned} f(x,t) &= e^{-t\Delta} f_0(x) = \sum_{k\geq 1} \langle f_0, \phi_k \rangle_{L^2(\mathcal{X})} e^{-t\lambda_k} \phi_k(x) \\ &= \int_{\mathcal{X}} f_0(x') \underbrace{\sum_{k\geq 1} e^{-t\lambda_k} \phi_k(x)\phi_k(x')}_{\text{heat kernel } h_t(x,x')} dx' \end{aligned}$$

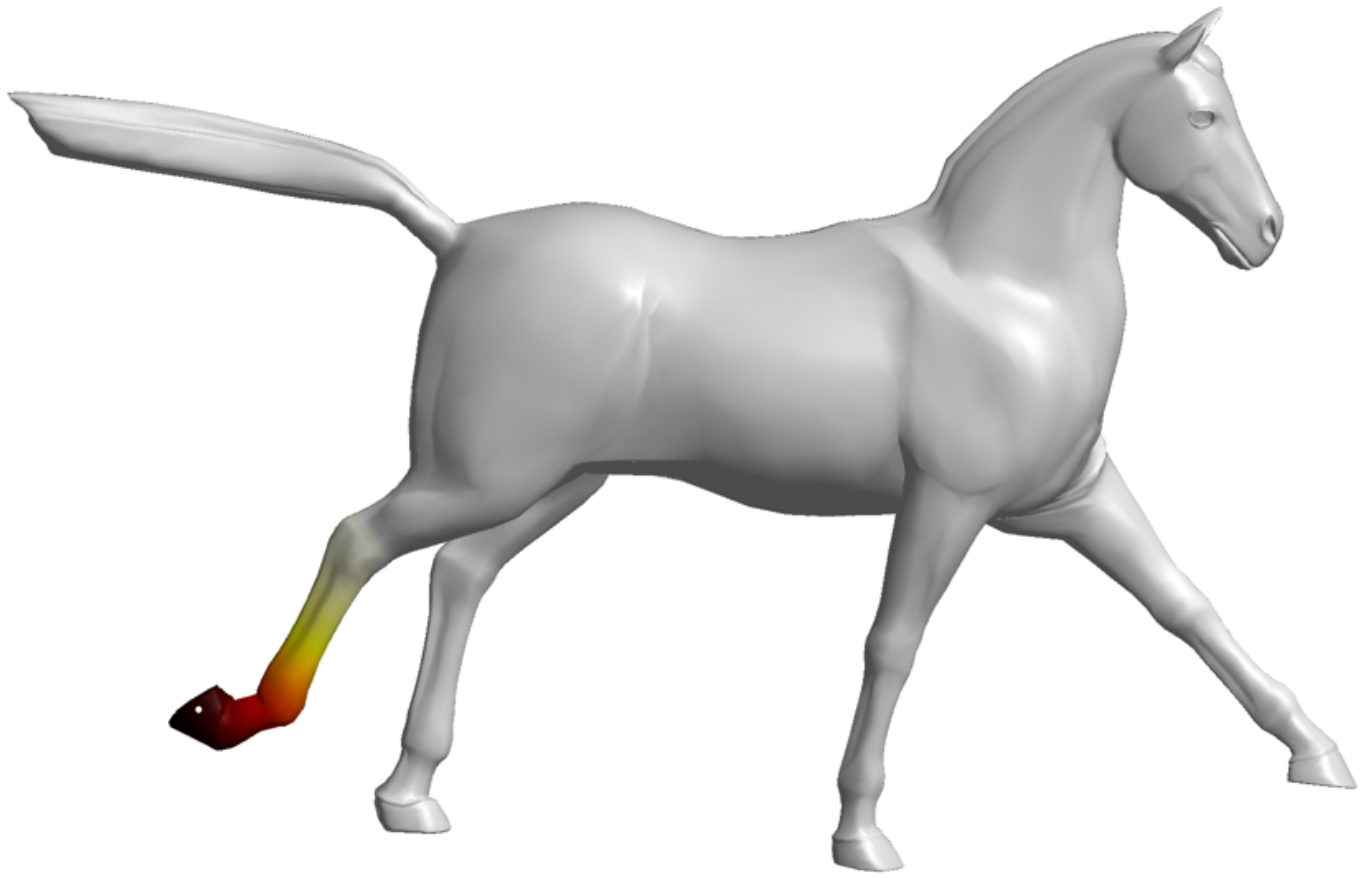
Heat kernels



Heat kernels



Heat kernels



Heat kernels

Interpretation of the heat kernel

Solution of the heat equation on a manifold $\mathcal{X}$ expressed in the Laplacian eigenbasis $\Delta\phi_k(x) = \lambda_k\phi_k(x)$

$$f(x,t) \quad = \quad \int_{\mathcal{X}} f_0(x') \underbrace{\sum_{k\geq 1} e^{-t\lambda_k}\phi_k(x)\phi_k(x')}_{\text{heat kernel } h_t(x,x')} dx'$$

Interpretation of the heat kernel

Solution of the heat equation on a Euclidean space $[-\pi, \pi]$ expressed in the Laplacian eigenbasis $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

$$f(x,t) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f_0(x') \underbrace{\sum_{k \geq 0} e^{-tk^2} e^{ikx} e^{-ikx'}}_{\text{heat kernel } h_t(x,x')} dx'$$

Interpretation of the heat kernel

Solution of the heat equation on a Euclidean space $[-\pi, \pi]$ expressed in the Laplacian eigenbasis $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

$$f(x,t) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f_0(x') \underbrace{\sum_{k \geq 0} e^{-tk^2} e^{ik(x-x')}}_{\text{heat kernel } h_t(x-x')} dx'$$

Interpretation of the heat kernel

Solution of the heat equation on a Euclidean space $[-\pi, \pi]$ expressed in the Laplacian eigenbasis $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

$$\begin{aligned} f(x,t) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} f_0(x') \underbrace{\sum_{k \geq 0} e^{-tk^2} e^{ik(x-x')}}_{\text{heat kernel } h_t(x-x')} dx' \\ &= (f_0 \star h_t)(x) \end{aligned}$$

Heat kernel = impulse response

Convolution: Euclidean space

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

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

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: 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 theorem: Fourier transform diagonalizes the convolution operator

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 theorem: Fourier transform diagonalizes the convolution operator $\Rightarrow$ convolution can be computed in the Fourier domain as

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

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 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

Convolution Theorem

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

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

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

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

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

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

Spectral 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

Spectral 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

Spectral 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

Spectral 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

Spectral 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

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

Spectral convolution

Spectral 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

Spectral 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

Spectral 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)
- Filter coefficients depend on basis $\phi_1, \ldots, \phi_n$

Spectral CNN

Convolution expressed in the spectral domain

$$\mathbf{g} = \mathbf{\Phi W \Phi}^\top \mathbf{f}$$

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

Bruna et al. 2014

Spectral CNN

Convolution expressed in the spectral domain

$$\mathbf{g} = \mathbf{\Phi W \Phi}^\top \mathbf{f}$$

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

☹ Filters are basis-dependent ⇒ do not generalize across domains

Bruna et al. 2014

Spectral CNN

Convolution expressed in the spectral domain

$$\mathbf{g} = \mathbf{\Phi W \Phi}^\top \mathbf{f}$$

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

☹ Filters are basis-dependent $\Rightarrow$ do not generalize across domains

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

Bruna et al. 2014

Spectral CNN

Convolution expressed in the spectral domain

$$\mathbf{g} = \mathbf{\Phi W \Phi}^\top \mathbf{f}$$

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

- ☹ Filters are basis-dependent $\Rightarrow$ do not generalize across domains
- ☹ $\mathcal{O}(n)$ 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 CNN

Convolution expressed in the spectral domain

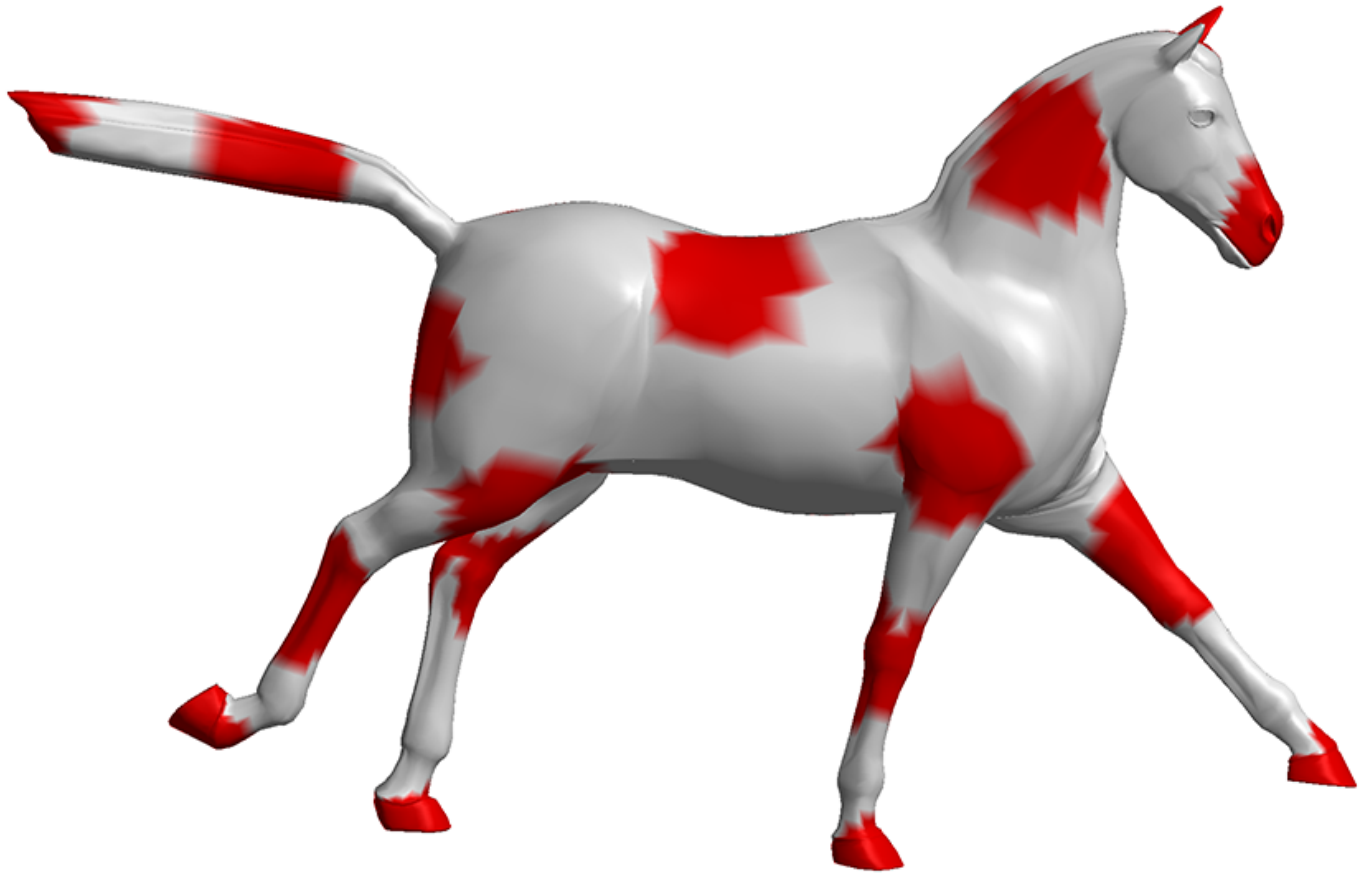
$$\mathbf{g} = \mathbf{\Phi W \Phi}^\top \mathbf{f}$$

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

- ☹ Filters are basis-dependent $\Rightarrow$ do not generalize across domains
- ☹ $\mathcal{O}(n)$ 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 $\mathbf{\Psi W \Psi}^\top \mathbf{f}$

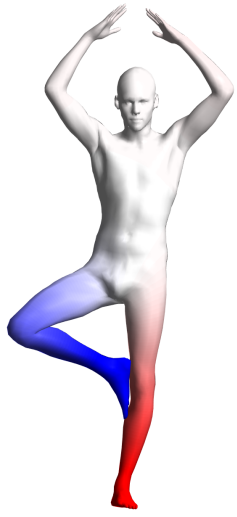
Basis dependence

High-frequency Laplacian eigenvector ϕ_{50}

Laplacian eigenbases on non-isometric domains



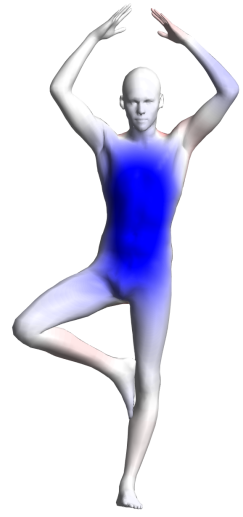
ϕ_2



ϕ_3



ϕ_{10}



ϕ_{15}



ϕ_{20}



ψ_2



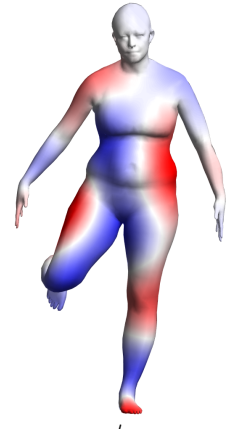
ψ_3



ψ_{10}

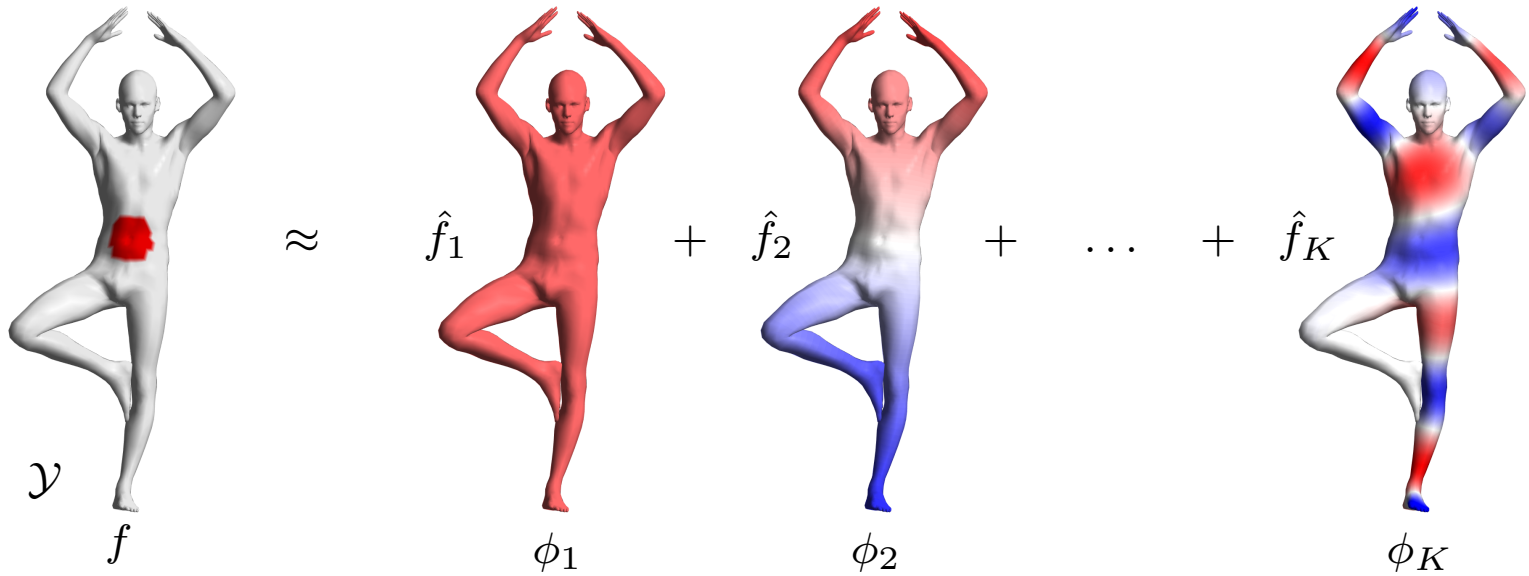


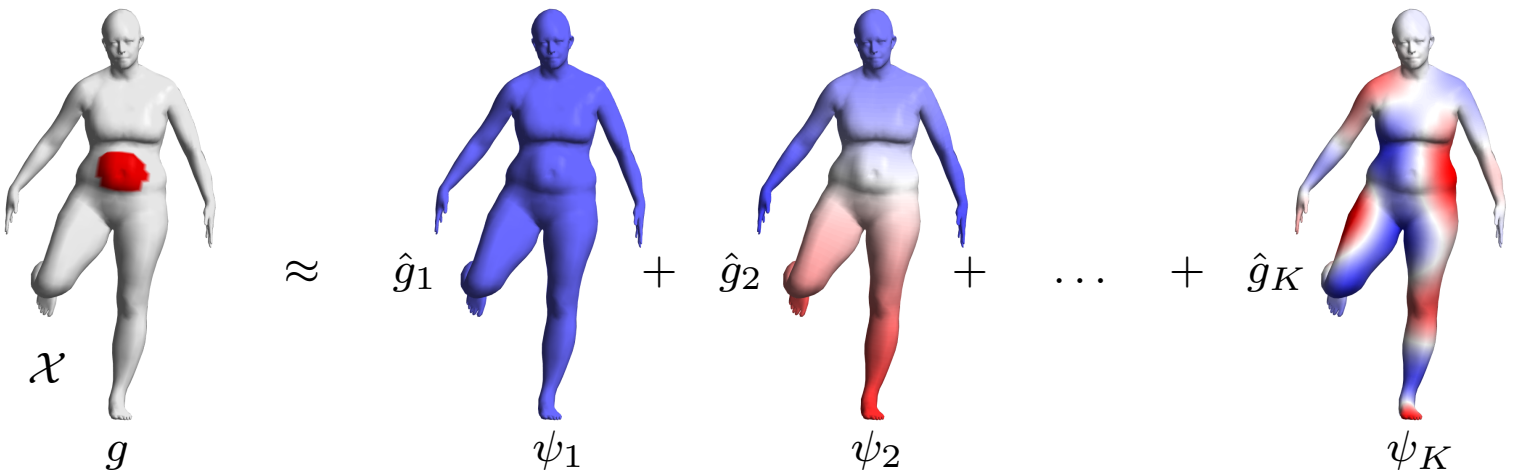
ψ_{15}



ψ_{20}

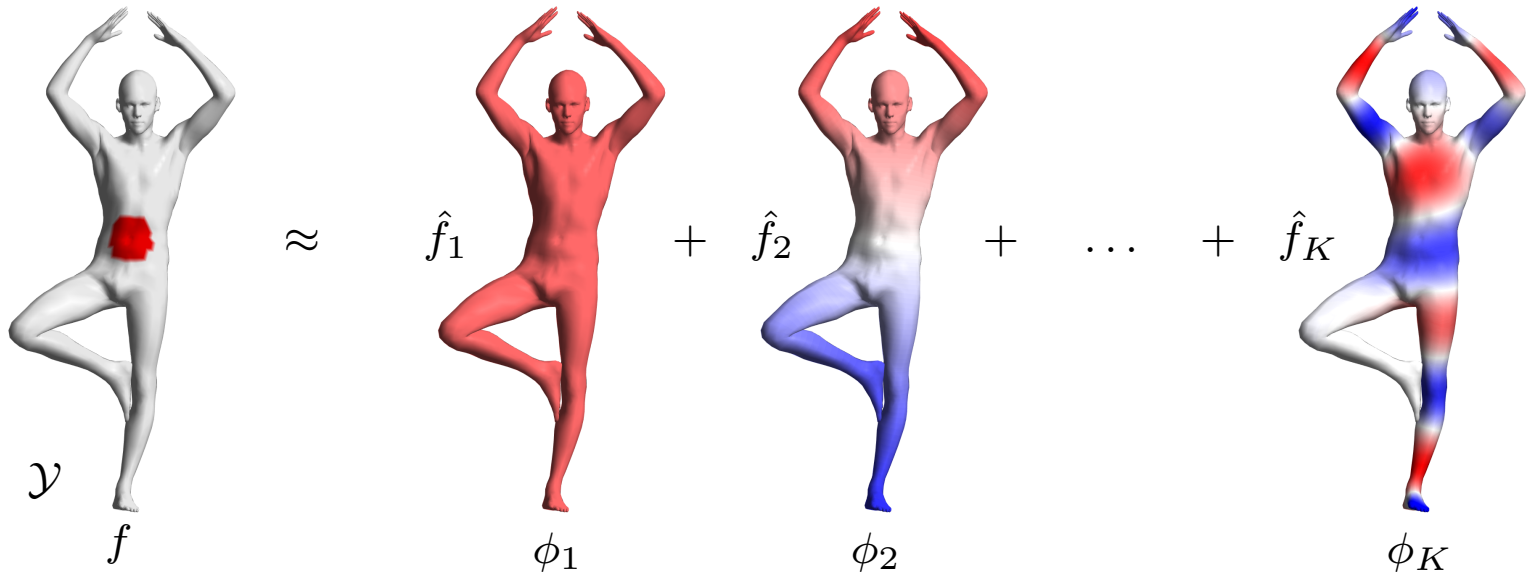
Functional maps





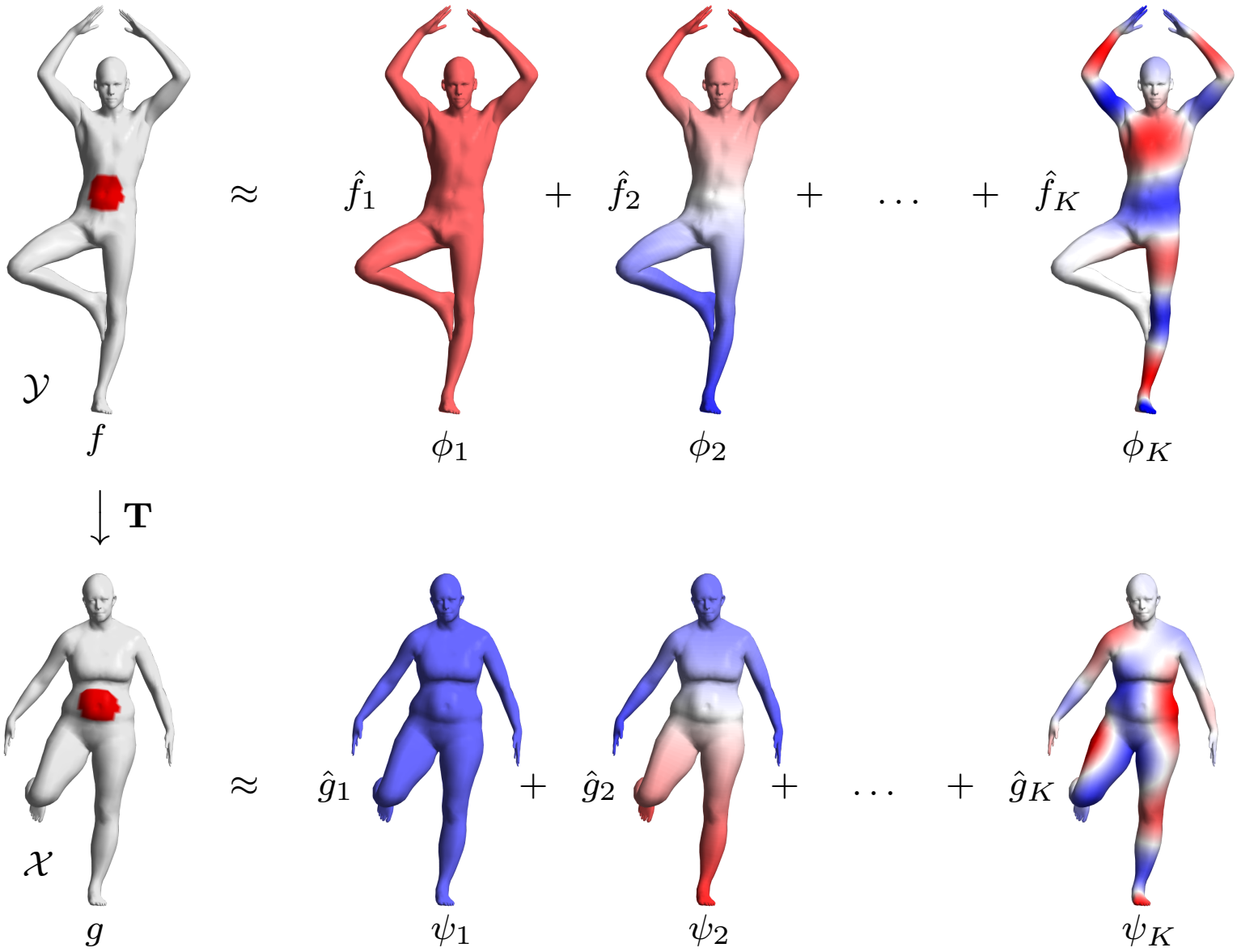
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Functional maps



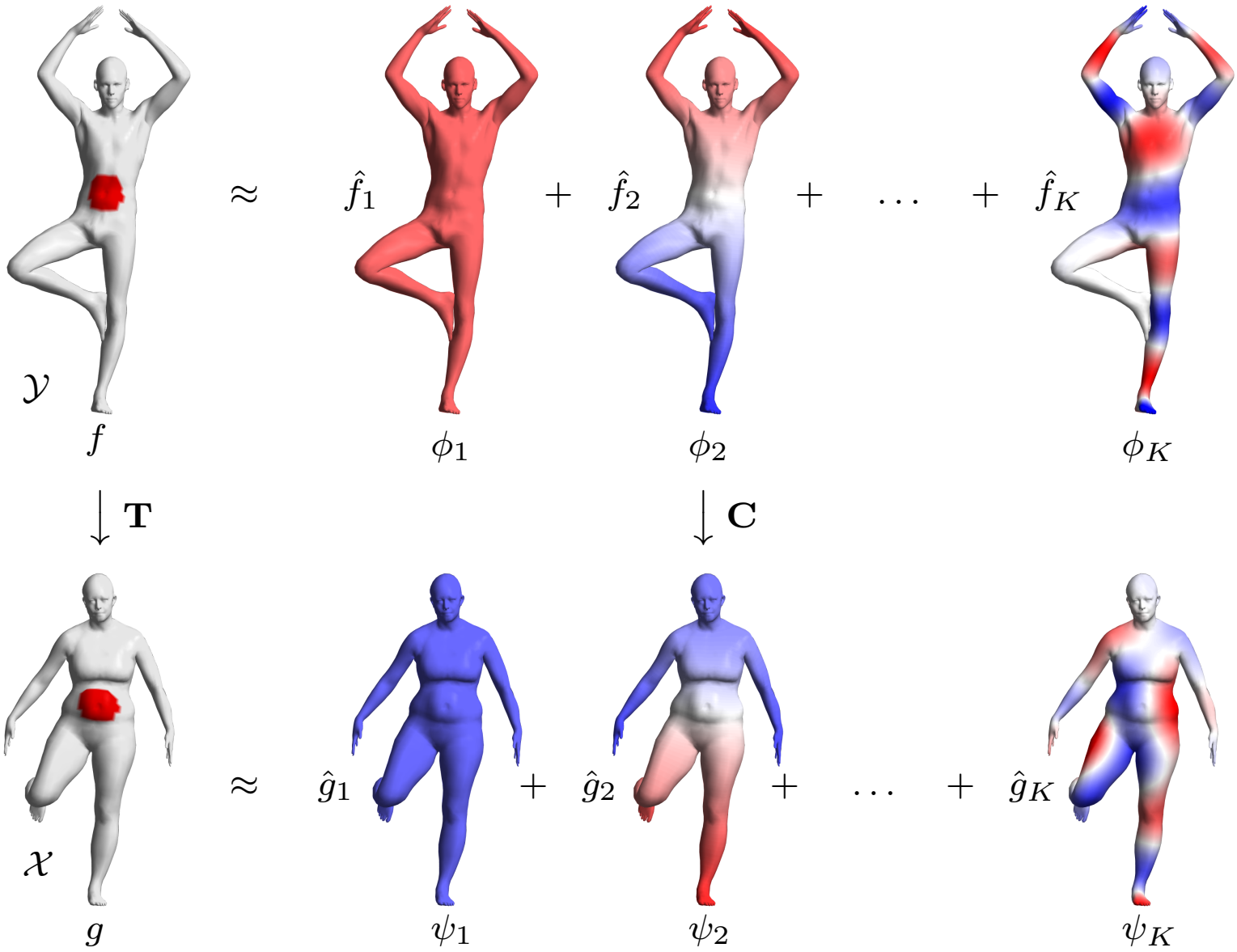
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Functional maps



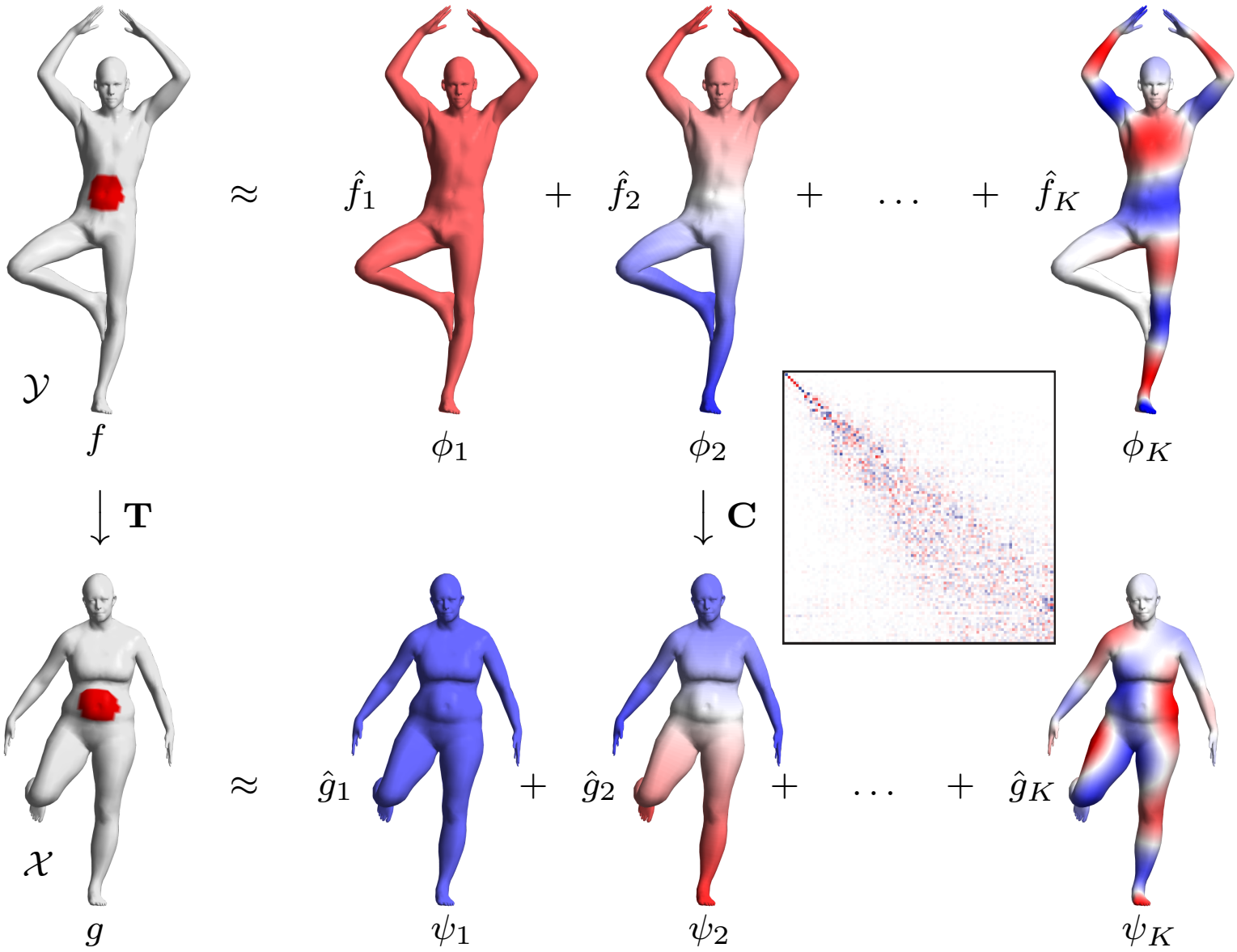
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Functional maps



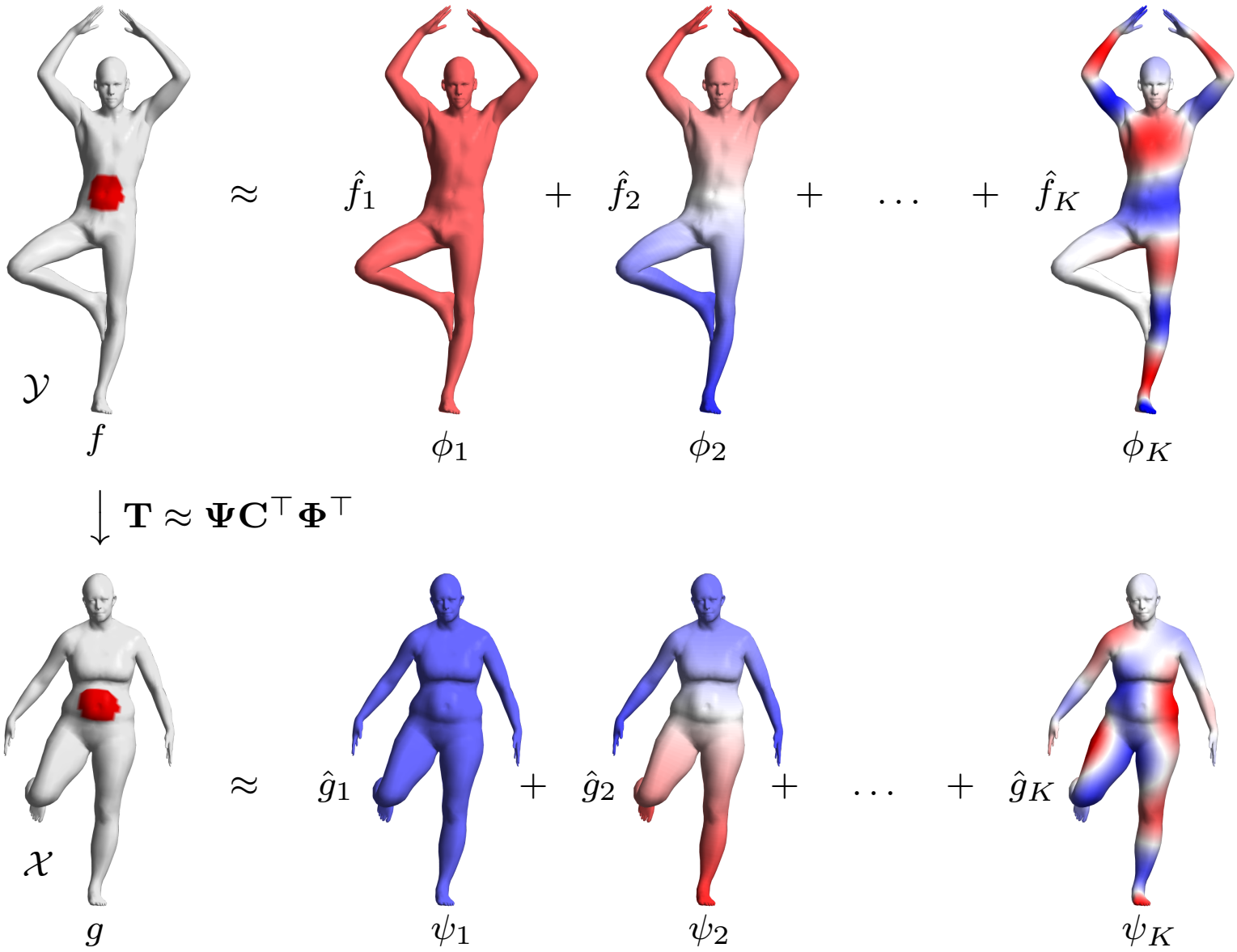
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Functional maps



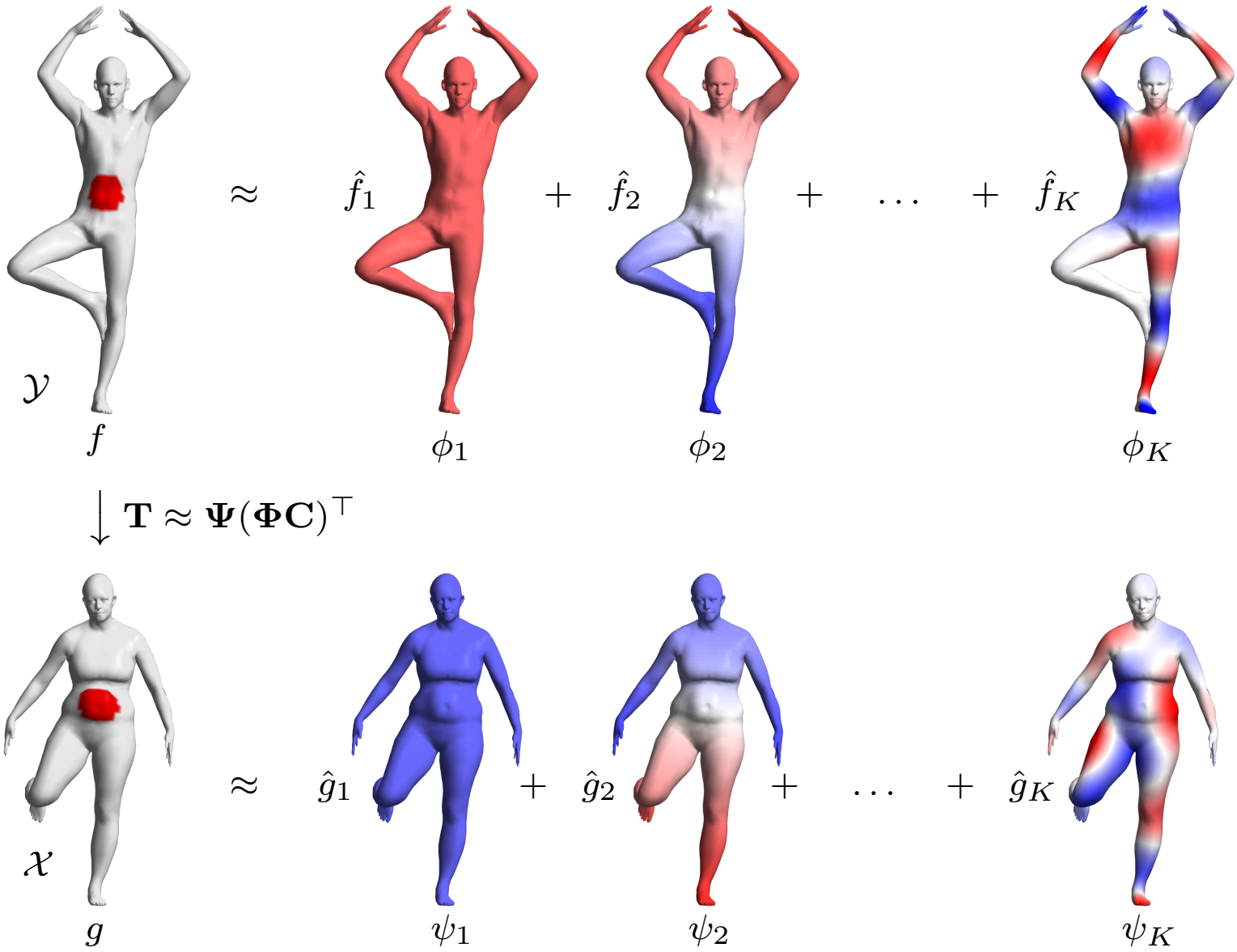
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Basis synchronization with functional maps



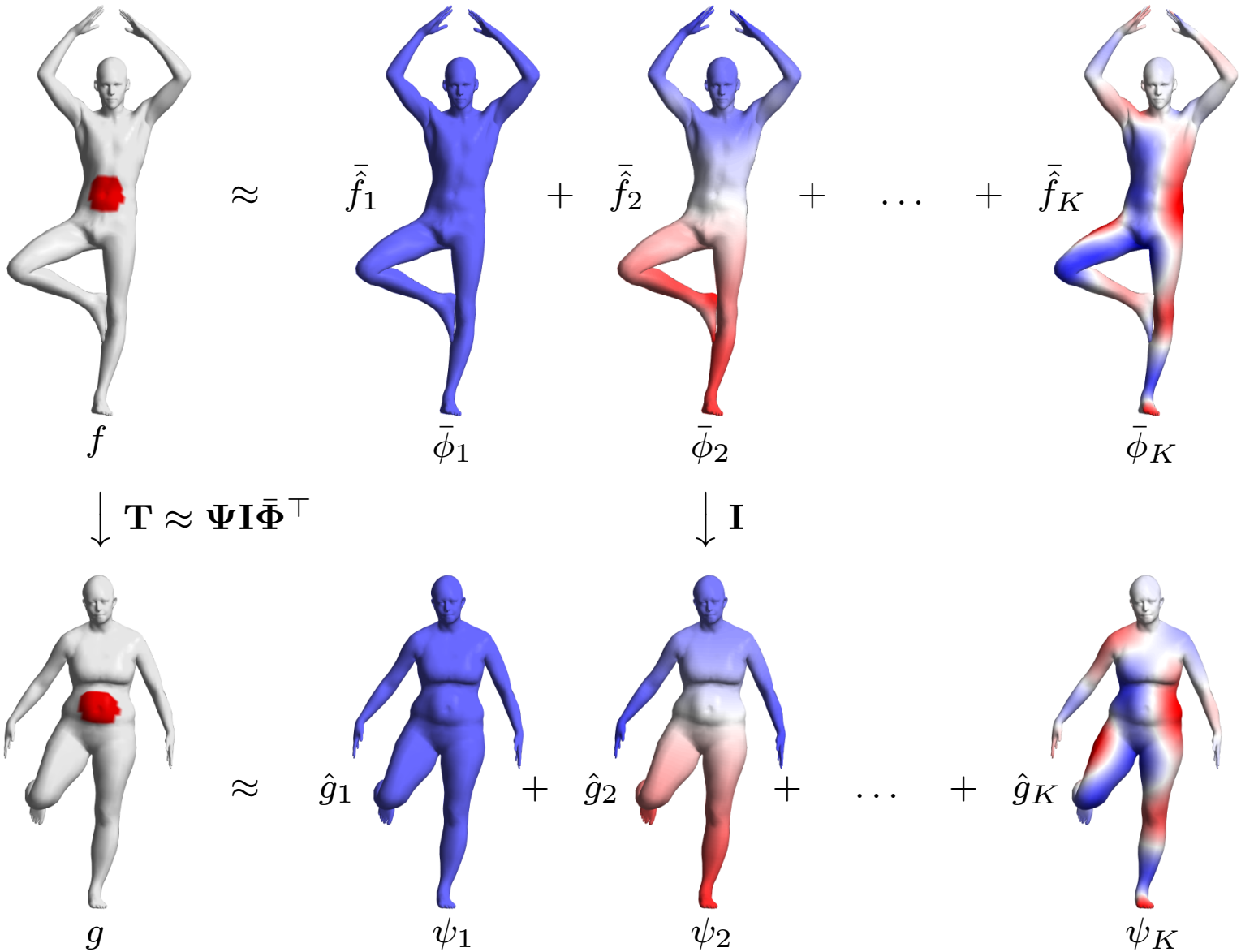
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Basis synchronization with functional maps



Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

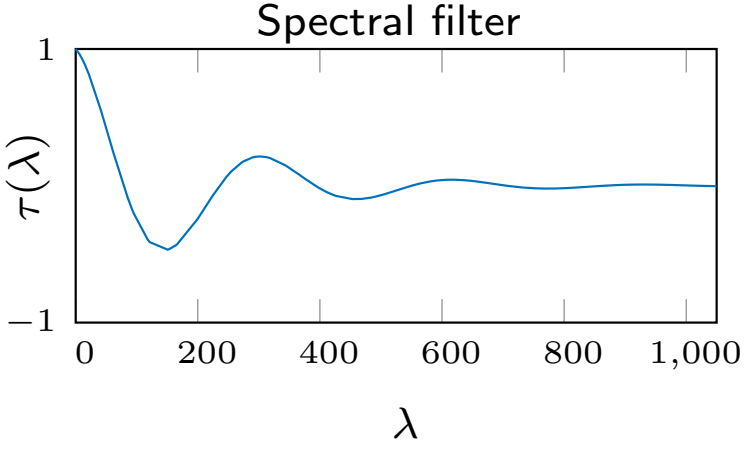
Basis synchronization with functional maps

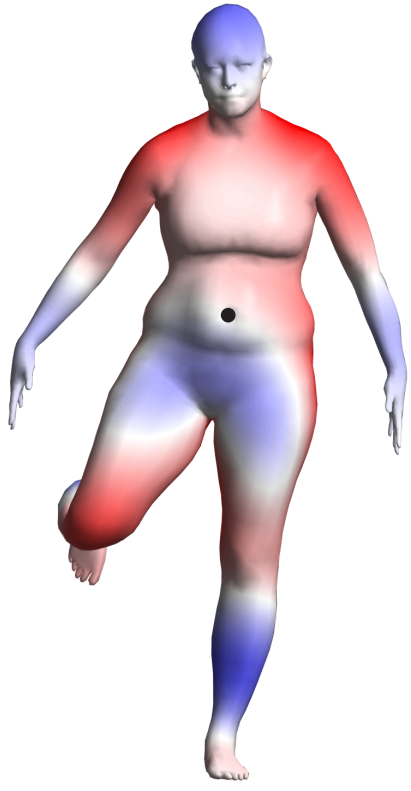


Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Filtering in different bases







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

$\boldsymbol{\Psi}\tau(\boldsymbol{\Lambda}_{\boldsymbol{\Psi}})\boldsymbol{\Psi}^\top\boldsymbol{\delta}_0$

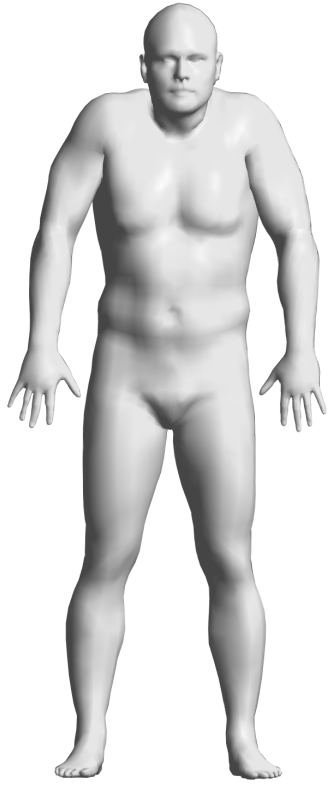
Apply spectral filter $\tau(\lambda)$ in different bases $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$

$\Rightarrow$ **different results!**

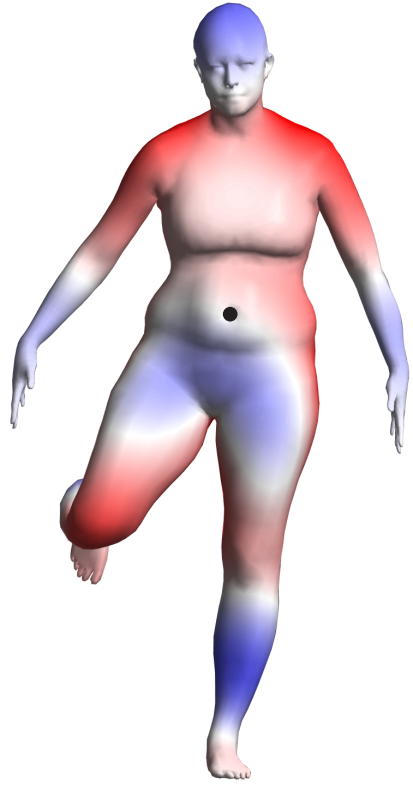
Filtering in different bases



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



Canonical shape
with basis $\boldsymbol{\Sigma}, \boldsymbol{\Lambda}$

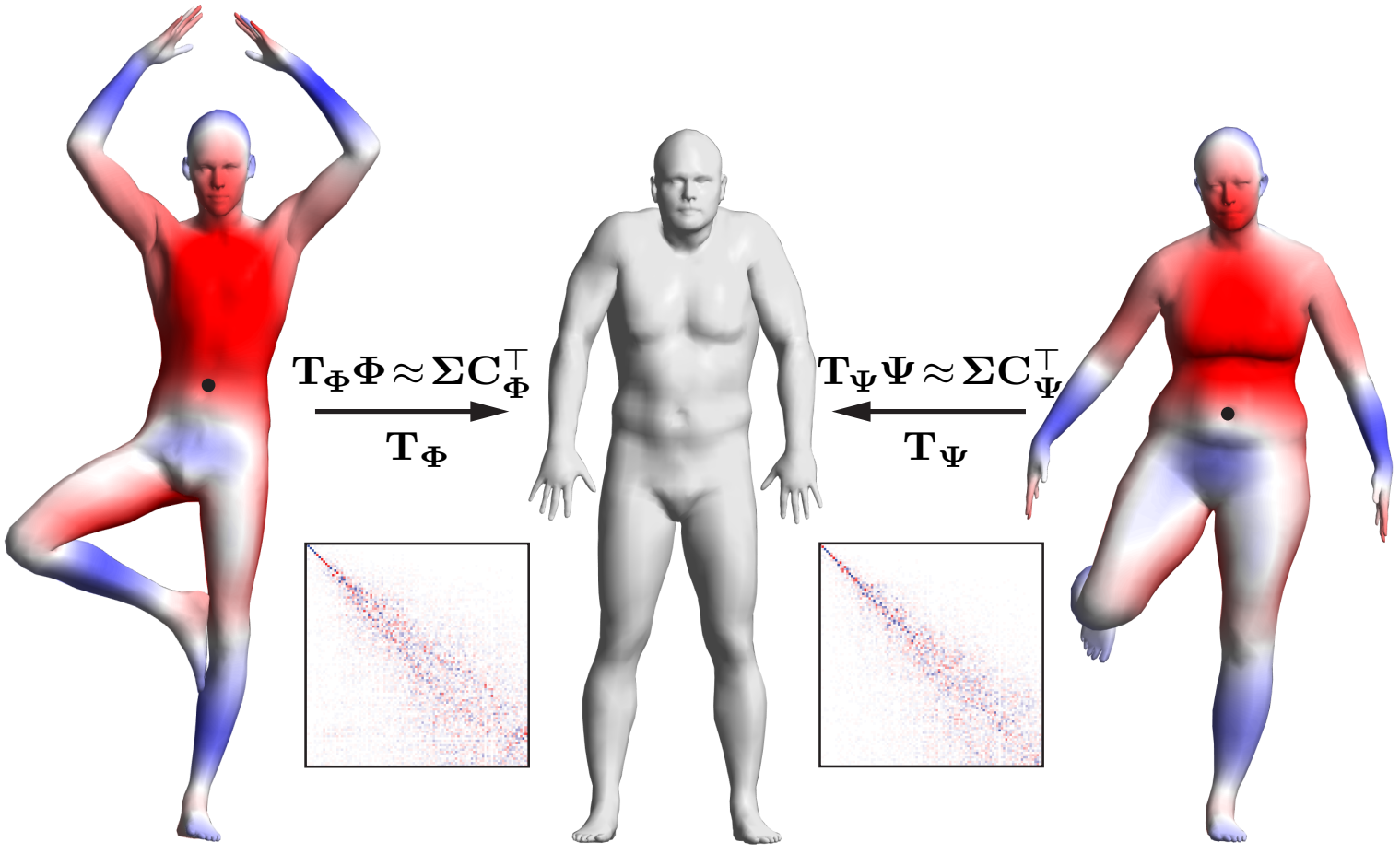


$\boldsymbol{\Psi}\tau(\boldsymbol{\Lambda}_{\boldsymbol{\Psi}})\boldsymbol{\Psi}^\top\boldsymbol{\delta}_0$

Apply spectral filter $\tau(\lambda)$ in different bases $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$

$\Rightarrow$ **different results!**

Filtering in synchronized bases



$\mathbf{\Phi}\mathbf{C}_{\mathbf{\Phi}}\tau(\mathbf{\Lambda})\mathbf{C}_{\mathbf{\Phi}}^{\top}\mathbf{\Phi}^{\top}\boldsymbol{\delta}_0$

Canonical shape with basis $\mathbf{\Sigma}, \mathbf{\Lambda}$

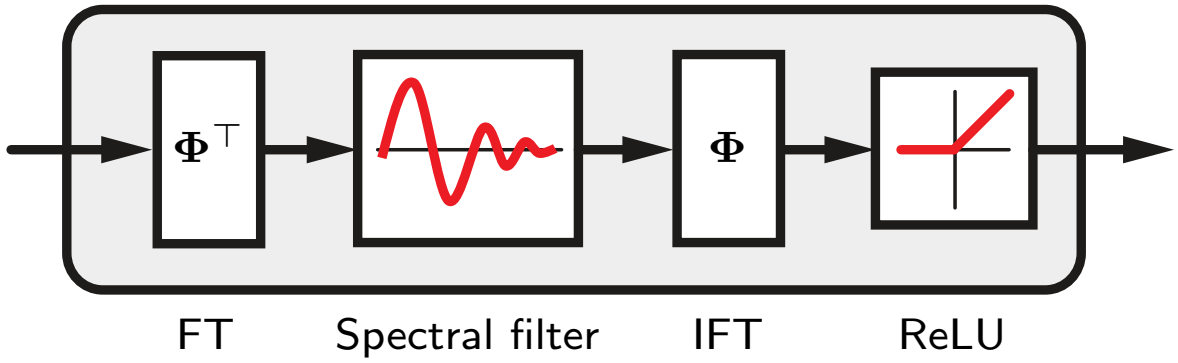
$\mathbf{\Psi}\mathbf{C}_{\mathbf{\Psi}}\tau(\mathbf{\Lambda})\mathbf{C}_{\mathbf{\Psi}}^{\top}\mathbf{\Psi}^{\top}\boldsymbol{\delta}_0$

Apply spectral filter $\tau(\lambda)$ in synchronized bases $\mathbf{\Phi}\mathbf{C}_{\mathbf{\Phi}}$ and $\mathbf{\Psi}\mathbf{C}_{\mathbf{\Psi}}$

$\Rightarrow$ **similar results!**

Yi et al. 2017

Spectral CNN



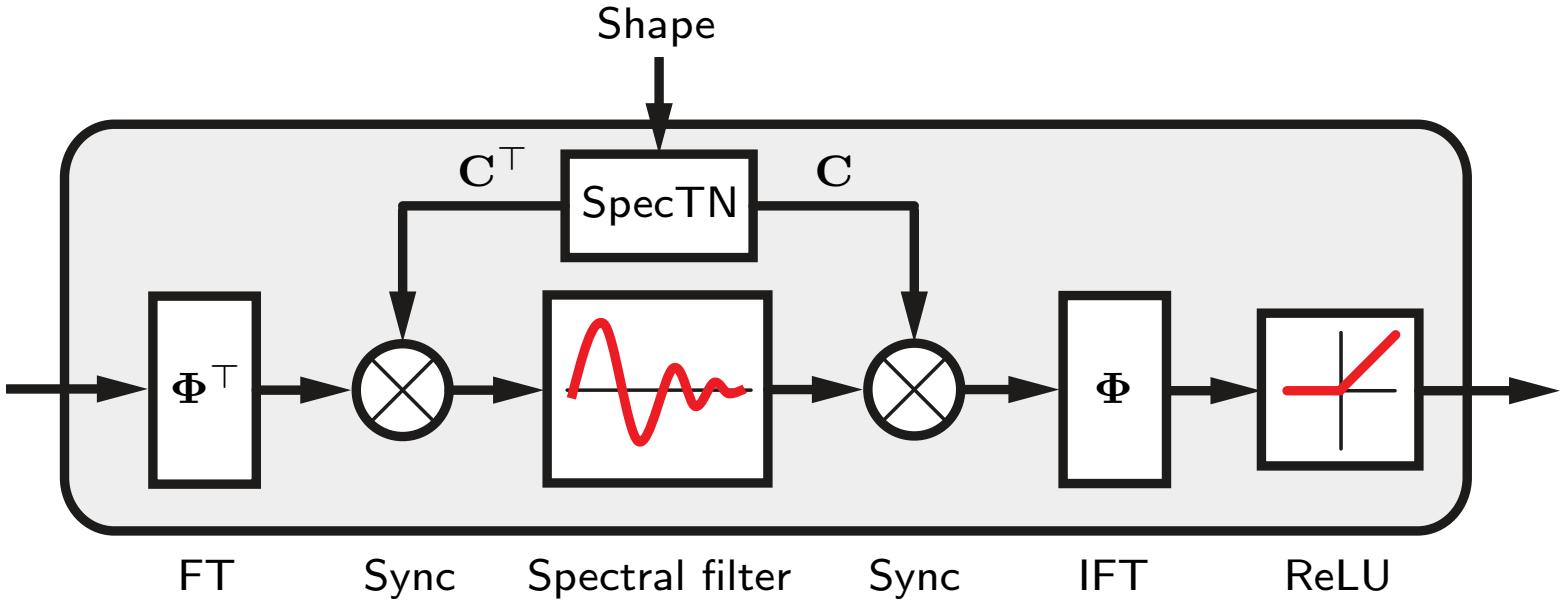
Convolutional filter of a Spectral CNN

☹ Fixed basis $\Rightarrow$ Does not generalize across domains

☺ Possible $\mathcal{O}(n)$ complexity avoiding explicit FT and IFT

Bruna et al. 2014

Spectral Transformer Network



Convolutional filter of a Spectral Transformer Network

☺ Basis synchronization allows generalization across domains

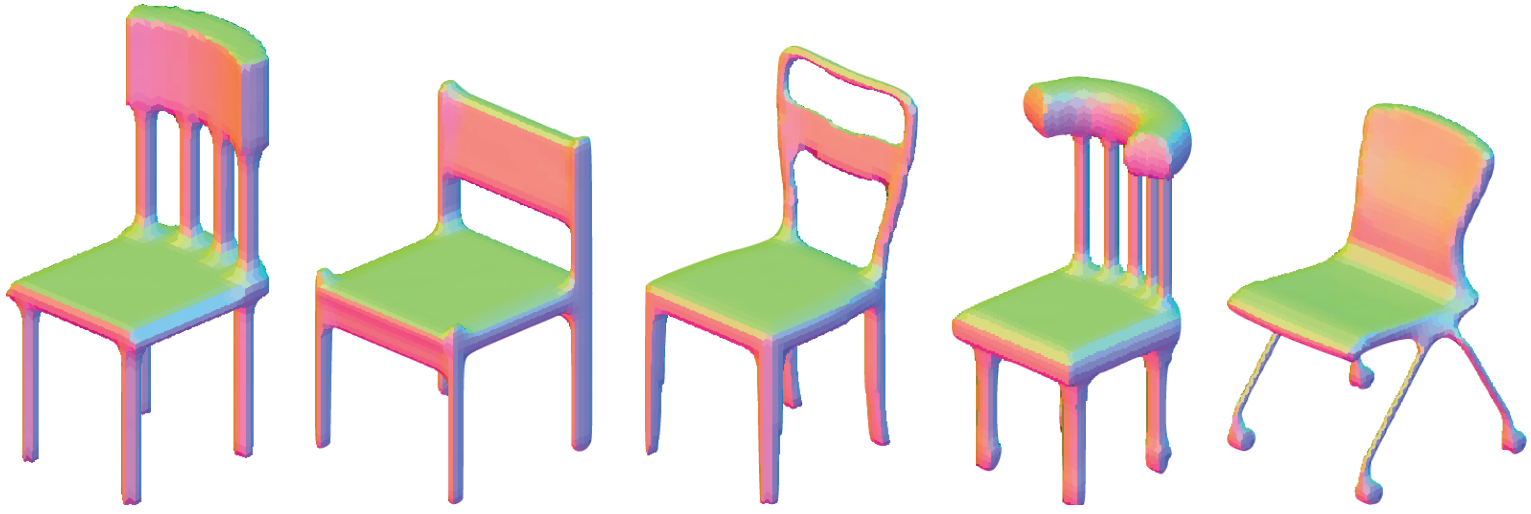
☹ Explicit FT and IFT

(Jaderberg et al. 2015); Yi et al. 2017

Example: normal prediction with SpecTN



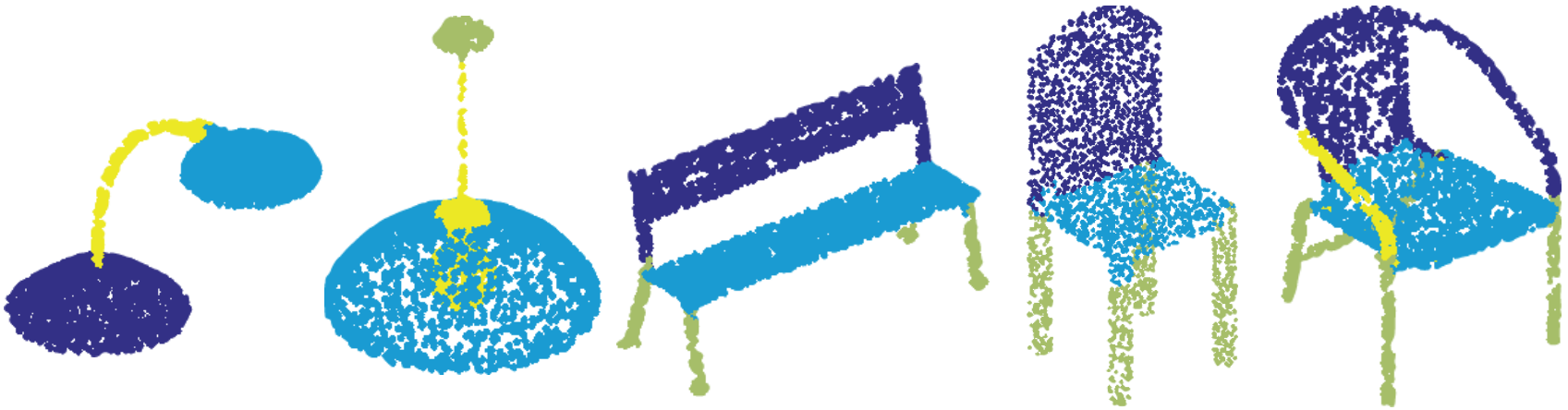
Predicted



Groundtruth

Yi et al. 2017

Example: shape segmentation with SpecTN



Predicted



Groundtruth

Yi et al. 2017

Localization and Smoothness

Vanishing moments: In the Euclidean setting

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

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

Localization and Smoothness

Vanishing moments: In the Euclidean setting

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

Localization in space = smoothness in frequency domain

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

Localization and Smoothness

Vanishing moments: In the Euclidean setting

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

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

Vanishing moments: In the Euclidean setting

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

Localization in space = smoothness in frequency domain

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

Application of the filter

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

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

Localization and Smoothness

Vanishing moments: In the Euclidean setting

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

Localization in space = smoothness in frequency domain

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

Application of the filter

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

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

Localization and Smoothness

Vanishing moments: In the Euclidean setting

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

Localization in space = smoothness in frequency domain

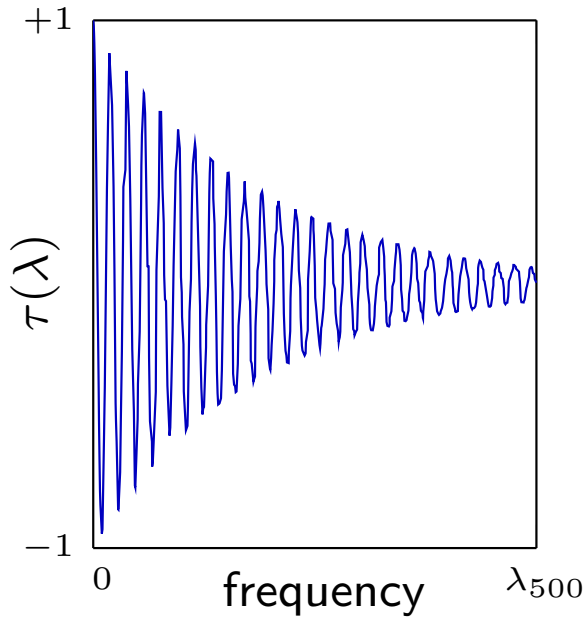
Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$

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

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

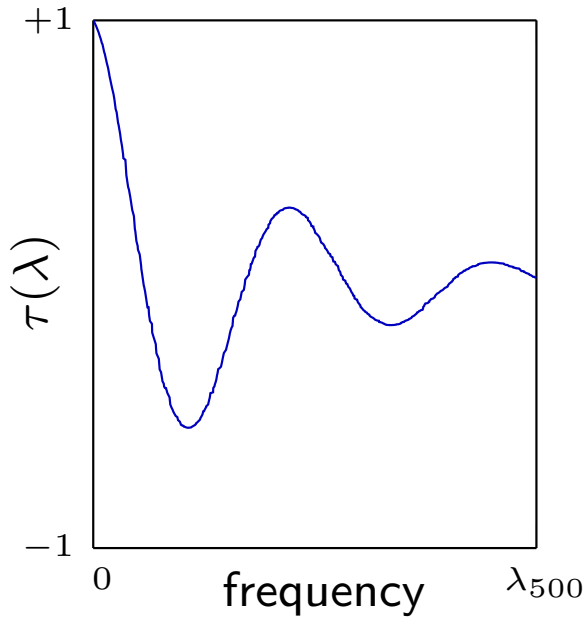
Bruna et al. 2014; Henaff, Bruna, LeCun 2015

Localization and smoothness



Non-smooth spectral filter (delocalized in space)

Localization and smoothness



Smooth spectral filter (localized in space)

Spectral graph CNN with polynomial filters (ChebNet)

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters

Defferrard, Bresson, Vandergheynst 2016

Spectral graph CNN with polynomial filters (ChebNet)

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters

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

Defferrard, Bresson, Vandergheynst 2016

Spectral graph CNN with polynomial filters (ChebNet)

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed r-hops support

Defferrard, Bresson, Vandergheynst 2016

Spectral graph CNN with polynomial filters (ChebNet)

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed r-hops support
- ☺ No explicit computation of $\boldsymbol{\Phi}^\top, \boldsymbol{\Phi} \Rightarrow \mathcal{O}(nr)$ computational complexity

Defferrard, Bresson, Vandergheynst 2016

Graph Convolutional Net: Simplified ChebNet

Use Chebychev polynomials of degree $r = 2$ and assume $\lambda_n \approx 2$

$$\begin{aligned} \tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} &= \alpha_0\mathbf{f} + \alpha_1(\boldsymbol{\Delta} - \mathbf{I})\mathbf{f} \\ &= \alpha_0\mathbf{f} - \alpha_1\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\mathbf{f} \end{aligned}$$

Kipf, Welling 2016

Graph Convolutional Net: Simplified ChebNet

Use Chebychev polynomials of degree $r = 2$ and assume $\lambda_n \approx 2$

$$\begin{aligned} \tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} &= \alpha_0\mathbf{f} + \alpha_1(\boldsymbol{\Delta} - \mathbf{I})\mathbf{f} \\ &= \alpha_0\mathbf{f} - \alpha_1\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\mathbf{f} \end{aligned}$$

Further constrain $\alpha = \alpha_0 = -\alpha_1$ to obtain a single-parameter filter

$$\tau_\alpha(\boldsymbol{\Delta})\mathbf{f} = \alpha(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2})\mathbf{f}$$

Kipf, Welling 2016

Graph Convolutional Net: Simplified ChebNet

Use Chebychev polynomials of degree $r = 2$ and assume $\lambda_n \approx 2$

$$\begin{aligned} \tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} &= \alpha_0\mathbf{f} + \alpha_1(\boldsymbol{\Delta} - \mathbf{I})\mathbf{f} \\ &= \alpha_0\mathbf{f} - \alpha_1\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\mathbf{f} \end{aligned}$$

Further constrain $\alpha = \alpha_0 = -\alpha_1$ to obtain a single-parameter filter

$$\tau_\alpha(\boldsymbol{\Delta})\mathbf{f} = \alpha(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2})\mathbf{f}$$

Caveat: The eigenvalues of $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ are now in $[0, 2]$

$\Rightarrow$ repeated application of the filter can result in numerical instability

Kipf, Welling 2016

Graph Convolutional Net: Simplified ChebNet

Use Chebychev polynomials of degree $r = 2$ and assume $\lambda_n \approx 2$

$$\begin{aligned} \tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} &= \alpha_0\mathbf{f} + \alpha_1(\boldsymbol{\Delta} - \mathbf{I})\mathbf{f} \\ &= \alpha_0\mathbf{f} - \alpha_1\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\mathbf{f} \end{aligned}$$

Further constrain $\alpha = \alpha_0 = -\alpha_1$ to obtain a single-parameter filter

$$\tau_\alpha(\boldsymbol{\Delta})\mathbf{f} = \alpha(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2})\mathbf{f}$$

Caveat: The eigenvalues of $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ are now in $[0, 2]$

$\Rightarrow$ repeated application of the filter can result in numerical instability

Remedy: Apply a renormalization

$$\tau_\alpha(\boldsymbol{\Delta})\mathbf{f} = \alpha\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{W}}\tilde{\mathbf{D}}^{-1/2}\mathbf{f}$$

where $\tilde{\mathbf{W}} = \mathbf{W} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \mathrm{diag}(\sum_{j\neq i}\tilde{w}_{ij})$.

Kipf, Welling 2016

Example: citation networks

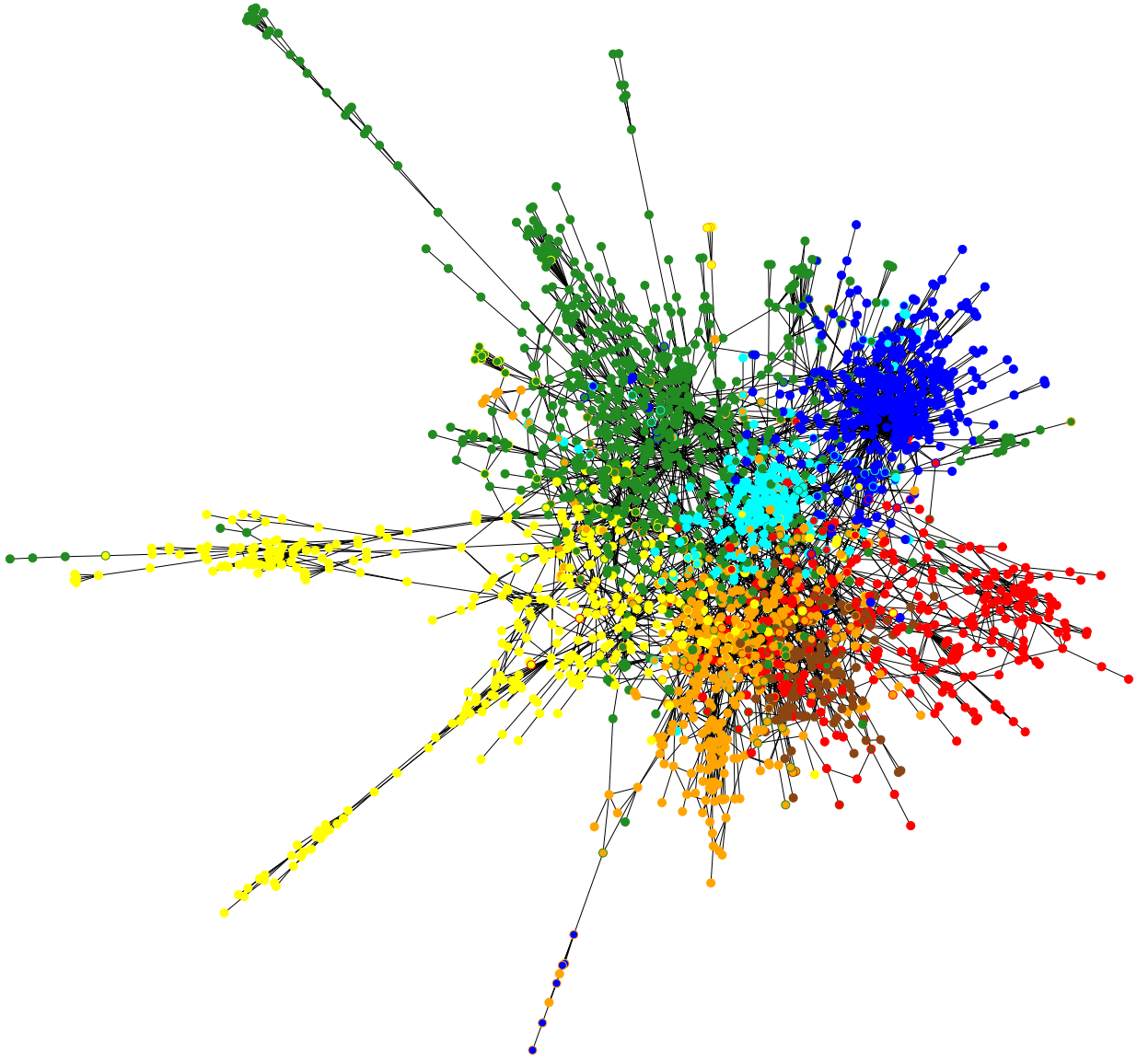


Figure: Monti et al. 2017

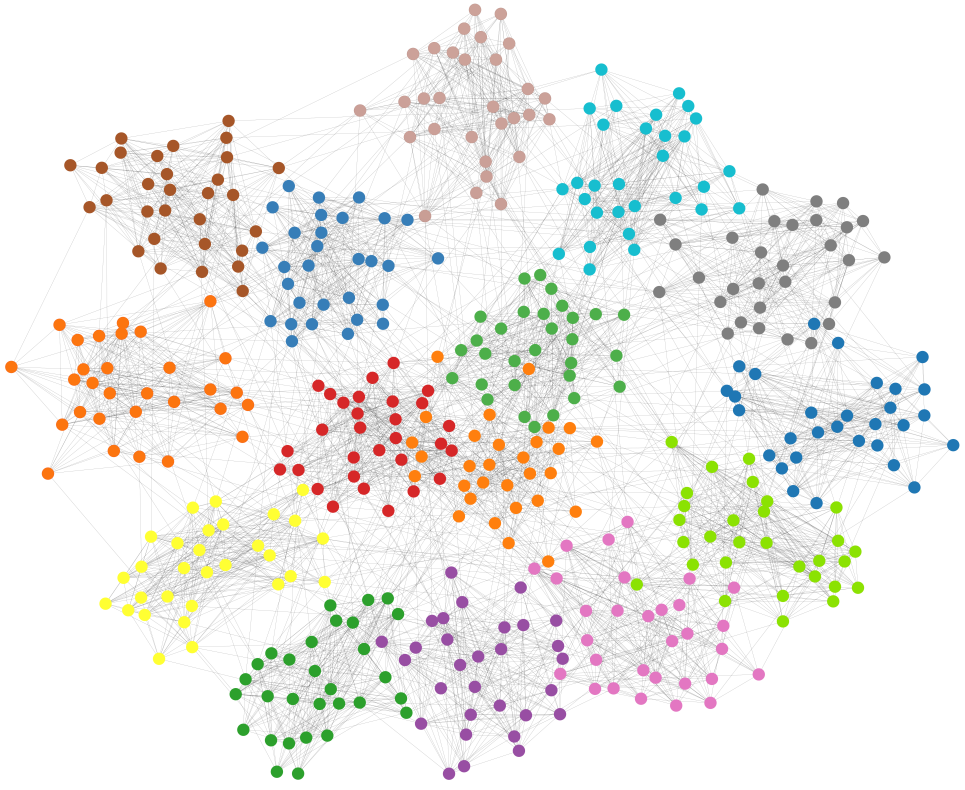
Example: citation networks

Method	**Cora**[1]	**PubMed**[2]
Manifold Regularization[3]	59.5%	70.7%
Semidefinite Embedding[4]	59.0%	71.1%
Label Propagation[5]	68.0%	63.0%
DeepWalk[6]	67.2%	65.3%
Planetoid[7]	75.7%	77.2%
GCN[8]	**81.6%**	**78.7%**

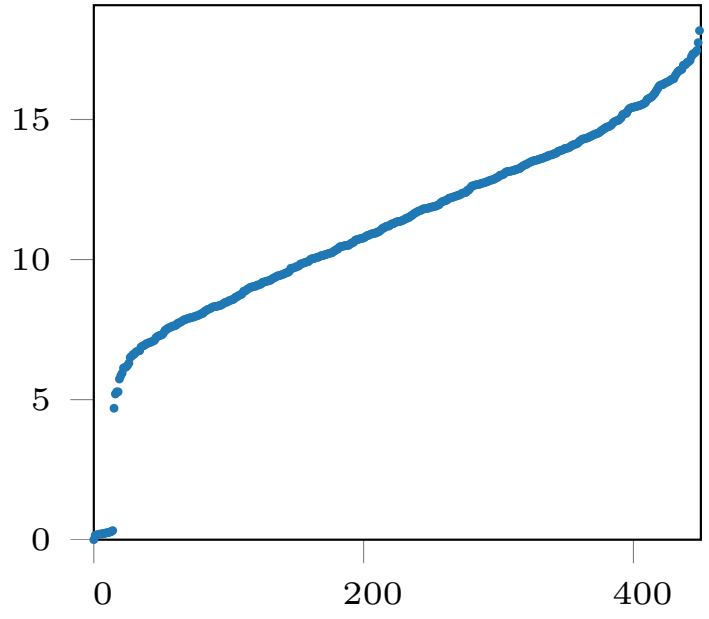
Classification accuracy of different methods on citation network datasets

Data: [1,2]Sen et al. 2008; methods: [3]Belkin et al. 2006; [4]Weston et al. 2012; [5]Zhu et al. 2003; [6]Perozzi et al. 2014; [7]Yang et al. 2016; [8]Kipf, Welling 2016

Community graph example



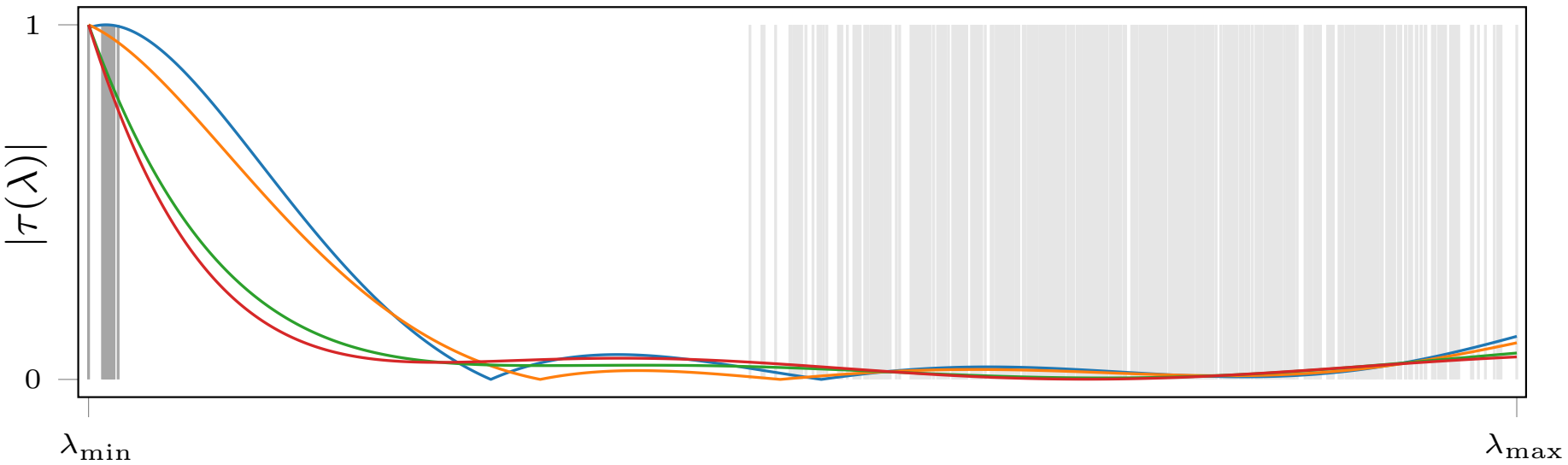
Synthetic graph with 15 communities



Laplacian eigenvalues

Levie et al. 2017

Community graph example



Example of Chebyshev filters learned on the 15-communities graph

Levie et al. 2017

Cayley transform

Cayley transform

$$C(\lambda) = \frac{\lambda - i}{\lambda + i}$$

is a smooth bijection from $\mathbb{R}$ to the complex unit circle

Levie et al. 2017

Cayley transform

Cayley transform

$$C(\lambda) = \frac{\lambda - i}{\lambda + i}$$

is a smooth bijection from $\mathbb{R}^+$ to the complex unit half-circle

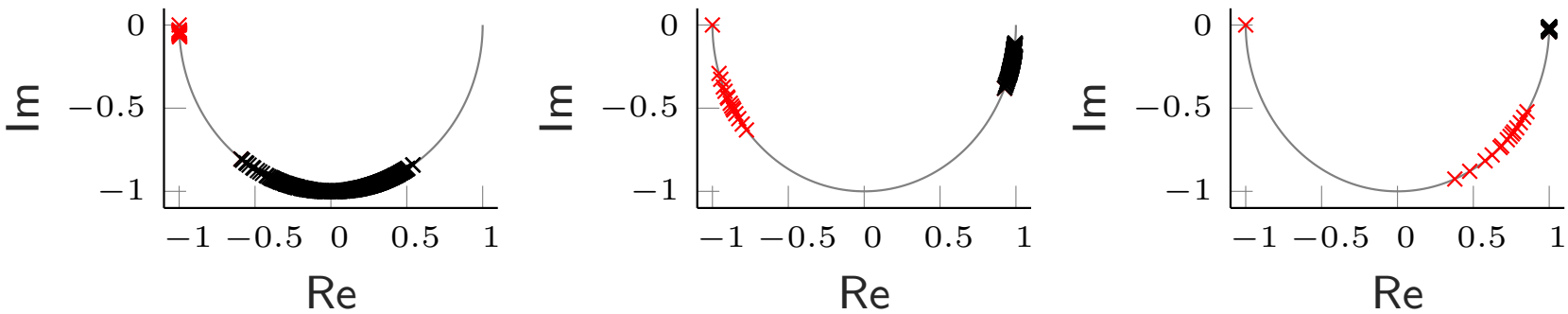
Levie et al. 2017

Cayley transform

Cayley transform with zoom

$$C(h\lambda) = \frac{h\lambda - i}{h\lambda + i}$$

is a smooth bijection from $\mathbb{R}^+$ to the complex unit half-circle allowing nonlinear 'magnification' of a region



Cayley transform $C(h\lambda)$ for (left-to-right) $h = 0.1$, 1, and 10 of the 15-communities graph Laplacian spectrum

Levie et al. 2017

Spectral graph CNN with Cayley polynomials (CayleyNet)

Represent spectral transfer function as a Cayley polynomial or order r

$$\tau_{\mathbf{c},h}(\lambda) = c_0 + 2\mathrm{Re}\Big\{ \sum_{j=1}^{r} c_j (h\lambda - i)^j (h\lambda + i)^{-j} \Big\}$$

where the filter parameters are the vector of real/complex coefficients $\mathbf{c} = (c_0, \ldots, c_r)^\top$ and the spectral zoom h

Levie et al. 2017

Spectral graph CNN with Cayley polynomials (CayleyNet)

Represent spectral transfer function as a Cayley polynomial or order r

$$\tau_{\mathbf{c},h}(\lambda) = c_0 + 2\mathrm{Re}\Big\{ \sum_{j=1}^{r} c_j (h\lambda - i)^j (h\lambda + i)^{-j} \Big\}$$

where the filter parameters are the vector of real/complex coefficients $\mathbf{c} = (c_0, \dots, c_r)^\top$ and the spectral zoom h

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

☺ Filters have guaranteed exponential spatial decay

☹ $\mathcal{O}(n^3)$ computational complexity with direct matrix inversion

Levie et al. 2017

Spectral graph CNN with Cayley polynomials (CayleyNet)

Represent spectral transfer function as a Cayley polynomial or order r

$$\tau_{\mathbf{c},h}(\lambda) = c_0 + 2\mathrm{Re}\Big\{ \sum_{j=1}^{r} c_j (h\lambda - i)^j (h\lambda + i)^{-j} \Big\}$$

where the filter parameters are the vector of real/complex coefficients $\mathbf{c} = (c_0, \ldots, c_r)^\top$ and the spectral zoom h

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed exponential spatial decay
- ☺ $\mathcal{O}(n)$ computational complexity with **Jacobi approximate matrix inversion** (assuming sparsely-connected graph)

Levie et al. 2017

Spectral graph CNN with Cayley polynomials (CayleyNet)

Represent spectral transfer function as a Cayley polynomial or order r

$$\tau_{\mathbf{c},h}(\lambda) = c_0 + 2\mathrm{Re}\Big\{ \sum_{j=1}^{r} c_j (h\lambda - i)^j (h\lambda + i)^{-j} \Big\}$$

where the filter parameters are the vector of real/complex coefficients $\mathbf{c} = (c_0, \ldots, c_r)^\top$ and the spectral zoom h

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed exponential spatial decay
- ☺ $\mathcal{O}(n)$ computational complexity with **Jacobi approximate matrix inversion** (assuming sparsely-connected graph)
- ☺ Spectral zoom property allowing to better localize in frequency

Levie et al. 2017

Spectral graph CNN with Cayley polynomials (CayleyNet)

Represent spectral transfer function as a Cayley polynomial or order r

$$\tau_{\mathbf{c},h}(\lambda) = c_0 + 2\mathrm{Re}\Big\{ \sum_{j=1}^{r} c_j (h\lambda - i)^j (h\lambda + i)^{-j} \Big\}$$

where the filter parameters are the vector of real/complex coefficients $\mathbf{c} = (c_0, \ldots, c_r)^\top$ and the spectral zoom h

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed exponential spatial decay
- ☺ $\mathcal{O}(n)$ computational complexity with **Jacobi approximate matrix inversion** (assuming sparsely-connected graph)
- ☺ Spectral zoom property allowing to better localize in frequency
- ☺ Richer class of filters than polynomials for the same order

Levie et al. 2017

Fast inversion

Application of Cayley filter $\tau(h\boldsymbol{\Delta})\mathbf{f}$ requires the solution or recursive linear system

$$\begin{aligned} \mathbf{y}_0 &= \mathbf{f} \\ (h\boldsymbol{\Delta} + i\mathbf{I})\mathbf{y}_j &= (h\boldsymbol{\Delta} - i\mathbf{I})\mathbf{y}_{j-1} \qquad\qquad j = 1, \ldots, r \end{aligned}$$

(Jacobi 1834); Levie et al. 2017

Fast inversion

Application of Cayley filter $\tau(h\boldsymbol{\Delta})\mathbf{f}$ requires the solution or recursive linear system

$$\begin{aligned} \mathbf{y}_0 &= \mathbf{f} \\ (h\boldsymbol{\Delta} + i\mathbf{I})\mathbf{y}_j &= (h\boldsymbol{\Delta} - i\mathbf{I})\mathbf{y}_{j-1} \qquad j = 1, \ldots, r \end{aligned}$$

(stable since $C(h\boldsymbol{\Delta})$ is unitary and has condition number $= 1$)

(Jacobi 1834); Levie et al. 2017

Fast inversion

Application of Cayley filter $\tau(h\mathbf{\Delta})\mathbf{f}$ requires the solution or recursive linear system

$$\begin{aligned} \mathbf{y}_0 &= \mathbf{f} \\ (h\mathbf{\Delta} + i\mathbf{I})\mathbf{y}_j &= (h\mathbf{\Delta} - i\mathbf{I})\mathbf{y}_{j-1} \qquad j = 1, \ldots, r \end{aligned}$$

Exact solution: $\mathcal{O}(n^3)$ complexity

(Jacobi 1834); Levie et al. 2017

Fast inversion

Application of Cayley filter $\tau(h\boldsymbol{\Delta})\mathbf{f}$ requires the solution or recursive linear system

$$\begin{aligned} \mathbf{y}_0 &= \mathbf{f} \\ (h\boldsymbol{\Delta} + i\mathbf{I})\mathbf{y}_j &= (h\boldsymbol{\Delta} - i\mathbf{I})\mathbf{y}_{j-1} \qquad j = 1, \ldots, r \end{aligned}$$

Approximate solution $\tilde{\mathbf{y}}_j \approx \mathbf{y}_j$ using K Jacobi iterations

$$\begin{aligned} \tilde{\mathbf{y}}_j^{(k+1)} &= \mathbf{J}\tilde{\mathbf{y}}_j^{(k)} + \mathrm{Diag}^{-1}(h\boldsymbol{\Delta} + i\mathbf{I})(h\boldsymbol{\Delta} - i\mathbf{I})\tilde{\mathbf{y}}_{j-1} \\ \tilde{\mathbf{y}}_j^{(0)} &= \mathrm{Diag}^{-1}(h\boldsymbol{\Delta} + i\mathbf{I})(h\boldsymbol{\Delta} - i\mathbf{I})\tilde{\mathbf{y}}_{j-1} \end{aligned}$$

with $\mathbf{J} = -\mathrm{Diag}^{-1}(h\boldsymbol{\Delta} + i\mathbf{I})\mathrm{Off}(h\boldsymbol{\Delta} + i\mathbf{I})$

(Jacobi 1834); Levie et al. 2017

Fast inversion

Application of Cayley filter $\tau(h\mathbf{\Delta})\mathbf{f}$ requires the solution or recursive linear system

$$\begin{aligned} \mathbf{y}_0 &= \mathbf{f} \\ (h\mathbf{\Delta} + i\mathbf{I})\mathbf{y}_j &= (h\mathbf{\Delta} - i\mathbf{I})\mathbf{y}_{j-1} \qquad j = 1, \dots, r \end{aligned}$$

Approximate solution $\tilde{\mathbf{y}}_j \approx \mathbf{y}_j$ using K Jacobi iterations

$$\begin{aligned} \tilde{\mathbf{y}}_j^{(k+1)} &= \mathbf{J}\tilde{\mathbf{y}}_j^{(k)} + \mathrm{Diag}^{-1}(h\mathbf{\Delta} + i\mathbf{I})(h\mathbf{\Delta} - i\mathbf{I})\tilde{\mathbf{y}}_{j-1} \\ \tilde{\mathbf{y}}_j^{(0)} &= \mathrm{Diag}^{-1}(h\mathbf{\Delta} + i\mathbf{I})(h\mathbf{\Delta} - i\mathbf{I})\tilde{\mathbf{y}}_{j-1} \end{aligned}$$

with $\mathbf{J} = -\mathrm{Diag}^{-1}(h\mathbf{\Delta} + i\mathbf{I})\mathrm{Off}(h\mathbf{\Delta} + i\mathbf{I})$

Application of the approximate Cayley filter

$$\sum_{j=0}^{r} c_j \tilde{\mathbf{y}}_j \approx \tau(h\mathbf{\Delta})\mathbf{f}$$

has $\mathcal{O}(rK|\mathcal{E}|)$ complexity

(Jacobi 1834); Levie et al. 2017

Fast inversion

Application of Cayley filter $\tau(h\mathbf{\Delta})\mathbf{f}$ requires the solution or recursive linear system

$$\begin{aligned} \mathbf{y}_0 &= \mathbf{f} \\ (h\mathbf{\Delta} + i\mathbf{I})\mathbf{y}_j &= (h\mathbf{\Delta} - i\mathbf{I})\mathbf{y}_{j-1} \qquad j = 1, \ldots, r \end{aligned}$$

Approximate solution $\tilde{\mathbf{y}}_j \approx \mathbf{y}_j$ using K Jacobi iterations

$$\begin{aligned} \tilde{\mathbf{y}}_j^{(k+1)} &= \mathbf{J}\tilde{\mathbf{y}}_j^{(k)} + \mathrm{Diag}^{-1}(h\mathbf{\Delta} + i\mathbf{I})(h\mathbf{\Delta} - i\mathbf{I})\tilde{\mathbf{y}}_{j-1} \\ \tilde{\mathbf{y}}_j^{(0)} &= \mathrm{Diag}^{-1}(h\mathbf{\Delta} + i\mathbf{I})(h\mathbf{\Delta} - i\mathbf{I})\tilde{\mathbf{y}}_{j-1} \end{aligned}$$

with $\mathbf{J} = -\mathrm{Diag}^{-1}(h\mathbf{\Delta} + i\mathbf{I})\mathrm{Off}(h\mathbf{\Delta} + i\mathbf{I})$

Application of the approximate Cayley filter

$$\sum_{j=0}^{r} c_j \tilde{\mathbf{y}}_j \approx \tau(h\mathbf{\Delta})\mathbf{f}$$

has $\mathcal{O}(rKn)$ complexity for a sparse graph

(Jacobi 1834); Levie et al. 2017

Error bound

Unnormalized Laplacian: $d = \max_j d_{jj}$ and $\kappa = \|\mathbf{J}\|_\infty = \frac{hd}{\sqrt{h^2 d^2 + 1}} < 1$

Normalized Laplacian: Assume that $(h\boldsymbol{\Delta} + i\mathbf{I})$ is dominant diagonal s.t. $\kappa = \|\mathbf{J}\|_\infty < 1$

(Jacobi 1834); Levie et al. 2017

Error bound

Unnormalized Laplacian: $d = \max_j d_{jj}$ and $\kappa = \|\mathbf{J}\|_\infty = \frac{hd}{\sqrt{h^2 d^2 + 1}} < 1$

Normalized Laplacian: Assume that $(h\mathbf{\Delta} + i\mathbf{I})$ is dominant diagonal s.t. $\kappa = \|\mathbf{J}\|_\infty < 1$

Theorem (approximation error) Under the above assumptions

$$\frac{\left\| \sum_{j=0}^{r} c_j \tilde{\mathbf{y}}_j - \tau(h\mathbf{\Delta})\mathbf{f} \right\|_2}{\|\tau(h\mathbf{\Delta})\mathbf{f}\|_2} < M\kappa^K$$

where $M = \sqrt{n} \sum_{j=1}^{r} j|c_j|$ for a general graph and $M = \sum_{j=1}^{r} j|c_j|$ for a regular graph

(Jacobi 1834); Levie et al. 2017

Exponential decay

Exponential decay on graphs $f \in L^p(\mathcal{V})$, $1 \leq p \leq \infty$ has exponential decay about vertex m if $\exists \gamma \in (0,1)$ and $c > 0$ such that for any k

$$\|f|_{\mathcal{N}_{k,m}^c}\|_{L^p(\mathcal{V})} \leq c\gamma^k \|f\|_{L^p(\mathcal{V})}$$

where $\mathcal{N}_{k,m}$ is the k-hop neighborhood of vertex m

Levie et al. 2017

Exponential decay

Exponential decay on graphs $f \in L^p(\mathcal{V})$, $1 \leq p \leq \infty$ has exponential decay about vertex m if $\exists \gamma \in (0,1)$ and $c > 0$ such that for any k

$$\|f|_{\mathcal{N}_{k,m}^c}\|_{L^p(\mathcal{V})} \leq c\gamma^k \|f\|_{L^p(\mathcal{V})}$$

where $\mathcal{N}_{k,m}$ is the k-hop neighborhood of vertex m

Compare to

Exponential decay in $\mathbb{R}$ $f(x)$ has exponential decay (about 0) if $\exists \gamma \in (0,1)$ and $c > 0$ such that for any $x \in \mathbb{R}$

$$|f(x)| \leq c\gamma^{-x}$$

Levie et al. 2017

Exponential decay

Exponential decay on graphs $f \in L^p(\mathcal{V})$, $1 \leq p \leq \infty$ has exponential decay about vertex m if $\exists \gamma \in (0,1)$ and $c > 0$ such that for any k

$$\|f|_{\mathcal{N}_{k,m}^c}\|_{L^p(\mathcal{V})} \leq c\gamma^k \|f\|_{L^p(\mathcal{V})}$$

where $\mathcal{N}_{k,m}$ is the k-hop neighborhood of vertex m

Compare to

Exponential decay in $\mathbb{R}$ $f(x)$ has exponential decay (about 0) if $\exists \gamma \in (0,1)$ and $\frac{c}{\|f\|_\infty} > 0$ such that for any $\rho > 0$

$$\|f|_{B_\rho^c}\|_\infty \leq \frac{c}{\|f\|_\infty} \gamma^{-\rho} \|f\|_\infty$$

where B_ρ is a ball of radius ρ about 0

Levie et al. 2017

Exponential decay

Exponential decay on graphs $f \in L^p(\mathcal{V})$, $1 \leq p \leq \infty$ has exponential decay about vertex m if $\exists \gamma \in (0,1)$ and $c > 0$ such that for any k

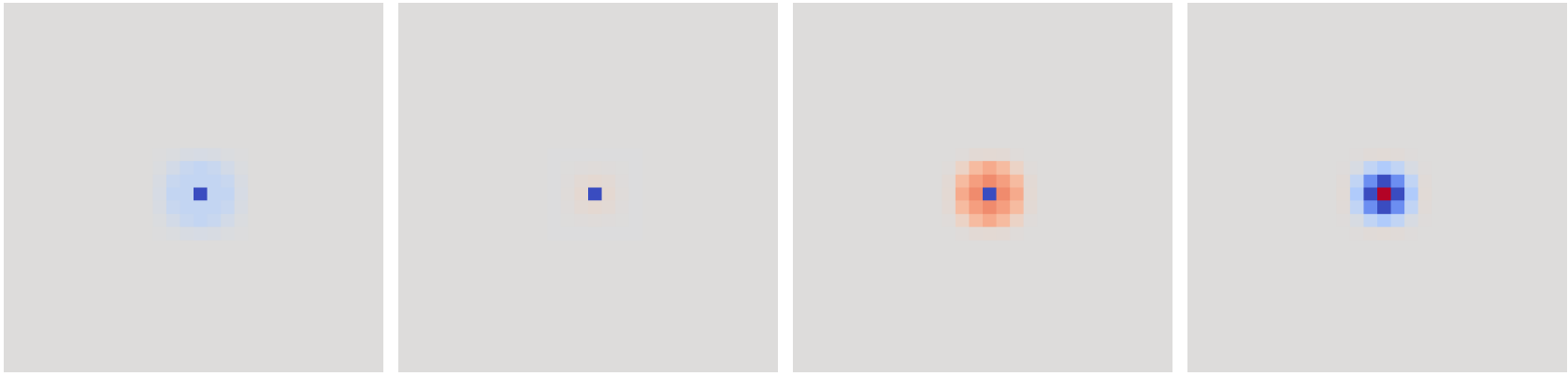
$$\|f|_{\mathcal{N}_{k,m}^c}\|_{L^p(\mathcal{V})} \leq c\gamma^k \|f\|_{L^p(\mathcal{V})}$$

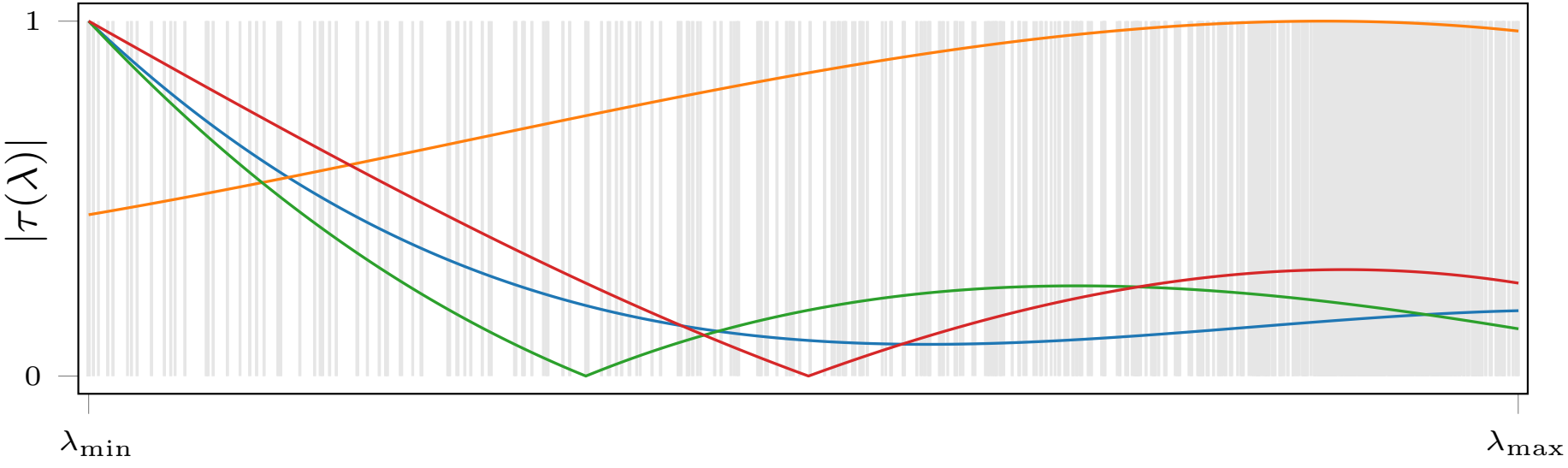
where $\mathcal{N}_{k,m}$ is the k-hop neighborhood of vertex m

Theorem (exponential decay of Cayley filters) Let $\tau(h\boldsymbol{\Delta})$ be a Cayley filter of order r and δ_m a delta-function at vertex m of the graph. Then, $\tau(h\boldsymbol{\Delta})\boldsymbol{\delta}_m$ has exponential decay about m with $p = 2$, $c = 2M/\|\tau(\boldsymbol{\Delta})\boldsymbol{\delta}_m\|_2$ and $\gamma = \kappa^{1/r}$

Levie et al. 2017

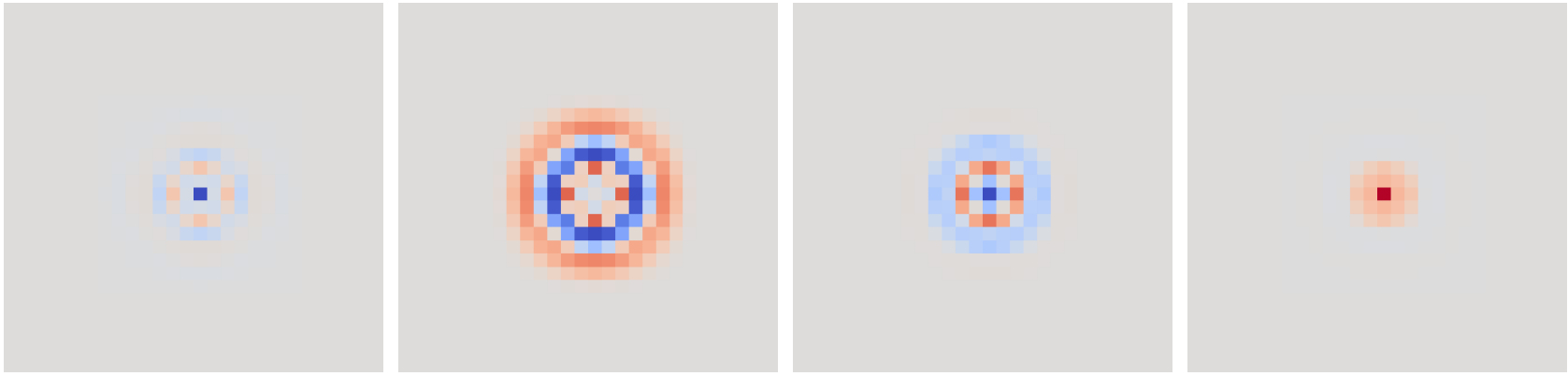
Chebyshev vs Cayley

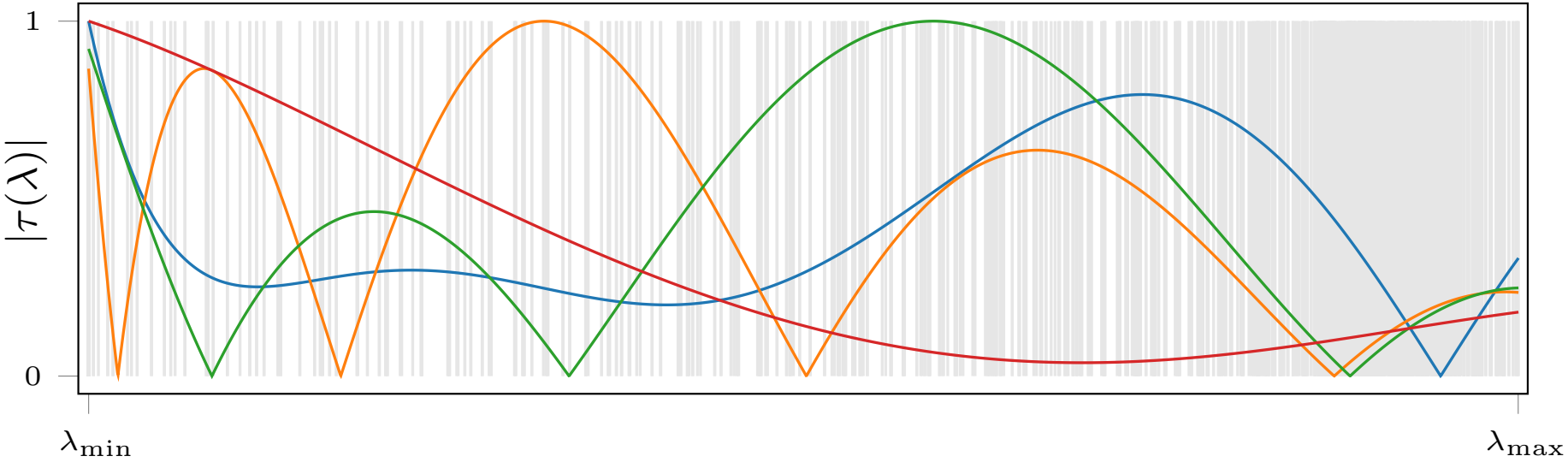




Example of Chebyshev filters (order $r = 3$) on Euclidean grid

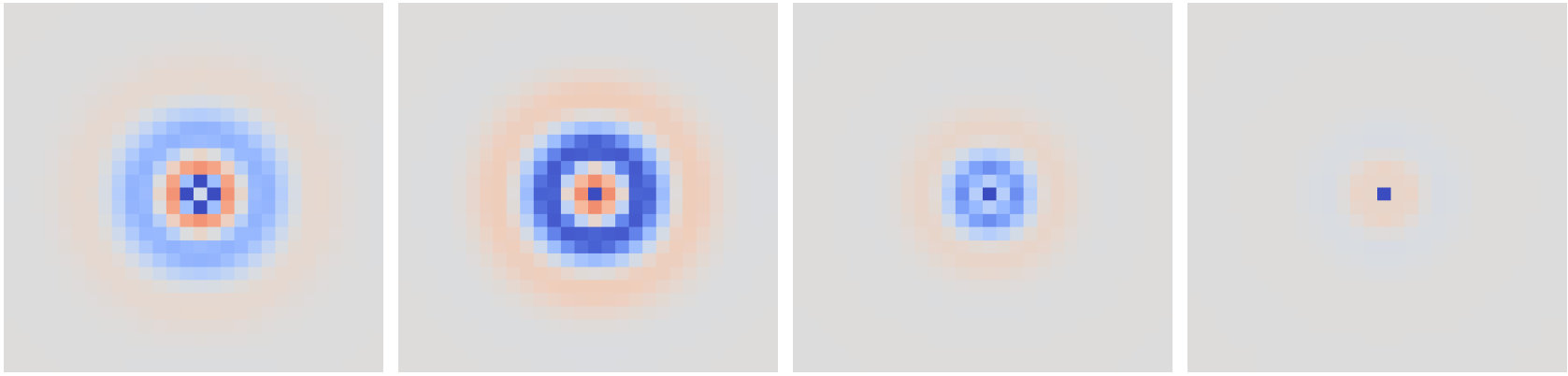
Chebyshev vs Cayley

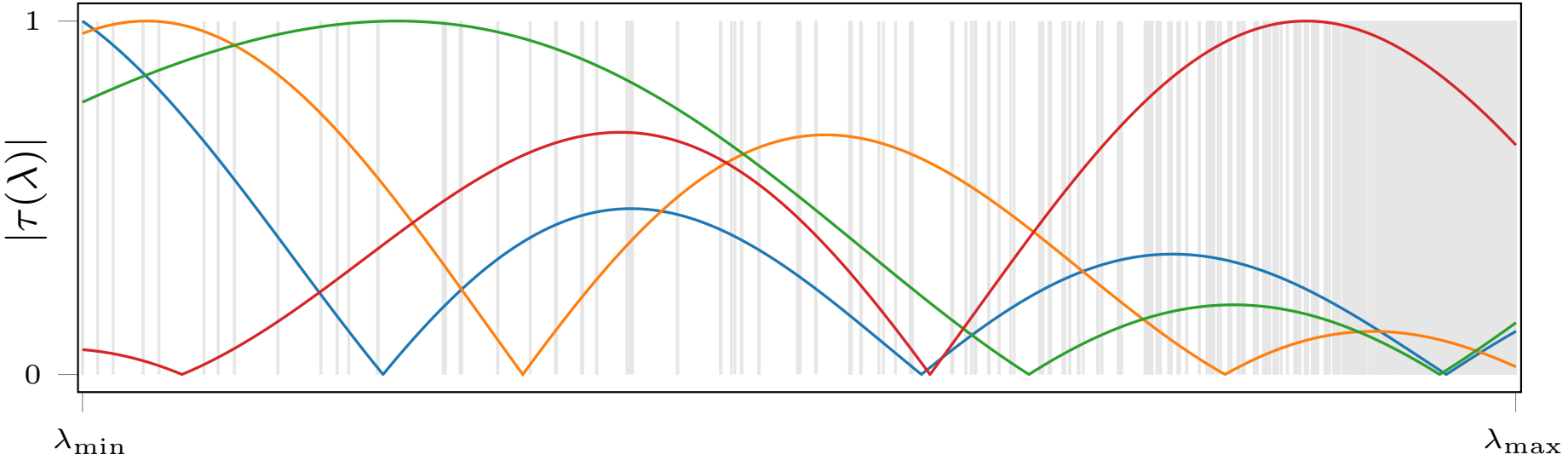




Example of Chebyshev filters (order $r = 7$) on Euclidean grid

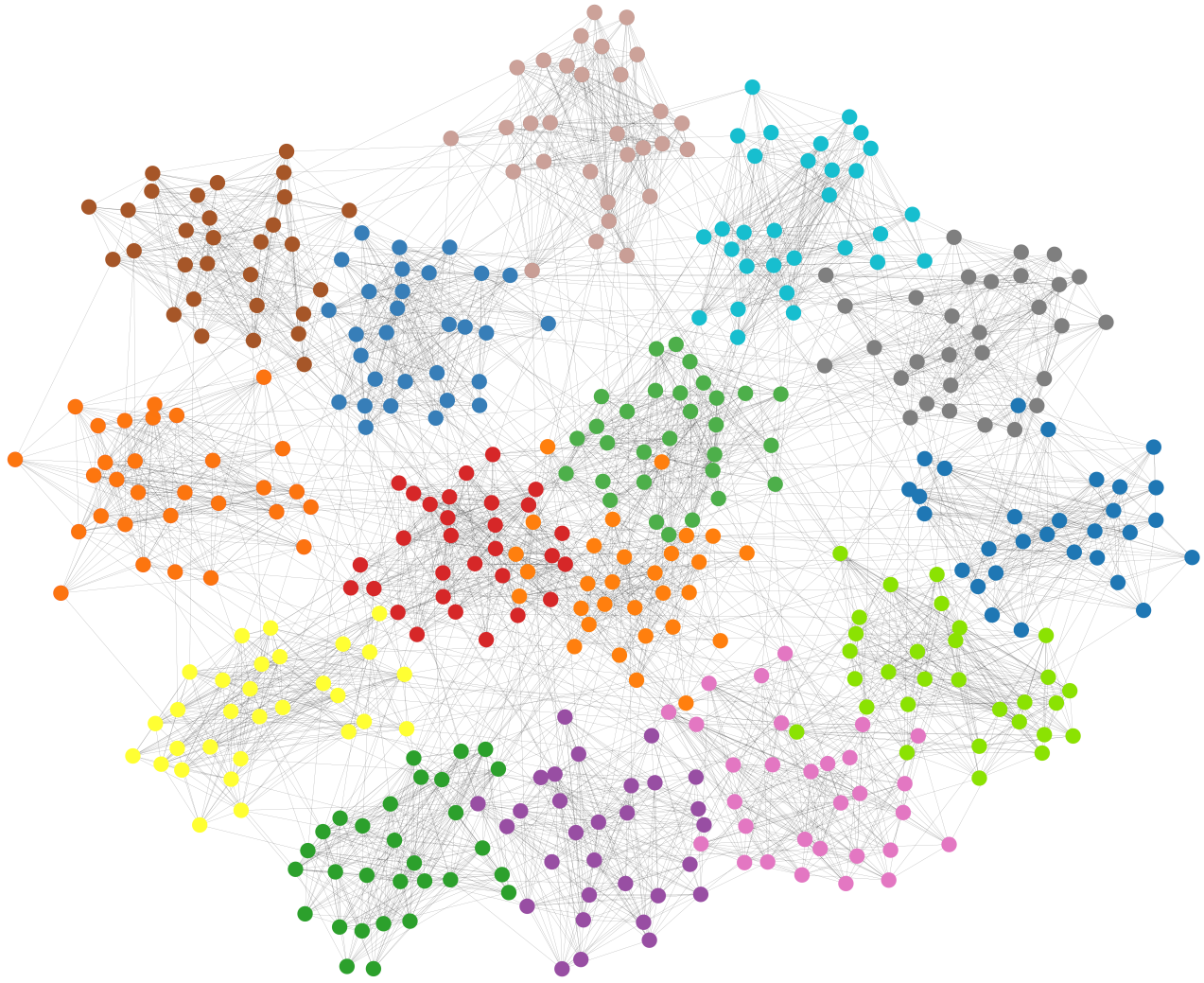
Chebyshev vs Cayley





Example of Cayley filters (order $r = 3$) on Euclidean grid

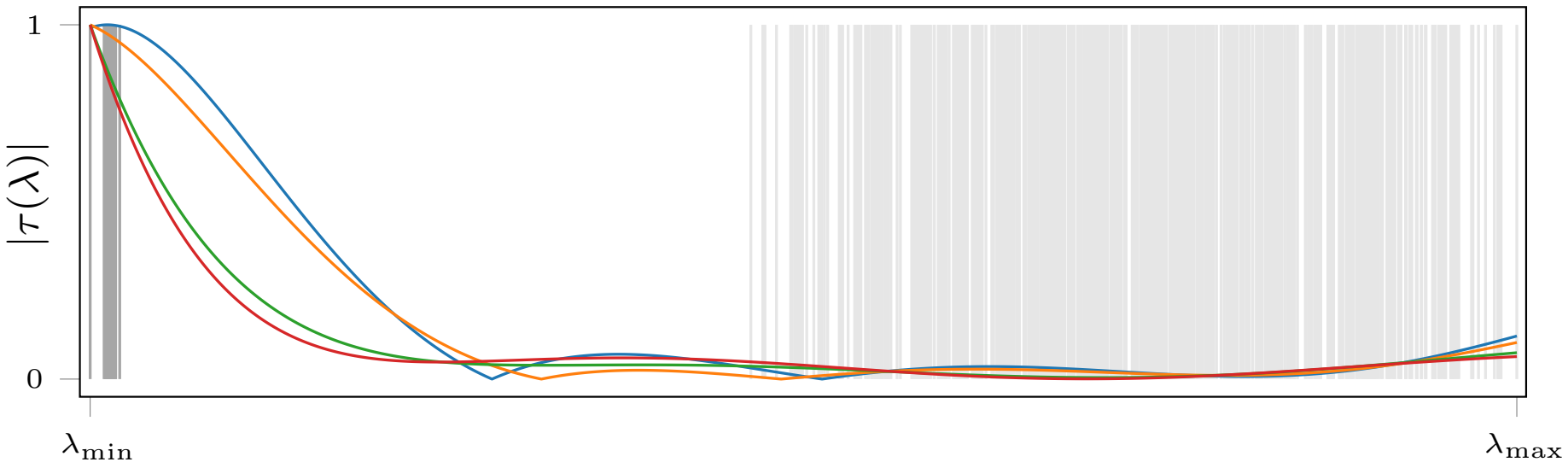
Chebyshev vs Cayley: community graph example



Synthetic graph with 15 communities

Levie et al. 2017

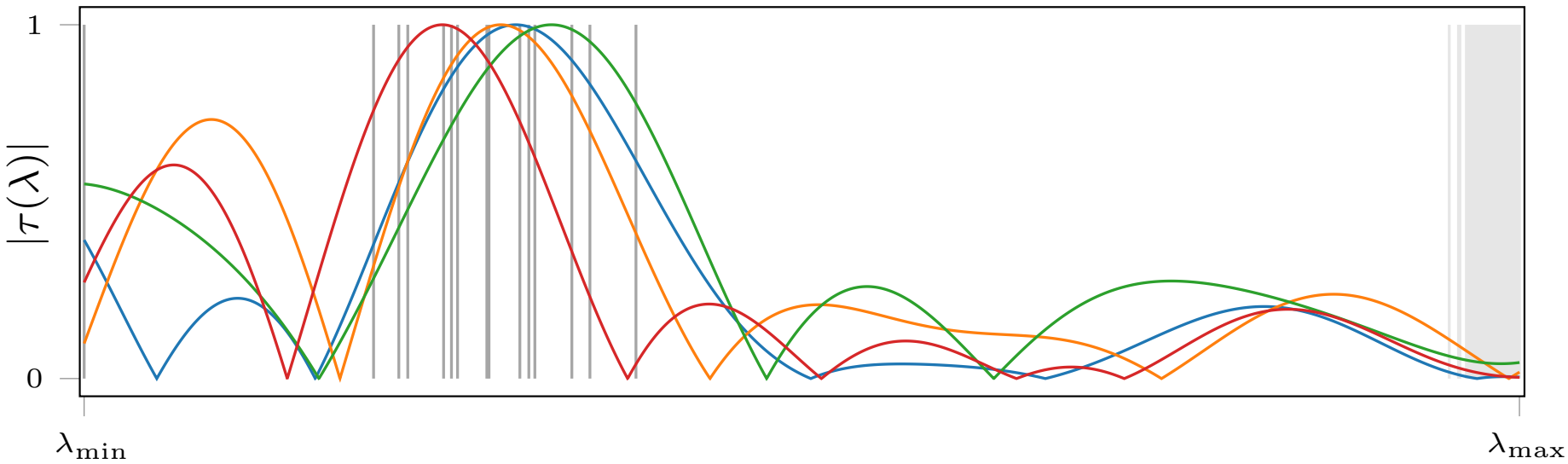
Chebyshev vs Cayley: community graph example



Example of Chebyshev filters learned on the 15-communities graph

Levie et al. 2017

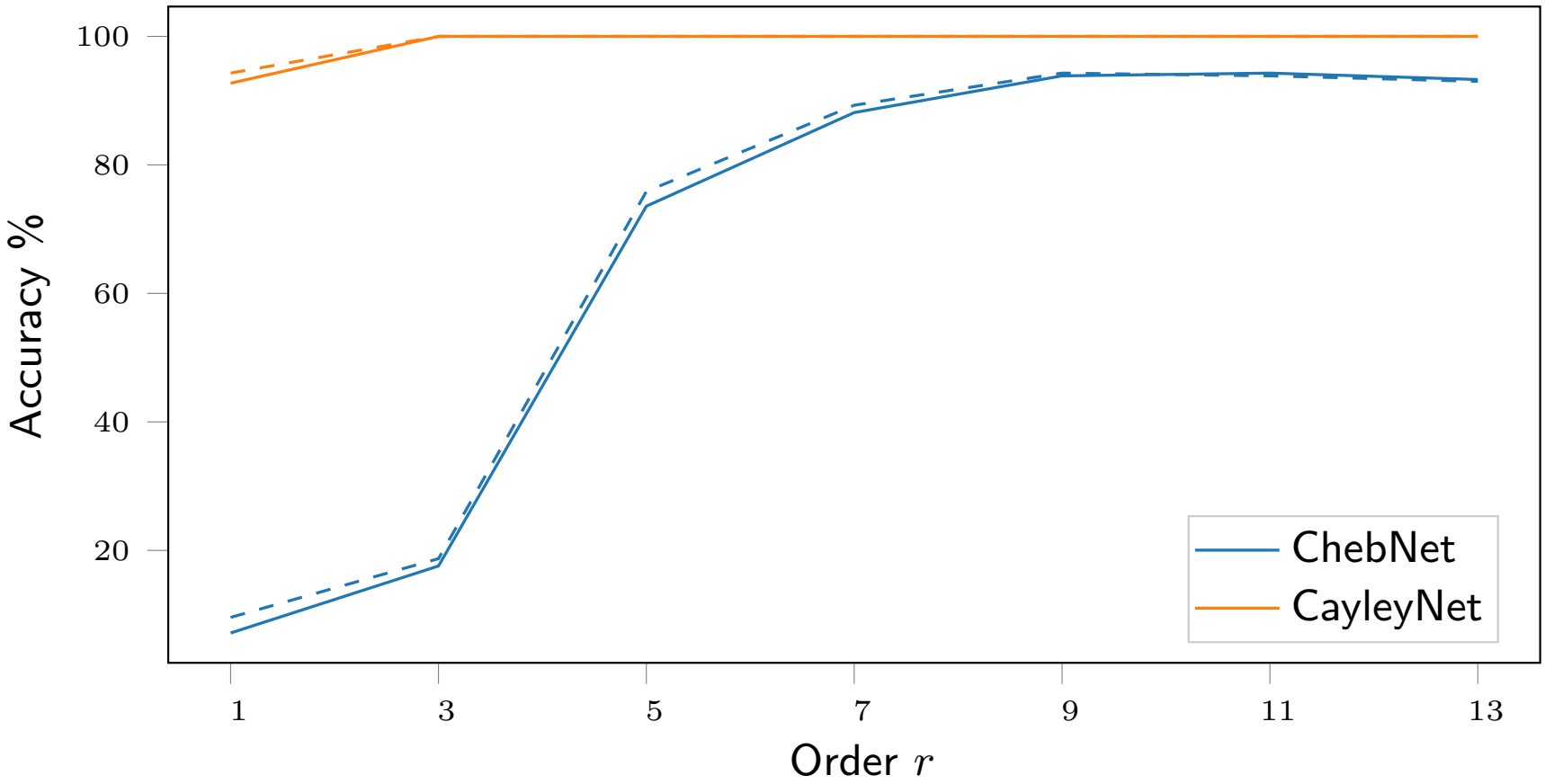
Chebyshev vs Cayley: community graph example



Example of Cayley filters learned on the 15-communities graph

Levie et al. 2017

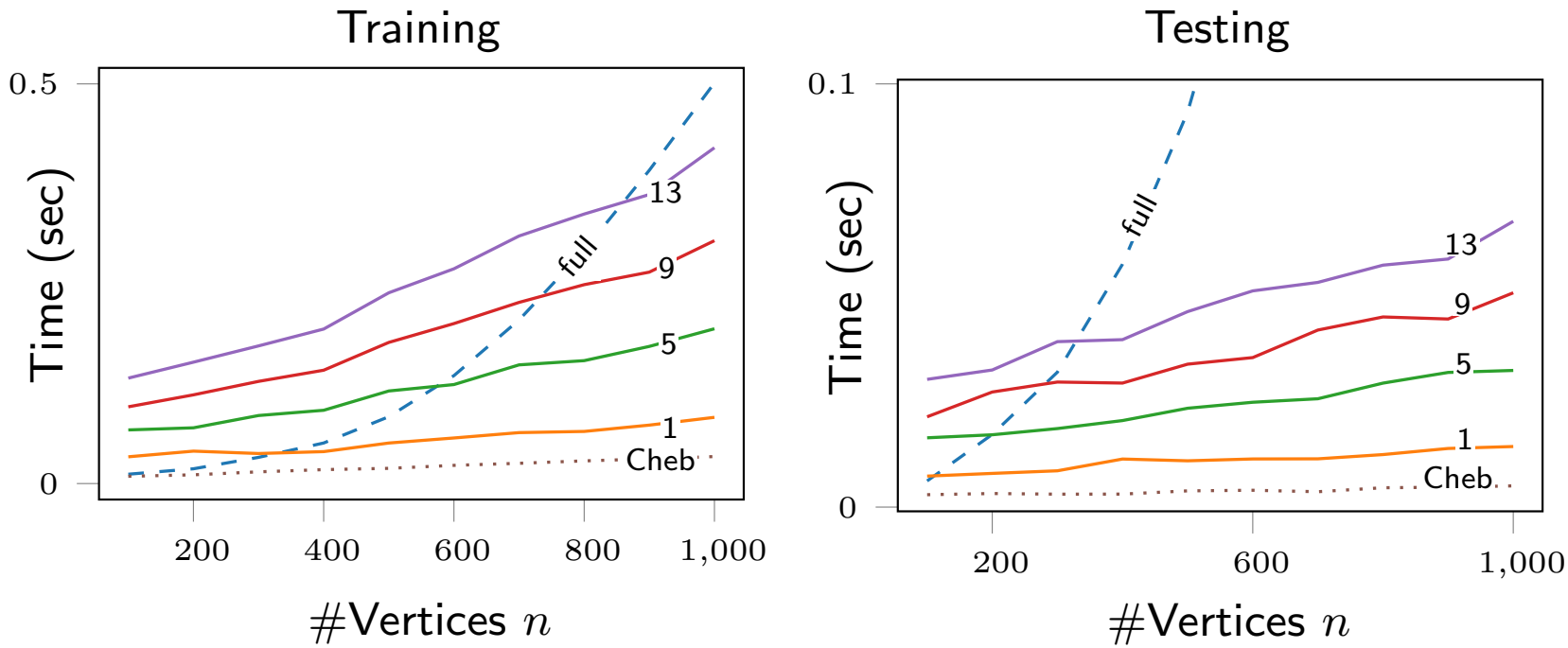
Chebyshev vs Cayley: community graph example



Community detection accuracy of ChebNet and CayleyNet on the synthetic 15-community graph

Levie et al. 2017

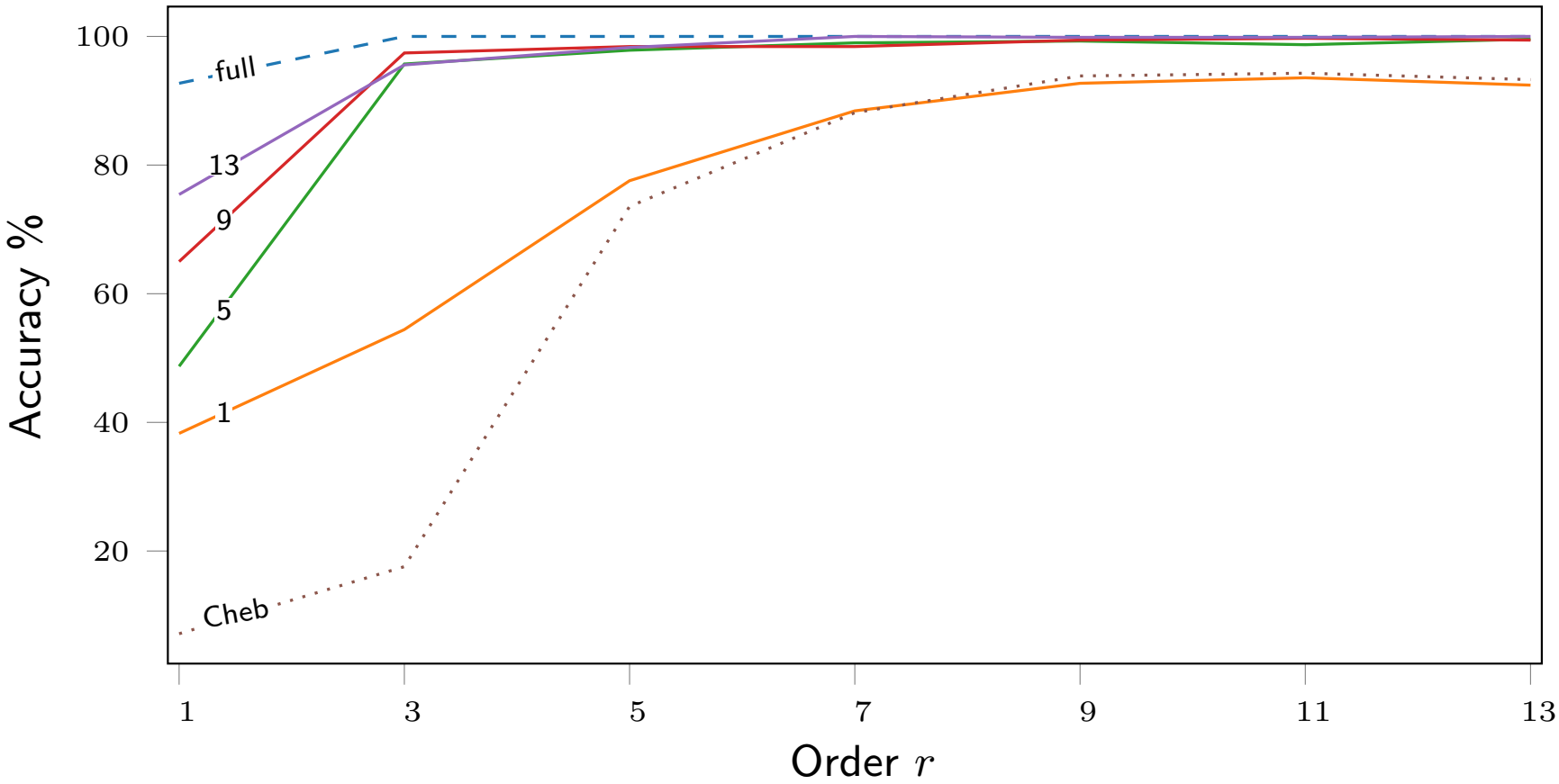
Computational complexity of approximate inversion



Training and testing computational complexity of CayleyNet with approximate Jacobi inversion for graphs of different size

Levie et al. 2017

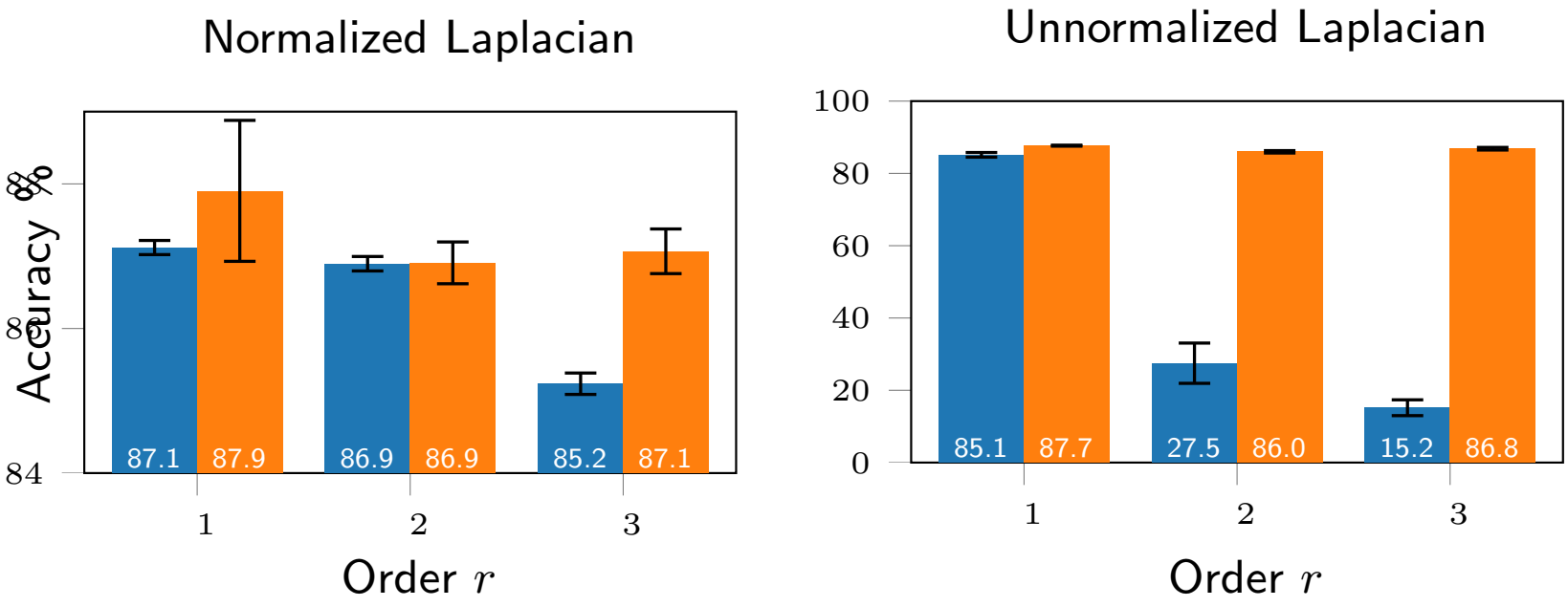
Accuracy of approximate inversion



Community detection accuracy of CayleyNet using approximate Jacobi inversion on the synthetic 15-community graph

Levie et al. 2017

Chebyshev vs Cayley: citation network example



Accuracy of ChebNet (blue) and CayleyNet (orange) on Cora dataset

Levie et al. 2017; data: Set et al. 2008

Limitations of spectral methods

- ☹ Poor generalization across non-isometric domains unless kernels are localized

Limitations of spectral methods

- ☹ Poor generalization across non-isometric domains unless kernels are localized (can be remedied to some extent with spectral transformer networks[1])

[1] Yi, Su et al. 2017

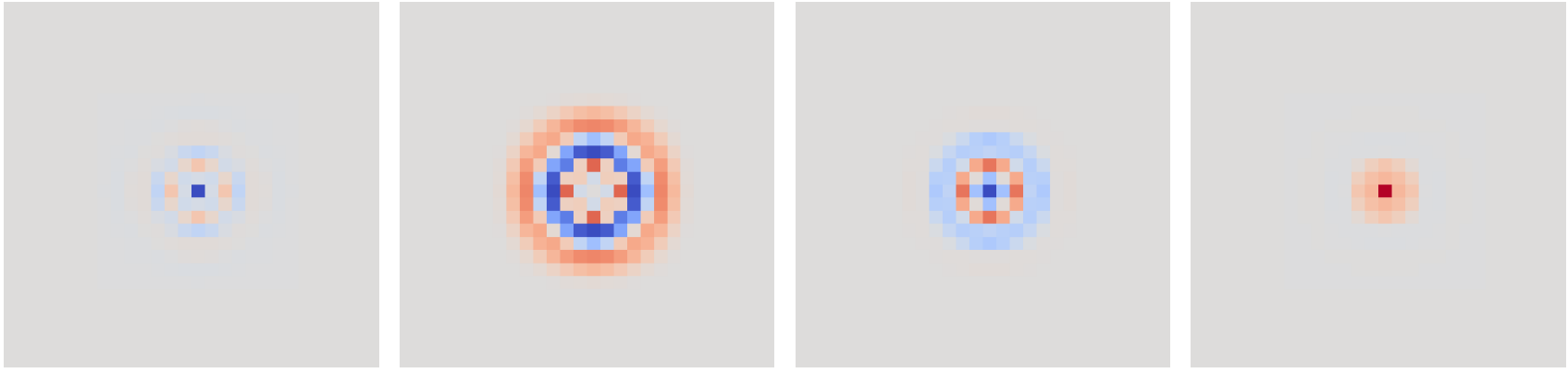
Limitations of spectral methods

- ☹ Poor generalization across non-isometric domains unless kernels are localized (can be remedied to some extent with spectral transformer networks[1])

- ☹ Spectral kernels are isotropic due to rotation invariance of the Laplacian

[1] Yi, Su et al. 2017

Only rotationally-symmetric kernels!



Example of Chebyshev filters (order $r = 7$) on Euclidean grid

Anisotropic kernels on manifolds

Scale t Orientation θ Elongation α

Examples of anisotropic heat kernels on a manifold

Boscaini et al. 2016

Limitations of spectral methods

- ☹ Poor generalization across non-isometric domains unless kernels are localized (can be remedied to some extent with spectral transformer networks[1])

- ☹ Spectral kernels are isotropic due to rotation invariance of the Laplacian (on manifolds, anisotropic Laplacians can be constructed[2])

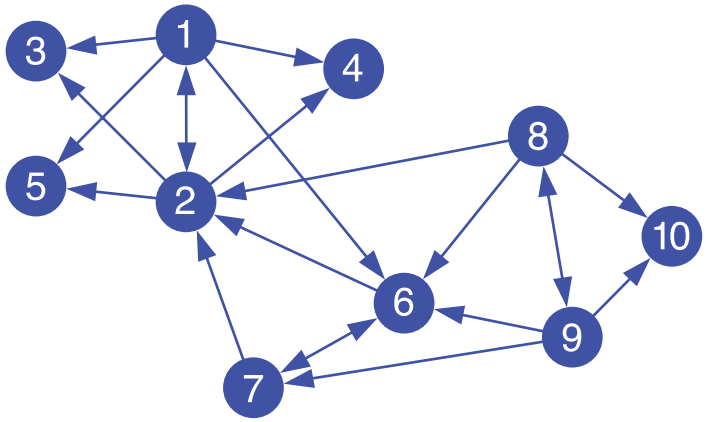
[1] Yi, Su, Guo, Guibas 2017; [2] Boscaini et al. 2016

Limitations of spectral methods

- ☹ Poor generalization across non-isometric domains unless kernels are localized (can be remedied to some extent with spectral transformer networks[1])
- ☹ Spectral kernels are isotropic due to rotation invariance of the Laplacian (on manifolds, anisotropic Laplacians can be constructed[2])
- ☹ Only undirected graphs, as symmetry of the Laplacian matrix is assumed

[1] Yi, Su, Guo, Guibas 2017; [2] Boscaini et al. 2016

Directed graphs



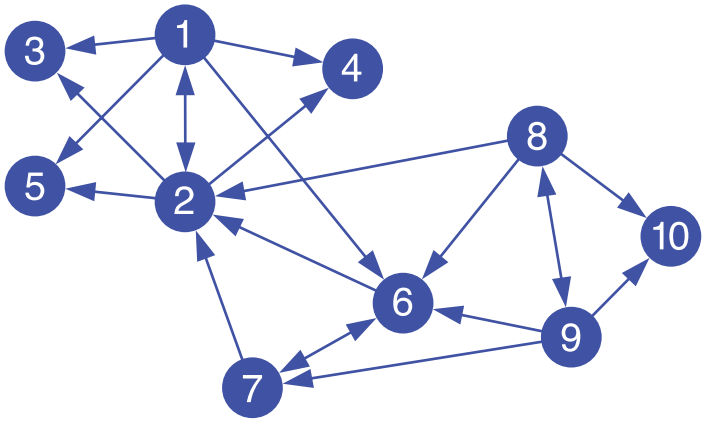
Directed graph

$$\begin{bmatrix}
. & 1 & 1 & 1 & 1 & 1 & . & . & . & . \\
1 & . & 1 & 1 & 1 & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . \\
. & 1 & . & . & . & . & 1 & . & . & . \\
. & 1 & . & . & . & 1 & . & . & . & . \\
. & 1 & . & . & . & 1 & . & . & 1 & 1 \\
. & . & . & . & . & 1 & 1 & 1 & . & 1 \\
. & . & . & . & . & . & . & . & . & .
\end{bmatrix}$$

Asymmetric adjacency matrix

Benson et al. 2016

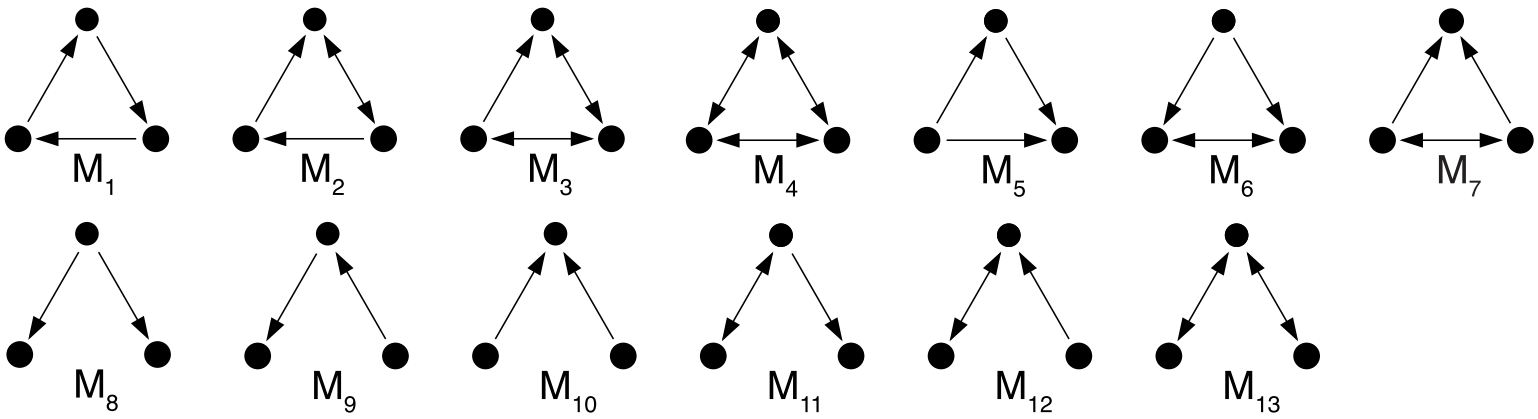
Motif-based graph analysis



Directed graph

$$\begin{bmatrix}
. & 1 & 1 & 1 & 1 & 1 & . & . & . & . \\
1 & . & 1 & 1 & 1 & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . \\
. & 1 & . & . & . & . & 1 & . & . & . \\
. & 1 & . & . & . & 1 & . & . & . & . \\
. & 1 & . & . & . & 1 & . & . & 1 & 1 \\
. & . & . & . & . & 1 & 1 & 1 & . & 1 \\
. & . & . & . & . & . & . & . & . & .
\end{bmatrix}$$

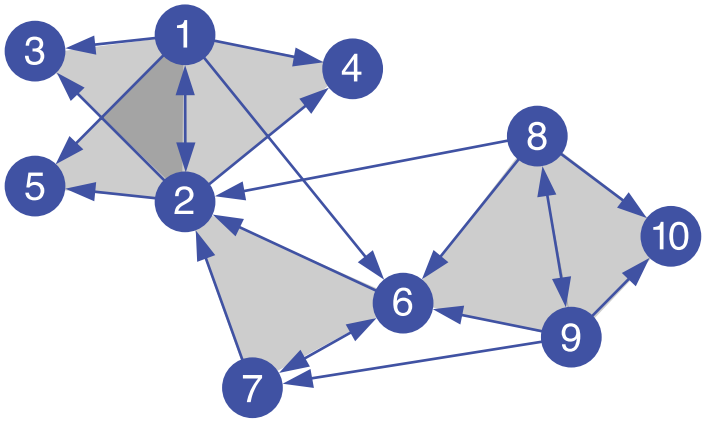
Asymmetric adjacency matrix



Graph motifs

Benson et al. 2016

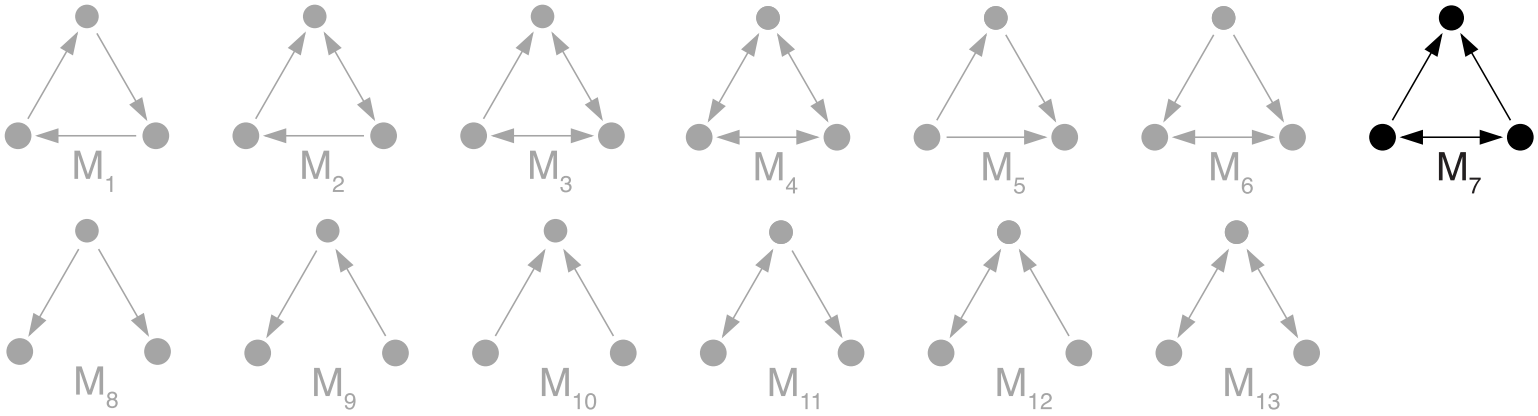
Motif-based graph analysis



Directed graph

$$\begin{bmatrix}
. & 3 & 1 & 1 & 1 & . & . & . & . & . \\
3 & . & 1 & 1 & 1 & 1 & 1 & . & . & . \\
1 & 1 & . & . & . & . & . & . & . & . \\
1 & 1 & . & . & . & . & . & . & . & . \\
1 & 1 & . & . & . & . & . & . & . & . \\
. & 1 & . & . & . & . & 1 & 1 & 1 & . \\
. & 1 & . & . & . & 1 & . & . & . & . \\
. & . & . & . & . & 1 & . & . & 2 & 1 \\
. & . & . & . & . & 1 & . & 2 & . & 1 \\
. & . & . & . & . & . & . & 1 & 1 & .
\end{bmatrix}$$

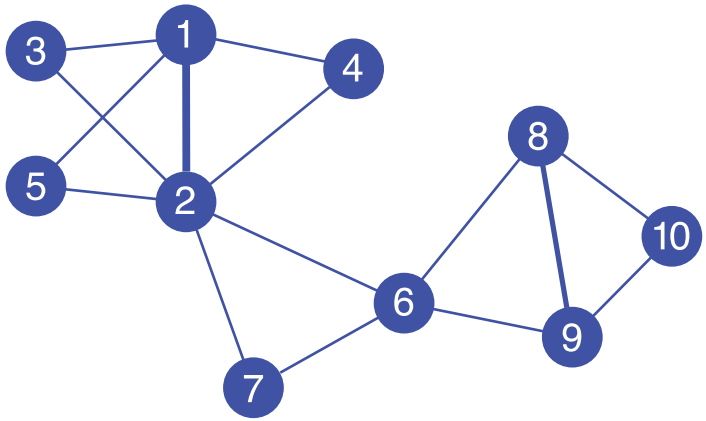
Motif adjacency matrix for M_7



Graph motifs

Benson et al. 2016

Motif Laplacians



Undirected weighted graph

$$\begin{bmatrix}
. & 3 & 1 & 1 & 1 & . & . & . & . & . \\
3 & . & 1 & 1 & 1 & 1 & 1 & . & . & . \\
1 & 1 & . & . & . & . & . & . & . & . \\
1 & 1 & . & . & . & . & . & . & . & . \\
1 & 1 & . & . & . & . & . & . & . & . \\
. & 1 & . & . & . & . & 1 & 1 & 1 & . \\
. & 1 & . & . & . & 1 & . & . & . & . \\
. & . & . & . & . & 1 & . & . & 2 & 1 \\
. & . & . & . & . & 1 & . & 2 & . & 1 \\
. & . & . & . & . & . & . & 1 & 1 & .
\end{bmatrix}$$

Motif adjacency matrix for M_7

Motif Laplacian for motif $k = 1, \ldots, K$

$$\tilde{\mathbf{\Delta}}_k = \mathbf{I} - \tilde{\mathbf{D}}_k^{-1/2} \tilde{\mathbf{W}}_k \tilde{\mathbf{D}}_k^{-1/2}$$

Benson et al. 2016

MotifNet: multivariate polynomial

Apply K-variate polynomial or order r to the motif Laplacians

$$\tau_{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\Delta}}_1, \ldots, \tilde{\boldsymbol{\Delta}}_K) = \alpha_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1,\ldots,k_j \in \{1,\ldots,K\}} \alpha_{k_1,\ldots,k_j} \tilde{\boldsymbol{\Delta}}_{k_1} \cdots \tilde{\boldsymbol{\Delta}}_{k_j}$$

where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_{K,\ldots,K})$ is the vector of filter parameters

Monti, Otness, Bronstein 2018

MotifNet: multivariate polynomial

Apply K-variate polynomial or order r to the motif Laplacians

$$\tau_{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\Delta}}_1, \ldots, \tilde{\boldsymbol{\Delta}}_K) = \alpha_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1,\ldots,k_j \in \{1,\ldots,K\}} \alpha_{k_1,\ldots,k_j} \tilde{\boldsymbol{\Delta}}_{k_1} \cdots \tilde{\boldsymbol{\Delta}}_{k_j}$$

where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_{K,\ldots,K})$ is the vector of filter parameters

☺ Explicitly accounts for directed graph structures

Monti, Otness, Bronstein 2018

MotifNet: multivariate polynomial

Apply K-variate polynomial or order r to the motif Laplacians

$$\tau_{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\Delta}}_1, \dots, \tilde{\boldsymbol{\Delta}}_K) = \alpha_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1,\dots,k_j \in \{1,\dots,K\}} \alpha_{k_1,\dots,k_j} \tilde{\boldsymbol{\Delta}}_{k_1} \cdots \tilde{\boldsymbol{\Delta}}_{k_j}$$

where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \dots, \alpha_{K,\dots,K})$ is the vector of filter parameters

☺ Explicitly accounts for directed graph structures

☺ Anisotropic kernels

Monti, Otness, Bronstein 2018

MotifNet: multivariate polynomial

Apply K-variate polynomial or order r to the motif Laplacians

$$\tau_{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\Delta}}_1, \dots, \tilde{\boldsymbol{\Delta}}_K) = \alpha_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1,\dots,k_j \in \{1,\dots,K\}} \alpha_{k_1,\dots,k_j} \tilde{\boldsymbol{\Delta}}_{k_1} \cdots \tilde{\boldsymbol{\Delta}}_{k_j}$$

where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \dots, \alpha_{K,\dots,K})$ is the vector of filter parameters

- ☺ Explicitly accounts for directed graph structures
- ☺ Anisotropic kernels
- ☺ $\mathcal{O}(1)$ parameters per layer

Monti, Otness, Bronstein 2018

MotifNet: multivariate polynomial

Apply K-variate polynomial or order r to the motif Laplacians

$$\tau_{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\Delta}}_1, \dots, \tilde{\boldsymbol{\Delta}}_K) = \alpha_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1,\dots,k_j \in \{1,\dots,K\}} \alpha_{k_1,\dots,k_j} \tilde{\boldsymbol{\Delta}}_{k_1} \cdots \tilde{\boldsymbol{\Delta}}_{k_j}$$

where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \dots, \alpha_{K,\dots,K})$ is the vector of filter parameters

☺ Explicitly accounts for directed graph structures

☺ Anisotropic kernels

☹ $\frac{1+K^{r+1}}{K-1}$ parameters per layer, intractable in practice

Monti, Otness, Bronstein 2018

MotifNet: multivariate polynomial

Apply K-variate polynomial or order r to the motif Laplacians

$$\tau_{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\Delta}}_1, \dots, \tilde{\boldsymbol{\Delta}}_K) = \alpha_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1,\dots,k_j \in \{1,\dots,K\}} \alpha_{k_1,\dots,k_j} \tilde{\boldsymbol{\Delta}}_{k_1} \cdots \tilde{\boldsymbol{\Delta}}_{k_j}$$

where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \dots, \alpha_{K,\dots,K})$ is the vector of filter parameters

- ☺ Explicitly accounts for directed graph structures
- ☺ Anisotropic kernels
- ☺ $Kr + 1$ parameters per layer using recurrent multivariate polynomials with dependent coefficients

Monti, Otness, Bronstein 2018

MotifNet: multivariate polynomial

Apply K-variate polynomial or order r to the motif Laplacians

$$\tau_{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\Delta}}_1, \dots, \tilde{\boldsymbol{\Delta}}_K) = \alpha_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1,\dots,k_j \in \{1,\dots,K\}} \alpha_{k_1,\dots,k_j} \tilde{\boldsymbol{\Delta}}_{k_1} \cdots \tilde{\boldsymbol{\Delta}}_{k_j}$$

where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \dots, \alpha_{K,\dots,K})$ is the vector of filter parameters

- ☺ Explicitly accounts for directed graph structures
- ☺ Anisotropic kernels
- ☺ $Kr + 1$ parameters per layer using recurrent multivariate polynomials with dependent coefficients
- ☺ Filters have guaranteed r-hops support

Monti, Otness, Bronstein 2018

MotifNet: multivariate polynomial

Apply K-variate polynomial or order r to the motif Laplacians

$$\tau_{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\Delta}}_1, \ldots, \tilde{\boldsymbol{\Delta}}_K) = \alpha_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1,\ldots,k_j \in \{1,\ldots,K\}} \alpha_{k_1,\ldots,k_j} \tilde{\boldsymbol{\Delta}}_{k_1} \cdots \tilde{\boldsymbol{\Delta}}_{k_j}$$

where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_{K,\ldots,K})$ is the vector of filter parameters

- ☺ Explicitly accounts for directed graph structures
- ☺ Anisotropic kernels
- ☺ $Kr + 1$ parameters per layer using recurrent multivariate polynomials with dependent coefficients
- ☺ Filters have guaranteed r-hops support
- ☺ $\mathcal{O}(n)$ computational complexity

Monti, Otness, Bronstein 2018

Example: directed citation networks

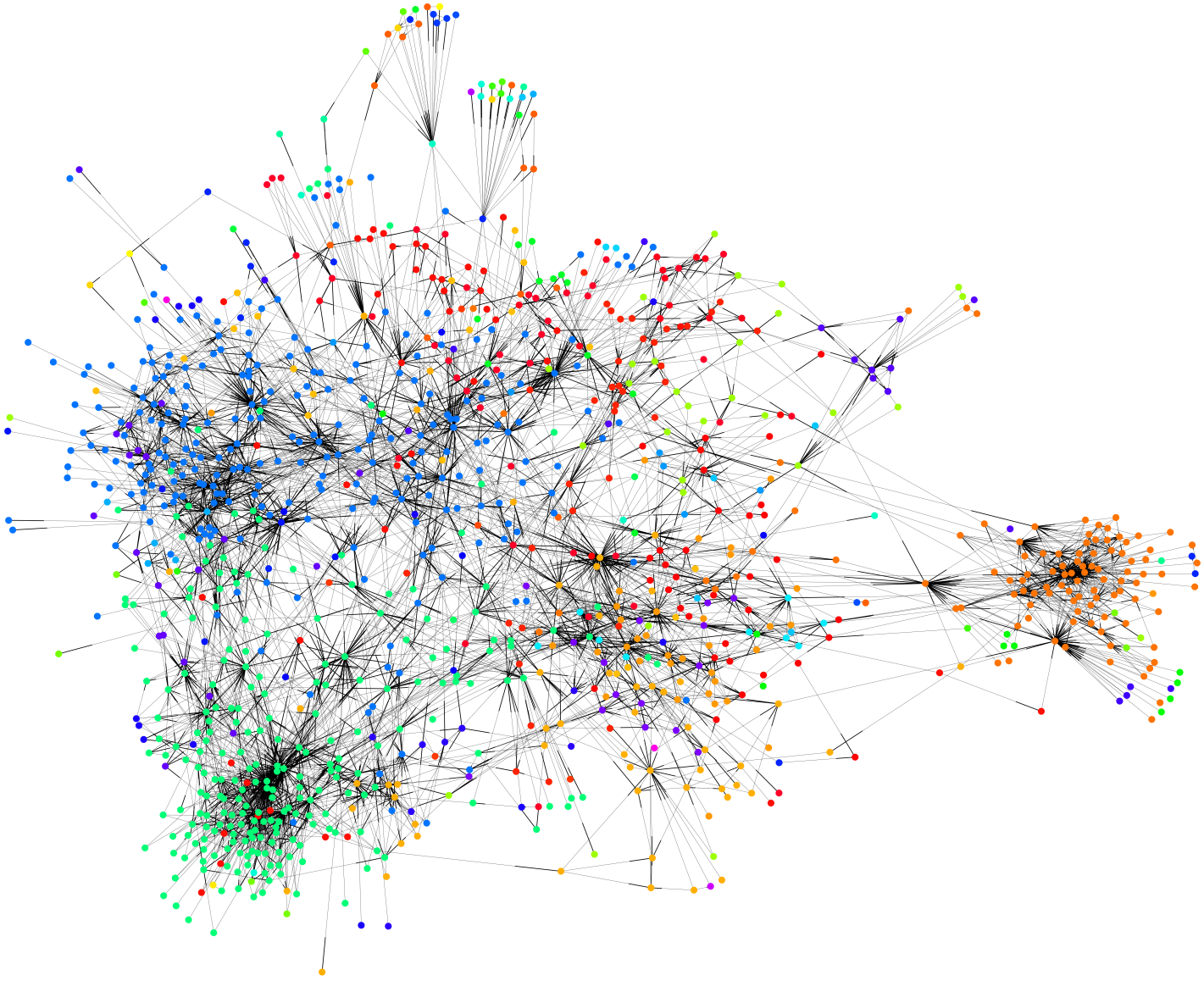
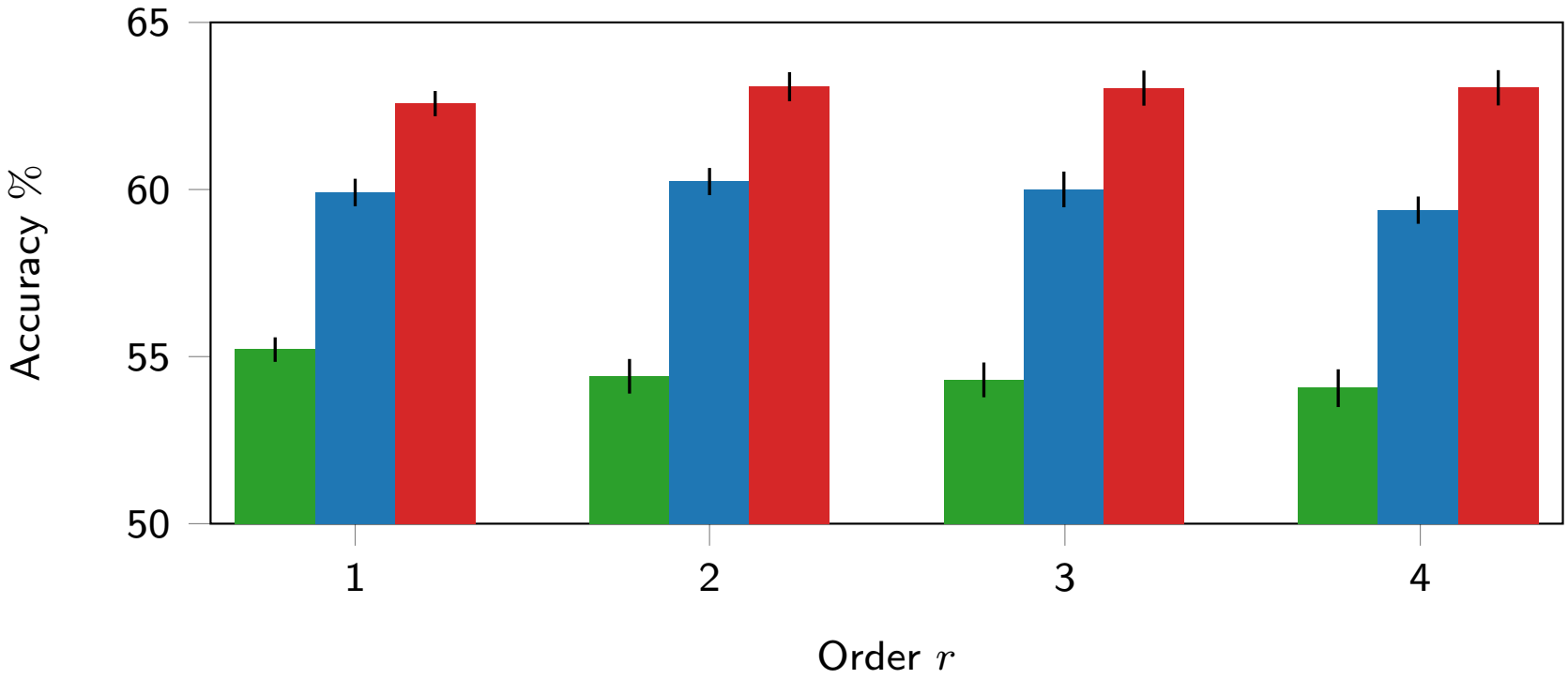


Figure: Monti, Otness, Bronstein 2018; data: Bojchevski, Günnemann 2017

Example: directed citation networks

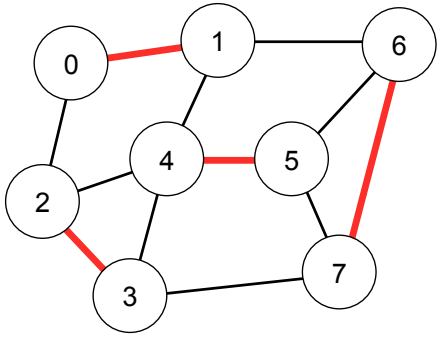


Classification accuracy on directed CORA obtained with ChebNet applied with directed adjacency matrix $\mathbf{W}$ (blue) and $\mathbf{W}^\top$ (green), and MotifNet-m (red)

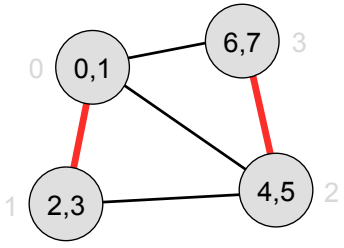
Monti, Otness, Bronstein 2018; data: Bojchevski, Günnemann, 2017

Graph pooling

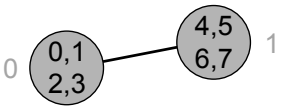
$\mathcal{G}^0 = \mathcal{G}$



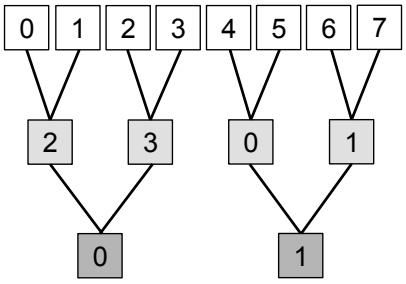
$\mathcal{G}^1$



$\mathcal{G}^2$



Coarsening structure

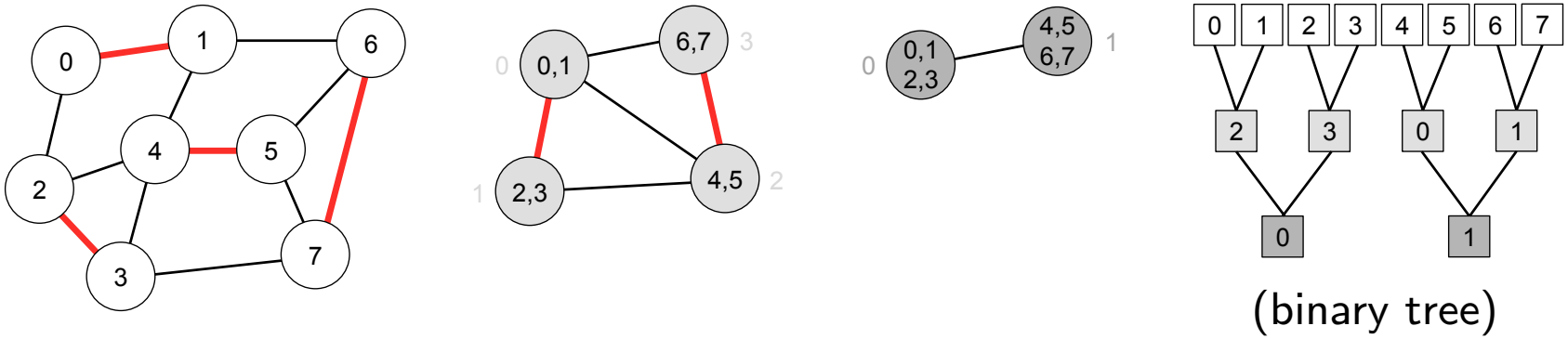


(binary tree)

- Produce a sequence of coarsened graphs

Graph pooling

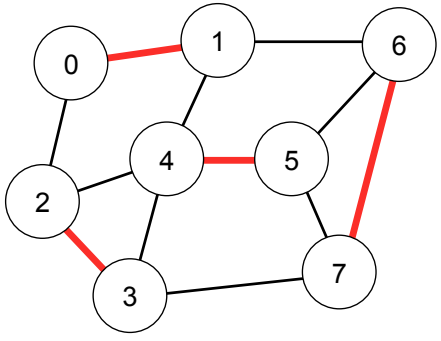
$\mathcal{G}^0 = \mathcal{G}$ $\mathcal{G}^1$ $\mathcal{G}^2$ Coarsening structure



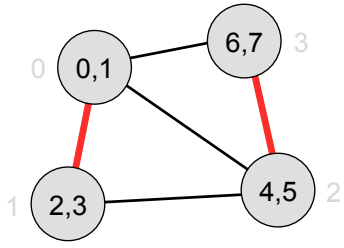
- Produce a sequence of coarsened graphs
- Max or average pooling of collapsed vertices

Graph pooling

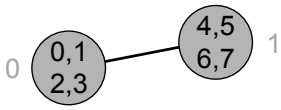
$\mathcal{G}^0 = \mathcal{G}$



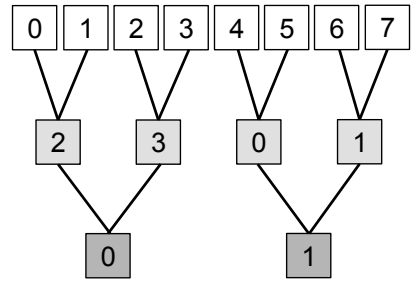
$\mathcal{G}^1$



$\mathcal{G}^2$



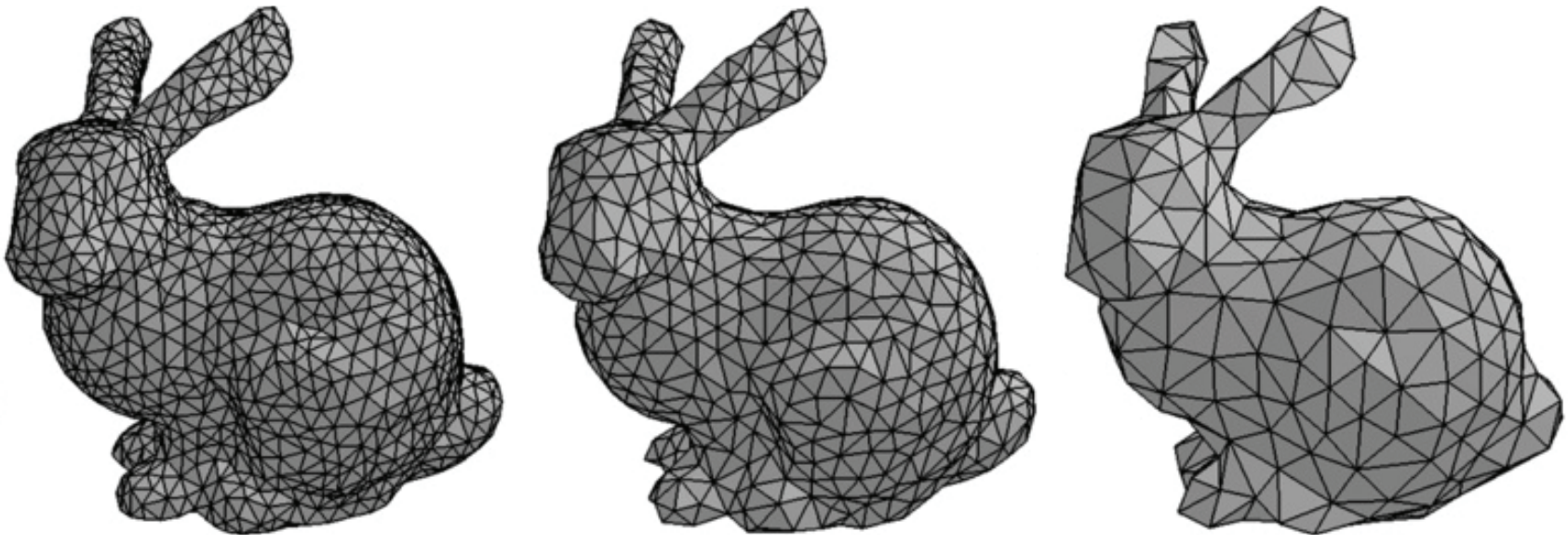
Coarsening structure



(binary tree)

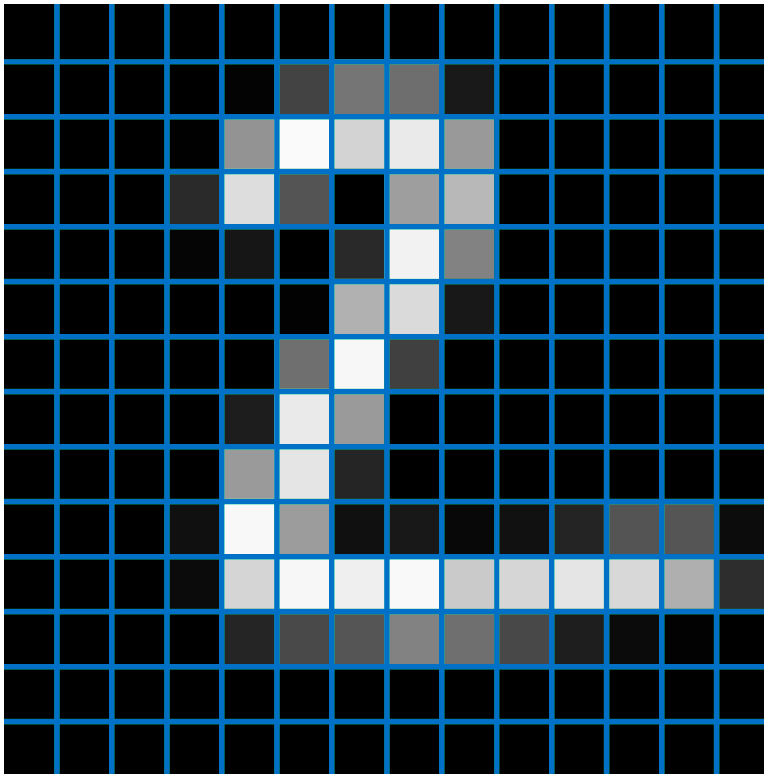
- Produce a sequence of coarsened graphs
- Max or average pooling of collapsed vertices
- Binary tree arrangement of node indices

Mesh pooling

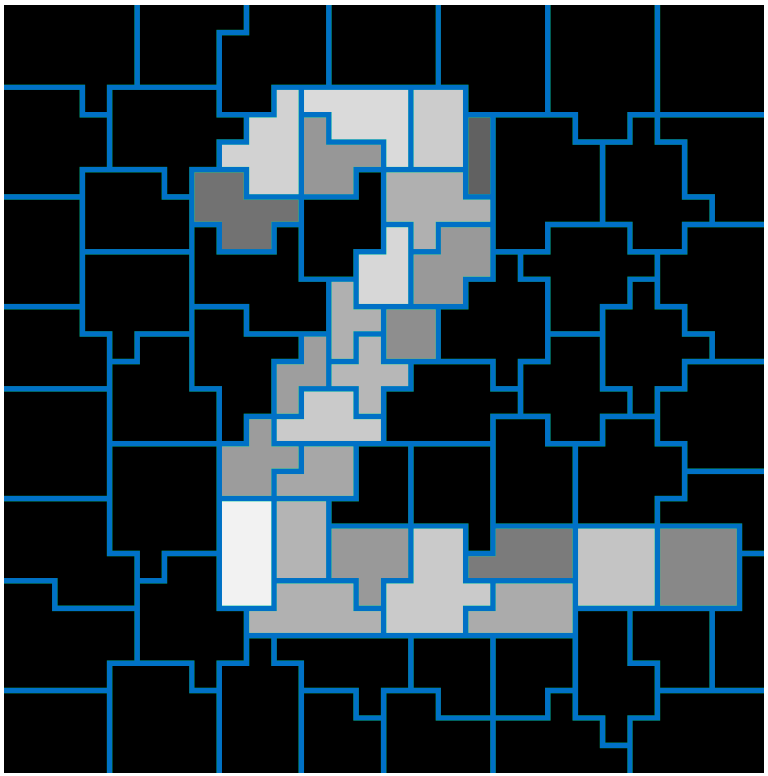


Example of progressive coarsening of a mesh

Example: MNIST digits classification



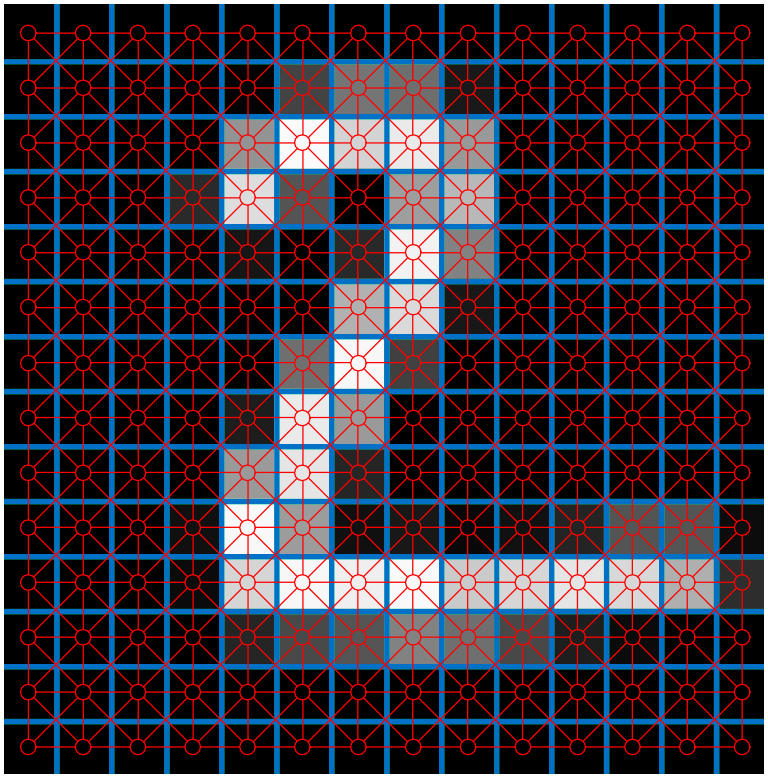
Regular grid



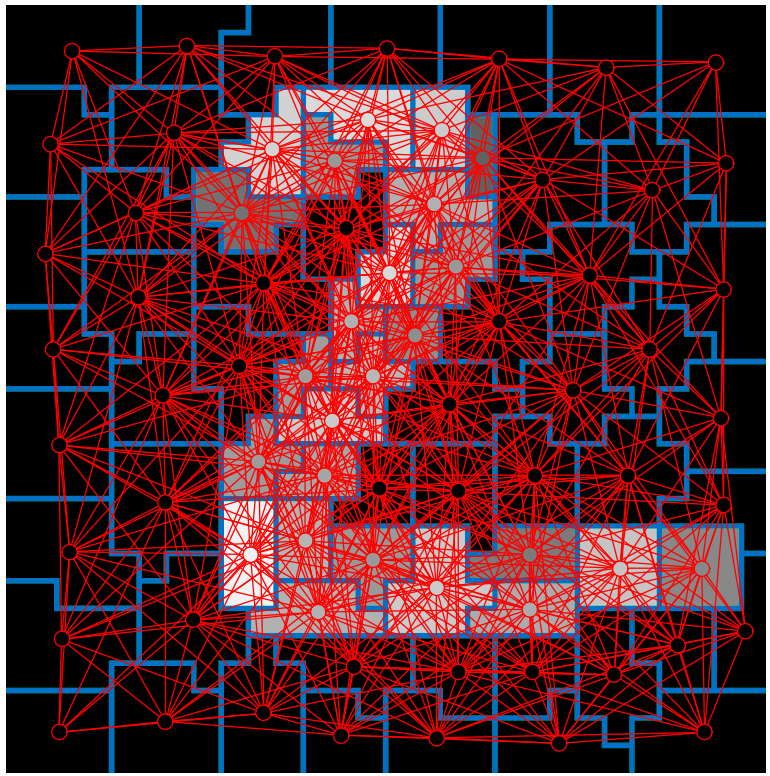
Superpixels

Monti et al. 2017

Example: MNIST digits classification



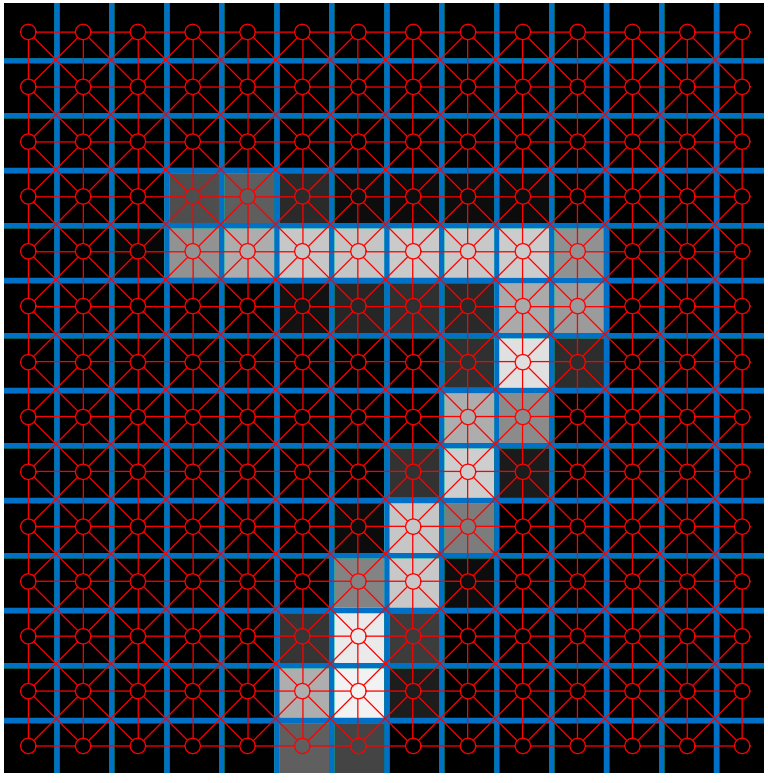
Regular grid



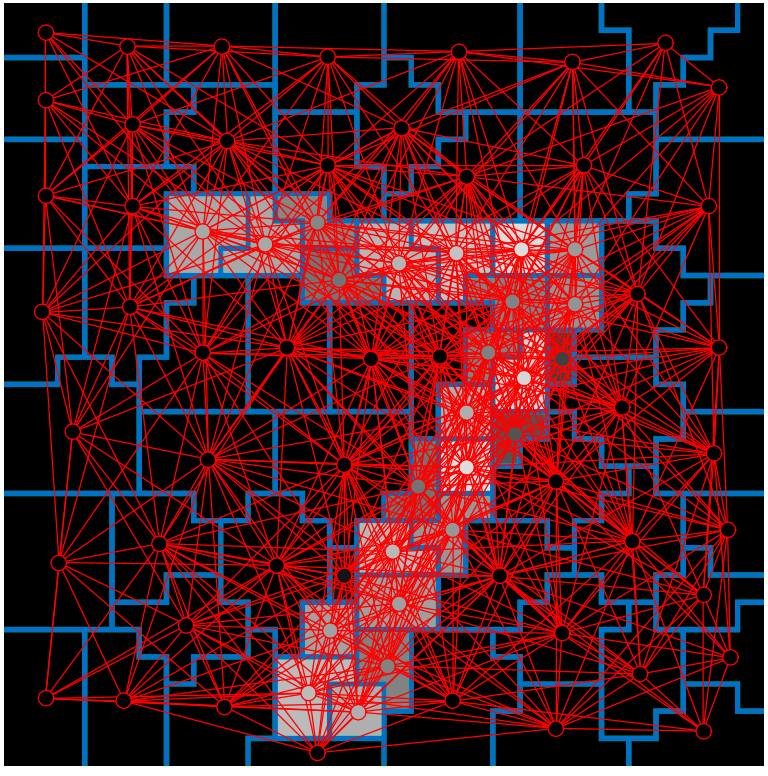
Superpixels

Monti et al. 2017

Example: MNIST digits classification



Regular grid
(fixed graph, different data)



Superpixels
(different graph and data)

Monti et al. 2017

Example: MNIST digits classification

Dataset	**LeNet-5**[1]	**ChebNet**[2]
*Full grid	99.33%	99.14%
$*\frac{1}{4}$ grid	98.59%	97.51%

Classification accuracy of different methods on MNIST dataset

*All images have the same graph

Monti et al. 2017; [1]LeCun et al. 1998; [2]Defferrard, Bresson, Vandergheynst 2016

Example: MNIST digits classification

Dataset	**LeNet-5**[1]	**ChebNet**[2]
*Full grid	99.33%	99.14%
$*\frac{1}{4}$ grid	98.59%	97.51%
300 Superpixels	-	**88.05%**
150 Superpixels	-	**80.94%**
75 Superpixels	-	**75.62%**

Classification accuracy of different methods on MNIST dataset

*All images have the same graph

Monti et al. 2017; [1]LeCun et al. 1998; [2]Defferrard, Bresson, Vandergheynst 2016

Different formulations of non-Euclidean CNNs

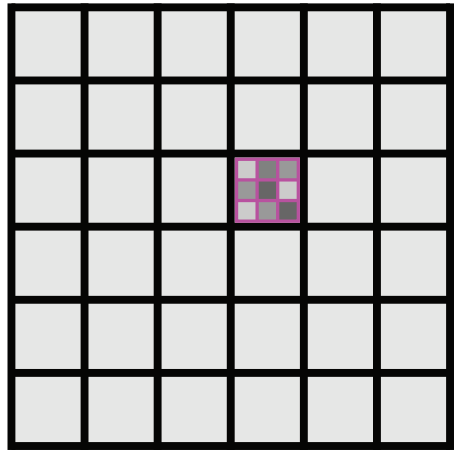


Spectral domain



Spatial domain





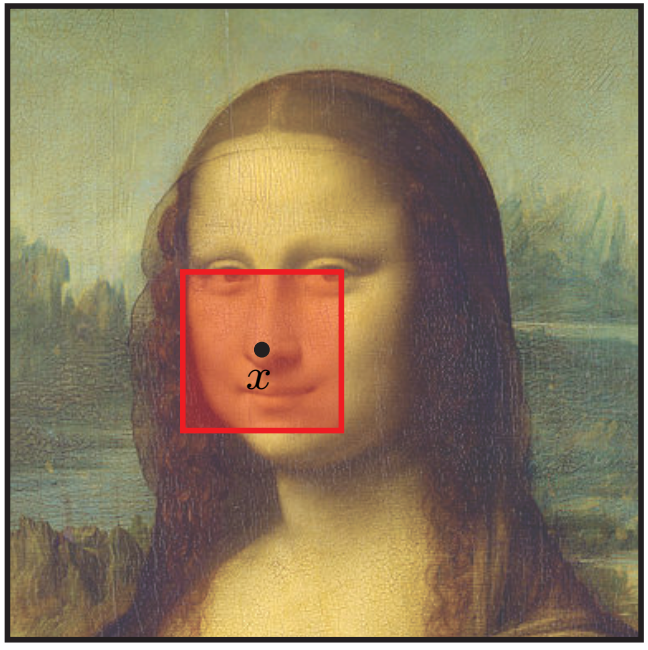
Parametric domain

Spatial domain (charting-based) geometric deep learning methods

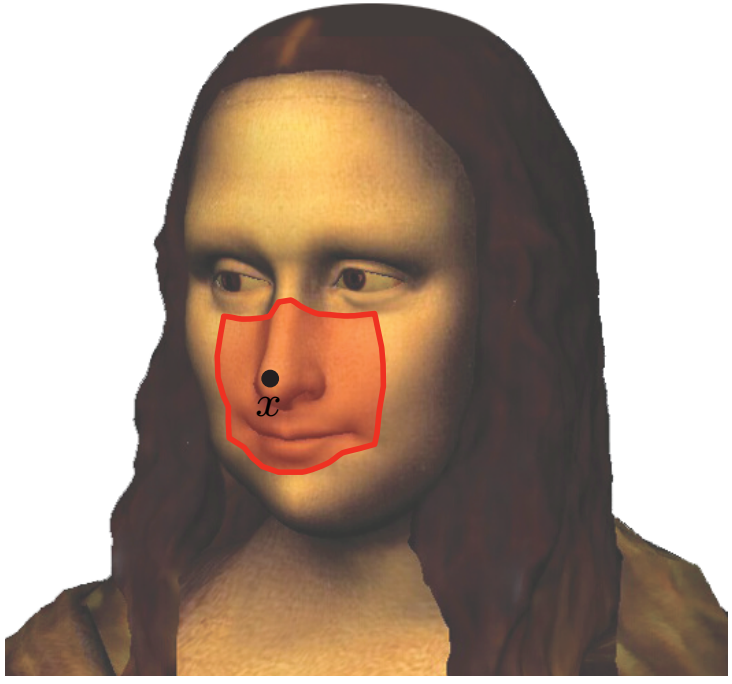
Convolution

Euclidean	**Non-Euclidean**
Spatial domain	
$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x-x')dx'$	?
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'	

Patch operators



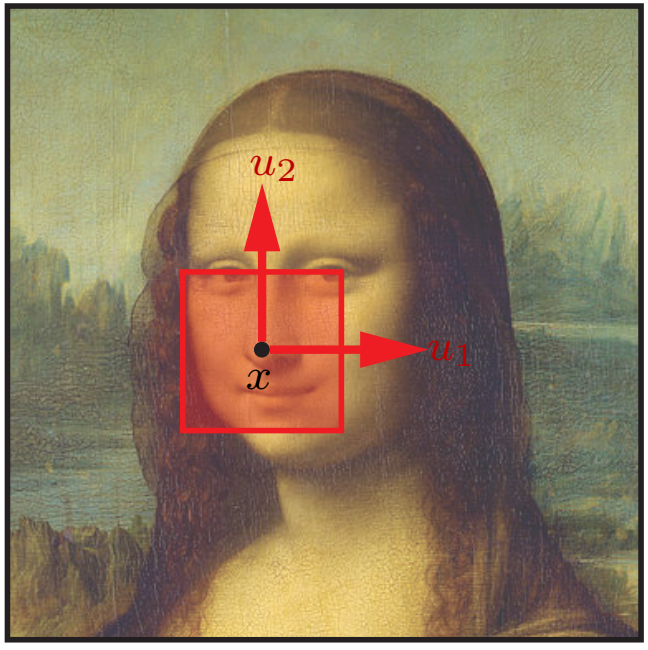
Image



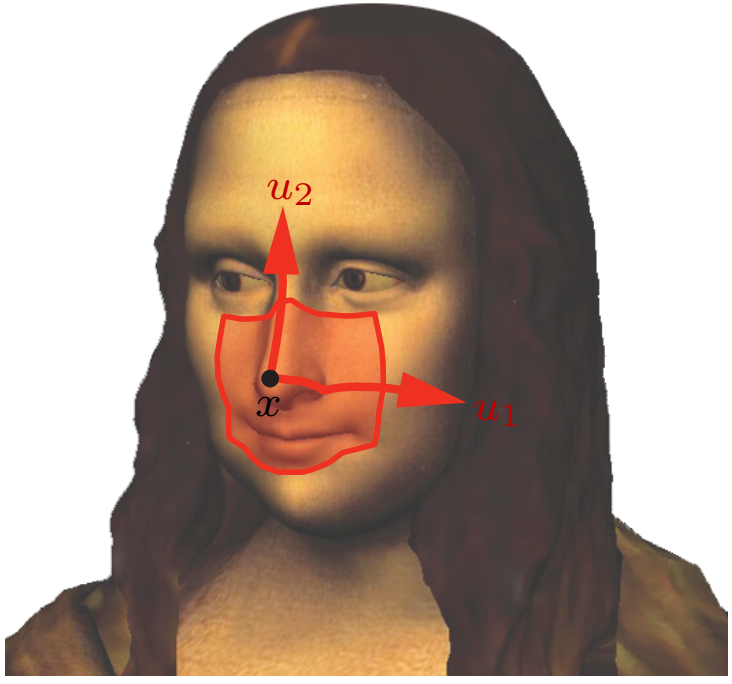
Manifold

Boscaini et al. 2015

Patch operators



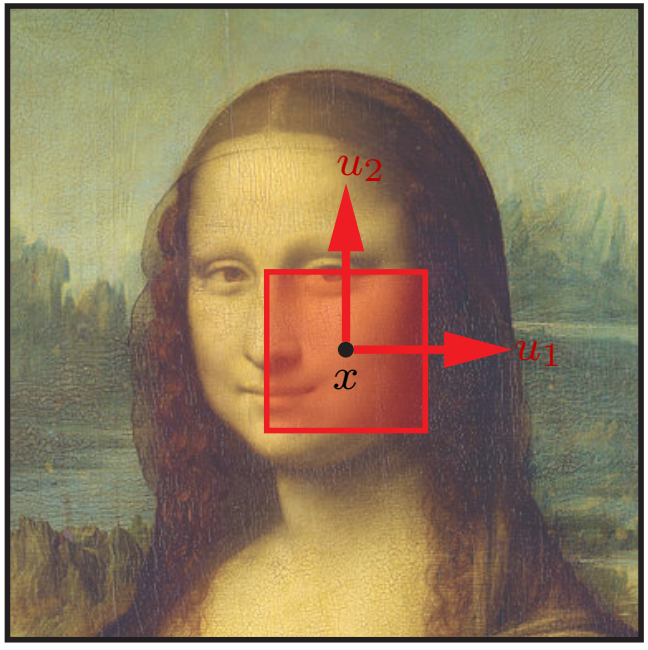
Image



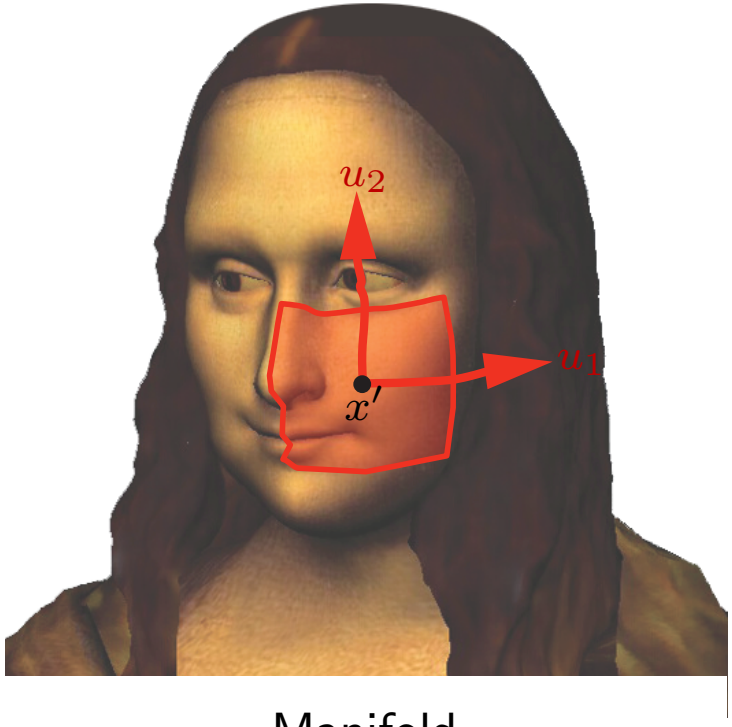
Manifold

Boscaini et al. 2015

Patch operators



Image

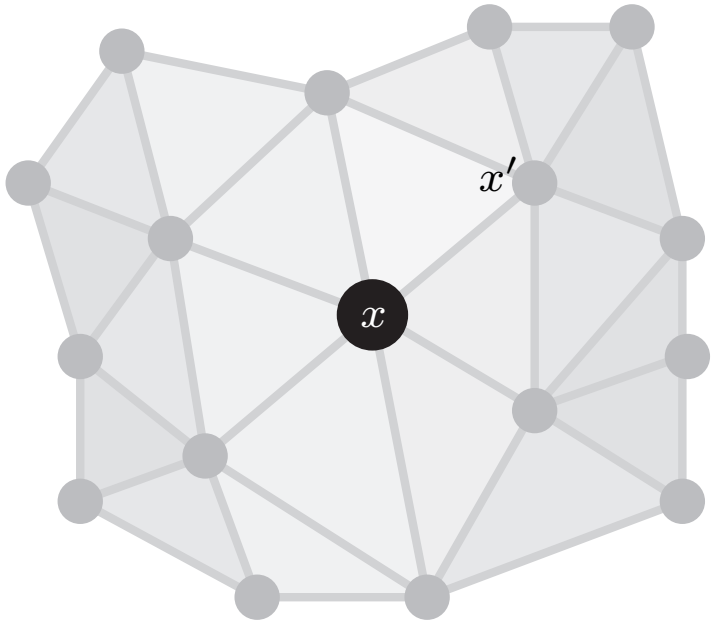


Manifold

Boscaini et al. 2015

Convolution on manifolds

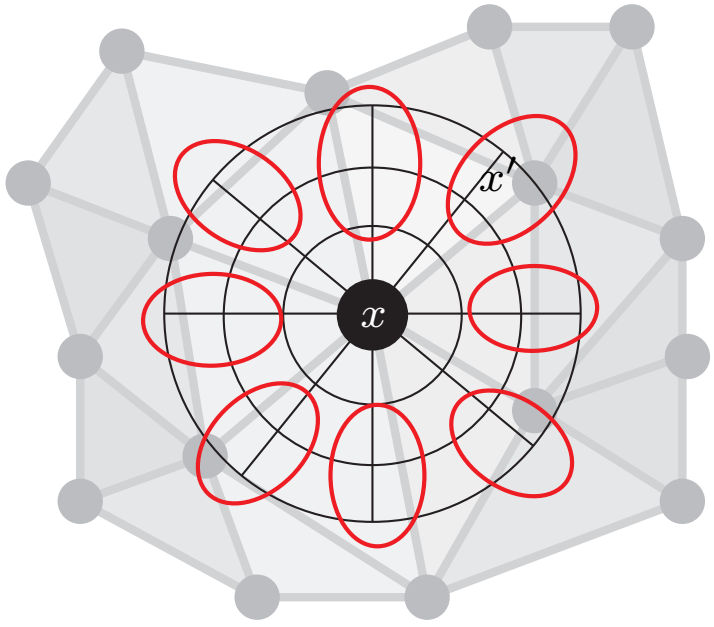
- Local system of coordinates $\mathbf{u}(x, x')$ around x' (e.g. geodesic polar)



Masci et al. 2015; Boscaini et al. 2016; Monti et al. 2017

Convolution on manifolds

- Local system of coordinates $\mathbf{u}(x, x')$ around x' (e.g. geodesic polar)
- Local weights $w_1(\mathbf{u}), \ldots, w_L(\mathbf{u})$ w.r.t. $\mathbf{u}$



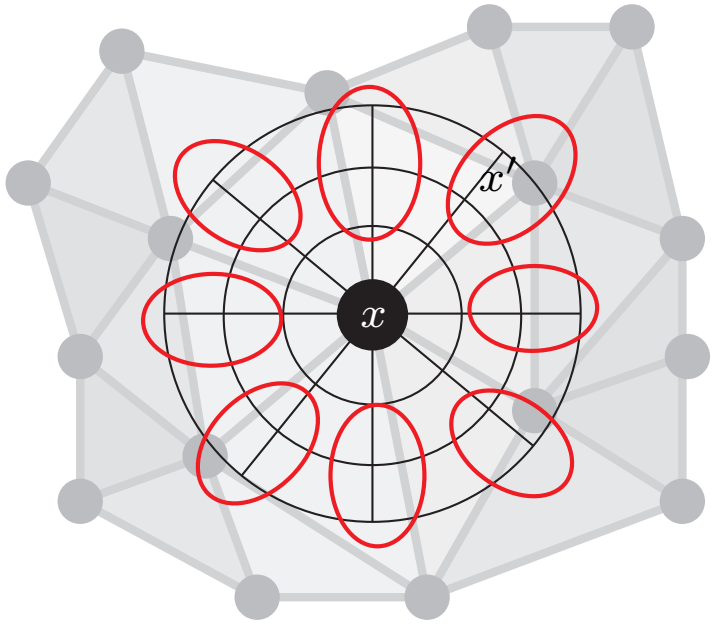
Masci et al. 2015; Boscaini et al. 2016; Monti et al. 2017

Convolution on manifolds

- Local system of coordinates $\mathbf{u}(x, x')$ around x' (e.g. geodesic polar)

- Local weights $w_1(\mathbf{u}), \dots, w_L(\mathbf{u})$ w.r.t. $\mathbf{u}$, e.g. Gaussians

$$w_\ell = \exp\left(-(\mathbf{u} - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}_\ell^{-1} (\mathbf{u} - \boldsymbol{\mu}_\ell)\right)$$



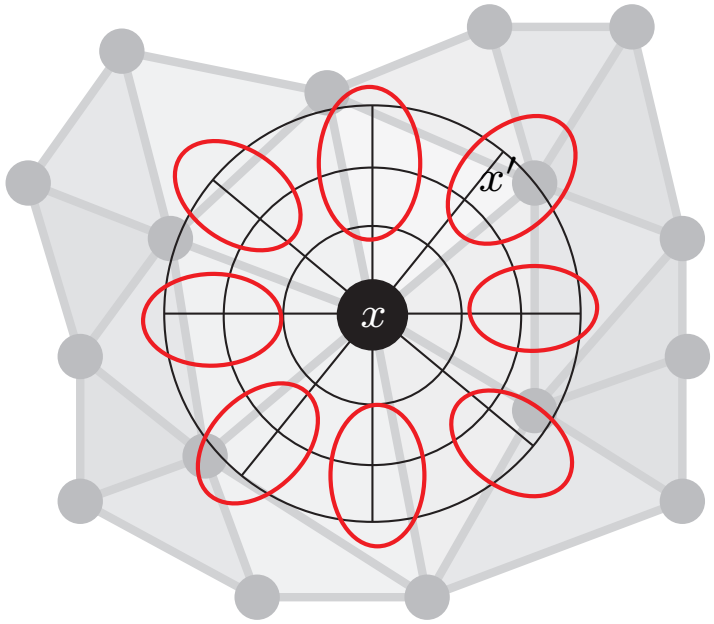
Masci et al. 2015; Boscaini et al. 2016; Monti et al. 2017

Convolution on manifolds

- Local system of coordinates $\mathbf{u}(x, x')$ around x' (e.g. geodesic polar)

- Local weights $w_1(\mathbf{u}), \dots, w_L(\mathbf{u})$ w.r.t. $\mathbf{u}$, e.g. Gaussians

$$w_\ell = \exp\left(-(\mathbf{u} - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}_\ell^{-1} (\mathbf{u} - \boldsymbol{\mu}_\ell)\right)$$



- Spatial convolution with filter g

$$(f \star g)(x) \propto \sum_{\ell=1}^{L} g_\ell \int_{\mathcal{X}} w_\ell(\mathbf{u}(x, x')) f(x')\, dx'$$

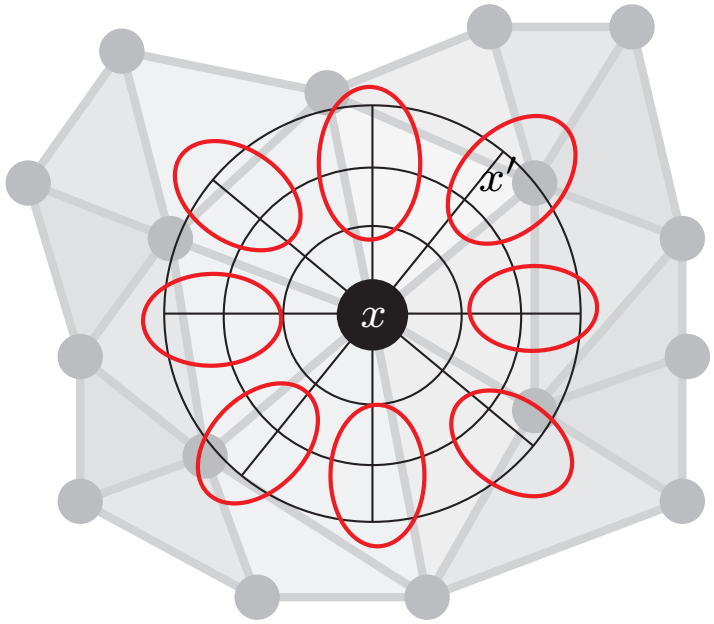
Masci et al. 2015; Boscaini et al. 2016; Monti et al. 2017

Convolution on manifolds

- Local system of coordinates $\mathbf{u}(x, x')$ around x' (e.g. geodesic polar)

- Local weights $w_1(\mathbf{u}), \dots, w_L(\mathbf{u})$ w.r.t. $\mathbf{u}$, e.g. Gaussians

  $$w_\ell = \exp\left(-(\mathbf{u} - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}_\ell^{-1} (\mathbf{u} - \boldsymbol{\mu}_\ell)\right)$$



- Spatial convolution with filter g

$$(f \star g)(x) \propto \sum_{\ell=1}^{L} g_\ell \underbrace{\int_{\mathcal{X}} w_\ell(\mathbf{u}(x, x')) f(x')\, dx'}_{\text{patch operator}}$$

Masci et al. 2015; Boscaini et al. 2016; Monti et al. 2017

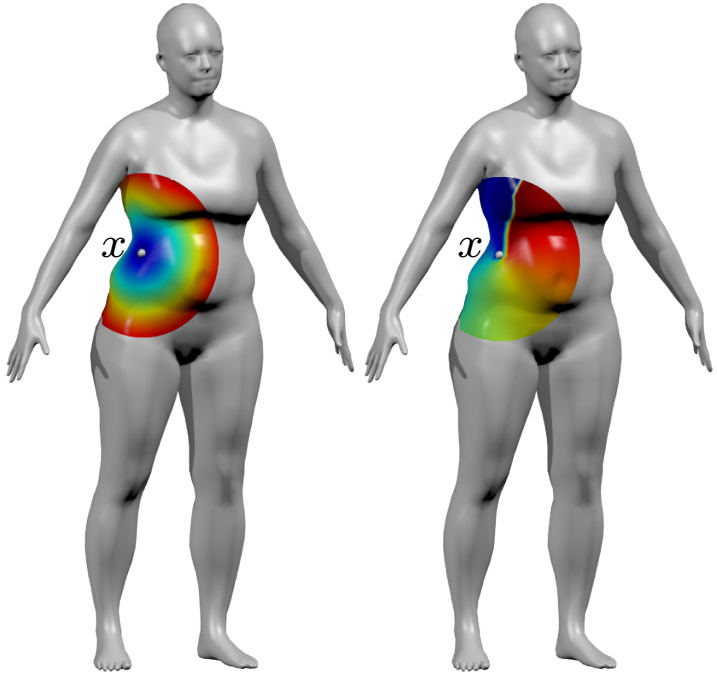
Geodesic polar coordinates

- Geodesic polar coordinates

$$\mathbf{u}(x, x') = (\rho(x, x'), \theta(x, x'))$$

$\rho(x, x')$ geodesic distance from x to x'

$\theta(x, x')$ direction of geodesic from x to x'



Masci et al. 2015; Monti et al. 2017

Geodesic polar coordinates

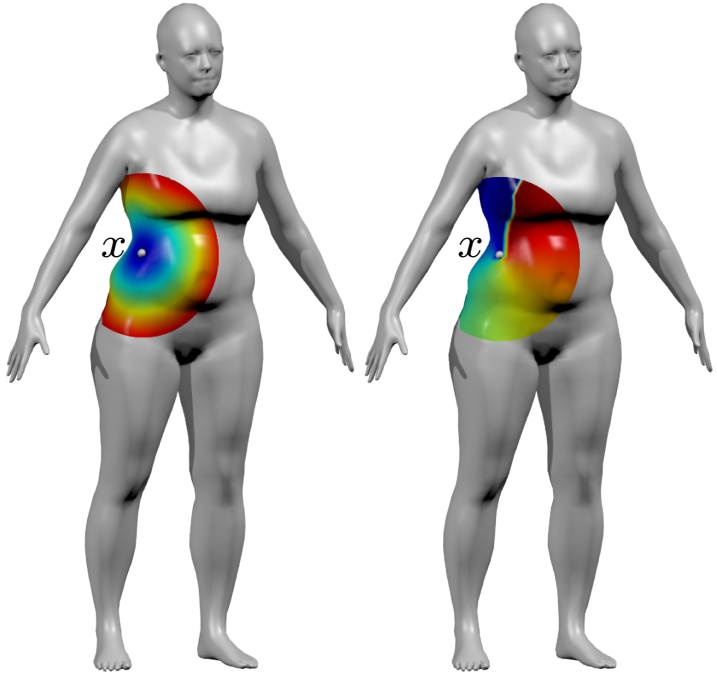
- Geodesic polar coordinates

$$\mathbf{u}(x, x') = (\rho(x, x'), \theta(x, x'))$$

$\rho(x, x')$ geodesic distance from x to x'

$\theta(x, x')$ direction of geodesic from x to x'

- Orientation ambiguity!



Masci et al. 2015; Monti et al. 2017

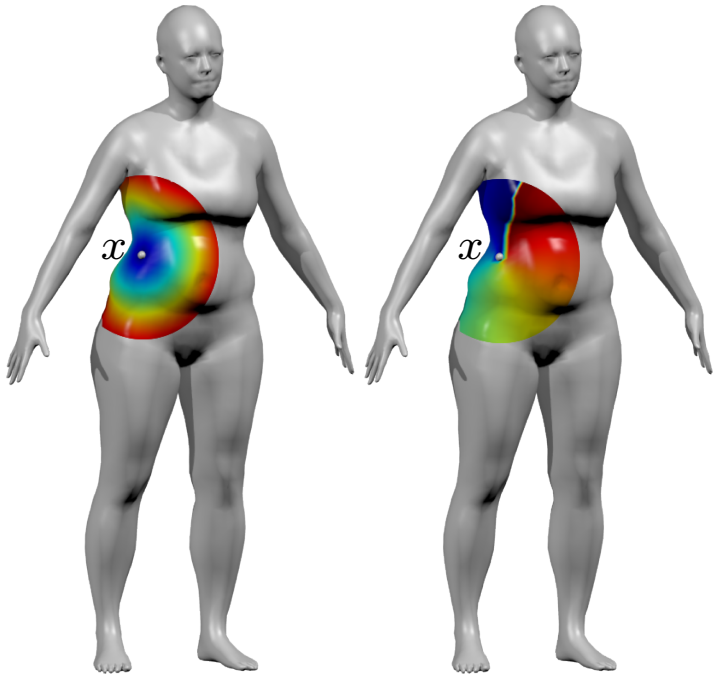
Geodesic polar coordinates

- Geodesic polar coordinates

$$\mathbf{u}(x, x') = (\rho(x, x'), \theta(x, x'))$$

$\rho(x, x')$ geodesic distance from x to x'

$\theta(x, x')$ direction of geodesic from x to x'



- Orientation ambiguity!
  - Canionical direction (e.g. intrinsic vector field, max curvature direction)

Masci et al. 2015; Monti et al. 2017

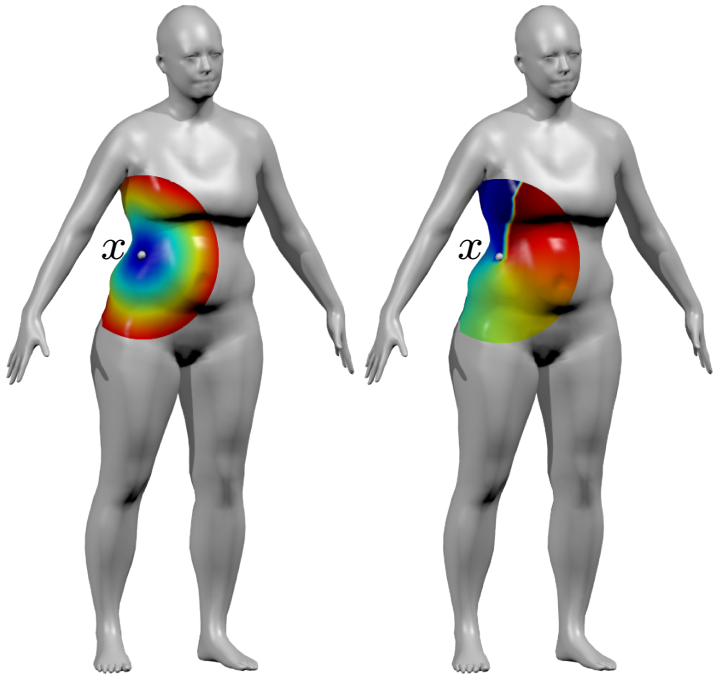
Geodesic polar coordinates

- Geodesic polar coordinates

$$\mathbf{u}(x, x') = (\rho(x, x'), \theta(x, x'))$$

$\rho(x, x')$ geodesic distance from x to x'

$\theta(x, x')$ direction of geodesic from x to x'



- Orientation ambiguity!
  - Canionical direction (e.g. intrinsic vector field, max curvature direction)
  - Angular max pooling: apply a rotating filter

$$(f \star g)(x) \propto \max_{\Delta\theta \in [0, 2\pi)} \sum_{\ell=1}^{L} g_\ell \int_{\mathcal{X}} w_\ell(\rho(x, x'), \theta(x, x') + \Delta\theta) f(x')\, dx'$$

Masci et al. 2015; Monti et al. 2017

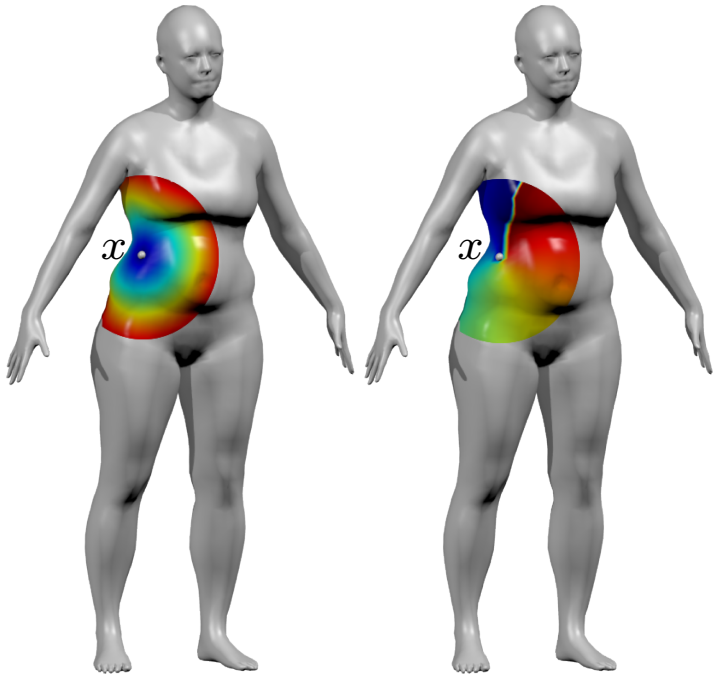
Geodesic polar coordinates

- Geodesic polar coordinates

$$\mathbf{u}(x, x') = (\rho(x, x'), \theta(x, x'))$$

$\rho(x, x')$ geodesic distance from x to x'

$\theta(x, x')$ direction of geodesic from x to x'



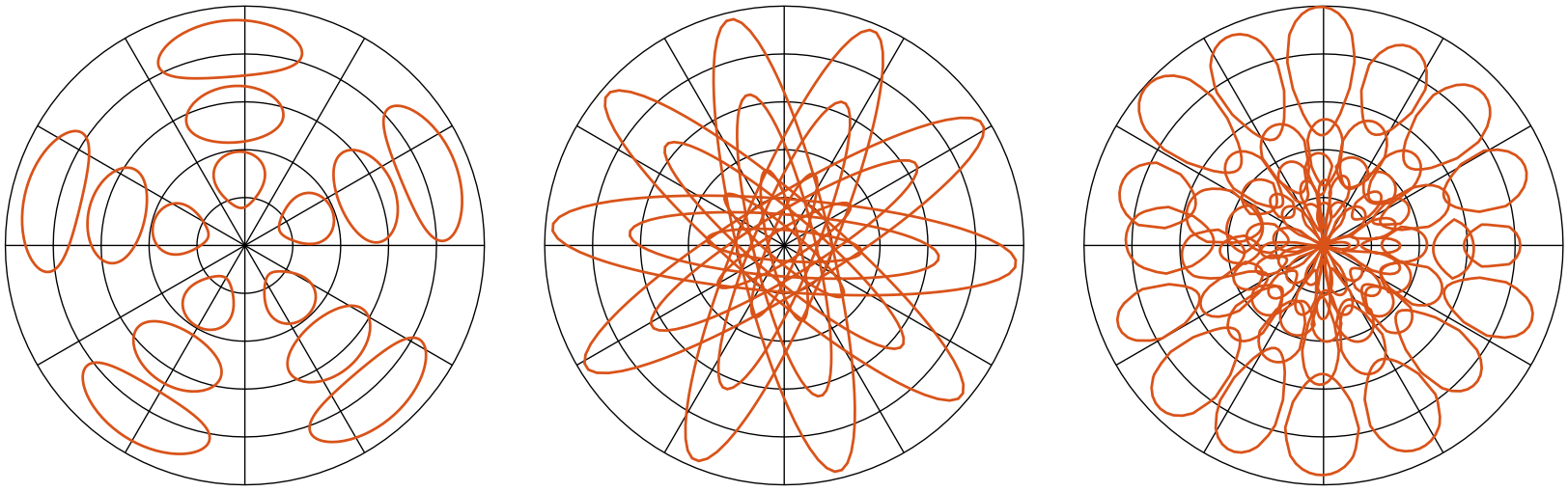
- Orientation ambiguity!
  - Canionical direction (e.g. intrinsic vector field, max curvature direction)
  - Angular max pooling: apply a rotating filter

$$(f \star g)(x) \propto \max_{\Delta\theta \in [0, 2\pi)} \sum_{\ell=1}^{L} g_\ell \int_{\mathcal{X}} w_\ell(\rho(x, x'), \theta(x, x') + \Delta\theta) f(x')\, dx'$$

  - Fourier transform magnitute w.r.t. θ

Masci et al. 2015; Monti et al. 2017

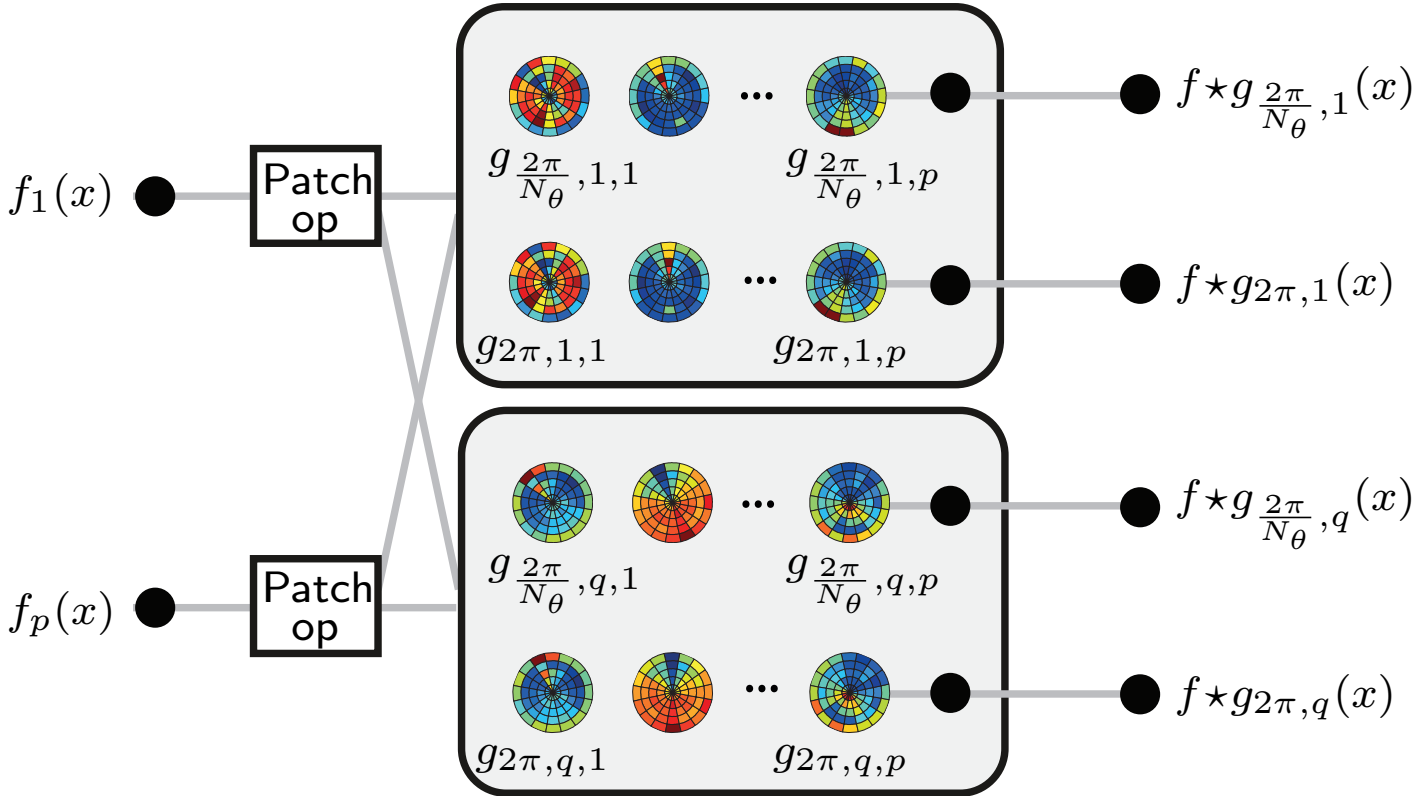
Patch operator weight functions



GCNN ACNN MoNet

Masci et al. 2015 (GCNN); Boscaini et al. 2016 (ACNN); Monti et al. 2017 (MoNet)

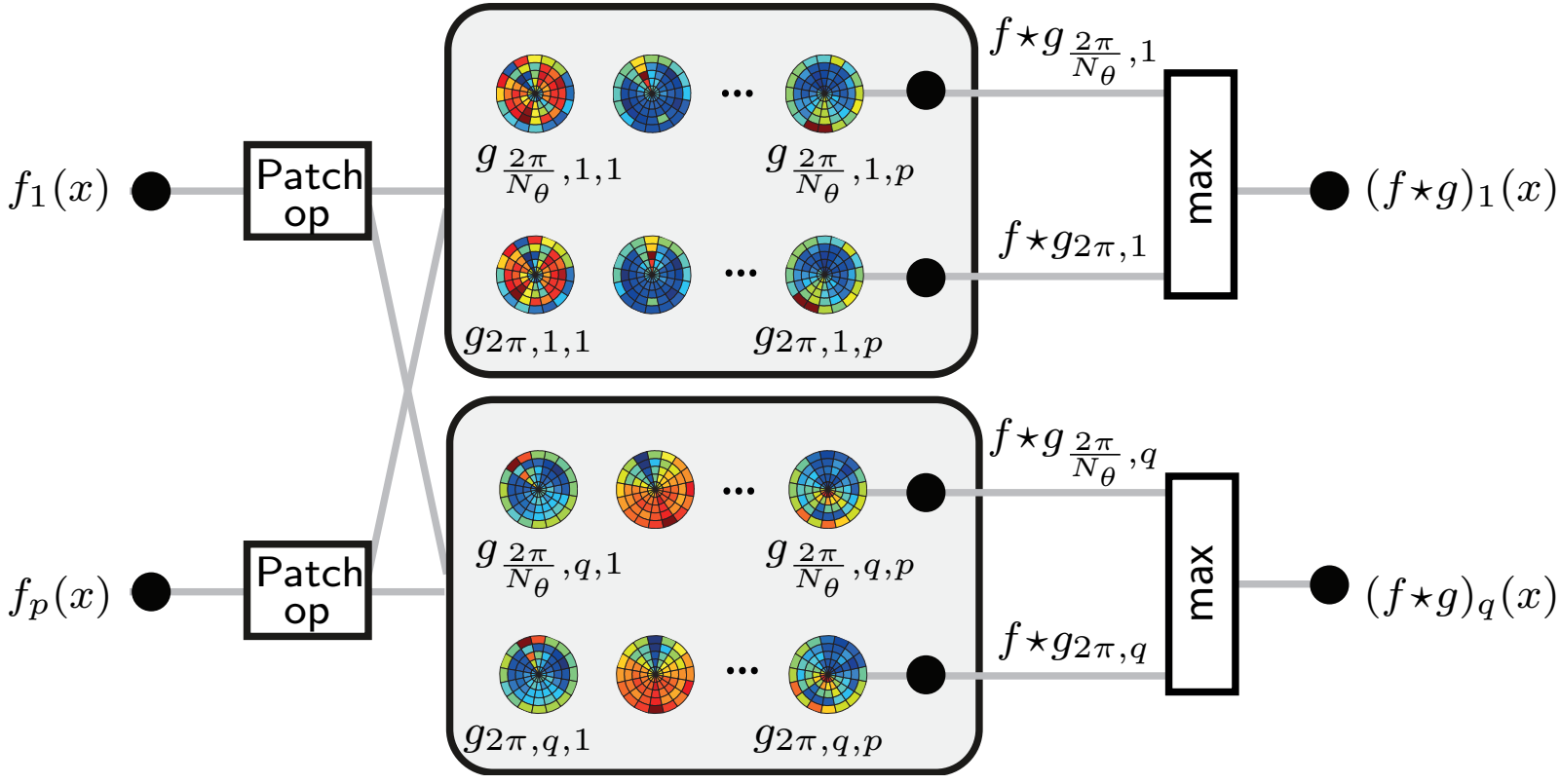
Geodesic convolution layer



Conv. layer $$(f_l \star g)_{\Delta\theta,l}(x) = \xi\left(\sum_{\ell=1}^{p}(f_\ell \star g_{\Delta\theta,l,\ell})(x)\right)$$

Masci et al. 2015

Geodesic convolution layer

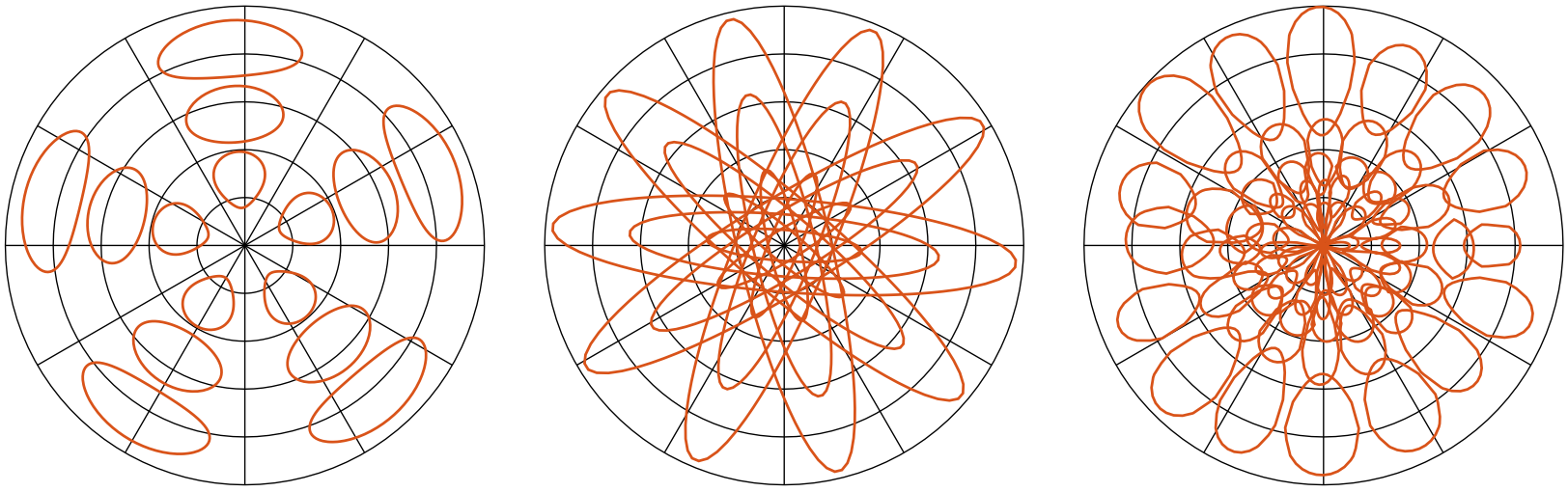


Conv. layer $$(f_l \star g)_{\Delta\theta,l}(x) = \xi\left(\sum_{\ell=1}^{p}(f_\ell \star g_{\Delta\theta,l,\ell})(x)\right)$$

Angular max pooling $$(f \star g)_l(x) = \max_{\Delta\theta}(f \star g)_{\Delta\theta,l}(x)$$

Masci et al. 2015

Patch operator weight functions



GCNN **ACNN** MoNet

Masci et al. 2015 (GCNN); Boscaini et al. 2016 (ACNN); Monti et al. 2017 (MoNet)

Homogeneous diffusion

$$f_t(x) = -\mathrm{div}(c\nabla f(x))$$

$c =$ thermal diffusivity constant describing heat conduction properties of the material (diffusion speed is equal everywhere)

Anisotropic diffusion

$$f_t(x) = -\mathrm{div}(\mathbf{A}(x)\nabla f(x))$$

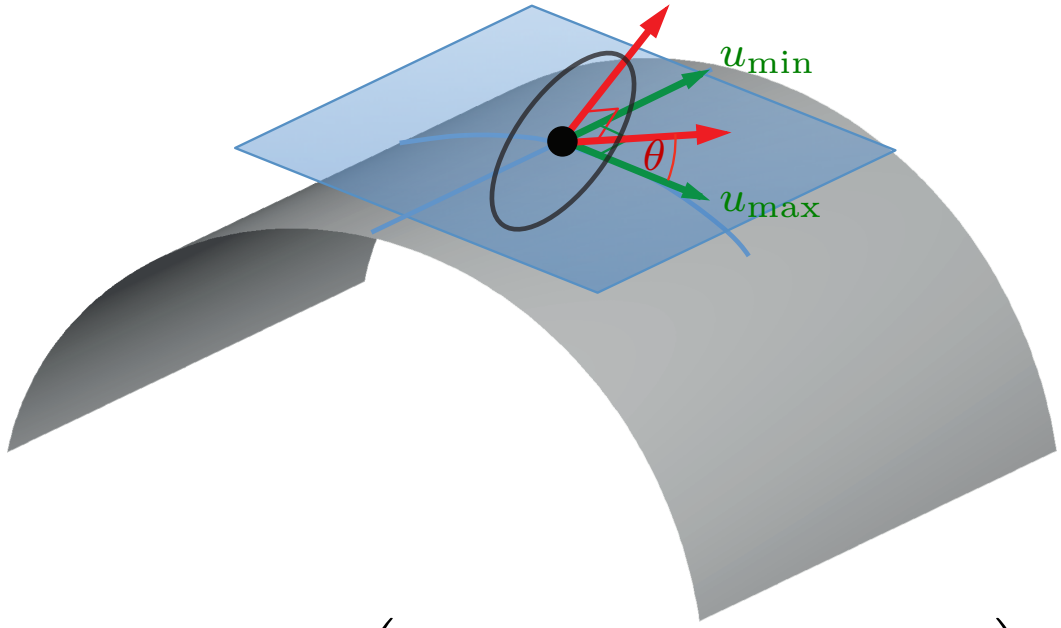
$\mathbf{A}(x) =$ heat conductivity tensor describing heat conduction properties of the material (diffusion speed is position + direction dependent)

Anisotropic diffusion

Isotropic

Anisotropic

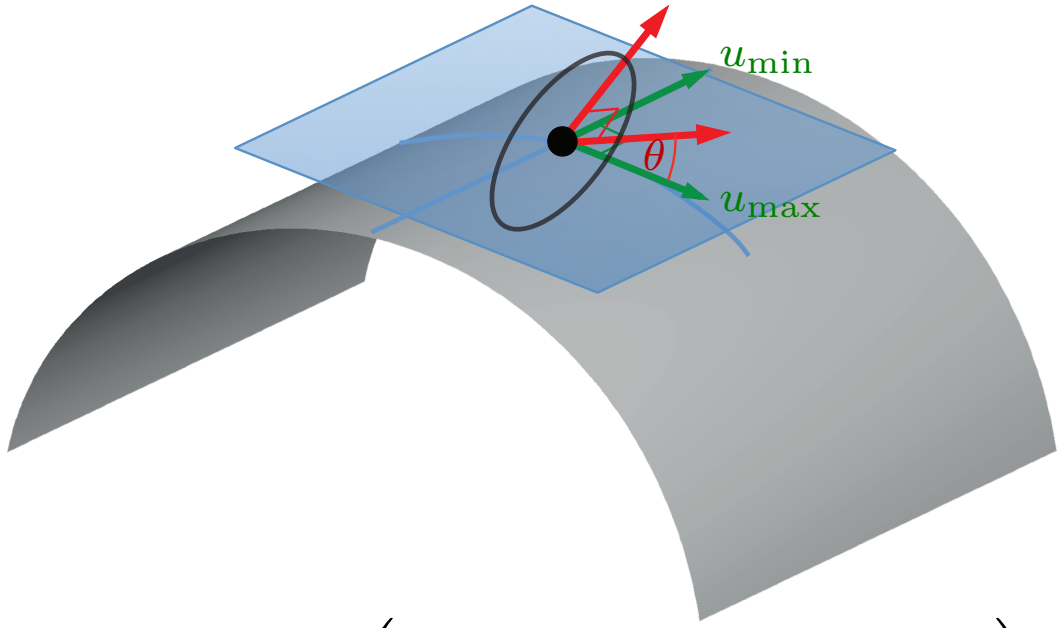
Anisotropic diffusion on manifolds



$$f_t(x) = -\mathrm{div}\left(\mathbf{R}_\theta \begin{pmatrix} \alpha & \\ & 1 \end{pmatrix} \mathbf{R}_\theta^\top \nabla f(x) \right)$$

Andreux et al. 2014; Boscaini et al. 2016

Anisotropic diffusion on manifolds



$$f_t(x) = -\mathrm{div}\Big(\underbrace{\mathbf{R}_\theta \begin{pmatrix} \alpha & \\ & 1 \end{pmatrix} \mathbf{R}_\theta^\top}_{\mathbf{D}_{\alpha\theta}(x)} \nabla f(x)\Big)$$

- Anisotropic Laplacian $\Delta_{\alpha\theta} f(x) = \mathrm{div}\left(D_{\alpha\theta}(x) \nabla f(x)\right)$
- $\theta =$ orientation w.r.t. max curvature direction
- $\alpha =$ 'elongation'

Andreux et al. 2014; Boscaini et al. 2016

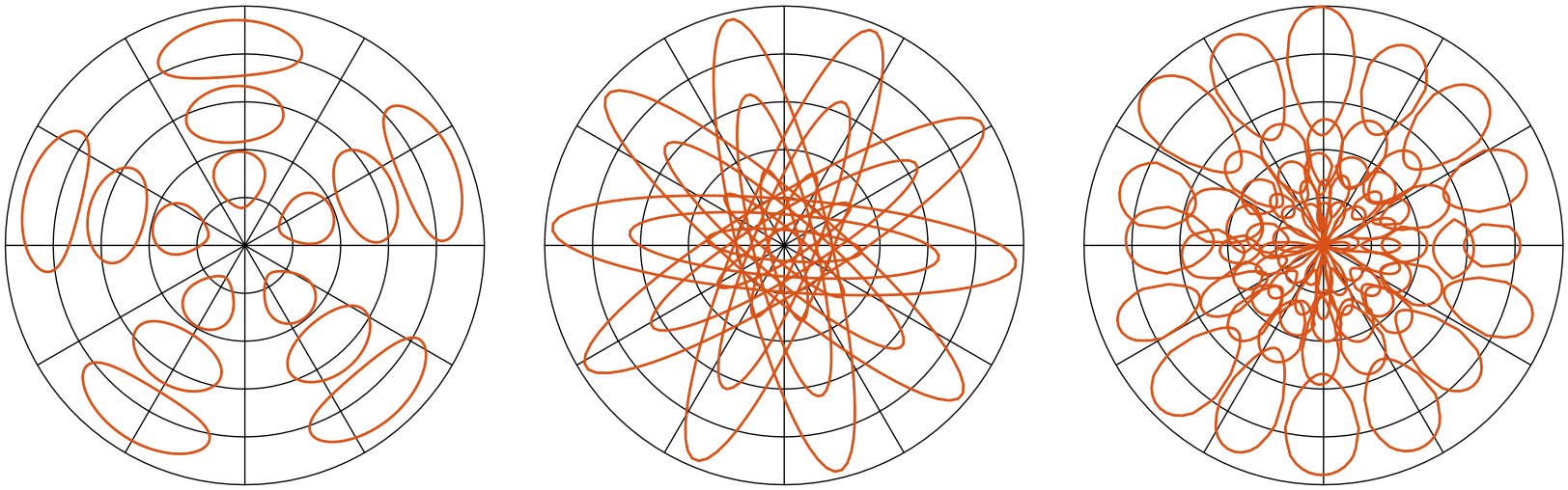
Anisotropic heat kernels

$$h_{\alpha\theta t}(x, x') = \sum_{k \geq 1} e^{-t\lambda_{\alpha\theta k}} \phi_{\alpha\theta k}(x)\phi_{\alpha\theta k}(x')$$

Scale t Orientation θ Elongation α

Boscaini et al. 2016

Patch operator weight functions



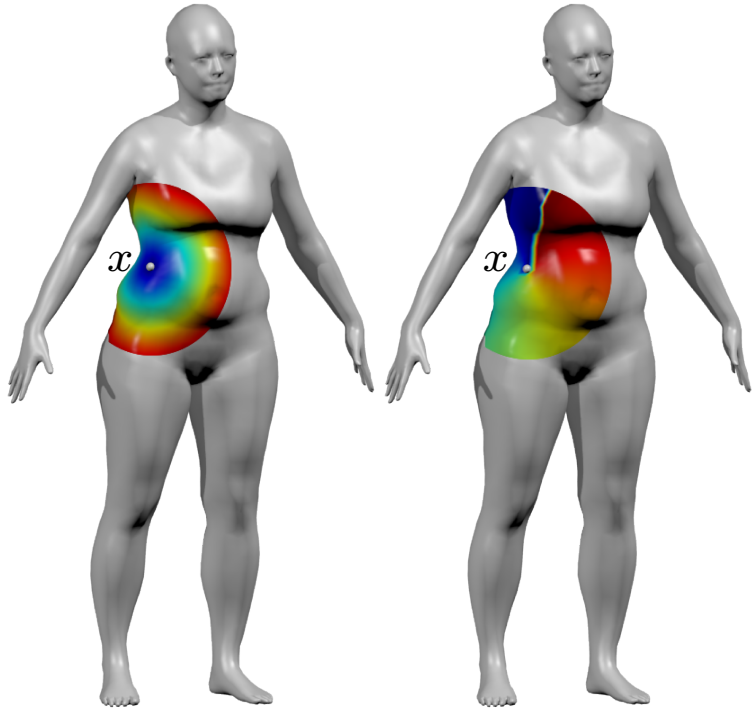
GCNN ACNN **MoNet**

Masci et al. 2015 (GCNN); Boscaini et al. 2016 (ACNN); Monti et al. 2016 (MoNet)

Learnable patch operator

- Geodesic polar coordinates

$$\mathbf{u}(x, y) = (\rho(x, x'), \theta(x, x'))$$



Monti et al. 2017

Learnable patch operator

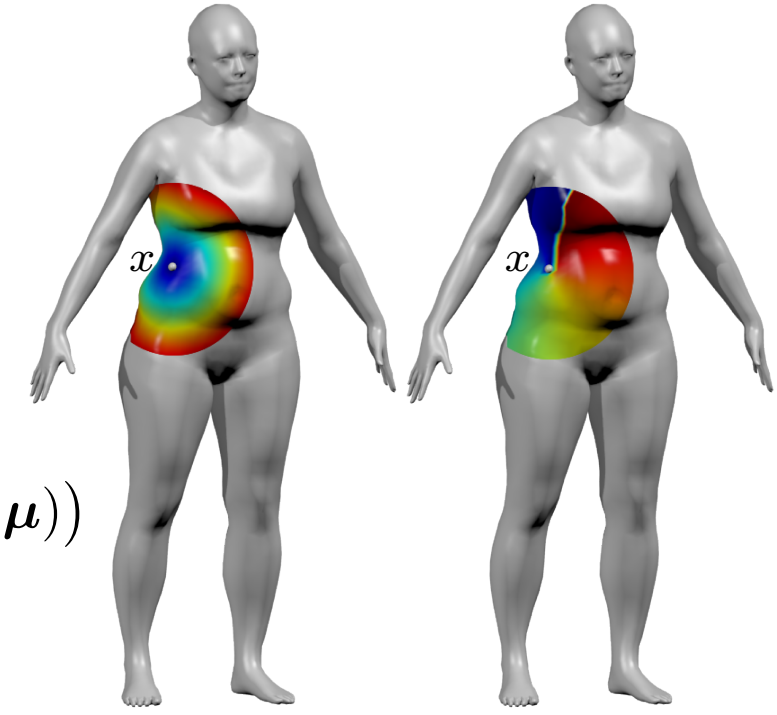
- Geodesic polar coordinates

$$\mathbf{u}(x, y) = (\rho(x, x'), \theta(x, x'))$$

- Gaussian weighting functions

$$w_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{u}) = \exp\left(-\tfrac{1}{2}(\mathbf{u} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{u} - \boldsymbol{\mu})\right)$$

with learnable covariance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{\mu}$



Monti et al. 2017

Learnable patch operator

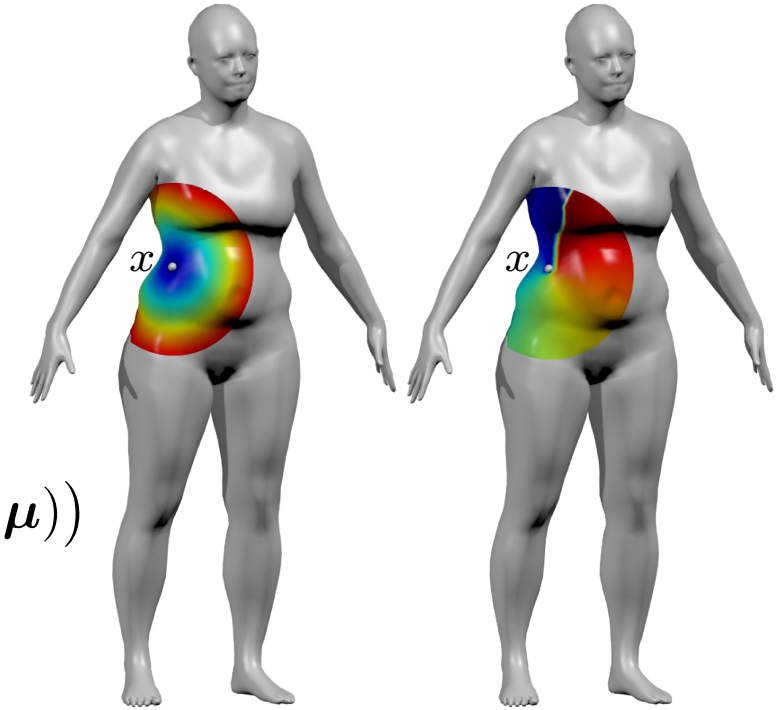
- Geodesic polar coordinates

$$\mathbf{u}(x, y) = (\rho(x, x'), \theta(x, x'))$$

- Gaussian weighting functions

$$w_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{u}) = \exp\left(-\tfrac{1}{2}(\mathbf{u} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{u} - \boldsymbol{\mu})\right)$$

with learnable covariance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{\mu}$



Spatial convolution

$$(f \star g)(x) \propto \sum_{\ell=1}^{L} g_\ell \int_{\mathcal{X}} w_{\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell}(\mathbf{u}(x, x'))\, f(x')\, dx'$$

where $g_1, \ldots, g_L$ are the spatial filter coefficients and $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_L$ and $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_L$ are patch operator parameters

Monti et al. 2017

Learnable patch operator

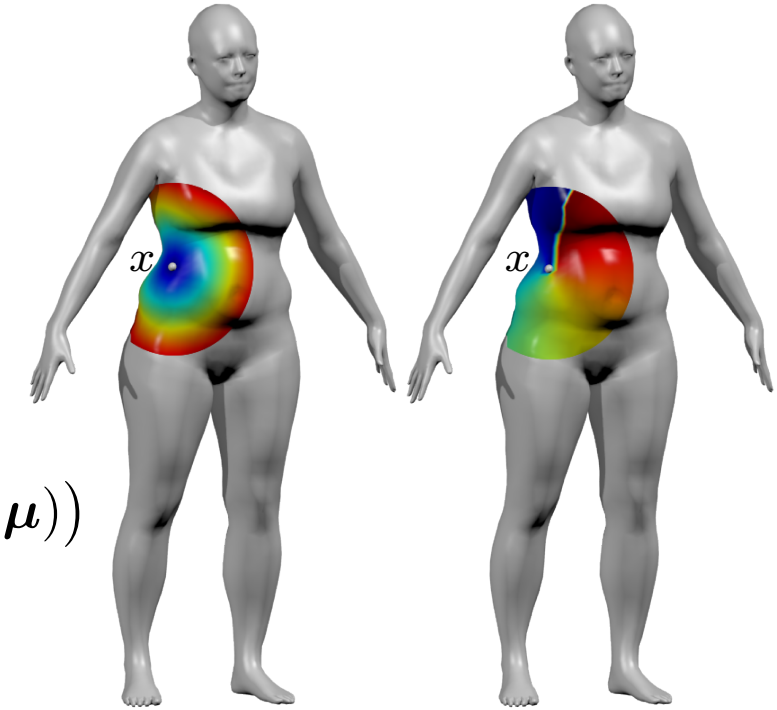
- Geodesic polar coordinates

$$\mathbf{u}(x, y) = (\rho(x, x'), \theta(x, x'))$$

- Gaussian weighting functions

$$w_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{u}) = \exp\left(-\tfrac{1}{2}(\mathbf{u} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{u} - \boldsymbol{\mu})\right)$$

with learnable covariance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{\mu}$



Spatial convolution

$$(f \star g)(x) \propto \int_{\mathcal{X}} \sum_{\ell=1}^{L} g_\ell w_{\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell}(\mathbf{u}(x, x'))\, f(x')\, dx'$$

where $g_1, \ldots, g_L$ are the spatial filter coefficients and $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_L$ and $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_L$ are patch operator parameters

Monti et al. 2017

Mixture Model Network (MoNet)

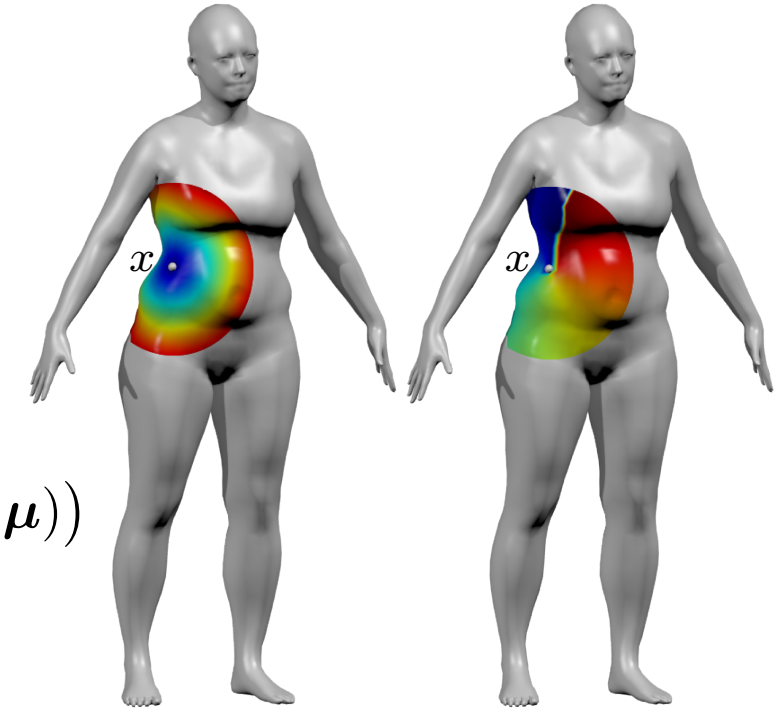
- Geodesic polar coordinates

$$\mathbf{u}(x,y) = (\rho(x,x'), \theta(x,x'))$$

- Gaussian weighting functions

$$w_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{u}) = \exp\big(-\tfrac{1}{2}(\mathbf{u}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{u}-\boldsymbol{\mu})\big)$$

with learnable covariance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{\mu}$



Spatial convolution

$$(f \star g)(x) \propto \int_{\mathcal{X}} \underbrace{\sum_{\ell=1}^{L} g_\ell w_{\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell}(\mathbf{u}(x,x'))}_{\text{Gaussian mixture}} f(x')\,dx'$$

where $g_1,\dots,g_L$ are the spatial filter coefficients and $\boldsymbol{\mu}_1,\dots,\boldsymbol{\mu}_L$ and $\boldsymbol{\Sigma}_1,\dots,\boldsymbol{\Sigma}_L$ are patch operator parameters

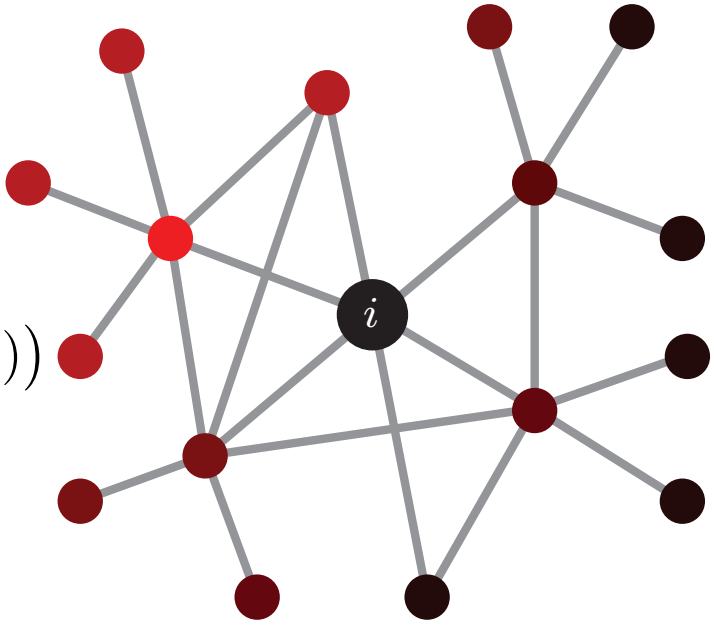
Monti et al. 2017

Mixture Model Network on graphs

- Local coordinates $\mathbf{u}_{ij}$, e.g. vertex degree, geodesic distance,...
- Gaussian weighting functions

  $w_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{u}) = \exp\big(-\frac{1}{2}(\mathbf{u}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{u}-\boldsymbol{\mu})\big)$

  with learnable covariance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{\mu}$



Local coordinates on graph

Spatial convolution

$$(f \star g)_i \propto \sum_{\ell=1}^{L} g_\ell \sum_{j=1}^{n} w_{\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell}(\mathbf{u}_{i,j}) f_j$$

where $g_1,\dots,g_L$ are the spatial filter coefficients and $\boldsymbol{\mu}_1,\dots,\boldsymbol{\mu}_L$ and $\boldsymbol{\Sigma}_1,\dots,\boldsymbol{\Sigma}_L$ are patch operator parameters

Monti et al. 2017

MoNet as generalization of previous methods

Method	**Coordinates** $\mathbf{u}(x, x')$	**Weight function** $w_{\mathbf{\Theta}}(\mathbf{u})$
CNN[1]	$\mathbf{u}(x') - \mathbf{u}(x)$	$\delta(\mathbf{u} - \mathbf{v})$ fixed parameters $\mathbf{\Theta} = \mathbf{v}$
GCN[2]	$\mathsf{deg}(x), \mathsf{deg}(x')$	$\left(1 - \lvert 1 - \frac{1}{\sqrt{u_1}}\rvert\right)\left(1 - \lvert 1 - \frac{1}{\sqrt{u_2}}\rvert\right)$
GCNN[3]	$\rho(x, x'), \theta(x, x')$	$\exp\left(-\frac{1}{2}(\mathbf{u} - \mathbf{v})^\top \begin{pmatrix} \sigma_\rho^2 & \\ & \sigma_\theta^2 \end{pmatrix}^{-1} (\mathbf{u} - \mathbf{v})\right)$ fixed parameters $\mathbf{\Theta} = (\mathbf{v}, \sigma_\rho, \sigma_\theta)$
ACNN[4]	$\rho(x, x'), \theta(x, x')$	$\exp\left(-t\mathbf{u}^\top \mathbf{R}_\varphi \begin{pmatrix} \alpha & \\ & 1 \end{pmatrix} \mathbf{R}_\varphi^\top \mathbf{u}\right)$ fixed parameters $\mathbf{\Theta} = (\alpha, \varphi, t)$
MoNet[5]	$\rho(x, x'), \theta(x, x')$	$\exp\left(-\frac{1}{2}(\mathbf{u} - \boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1} (\mathbf{u} - \boldsymbol{\mu})\right)$ learnable parameters $\mathbf{\Theta} = (\boldsymbol{\mu}, \mathbf{\Sigma})$

Some CNN models can be considered as particular settings of MoNet with weighting functions of different form

Methods: [1]LeCun et al. 1998; [2]Kipf, Welling 2016; [3]Masci et al. 2015; [4]Boscaini et al. 2016; [5]Monti et al. 2017

Spectral vs Spatial methods

ChebNet filter

$$\mathbf{h} = \tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f}$$

Spatial filter

$$\mathbf{h} = (\mathcal{D}\mathbf{f})\mathbf{g}$$

Defferrard, Bresson, Vandergheynst 2016; Masci et al. 2015; Monti et al. 2017

Spectral vs Spatial methods

ChebNet filter

$$\mathbf{h} = \sum_{\ell=0}^{r} \alpha_\ell \boldsymbol{\Delta}^\ell \mathbf{f}$$

Spatial filter

$$\mathbf{h} = (\mathcal{D}\mathbf{f})\mathbf{g}$$

Defferrard, Bresson, Vandergheynst 2016; Masci et al. 2015; Monti et al. 2017

Spectral vs Spatial methods

ChebNet filter

$$h_i = \sum_{\ell=0}^{r} \alpha_\ell (\boldsymbol{\Delta}^\ell \mathbf{f})_i$$

Spatial filter

$$h_i = \sum_{\ell=1}^{L} g_\ell (\mathbf{W}_\ell \mathbf{f})_i$$

ChebNet is a particular setting of spatial convolution with local weighting functions given by the powers of the Laplacian $\mathbf{W}_\ell = \boldsymbol{\Delta}^\ell$

Defferrard, Bresson, Vandergheynst 2016; Masci et al. 2015; Monti et al. 2017

Graph Attention Networks (GAT)

Main idea: neighborhood average

$$\mathbf{f}_i' = \sum_{j:(i,j)\in\mathcal{E}} \alpha_{ij}\mathbf{f}_j$$

weighted by attention score

$$\alpha_{ij} = \frac{e^{\xi([\mathbf{f}_i\mathbf{W}, \mathbf{f}_j\mathbf{W}]\mathbf{a})}}{\sum_{k:(i,k)\in\mathcal{E}} e^{\xi([\mathbf{f}_i\mathbf{W}, \mathbf{f}_k\mathbf{W}]\mathbf{a})}}$$

which is a learnable transformation of the local features with learnable parameters $\mathbf{W}$, $\mathbf{a}$

Veličković et al. 2018

Graph Attention Networks (GAT)

Main idea: neighborhood average

$$\mathbf{f}_i' = \sum_{j:(i,j)\in\mathcal{E}} \alpha_{ij}\mathbf{f}_j$$

weighted by attention score

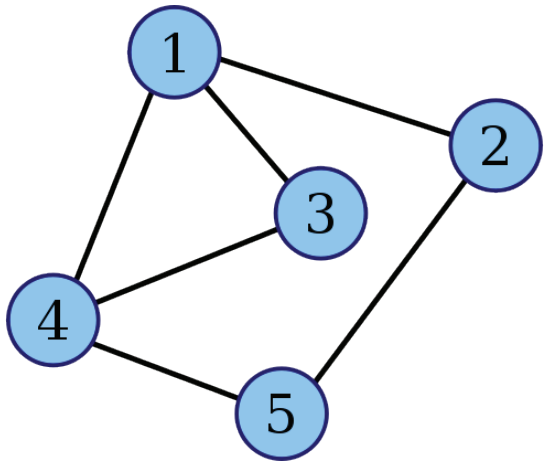
$$\alpha_{ij} = \frac{e^{\xi([\mathbf{f}_i\mathbf{W},\mathbf{f}_j\mathbf{W}]\mathbf{a})}}{\sum_{k:(i,k)\in\mathcal{E}} e^{\xi([\mathbf{f}_i\mathbf{W},\mathbf{f}_k\mathbf{W}]\mathbf{a})}}$$

which is a learnable transformation of the local features with learnable parameters $\mathbf{W}$, $\mathbf{a}$

Particular case of MoNet-type architectures!

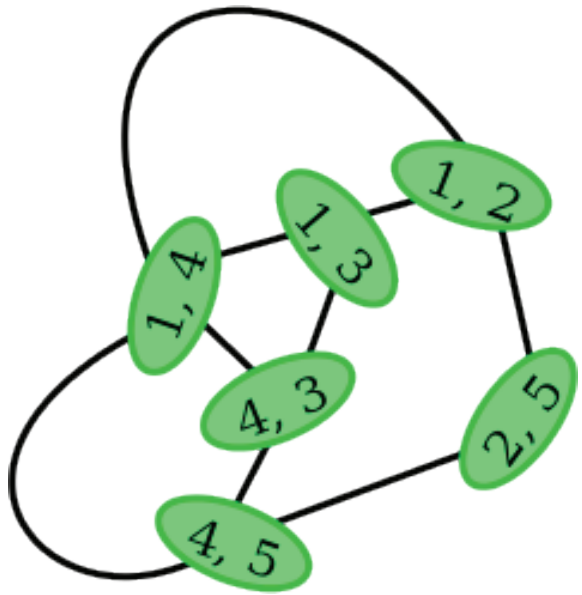
Veličković et al. 2018

Primal and Dual graphs



Primal graph

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



Dual or line graph

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

Harary, Norman 1960

Dual/Primal Graph CNN (DPGCNN)

Alternate GAT-type convolutions applied on primal and dual graphs

- Dual convolution on $\tilde{\mathcal{G}}$:

$$\tilde{\mathbf{f}}'_{ij} = \xi\left(\sum_{r\in\mathcal{N}_i}\tilde{\alpha}_{ij,ir}\tilde{\mathbf{f}}_{ir}\tilde{\mathbf{W}} + \sum_{t\in\mathcal{N}_j}\tilde{\alpha}_{ij,tj}\tilde{\mathbf{f}}_{tj}\tilde{\mathbf{W}}\right)$$

$$\tilde{\alpha}_{ij,ik} = \frac{e^{\xi([\tilde{\mathbf{f}}_{ij}\tilde{\mathbf{W}},\tilde{\mathbf{f}}_{ik}\tilde{\mathbf{W}}]\tilde{\mathbf{a}})}}{\sum_{r\in\mathcal{N}_i}e^{\xi([\tilde{\mathbf{f}}_{ij}\tilde{\mathbf{W}},\tilde{\mathbf{f}}_{ir}\tilde{\mathbf{W}}]\tilde{\mathbf{a}})} + \sum_{t\in\mathcal{N}_j}e^{\xi([\tilde{\mathbf{f}}_{ij}\tilde{\mathbf{W}},\tilde{\mathbf{f}}_{tj}\tilde{\mathbf{W}}]\tilde{\mathbf{a}})}}$$

- Primal convolution on $\mathcal{G}$:

$$\mathbf{f}'_i = \xi\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{f}_j\mathbf{W}\right) \qquad \alpha_{ij} = \frac{e^{\xi(\tilde{\mathbf{f}}'_{ij}\mathbf{a})}}{\sum_{k\in\mathcal{N}_i}e^{\xi(\tilde{\mathbf{f}}'_{ik}\mathbf{a})}}$$

Monti et al. 2018

Example: citation networks

Method	**Cora**[1]	**CiteSeer**[2]
Manifold Regularization[3]	59.5%	60.1%
Semidefinite Embedding[4]	59.0%	59.6%
Label Propagation[5]	68.0%	45.3%
DeepWalk[6]	67.2%	43.2%
Planetoid[7]	75.7%	64.7%
GCN[8]	81.6%	70.3%
MoNet[9]	81.7%	–
GAT[10]	83.0%	72.5%
DPGCN[11]	**83.3%**	**72.6%**

Classification accuracy of different methods on citation network datasets

Data: [1,2]Sen et al. 2008; methods: [3]Belkin et al. 2006; [4]Weston et al. 2012; [5]Zhu et al. 2003; [6]Perozzi et al. 2014; [7]Yang et al. 2016; [8]Kipf, Welling 2016 ; [9]Monti et al. 2017; [10]Veličković et al. 2018; [11]Monti et al. 2018

Different formulations of non-Euclidean CNNs

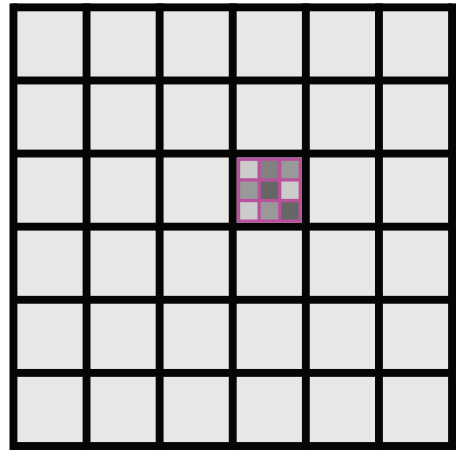


Spectral domain



Spatial domain



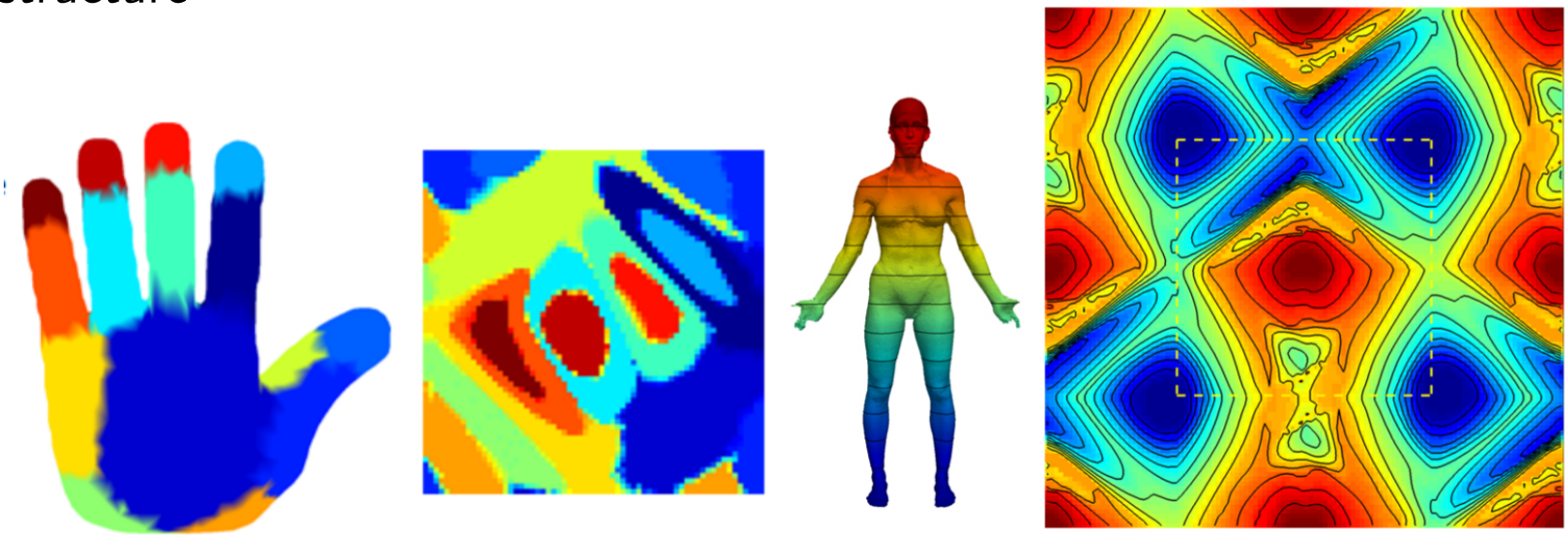


Parametric domain

Parametric domain geometric deep learning methods

Global parametrization

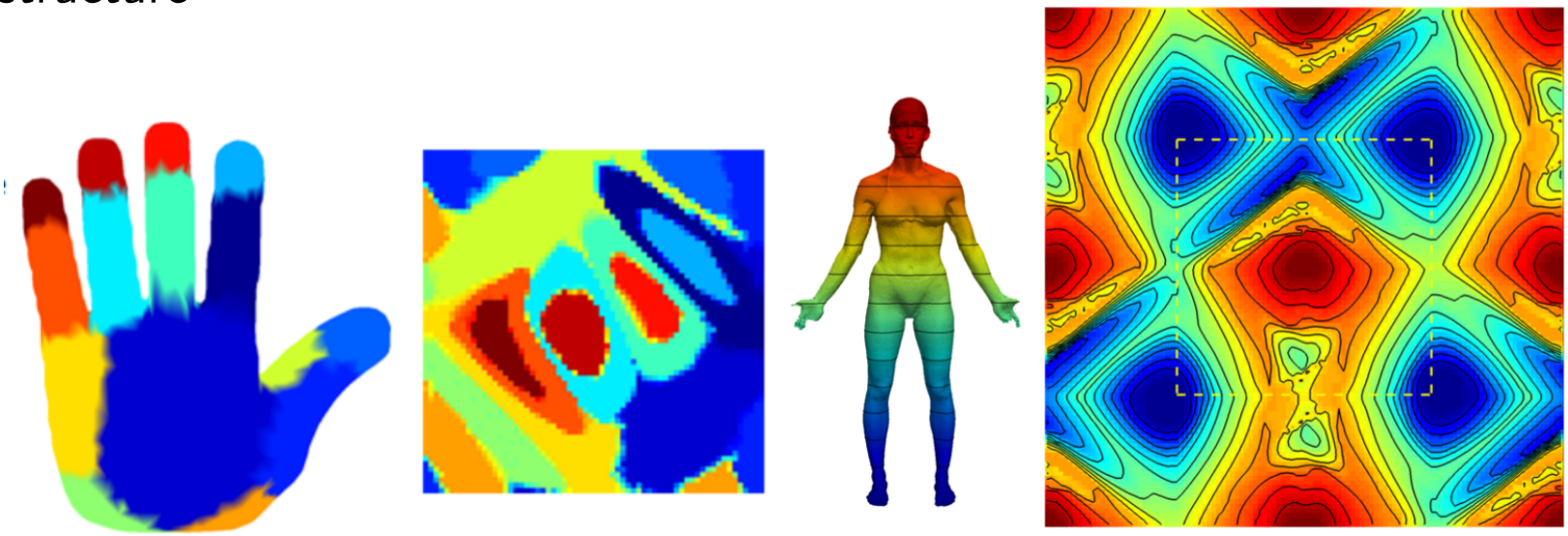
Map the input surface to some parametric domain with shift-invariant structure



Sinha et al. 2016; Maron et al. 2017; Ezuz et al. 2017

Global parametrization

Map the input surface to some parametric domain with shift-invariant structure

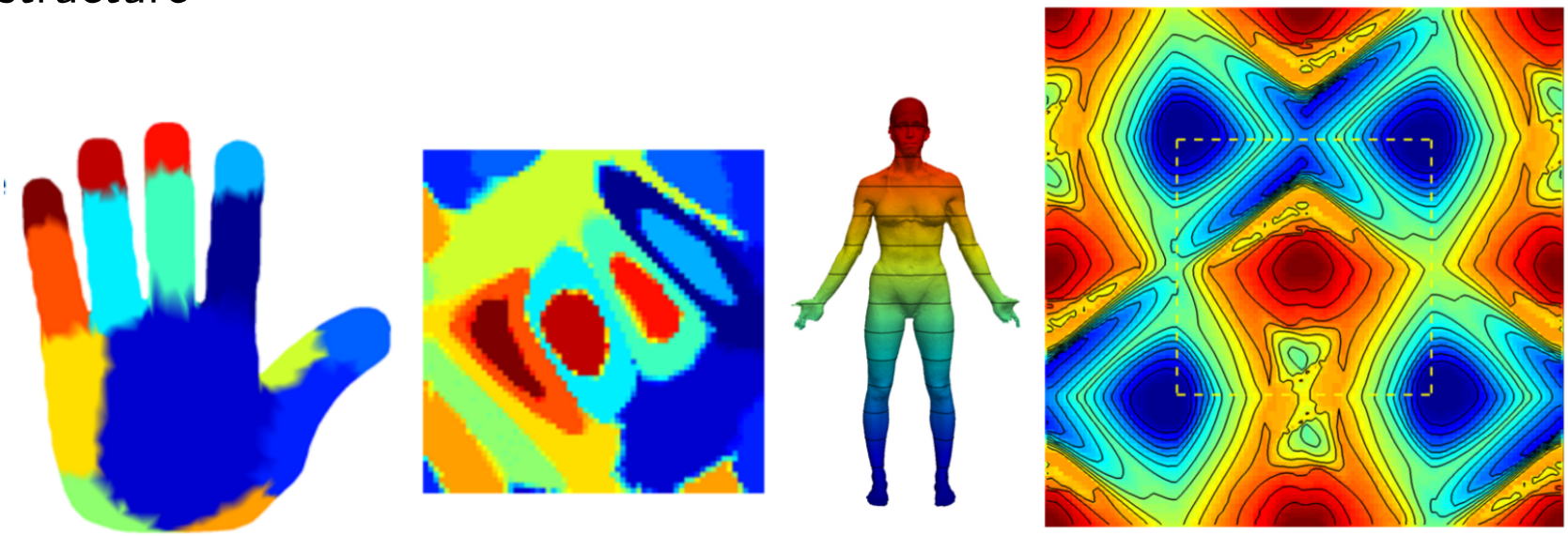


☺ Allows to use standard CNNs (pull back convolution from the parametric space)

Sinha et al. 2016; Maron et al. 2017; Ezuz et al. 2017

Global parametrization

Map the input surface to some parametric domain with shift-invariant structure



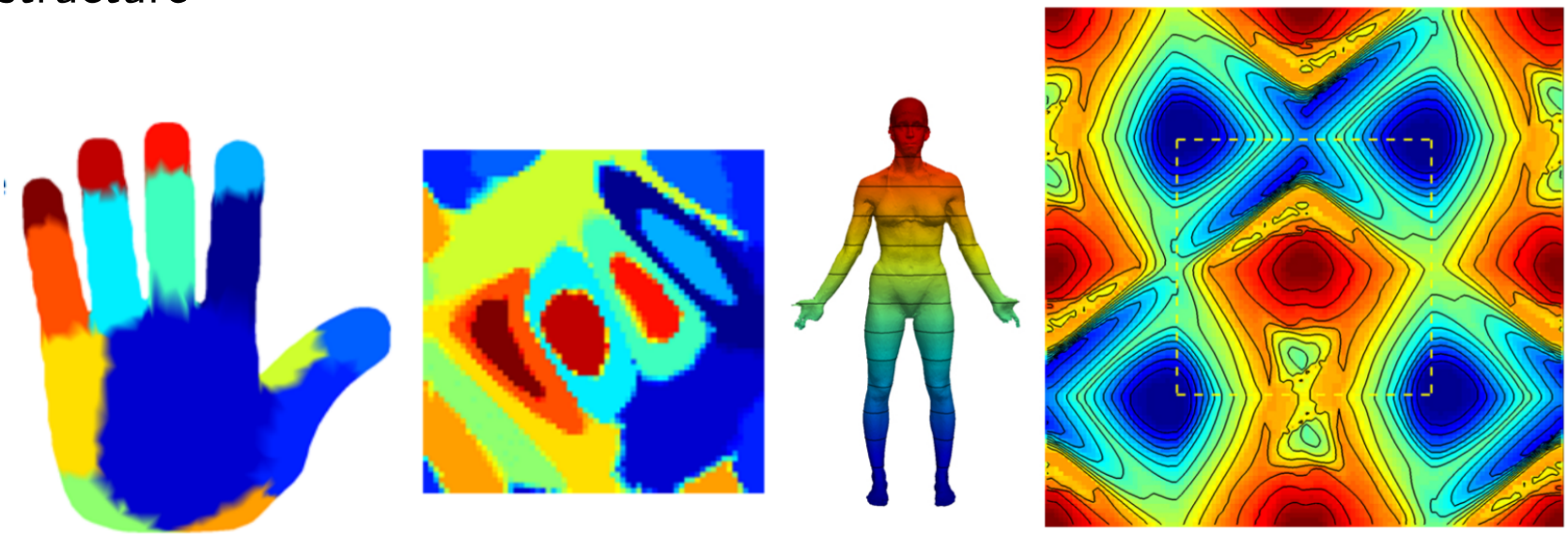
☺ Allows to use standard CNNs (pull back convolution from the parametric space)

☺ Guaranteed invariance to some classes of transformations

Sinha et al. 2016; Maron et al. 2017; Ezuz et al. 2017

Global parametrization

Map the input surface to some parametric domain with shift-invariant structure

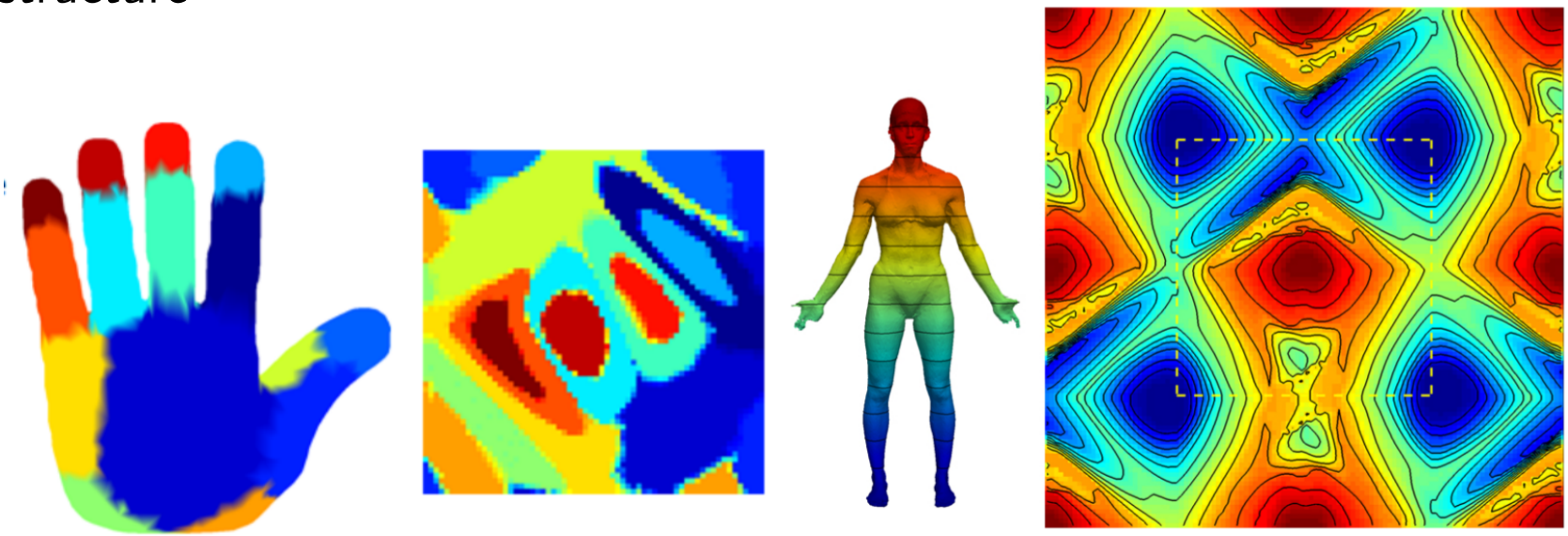


- ☺ Allows to use standard CNNs (pull back convolution from the parametric space)
- ☺ Guaranteed invariance to some classes of transformations
- ☹ Parametrization may not be unique

Sinha et al. 2016; Maron et al. 2017; Ezuz et al. 2017

Global parametrization

Map the input surface to some parametric domain with shift-invariant structure



- ☺ Allows to use standard CNNs (pull back convolution from the parametric space)
- ☺ Guaranteed invariance to some classes of transformations
- ☹ Parametrization may not be unique
- ☹ Embedding may introduce distortion

Sinha et al. 2016; Maron et al. 2017; Ezuz et al. 2017

Translation invariance on manifolds

Translation on manifold = locally Euclidean translation

Translation invariance on manifolds

Translation on manifold = locally Euclidean translation = flow along a non-vanishing vector field

Translation invariance on manifolds

Translation on manifold = locally Euclidean translation = flow along a non-vanishing vector field

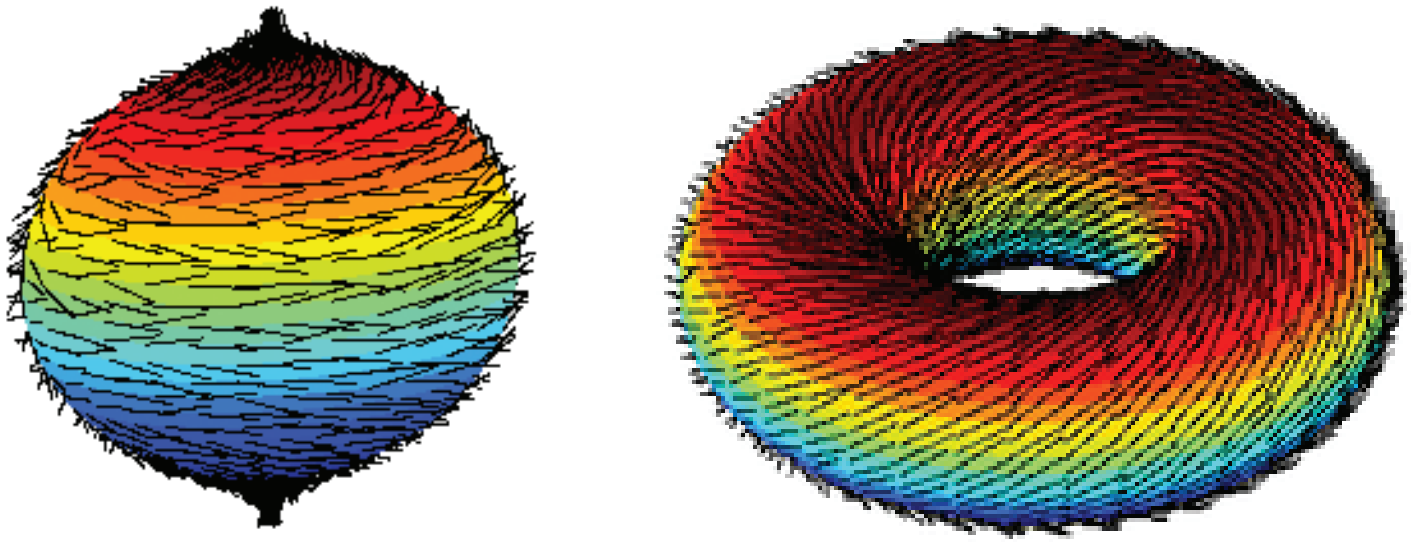
Poincaré-Hopf Theorem Non-vanishing vector field on a closed orientable compact 2-manifold implies manifold of genus 1 (torus)

Poincaré 1881; Hopf 1926

Translation invariance on manifolds

Translation on manifold = locally Euclidean translation = flow along a non-vanishing vector field

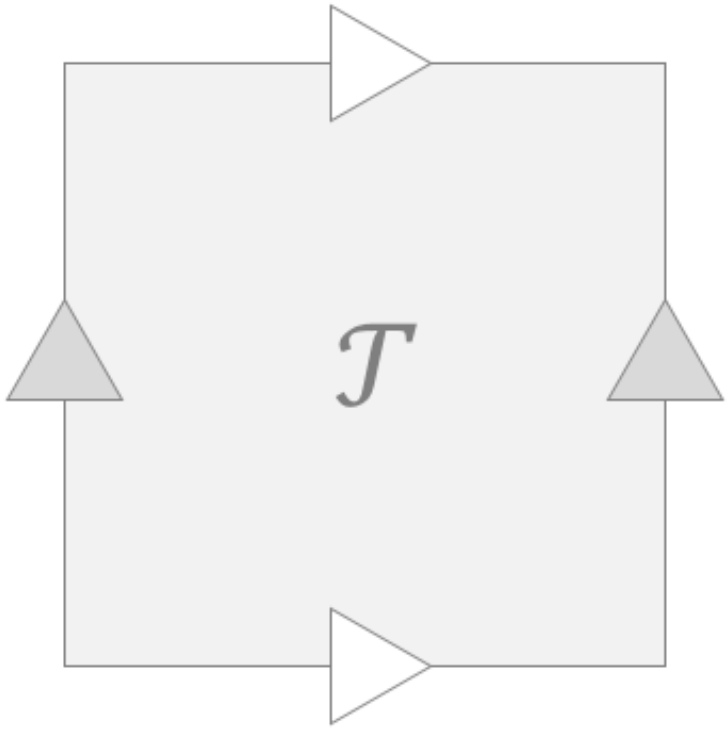
Poincaré-Hopf Theorem Non-vanishing vector field on a closed orientable compact 2-manifold implies manifold of genus 1 (torus)

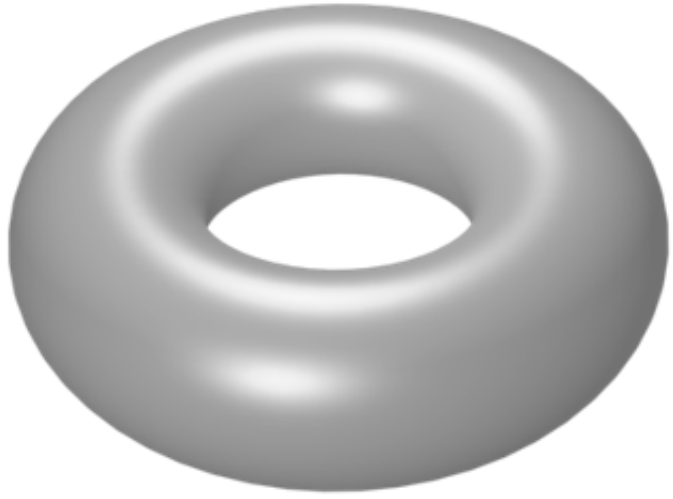


'Hairy Ball Theorem' states that a sphere cannot be combed

Poincaré 1881; Hopf 1926

Translation invariance on the torus

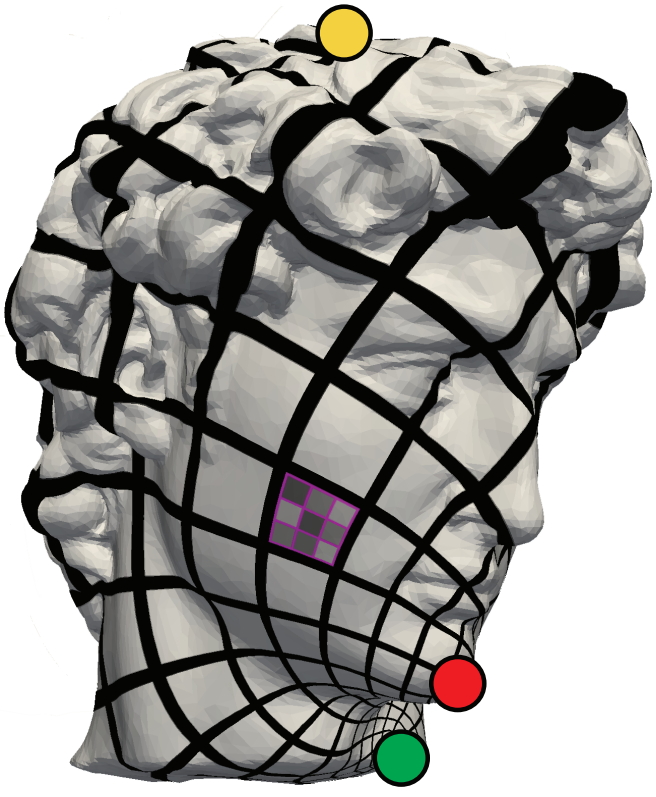


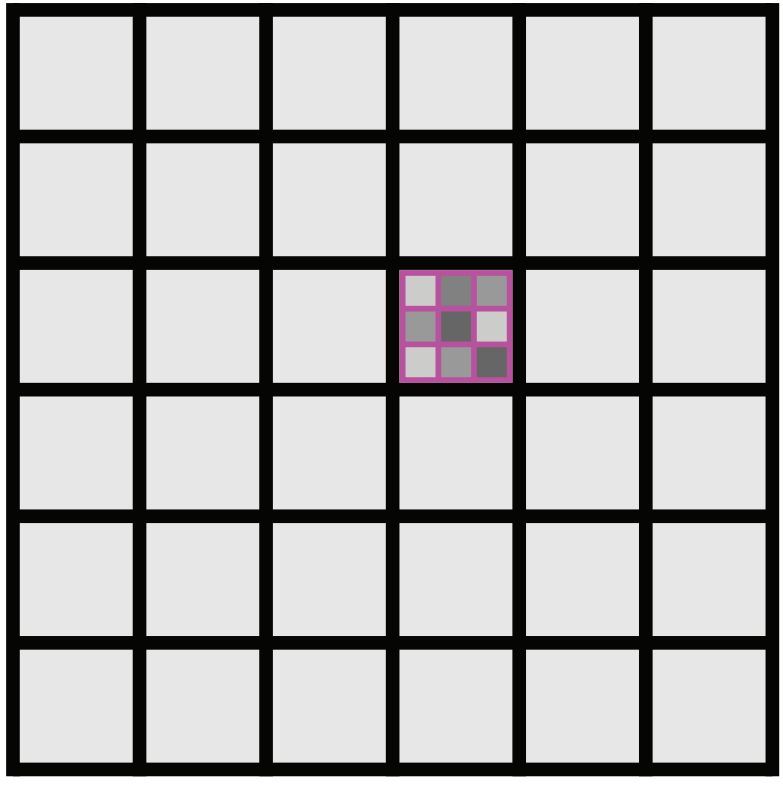


Torus is the only closed orientable surface admitting a translation group

Convolution on torus

For any triplet of points on $\mathcal{X}$, construct conformal homeomorphism from the 4-cover $\mathcal{X}^4$ to $\mathcal{T}$ using orbifold-Tutte method





Maron et al. 2017

Conformal zoom

- Embedding depends on the choice of the triplets of points

Maron et al. 2017

Conformal zoom

- Embedding depends on the choice of the triplets of points
- 'Conformal zoom' effect

Maron et al. 2017

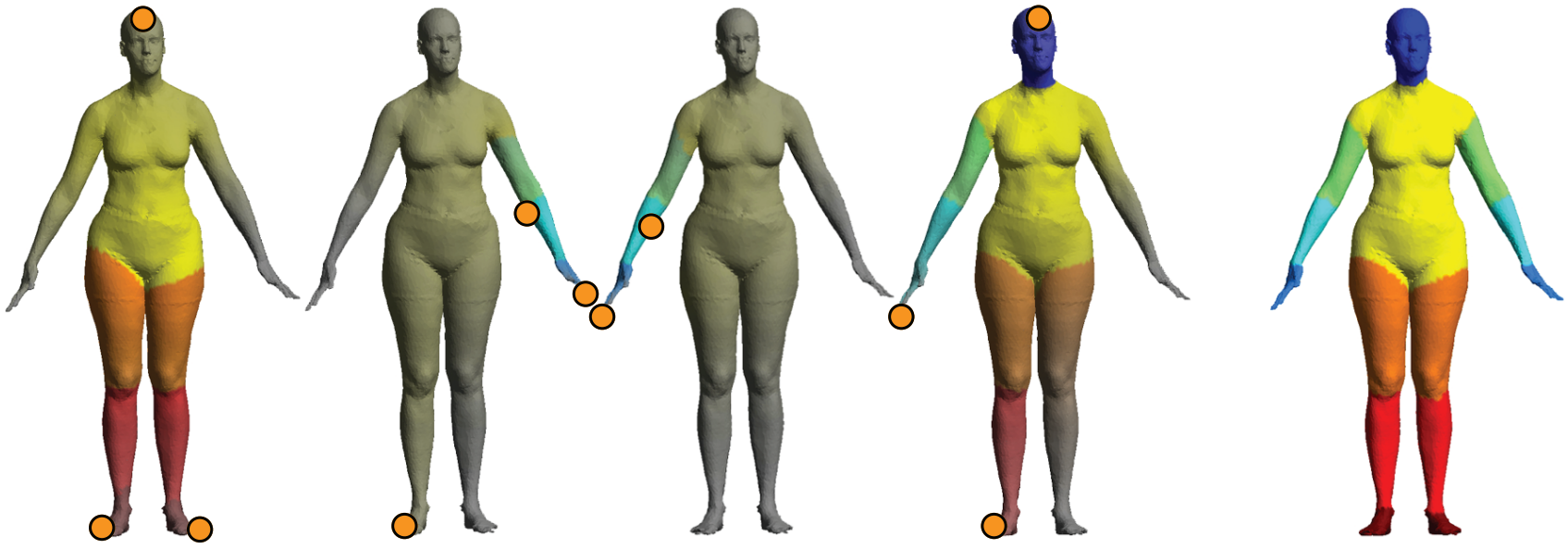
Conformal zoom

- Embedding depends on the choice of the triplets of points
- 'Conformal zoom' effect
- Choose multiple triples and aggregate results in training / test phase

Maron et al. 2017

Conformal zoom

- Embedding depends on the choice of the triplets of points
- 'Conformal zoom' effect
- Choose multiple triples and aggregate results in training / test phase

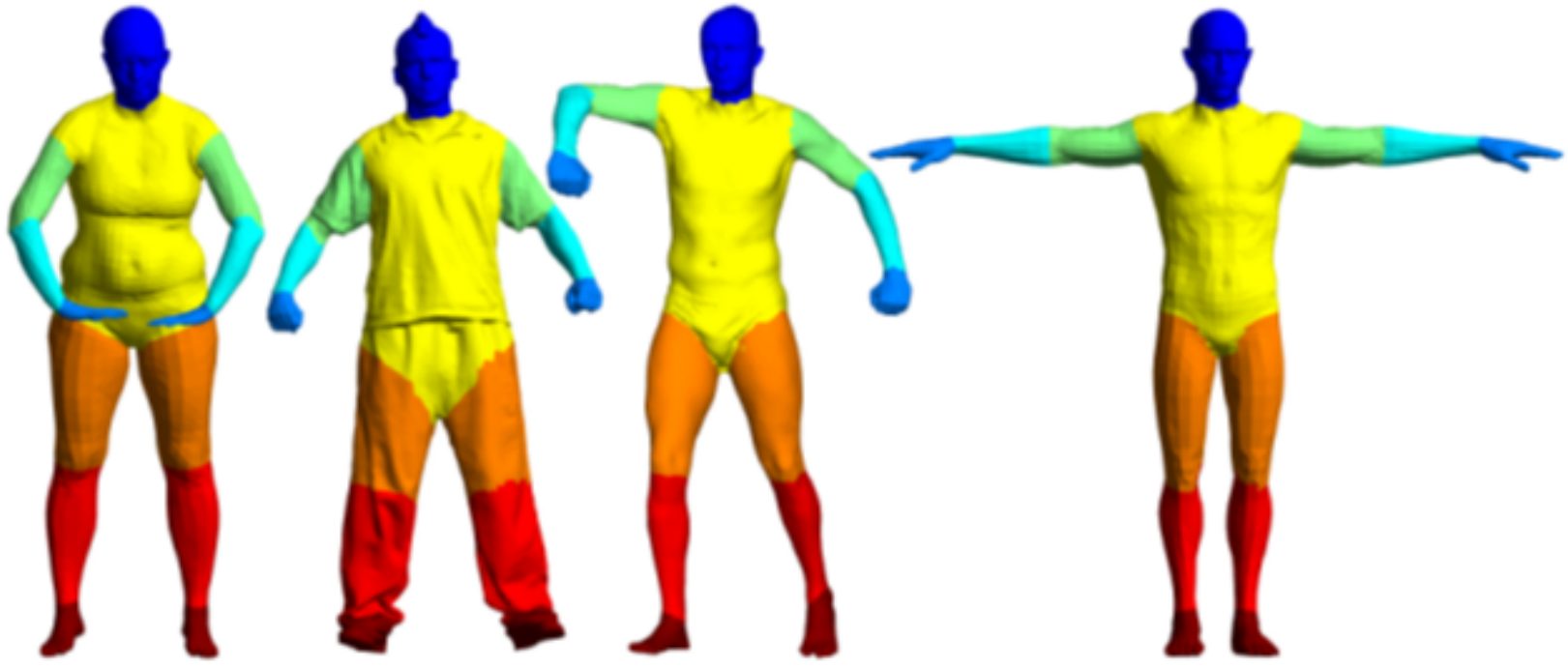


different triplets

aggregated result

Maron et al. 2017

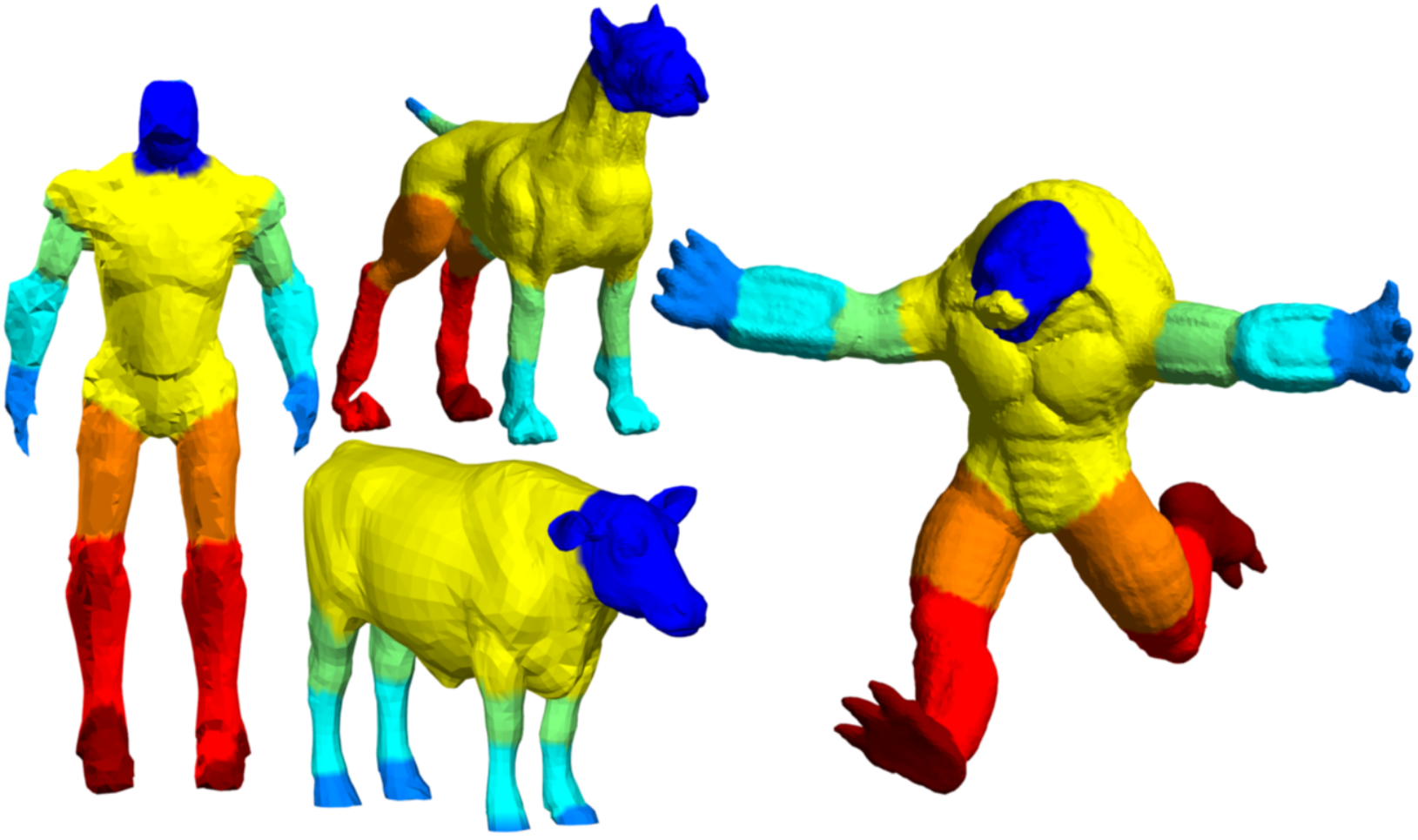
Example: shape segmentation with Toric CNN



Examples of shape segmentation obtained with Toric CNN

Maron et al. 2017

Example: shape segmentation with Toric CNN



Examples of shape segmentation obtained with Toric CNN

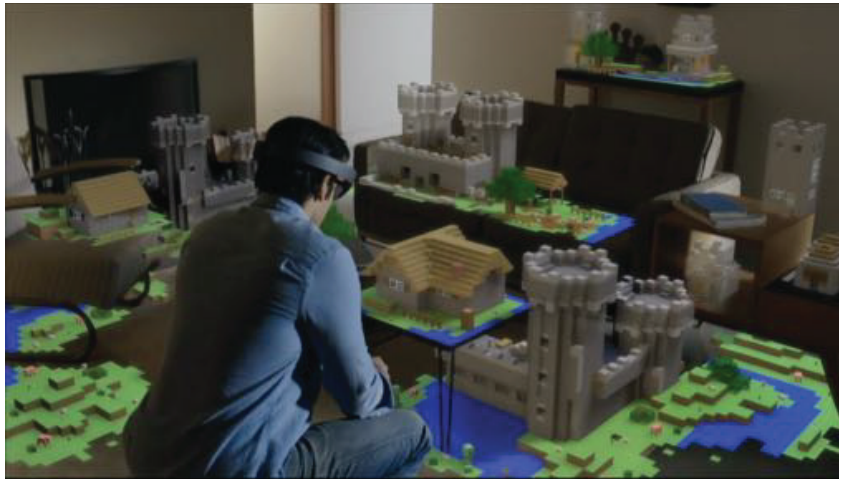
Maron et al. 2017

Application in Computer Graphics and 3D Vision

Application dealing with 3D data



Computer graphics



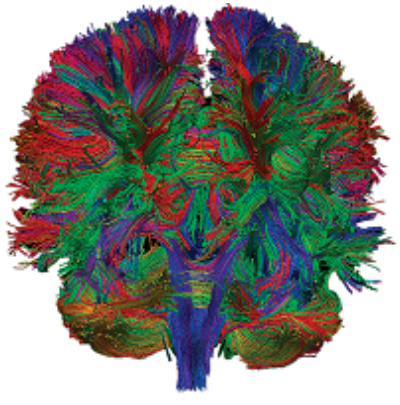
Virtual/augmented reality



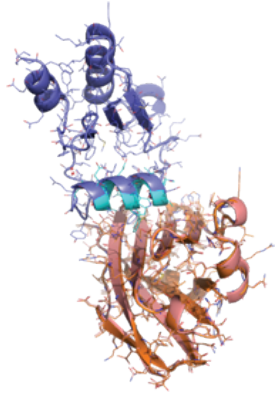
Robotics



Autonomous driving



Medicine



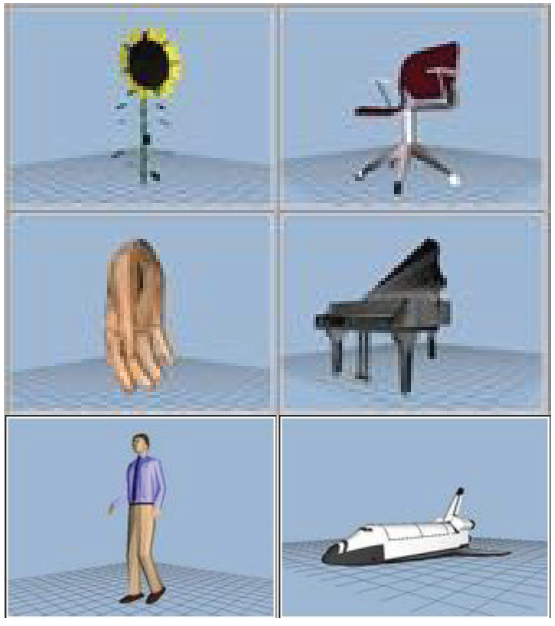
Drug design

3D data

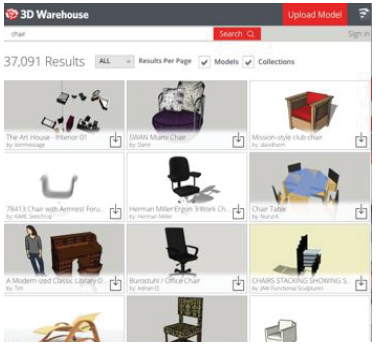




1998



2008



Clara.io

yobi 3D

Sketchfab

Google SketchUp

TINKERCAD



KINECT for XBOX 360



REALSENSE TECHNOLOGY

biobank uk

2018



>$10K

2005



$100

2010



$20

2014



???

2017

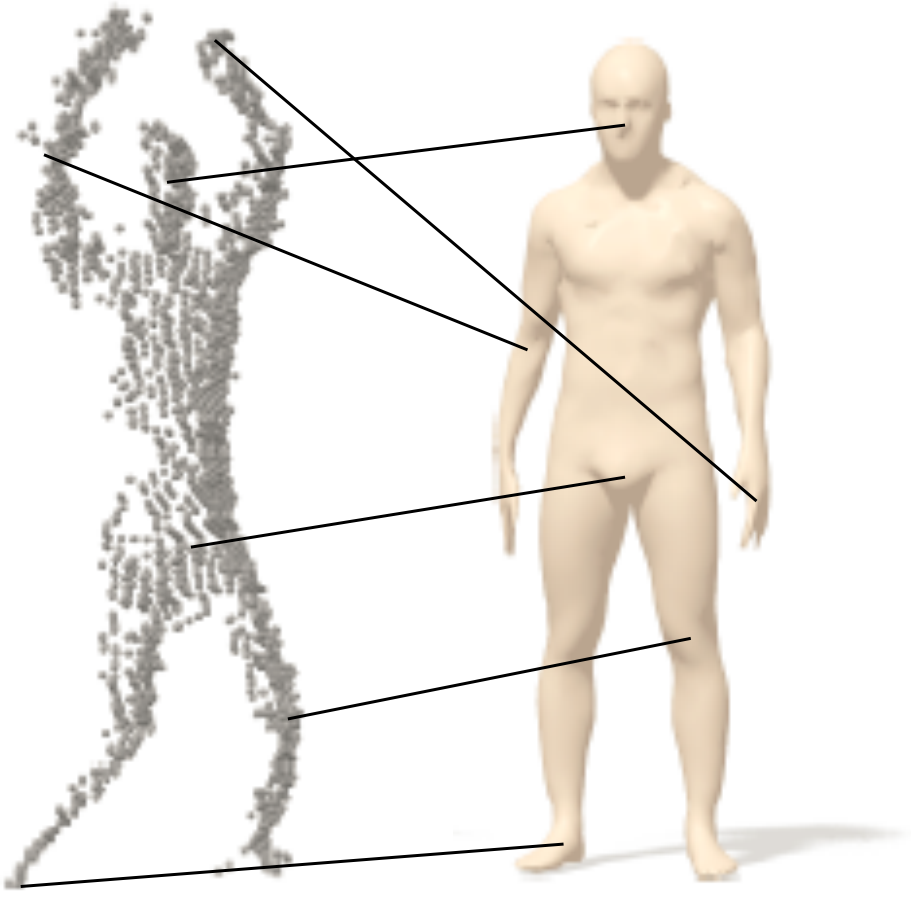
Faceshift (acquired by Apple in 2015)

Analysis and synthesis



Input
3D scan

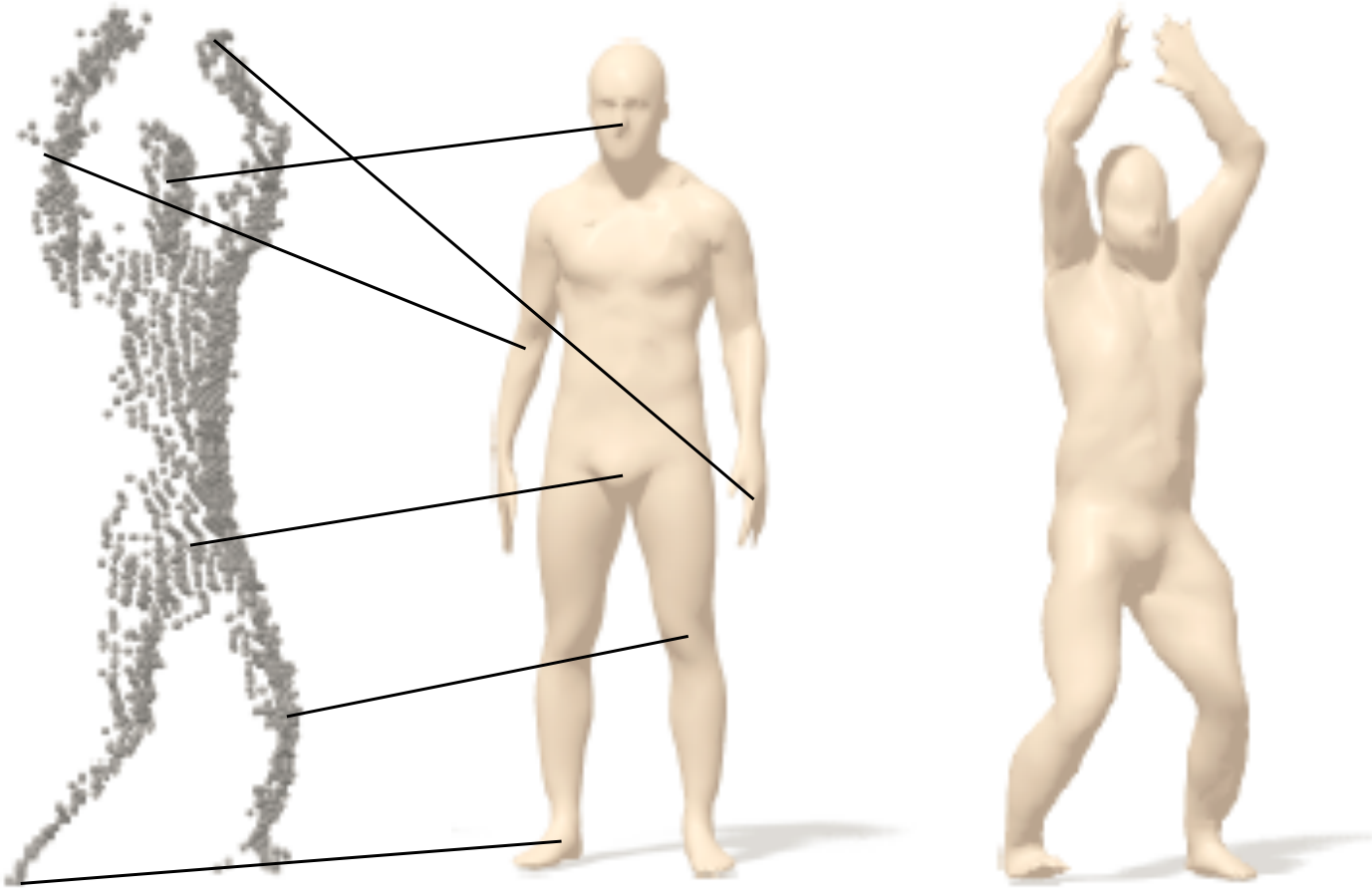
Analysis and synthesis



Input
3D scan

Correspondence to
Reference shape

Analysis and synthesis

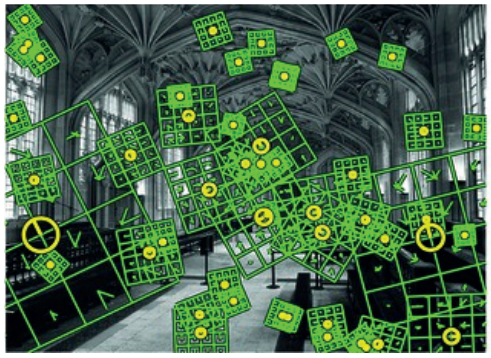


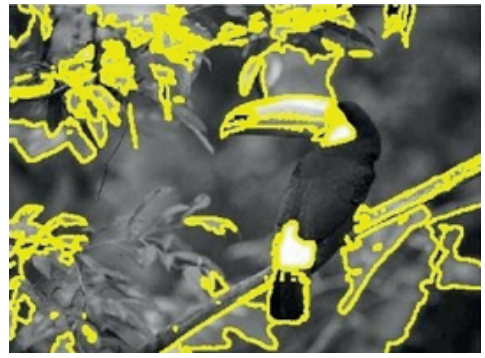
Input
3D scan

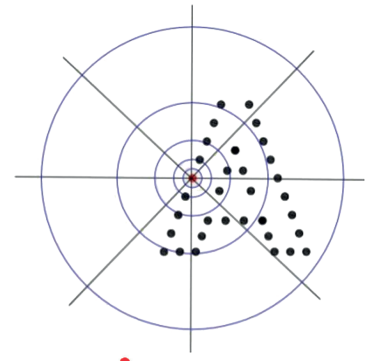
Correspondence to
Reference shape

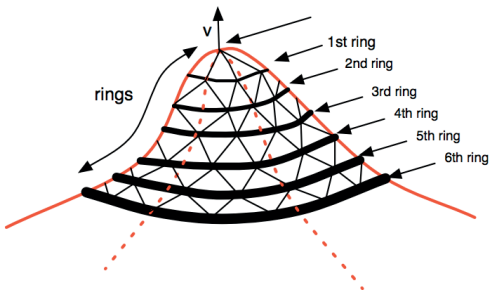
Deformation of
reference shape

3D feature descriptors





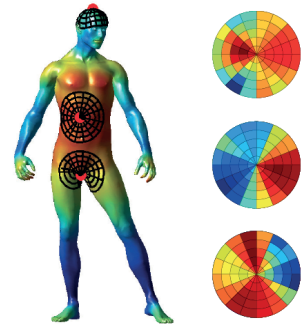




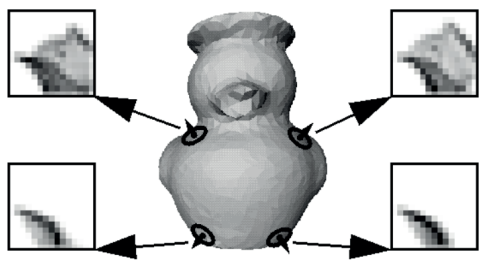
SIFT[1] / MeshHOG[2]



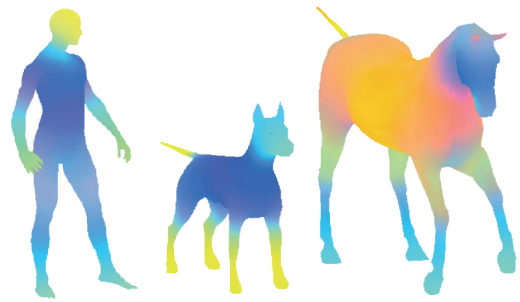
MSER[3] / ShapeMSER[4]



(Intrinsic[6]) Shape context[5]



Spin image[7]

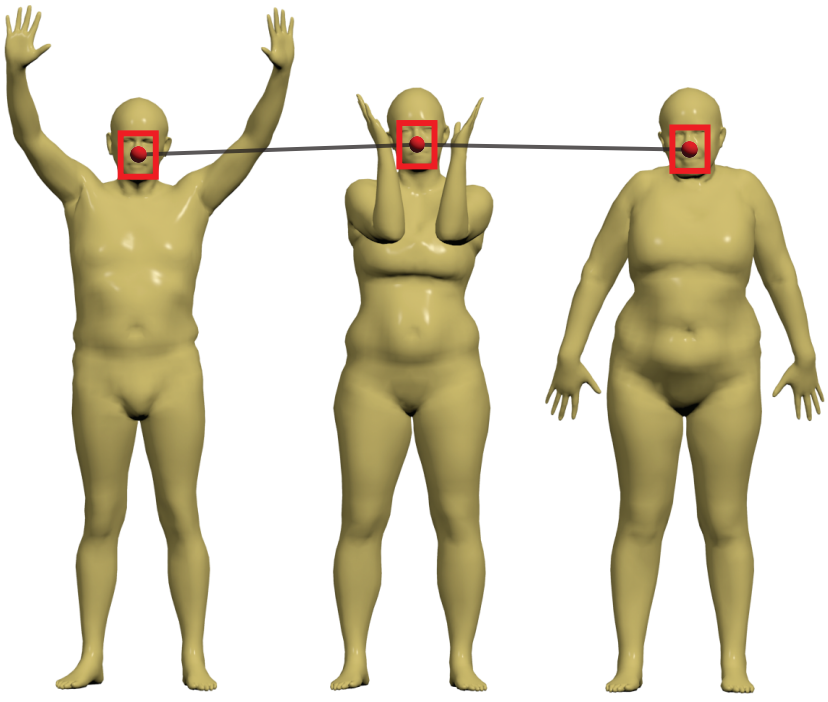


Heat kernel signature[8]

[1]Lowe 2004; [2]Zaharescu et al. 2009; [3]Matas et al. 2002; [4]Litman et al. 2010;
[5]Belongie et al. 2000; [6]Kokkinos et al. 2012; [7]Johnson et al. 1999; [8]Sun et al. 2009

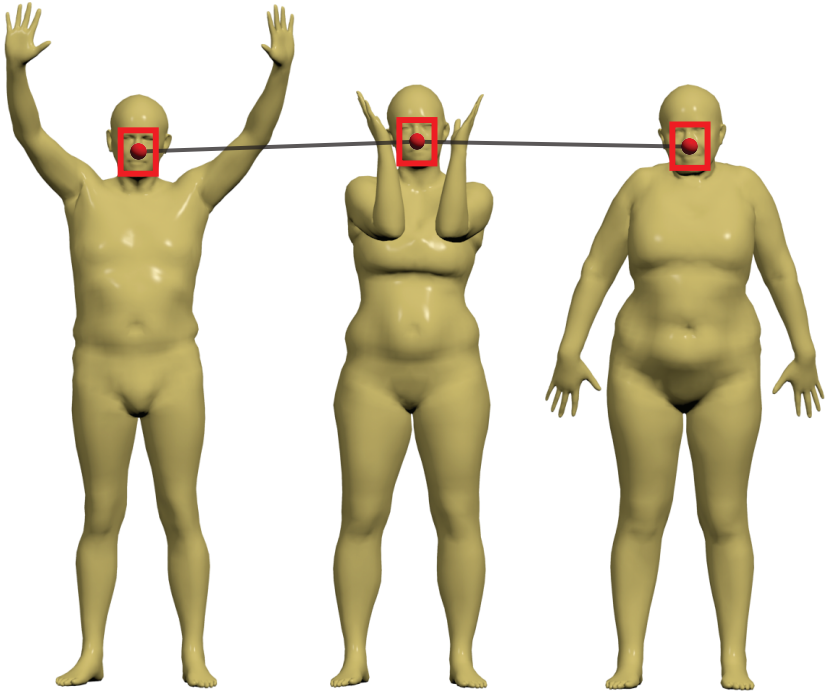
Task-specific features

Correspondence

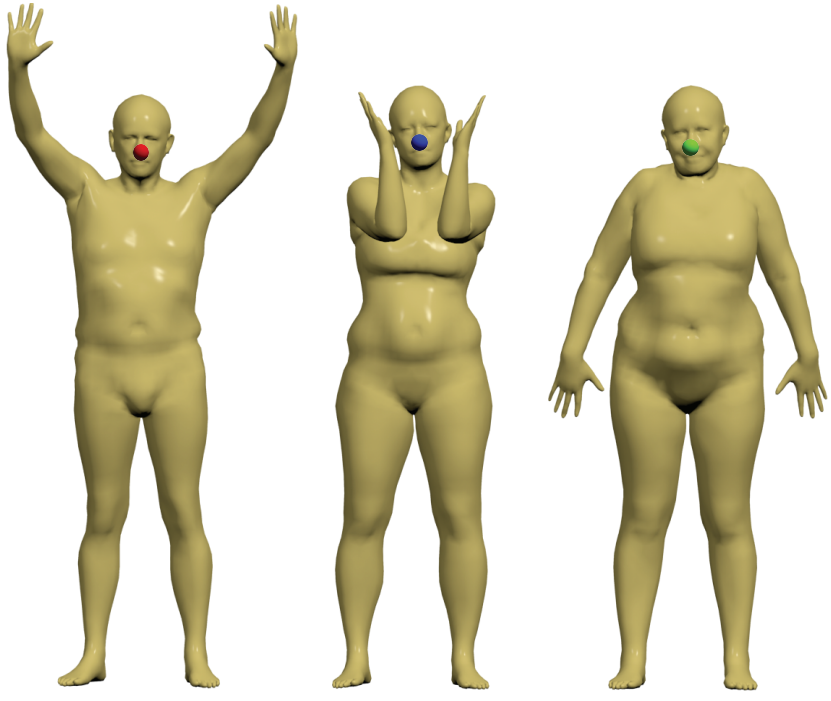


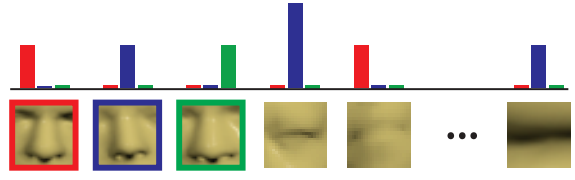
Task-specific features

Correspondence



Similarity

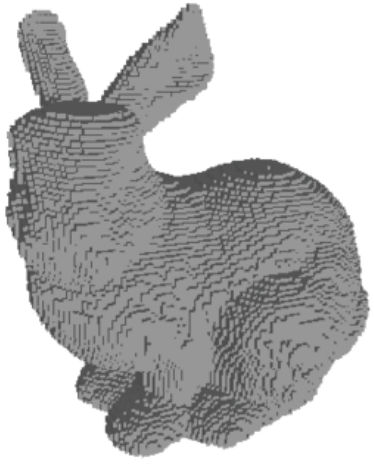




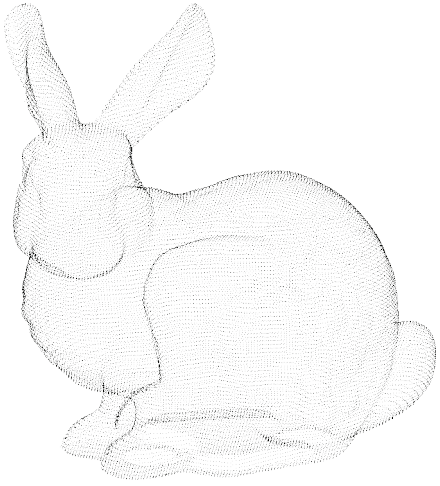
Shape representation



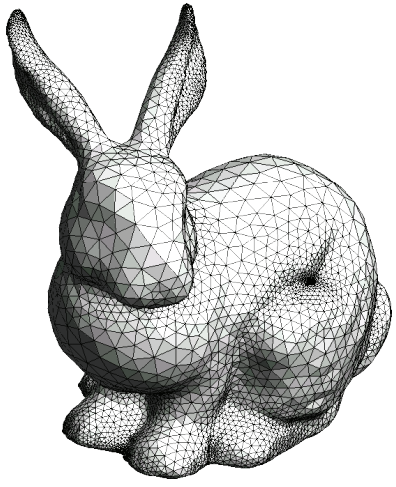
Image-based



Volumetric

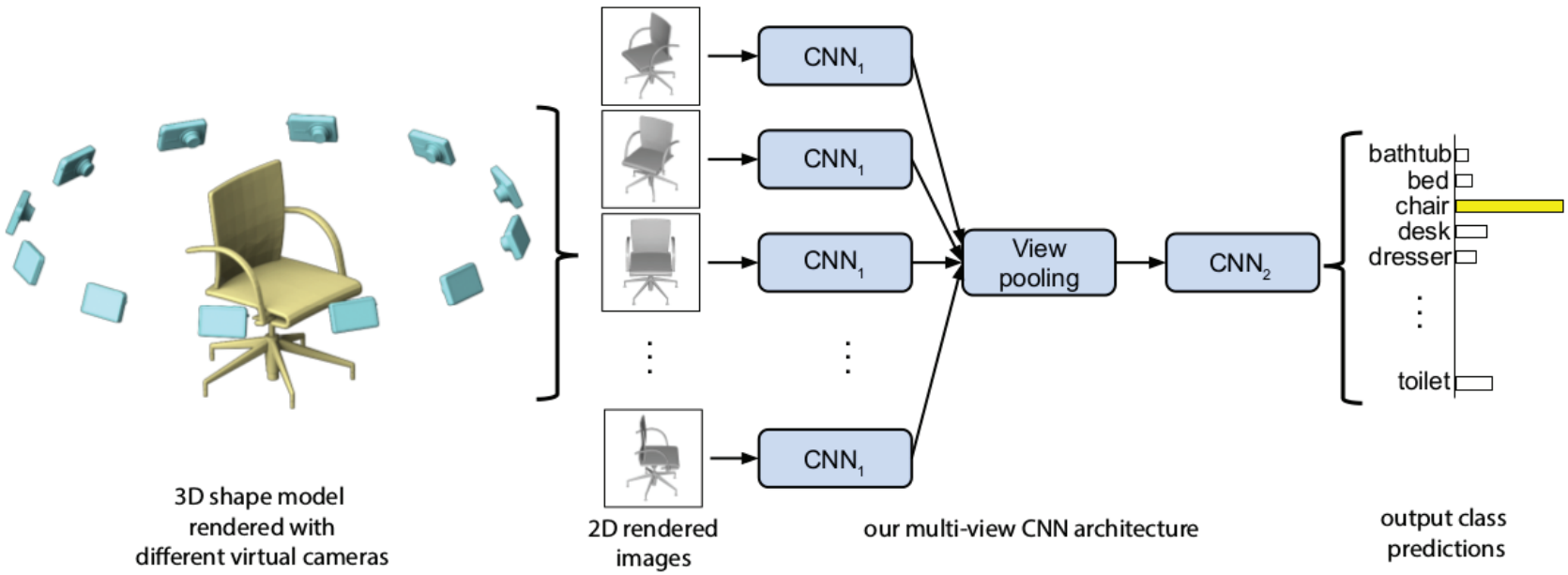


Point-based



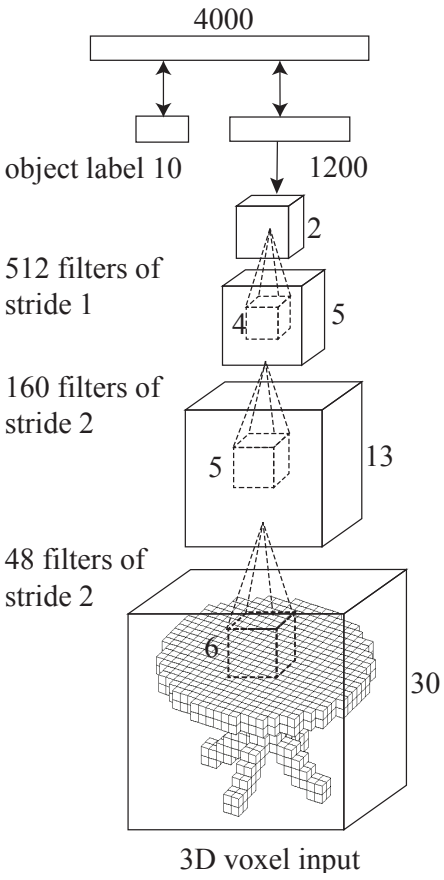
Surface-based

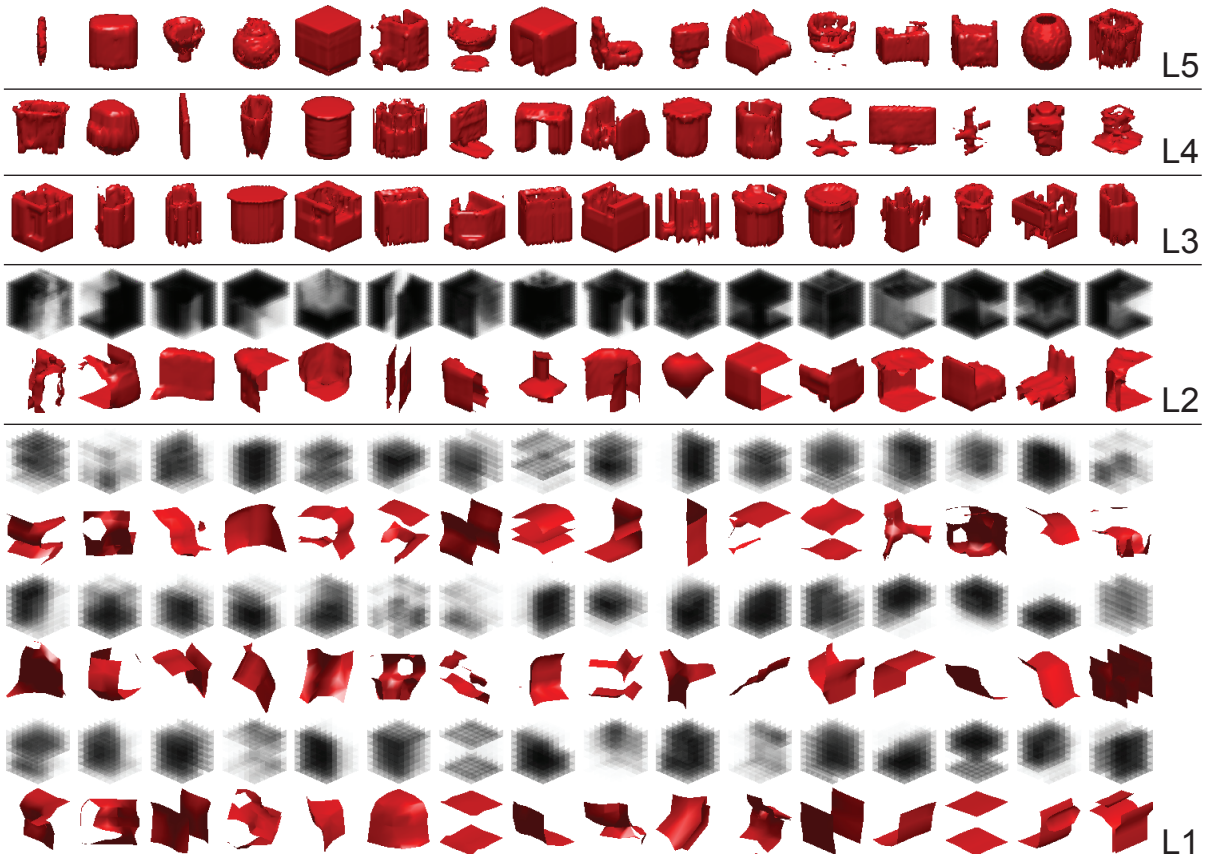
Multi-view CNN



Su, Maji et al. 2015

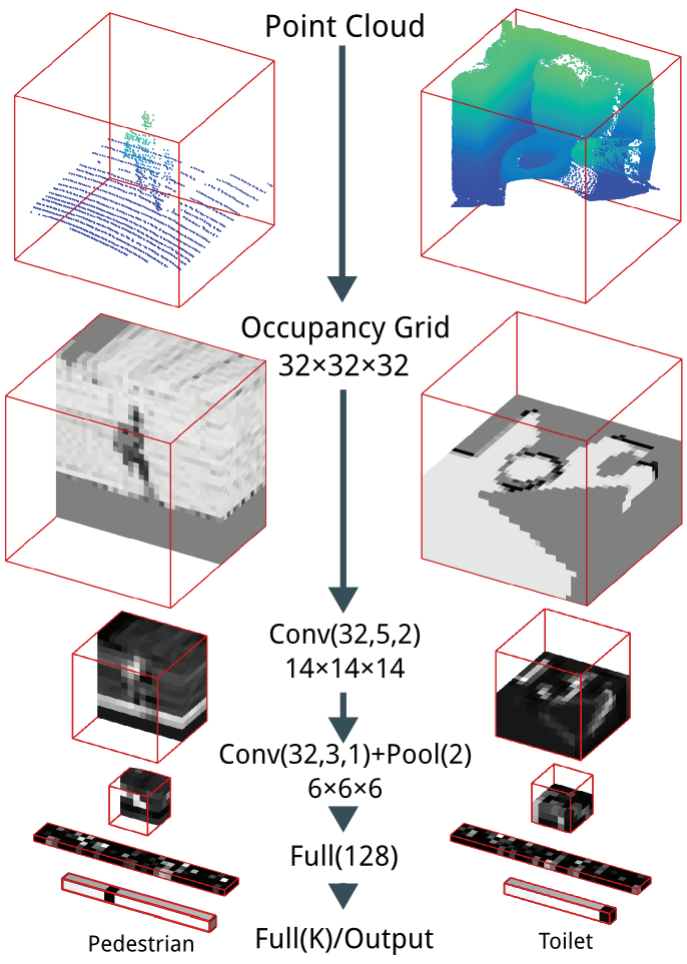
Volumetric methods: 3D ShapeNet

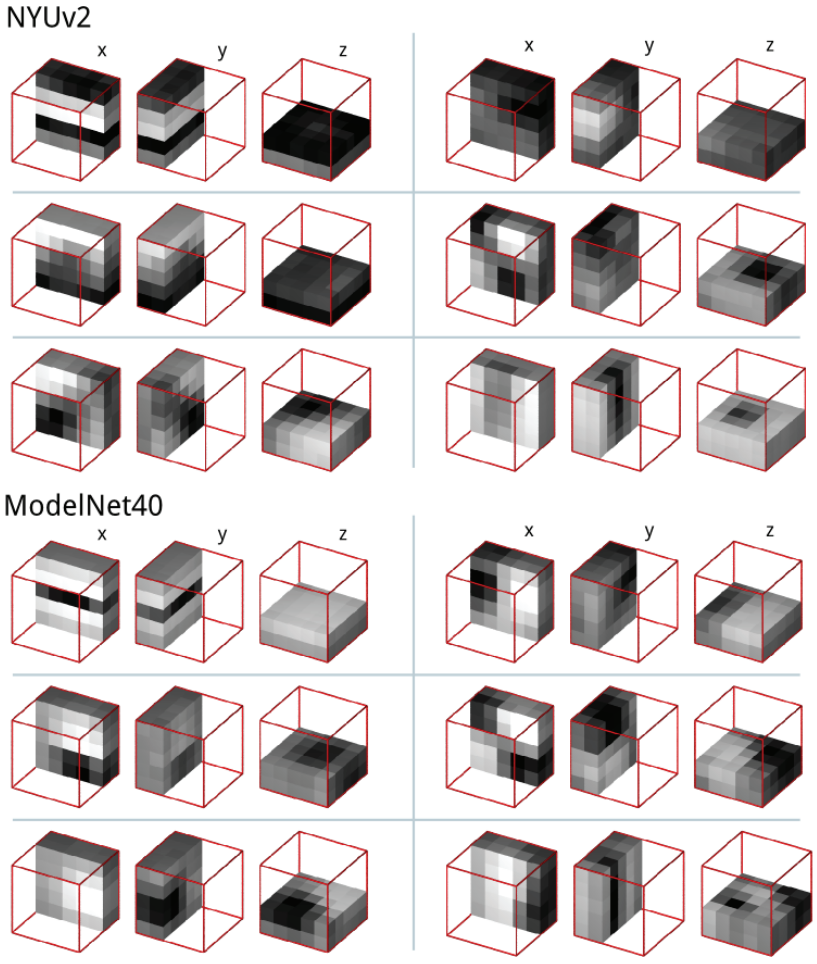




Wu et al. 2015

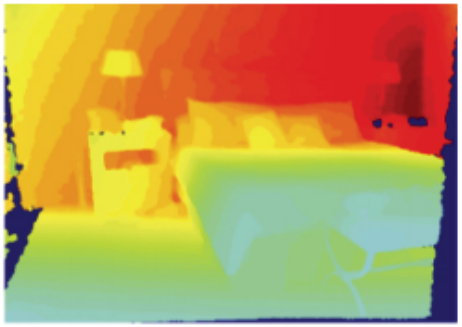
Volumetric methods: VoxNet





Maturana, Scherer 2015

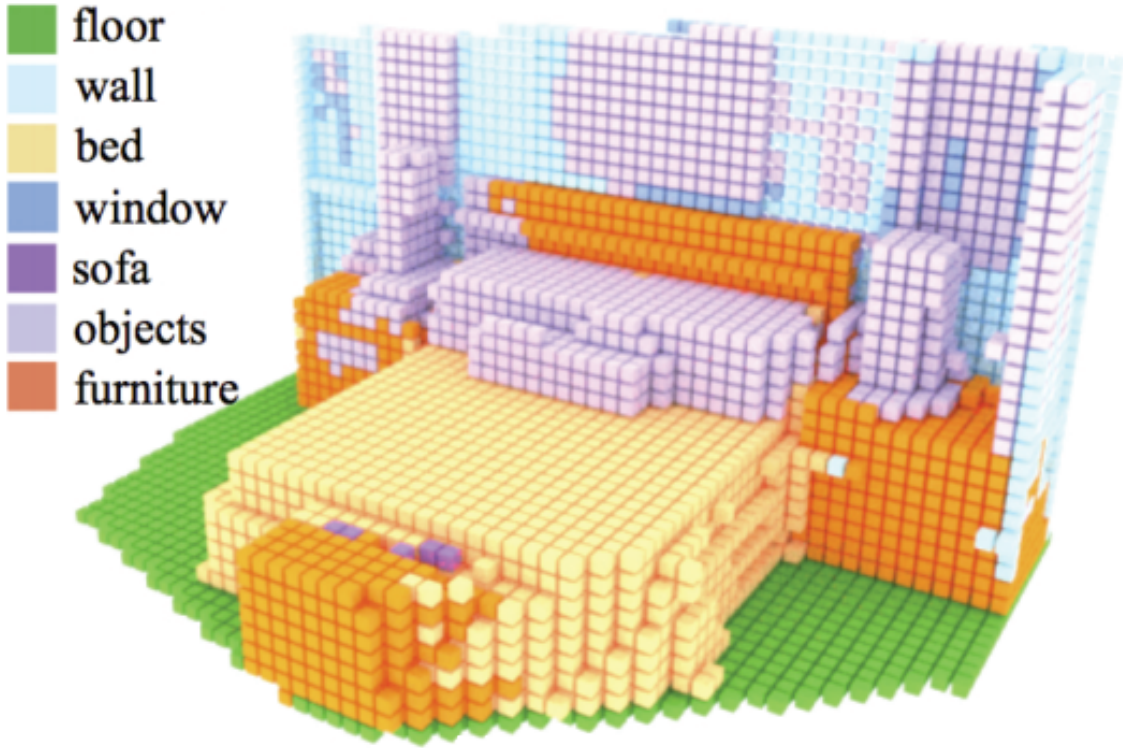
Volumetric semantic segmentation



Depth image



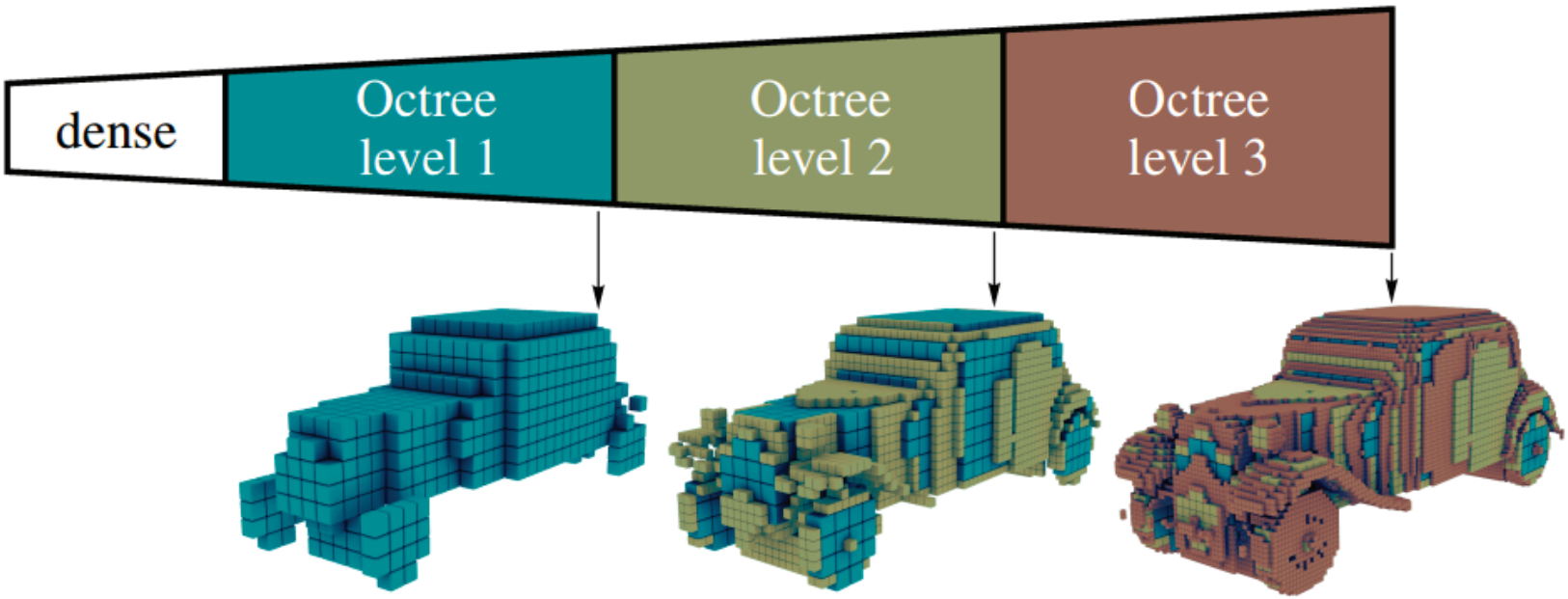
Visible surface



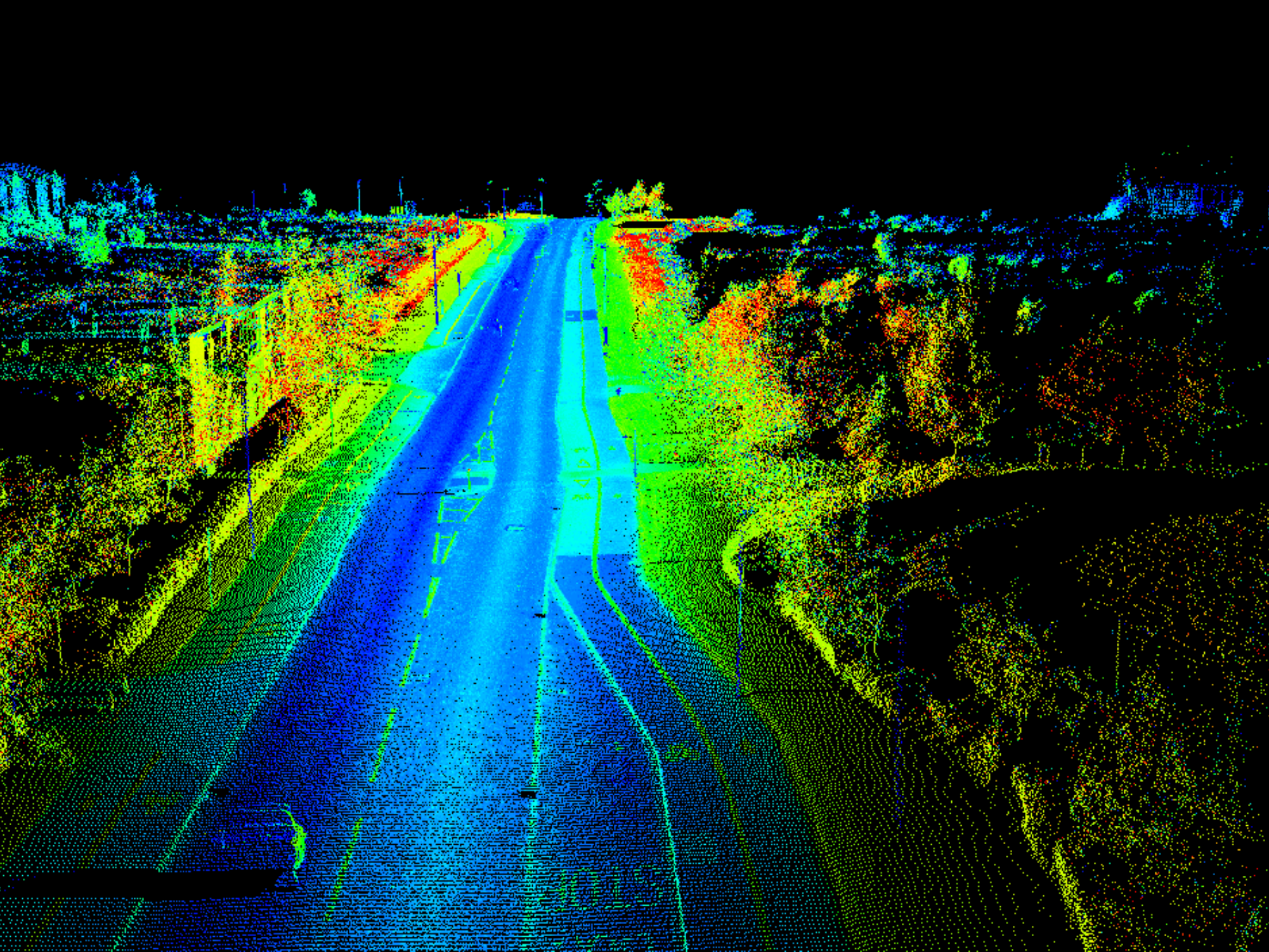
Semantic segmentation

Song et al. 2017 (slide credit: Niloy Mitra)

Efficient volumetric data structures



Tatarchenko, Dosovitskiy, Brox 2017



PointNet: learning on sets



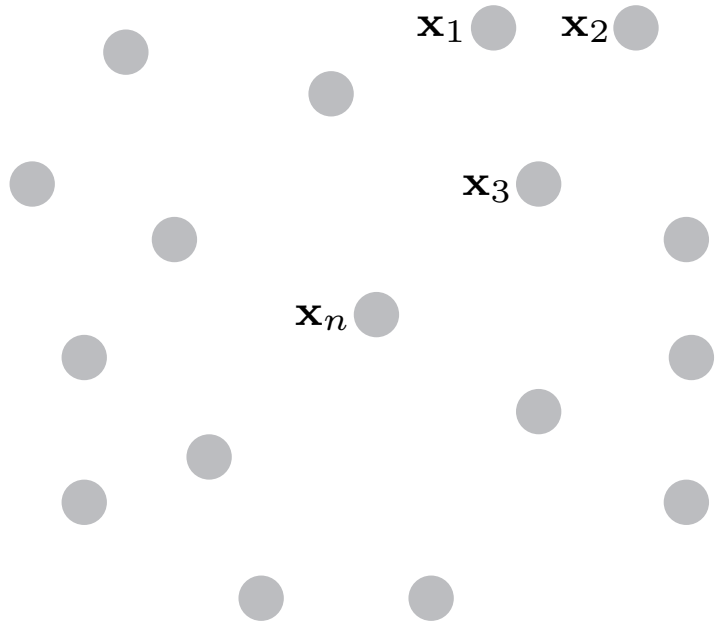
(Vinyals, Bengio, Kudlur 2015); Qi et al. 2017

PointNet: learning on sets

- Permutation-invariant function

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = f(\mathbf{x}_{\pi_1}, \ldots, \mathbf{x}_{\pi_n})$$

where $\mathbf{x}_i \in \mathbb{R}^d$ is feature at vertex i



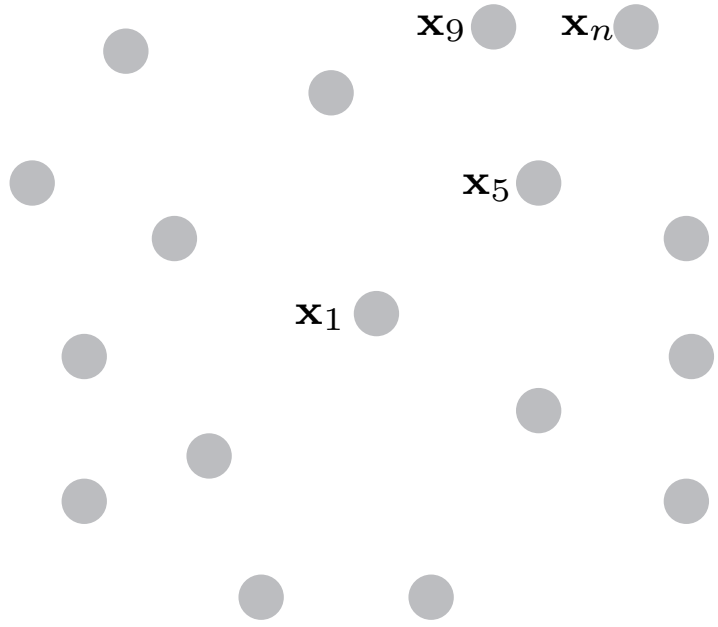
(Vinyals, Bengio, Kudlur 2015); Qi et al. 2017

PointNet: learning on sets

- Permutation-invariant function

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = f(\mathbf{x}_{\pi_1}, \ldots, \mathbf{x}_{\pi_n})$$

where $\mathbf{x}_i \in \mathbb{R}^d$ is feature at vertex i



(Vinyals, Bengio, Kudlur 2015); Qi et al. 2017

PointNet: learning on sets

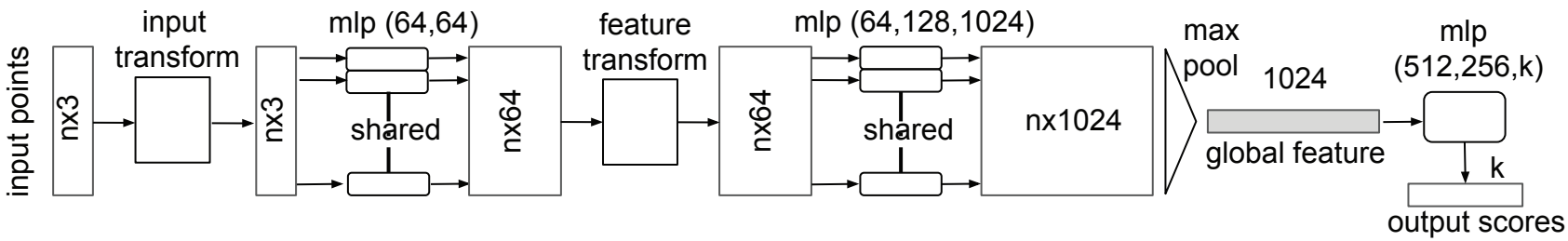


- Permutation-invariant function

  $$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = f(\mathbf{x}_{\pi_1}, \ldots, \mathbf{x}_{\pi_n})$$

  where $\mathbf{x}_i \in \mathbb{R}^d$ is feature at vertex i

- Shared function $h_{\mathbf{\Theta}}(\cdot)$ applied to each point + permutation-invariant aggregation ($\max$ or $\sum$)



(Vinyals, Bengio, Kudlur 2015); Qi et al. 2017

PointNet: learning on sets



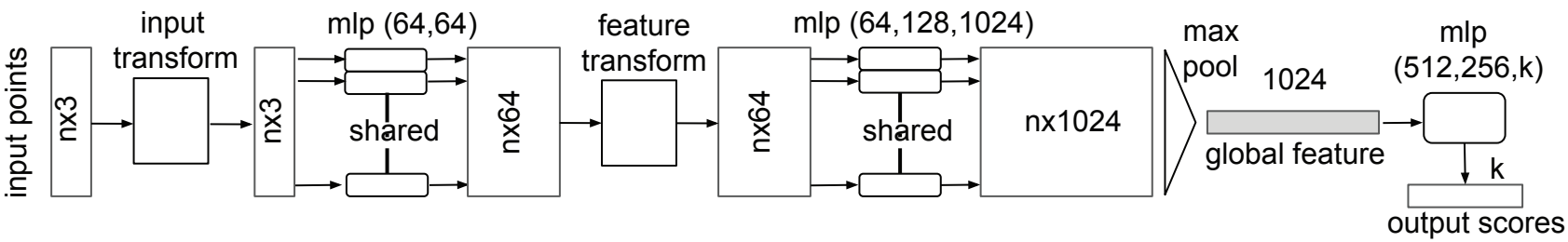
- Permutation-invariant function

  $$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = f(\mathbf{x}_{\pi_1}, \ldots, \mathbf{x}_{\pi_n})$$

  where $\mathbf{x}_i \in \mathbb{R}^d$ is feature at vertex i

- Shared function $h_{\mathbf{\Theta}}(\cdot)$ applied to each point + permutation-invariant aggregation ($\max$ or $\sum$)

- Spatial transformer units



(Vinyals, Bengio, Kudlur 2015); Qi et al. 2017

PointNet: learning on sets



- Permutation-invariant function

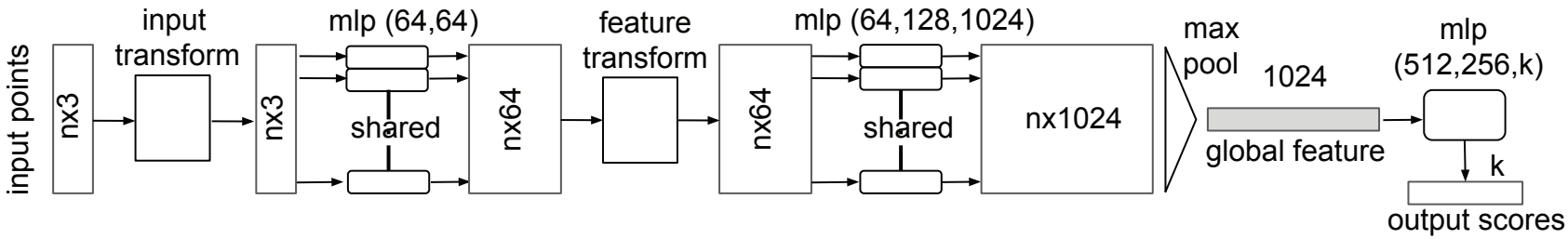
  $$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = f(\mathbf{x}_{\pi_1}, \ldots, \mathbf{x}_{\pi_n})$$

  where $\mathbf{x}_i \in \mathbb{R}^d$ is feature at vertex i

- Shared function $h_{\boldsymbol{\Theta}}(\cdot)$ applied to each point + permutation-invariant aggregation ($\max$ or $\sum$)

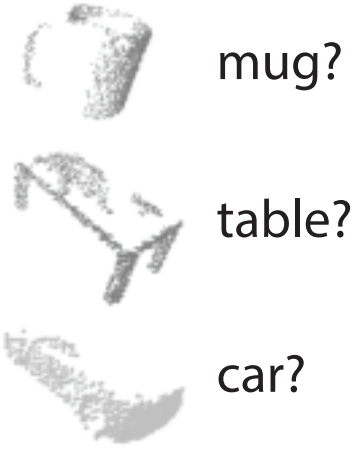
- Spatial transformer units

- Local grouping (PointNet++, PCPNet)



(Vinyals, Bengio, Kudlur 2015); Qi et al. 2017; Qi, Yi et al. 2017; Guerrero et al. 2018

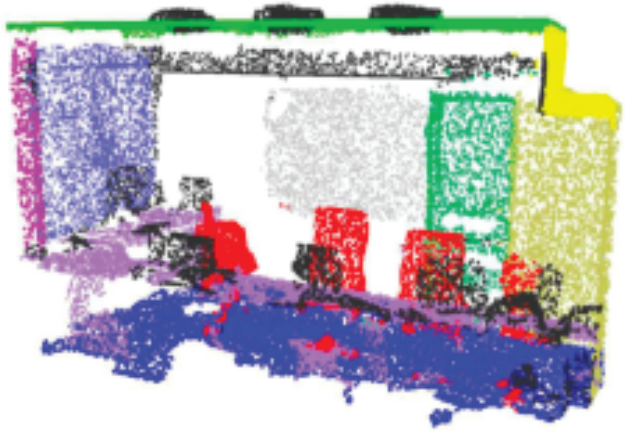
PointNet applications



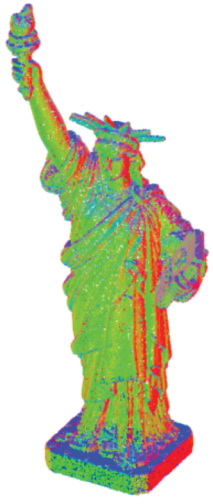
Object recognition



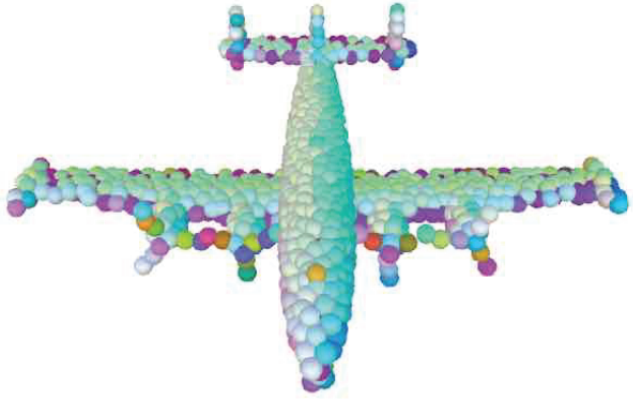
Part segmentation



Semantic segmentation



Curvature



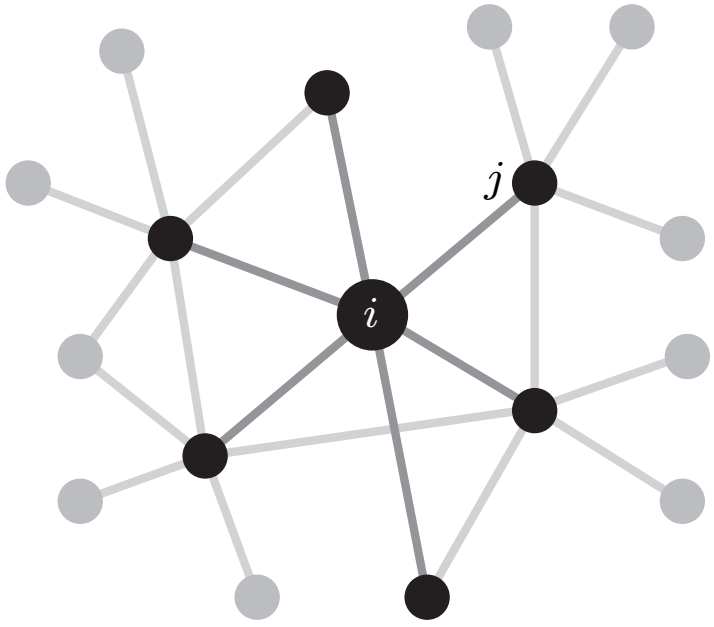
Normals



Point cloud generation

Graph-based edge convolution

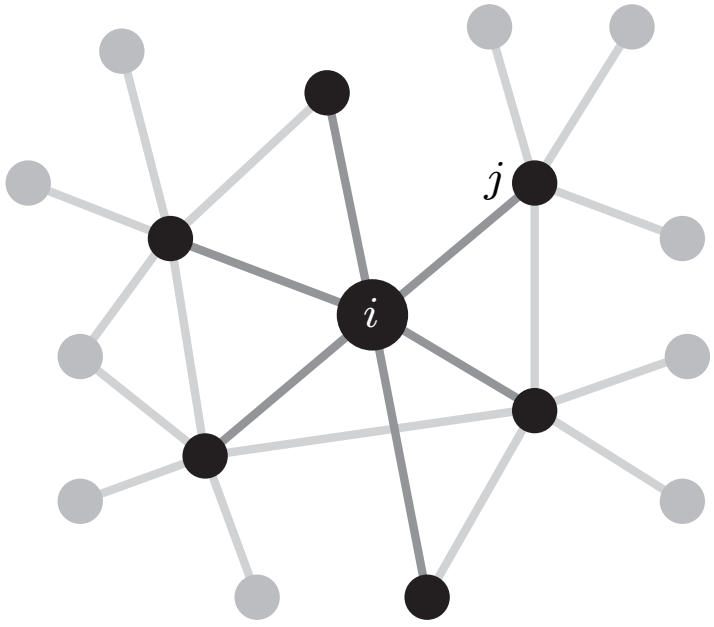
- Local neighborhood structure modeled as a graph



Wang et al. 2018

Graph-based edge convolution

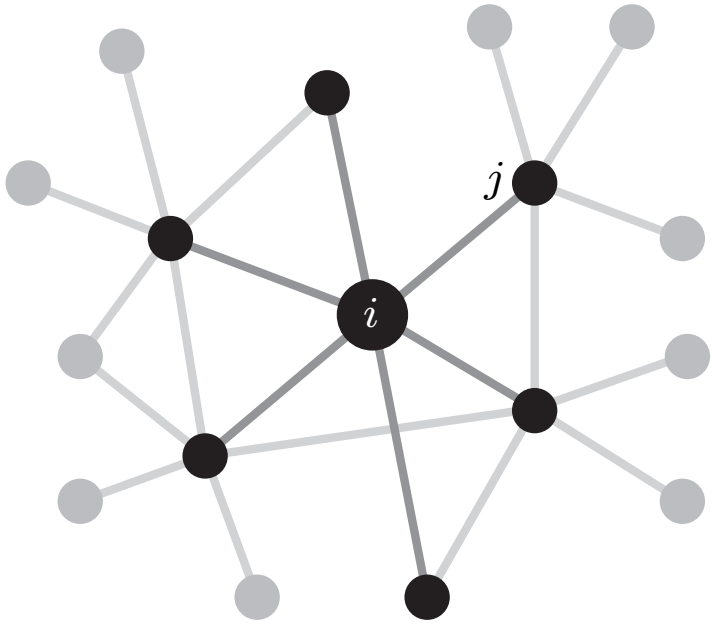
- Local neighborhood structure modeled as a graph
- Edge feature function $h_{\mathbf{\Theta}}(\cdot, \cdot)$ parametrized by $\mathbf{\Theta}$



Wang et al. 2018

Graph-based edge convolution

- Local neighborhood structure modeled as a graph
- Edge feature function $h_{\mathbf{\Theta}}(\cdot, \cdot)$ parametrized by $\mathbf{\Theta}$
- Permutation-invariant aggregation operator $\square$ (e.g. $\sum$ or $\max$) on the neighborhood of i

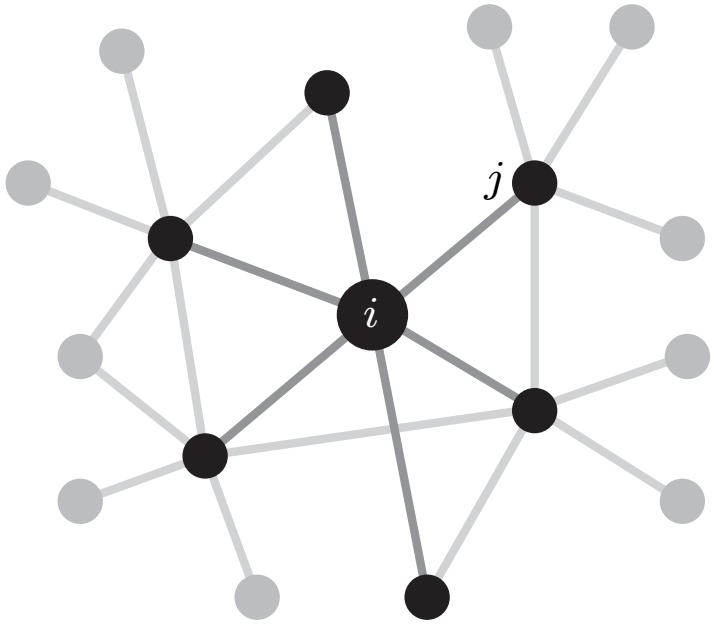


Wang et al. 2018

Graph-based edge convolution

- Local neighborhood structure modeled as a graph
- Edge feature function $h_{\mathbf{\Theta}}(\cdot, \cdot)$ parametrized by $\mathbf{\Theta}$
- Permutation-invariant aggregation operator $\square$ (e.g. $\sum$ or $\max$) on the neighborhood of i
- Edge convolution (EdgeConv)

$$\mathbf{x}_i' = \square_{j} h_{\mathbf{\Theta}}(\mathbf{x}_i, \mathbf{x}_j)$$

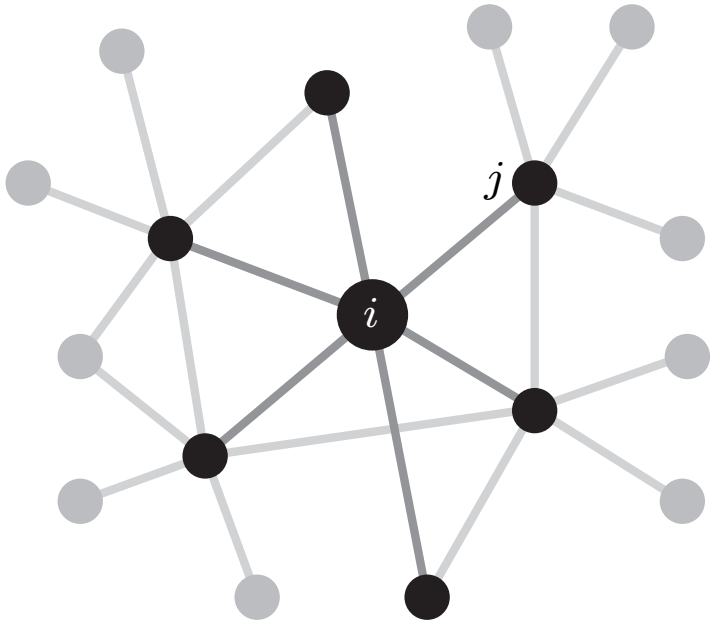


Wang et al. 2018

Graph-based edge convolution

- Local neighborhood structure modeled as a graph
- Edge feature function $h_{\mathbf{\Theta}}(\cdot, \cdot)$ parametrized by $\mathbf{\Theta}$
- Permutation-invariant aggregation operator $\square$ (e.g. $\sum$ or $\max$) on the neighborhood of i
- Edge convolution (EdgeConv)

$$\mathbf{x}_i' = \square_j h_{\mathbf{\Theta}}(\mathbf{x}_i, \mathbf{x}_j)$$



Learnable local (nonlinear) operator

Wang et al. 2018

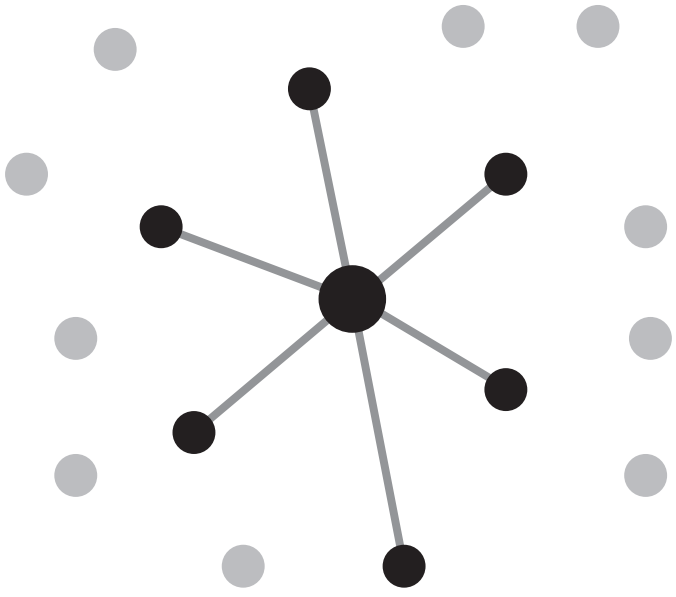
Particular cases

Method	**Aggregation** $\square$	**Edge feature** $h(\mathbf{x}_i, \mathbf{x}_j)$
Laplacian	$\sum$	$w_{ij}(\mathbf{x}_j - \mathbf{x}_i)$
PointNet[1]	–	$h(\mathbf{x}_i)$
PointNet++[2]	$\max$	$h(\mathbf{x}_i)$
MoNet[3]	$\sum$	$\sum_\ell g_\ell w_\ell(\mathbf{u}_{ij})\mathbf{x}_j$
PCNN[4]	$\sum$	$\sum_{\ell m} c(\mathbf{x}_i \cdot \mathbf{k}_{\ell m}) w_i q_{\mathbf{\Theta}_\ell}(\mathbf{x}_i, \mathbf{x}_j)$

Wang et al. 2018; [1]Qi et al. 2017; [2]Qi, Su et al. 2017; [3]Monti et al. 2017; [4]Atzmon et al. 2018

Dynamic Graph CNN (DynGCNN)

Construct k-NN graph in feature space and update it after each layer

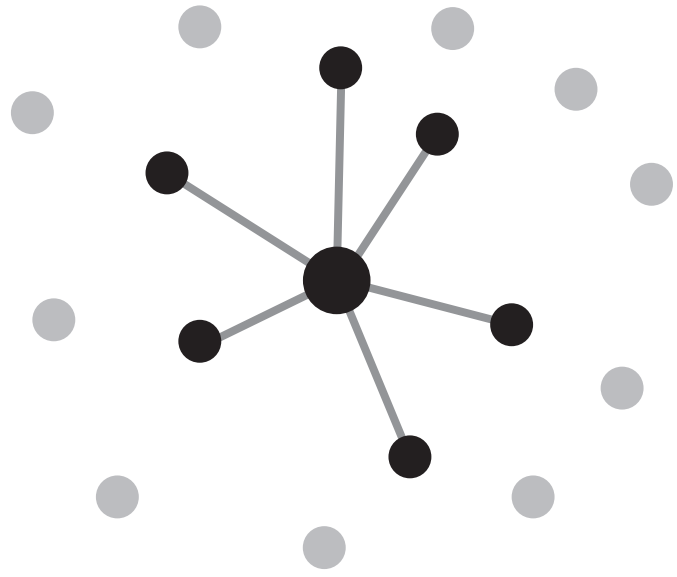


Layer l

Features $\mathbf{x}_1^{(l)}, \dots, \mathbf{x}_n^{(l)} \in \mathbb{R}^{d_l}$

k-NN graph $\mathcal{G}^{(l)}$

$h^{(l)} : \mathbb{R}^{d_l} \times \mathbb{R}^{d_l} \to \mathbb{R}^{d_{l+1}}$



Layer $l+1$

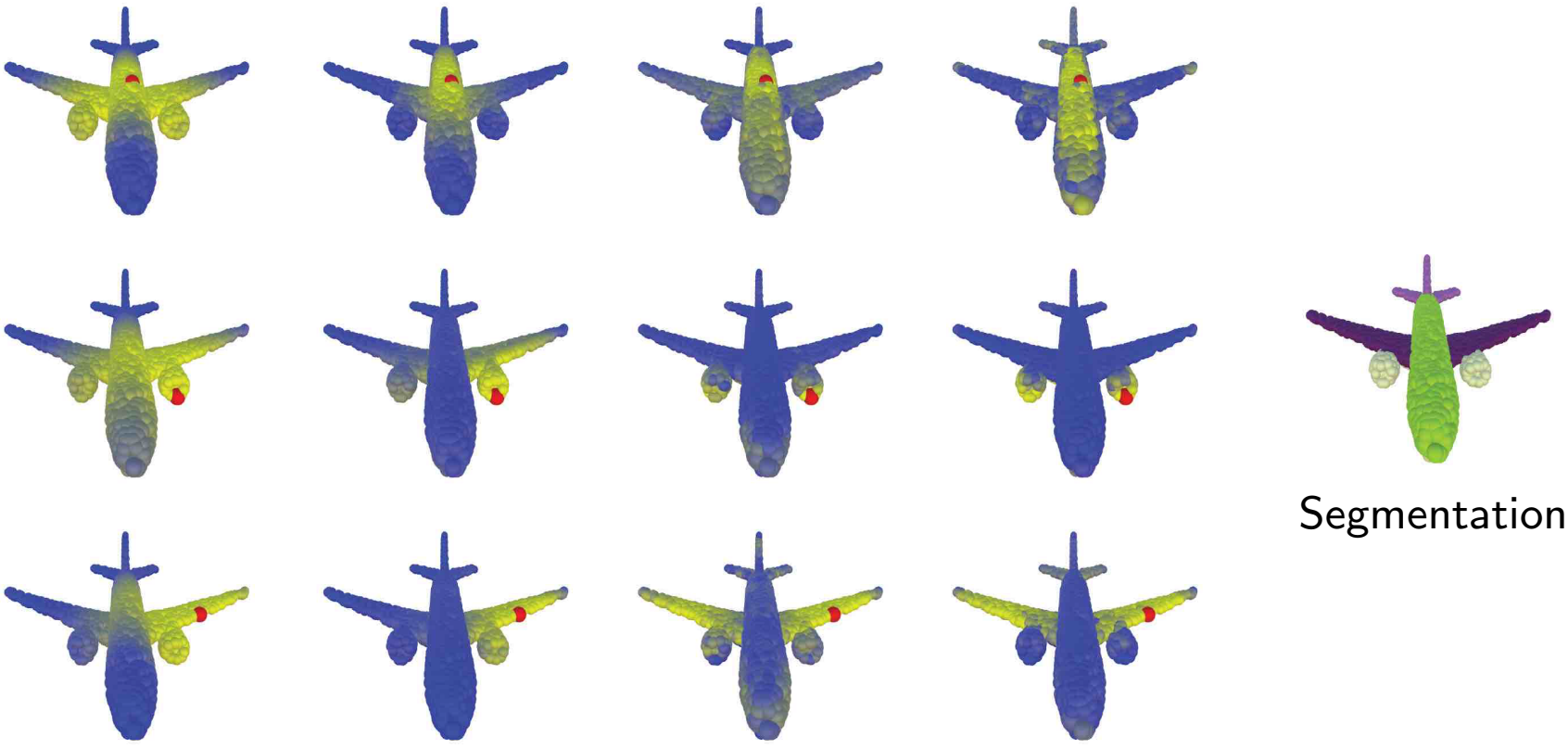
Features $\mathbf{x}_1^{(l+1)}, \dots, \mathbf{x}_n^{(l+1)} \in \mathbb{R}^{d_{l+1}}$

k-NN graph $\mathcal{G}^{(l+1)}$

$h^{(l+1)} : \mathbb{R}^{d_{l+1}} \times \mathbb{R}^{d_{l+1}} \to \mathbb{R}^{d_{l+2}}$

Wang et al. 2018

Learning semantic features



Left: Distance from red point in the feature space of different DynGCNN layers
Right: semantic segmentation results

Wang et al. 2018

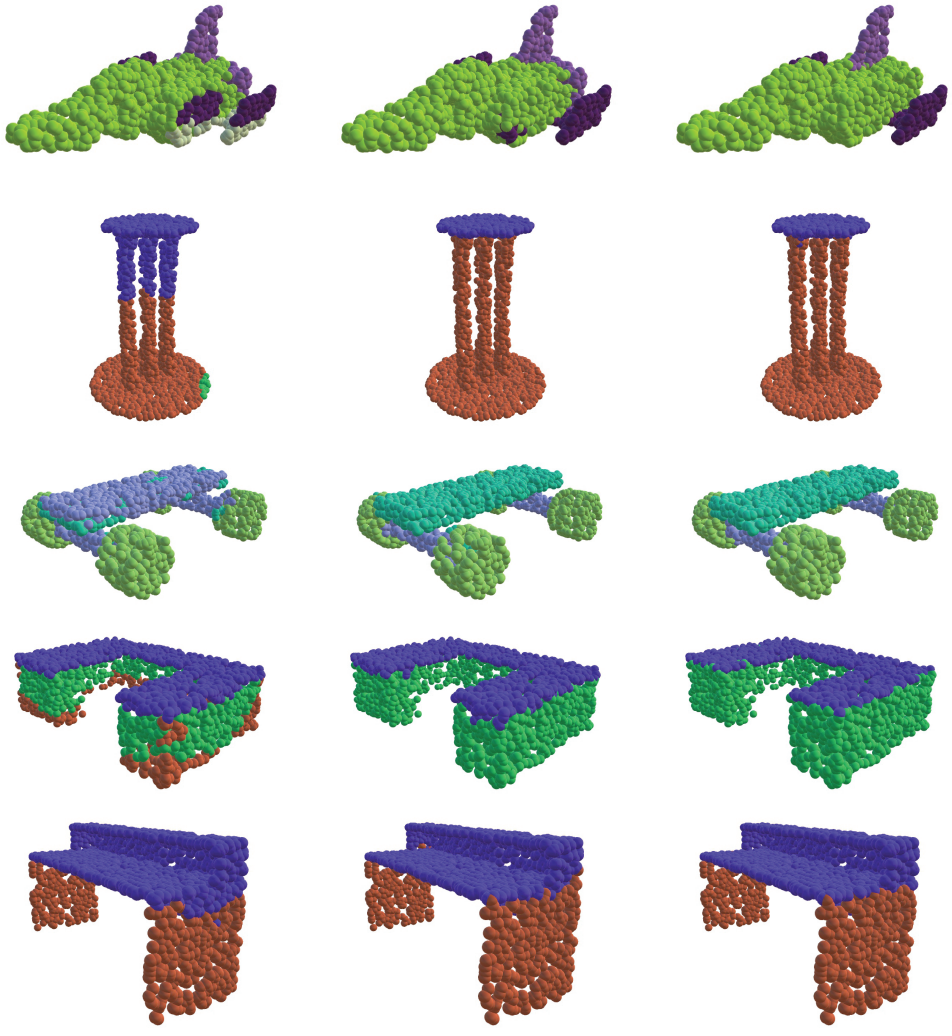
Shape classification (ModelNet40)

Method	Mean class accuracy	Overall accuracy
3DShapeNet[1]	77.3%	84.7%
VoxNet[2]	83.0%	85.9%
Subvolume[3]	86.0%	89.2%
ECC[4]	83.2%	87.4%
PointNet[5]	86.0%	89.2%
PointNet++[6]	–	90.7%
Kd-Net[7]	–	91.8%
DynGCNN (baseline) [8]	88.8%	91.2%
DynGCNN[8]	**90.2%**	**92.2%**

Classification accuracy of different methods on ModelNet40

Methods: [1]Wu et al. 2015; [2]Maturana et al. 2015; Qi et al. 2016; [4]Simonovsky, Komodakis 2017; [5]Qi et al. 2017; [6]Qi, Su et al. 2017; [7]Klokov, Lempitsky 2017; [8]Wang et al. 2018; data: Wu et al. 2015 (ModelNet)

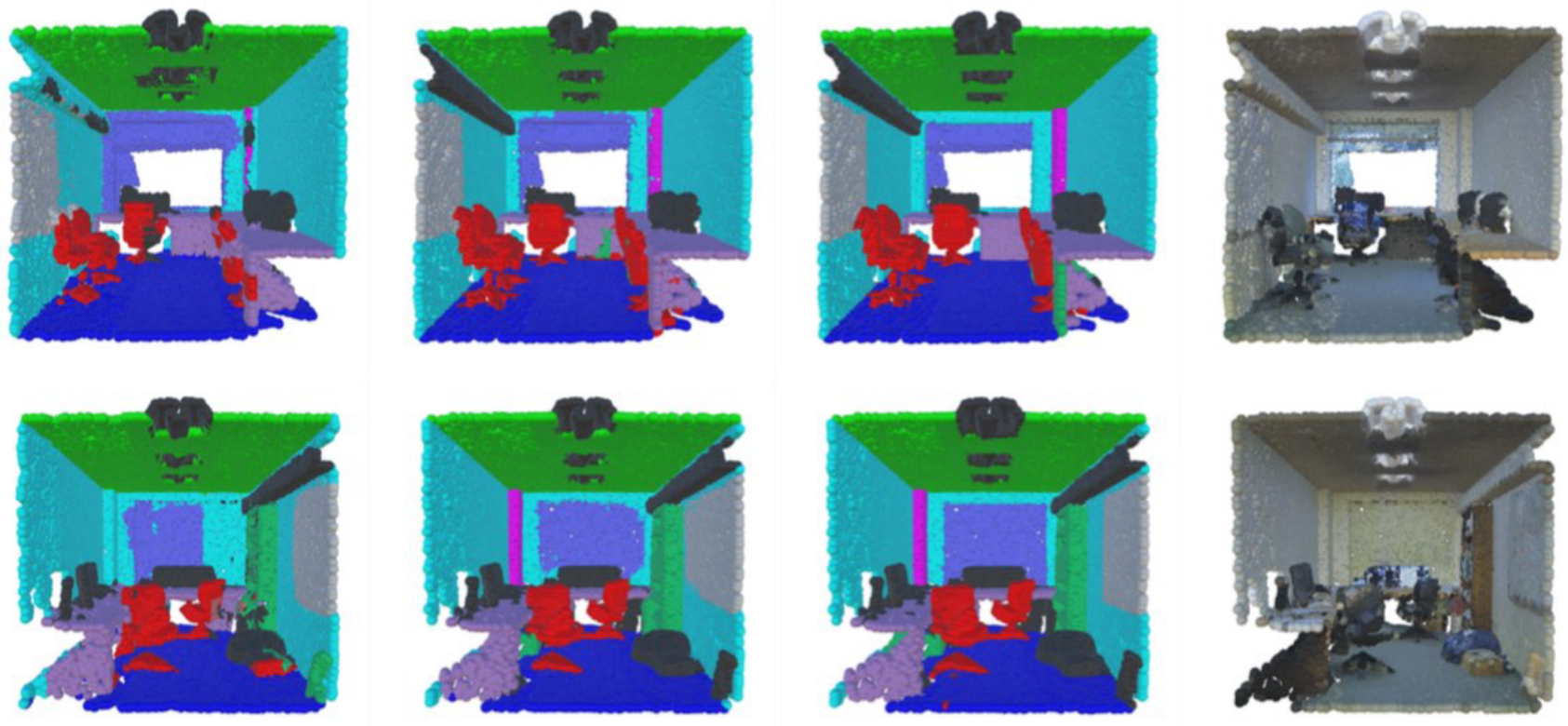
Semantic segmentation: synthetic (ShapeNet)



PointNet **DynGCNN** Groundtruth

Methods: Qi et al. 2017 (PointNet); Wang et al. 2018 (DynGCNN); data: Yi et al. 2016 (ShapeNet)

Semantic segmentation: indoor scans (S3DIS)



PointNet **DynGCNN** Groundtruth Scene

Results of semantic segmentation of point cloud+RGB data using different architectures

Methods: Qi et al. 2017 (PointNet); Wang et al. 2018 (DynGCNN); data: Armeni et al. 2016 (S3DIS)

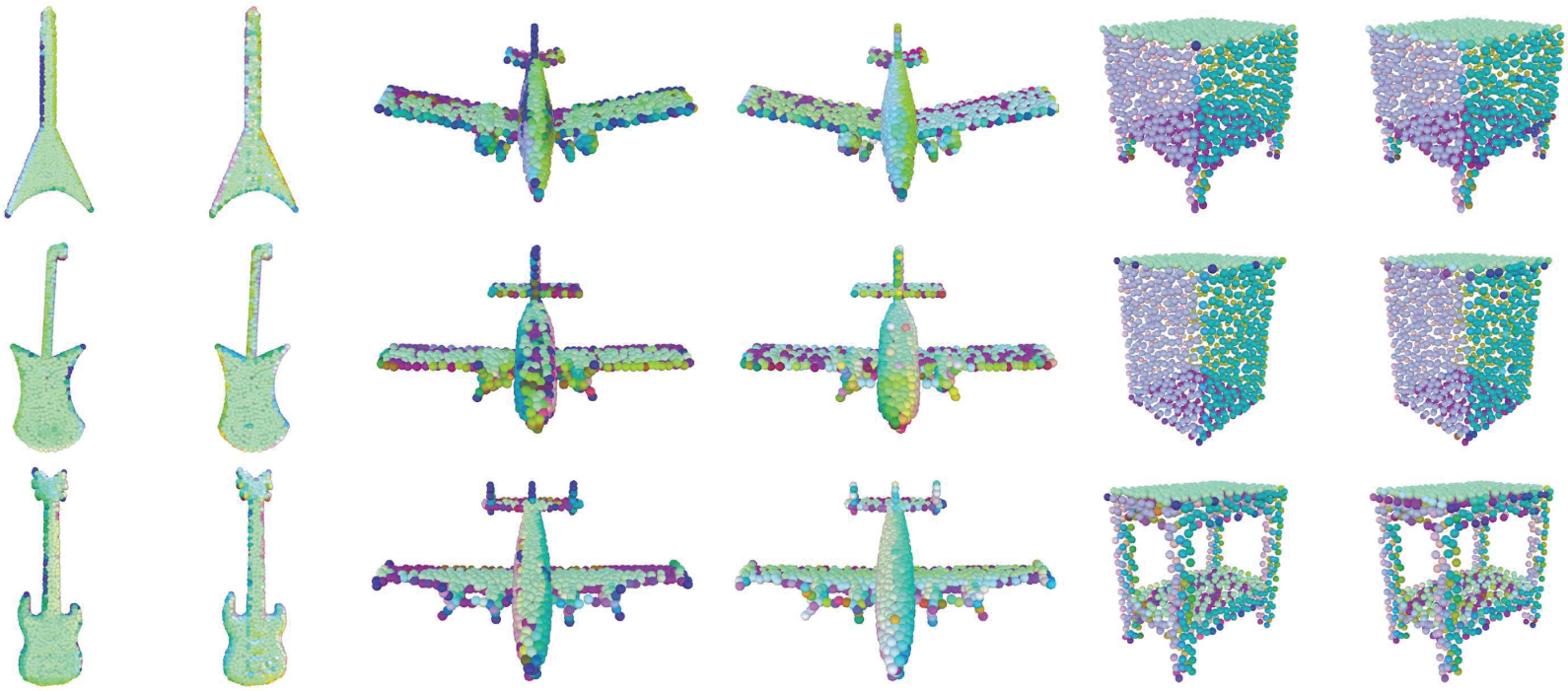
Shape segmentation: indoor scans (S3DIS)

Method	Mean IoU	Overall accuracy
PointNet (Baseline)[1]	20.1%	53.2%
PointNet[1]	47.6%	78.5%
MS + CU(2)[2]	47.8%	79.2%
G + RCU[2]	49.7%	81.1%
DynGCNN[3]	**56.1%**	**84.1%**

S3DIS indoor scene semantic segmentation accuracy

Methods: [1]Qi et al. 2017; [2]Engelmann et al. 2017 [3]Wang et al. 2018; data: Armeni et al. 2016 (S3DIS)

Surface normal prediction



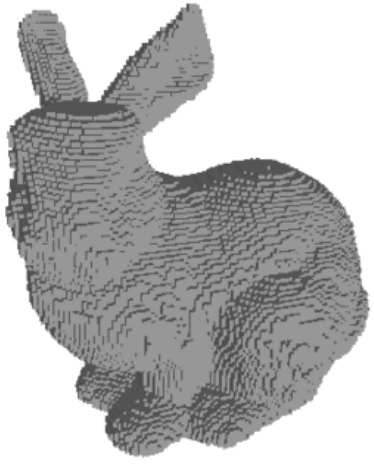
Surface normal predicted using DynGCNN (odd columns) and groundtruth (even columns). Normal direction is color-coded

Wang et al. 2018; data: Wu et al. 2015 (ModelNet)

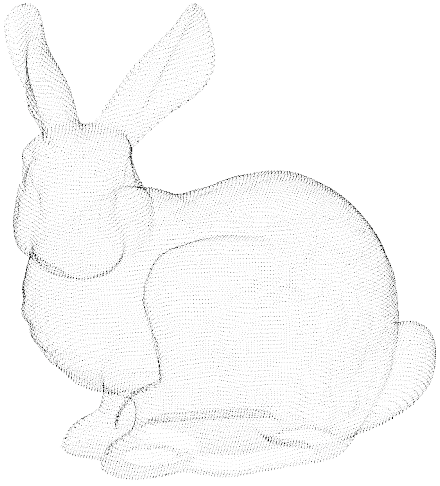
Shape representation



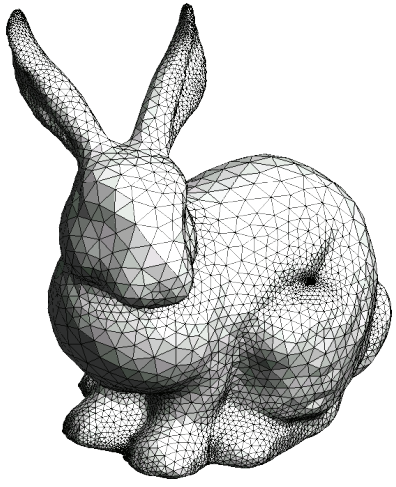
Image-based



Volumetric



Point-based



Surface-based

Extrinsic vs Intrinsic CNNs

Extrinsic

Intrinsic

Different formulations of non-Euclidean CNNs

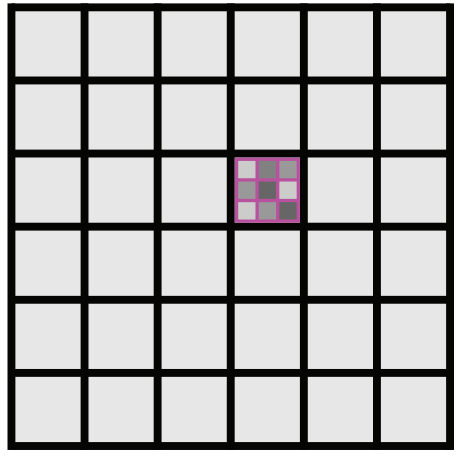


Spectral domain



Spatial domain





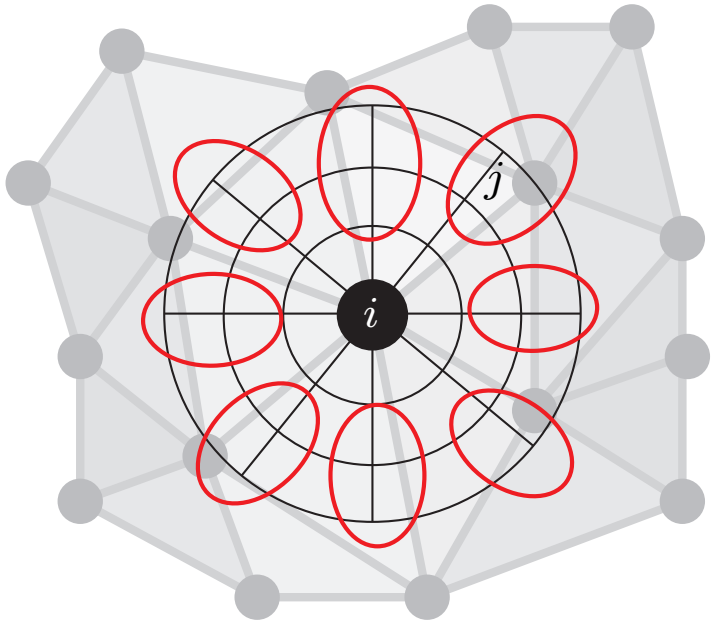
Parametric domain

Convolution on meshes

- Local system of coordinates $\mathbf{u}_{ij}$ around i (e.g. geodesic polar)

- Local weights $w_1(\mathbf{u}), \dots, w_L(\mathbf{u})$ w.r.t. $\mathbf{u}$, e.g. Gaussians

  $w_\ell = \exp\left(-(\mathbf{u} - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}_\ell^{-1} (\mathbf{u} - \boldsymbol{\mu}_\ell)\right)$



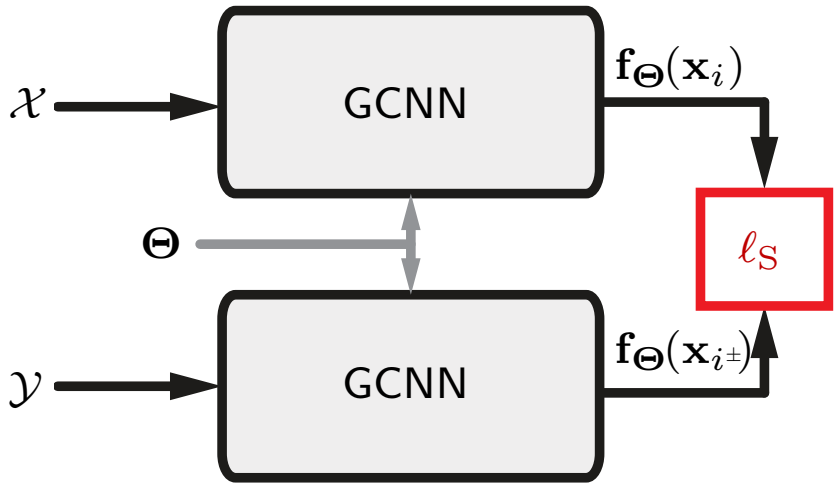
- Spatial convolution with filter g

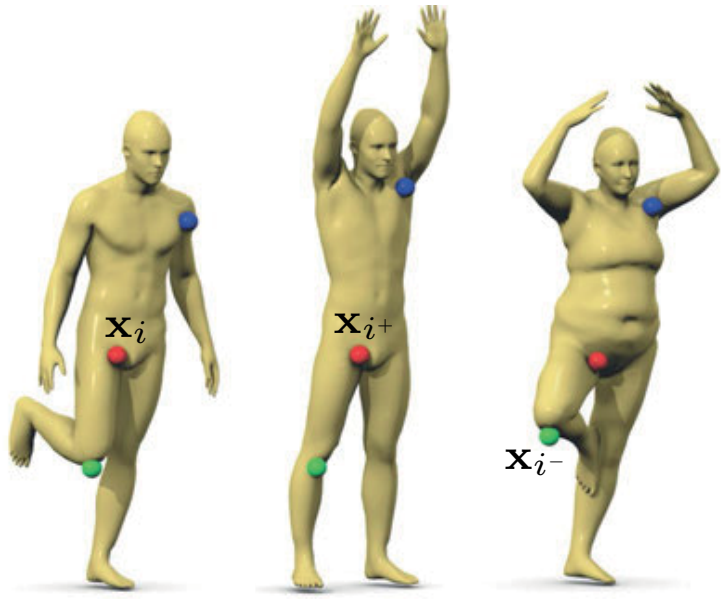
$$\mathbf{x}'_i \propto \sum_{\ell=1}^{L} g_\ell \underbrace{\sum_{j=1}^{n} w_\ell(\mathbf{u}_{ij}) \mathbf{x}_j}_{\text{patch operator}}$$

where $\mathbf{x}_i \in \mathbb{R}^d$ is feature at vertex i

Masci et al. 2015; Boscaini et al. 2016; Monti et al. 2017

Learning local descriptors with intrinsic CNNa





Training set	positive (i, i^+) and negative (i, i^-) pairs of points
Siamese net	two net instances with shared parameters Θ
Poitwise feature cost	$\ell_S(\Theta) = \gamma \sum_{i,i^+} \|\mathbf{f}_\Theta(\mathbf{x}_i) - \mathbf{f}_\Theta(\mathbf{x}_{i^+})\|_2^2$
	$+ (1-\gamma) \sum_{i,i^-} \left[\mu - \|\mathbf{f}_\Theta(\mathbf{x}_i) - \mathbf{f}_\Theta(\mathbf{x}_{i^-})\|_2^2\right]_+$

Masci et al. 2015

HKS descriptor



Distance in the space of local Heat Kernel Signature (HKS) features
(shown is distance from a point on the shoulder marked in white)

Descriptor: Sun, Ovsjanikov, Guibas 2009 (HKS); data: Bronstein, Bronstein, Kimmel 2008 (TOSCA); Anguelov et al. 2005 (SCAPE); Bogo et al. 2014 (FAUST)

WKS descriptor



Distance in the space of local Wave Kernel Signature (WKS) features
(shown is distance from a point on the shoulder marked in white)

Descriptor: Aubry, Schlickewei, Cremers 2011 (WKS); data: Bronstein, Bronstein, Kimmel 2008 (TOSCA); Anguelov et al. 2005 (SCAPE); Bogo et al. 2014 (FAUST)

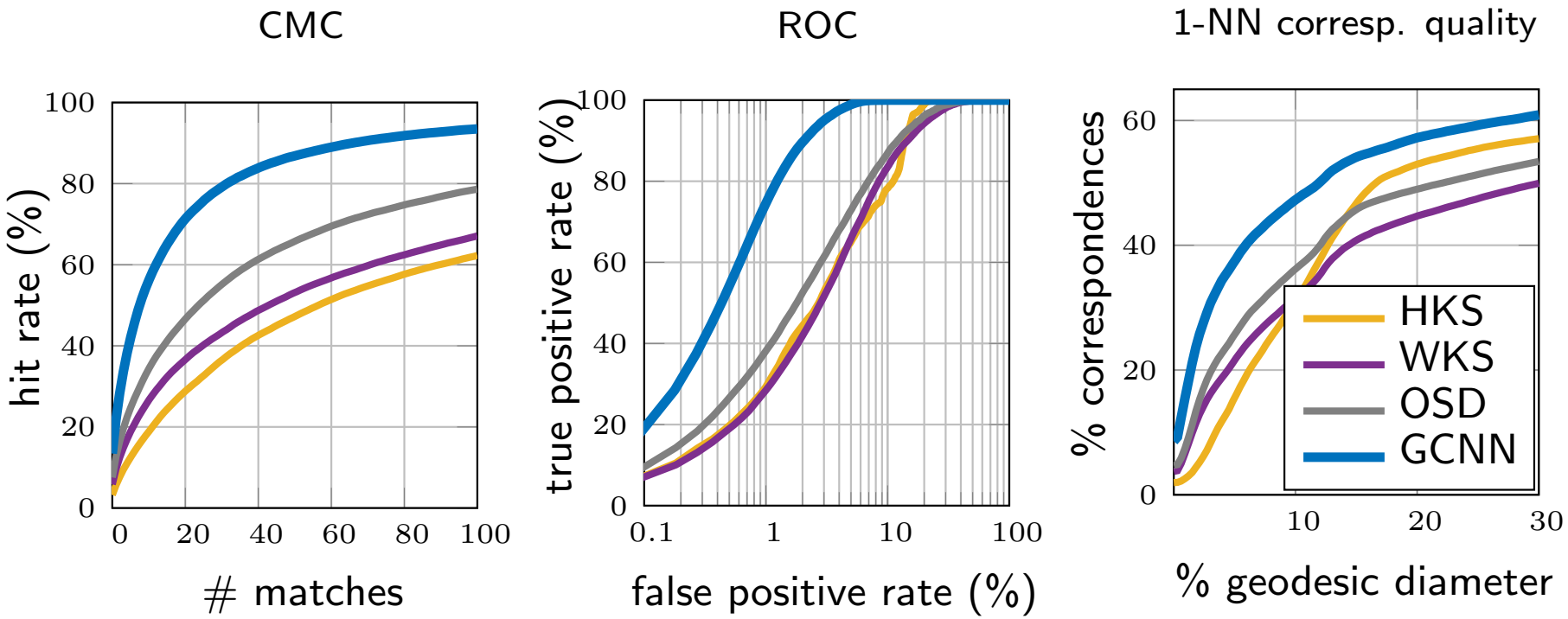
Descriptor learned with GCNN



Distance in the space of local GCNN features
(shown is distance from a point on the shoulder marked in white)

Descriptor: Masci et al. 2015 (GCNN); data: Bronstein, Bronstein, Kimmel 2008 (TOSCA); Anguelov et al. 2005 (SCAPE); Bogo et al. 2014 (FAUST)

Descriptor quality comparison

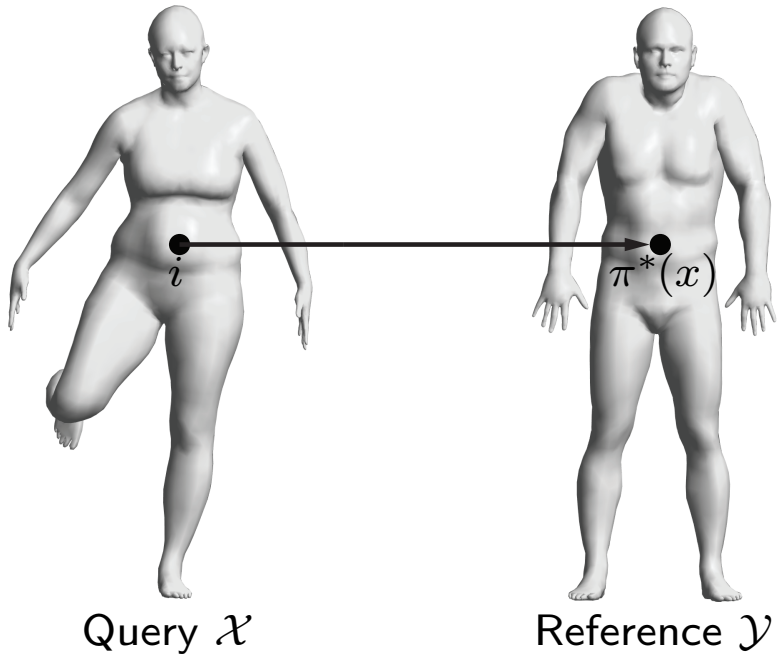


Descriptor performance using symmetric Princeton benchmark
(training and testing: disjoint subsets of FAUST)

Methods: Sun et al. 2009 (HKS); Aubry et al. 2011 (WKS); Litman, B 2014 (OSD); Masci et al. 2015 (GCNN); data: Bogo et al. 2014 (FAUST); benchmark: Kim et al. 2011

Learning deformation-invariant correspondence

- Groundtruth correspondence $\pi^* : \mathcal{X} \to \mathcal{Y}$ from query shape $\mathcal{X}$ to some reference shape $\mathcal{Y}$
- Correspondence = label each query vertex $i \in \{1, \ldots, n\}$ as reference vertex $\pi_i \in \{1, \ldots, m\}$



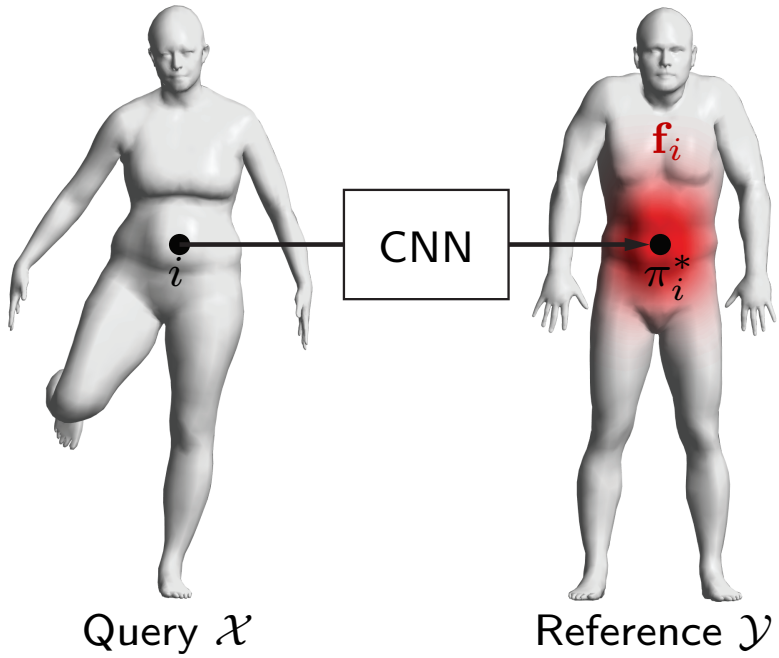
Rodolà et al. 2014; Masci et al. 2015

Learning deformation-invariant correspondence

- Groundtruth correspondence $\pi^* : \mathcal{X} \to \mathcal{Y}$ from query shape $\mathcal{X}$ to some reference shape $\mathcal{Y}$
- Correspondence = label each query vertex $i \in \{1, \ldots, n\}$ as reference vertex $\pi_i \in \{1, \ldots, m\}$
- Net output at i after softmax layer

$$\mathbf{f}_{\boldsymbol{\Theta}}(\mathbf{x}_i) = (f_{i1}, \ldots, f_{im})$$

$=$ probability distribution on $\mathcal{Y}$



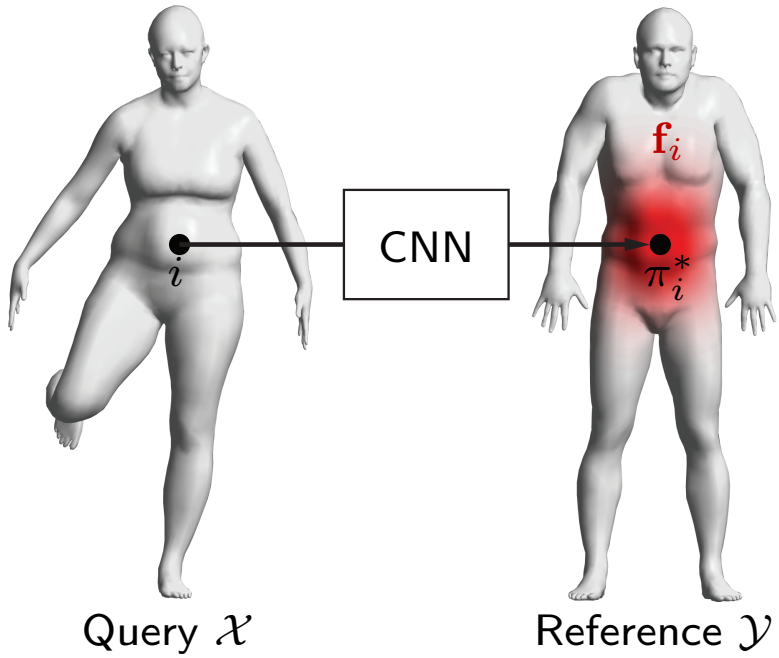
Rodolà et al. 2014; Masci et al. 2015

Learning deformation-invariant correspondence

- Groundtruth correspondence $\pi^* : \mathcal{X} \to \mathcal{Y}$ from query shape $\mathcal{X}$ to some reference shape $\mathcal{Y}$
- Correspondence = label each query vertex $i \in \{1, \dots, n\}$ as reference vertex $\pi_i \in \{1, \dots, m\}$
- Net output at i after softmax layer

$$\mathbf{f}_{\boldsymbol{\Theta}}(\mathbf{x}_i) = (f_{i1}, \dots, f_{im})$$

$=$ probability distribution on $\mathcal{Y}$

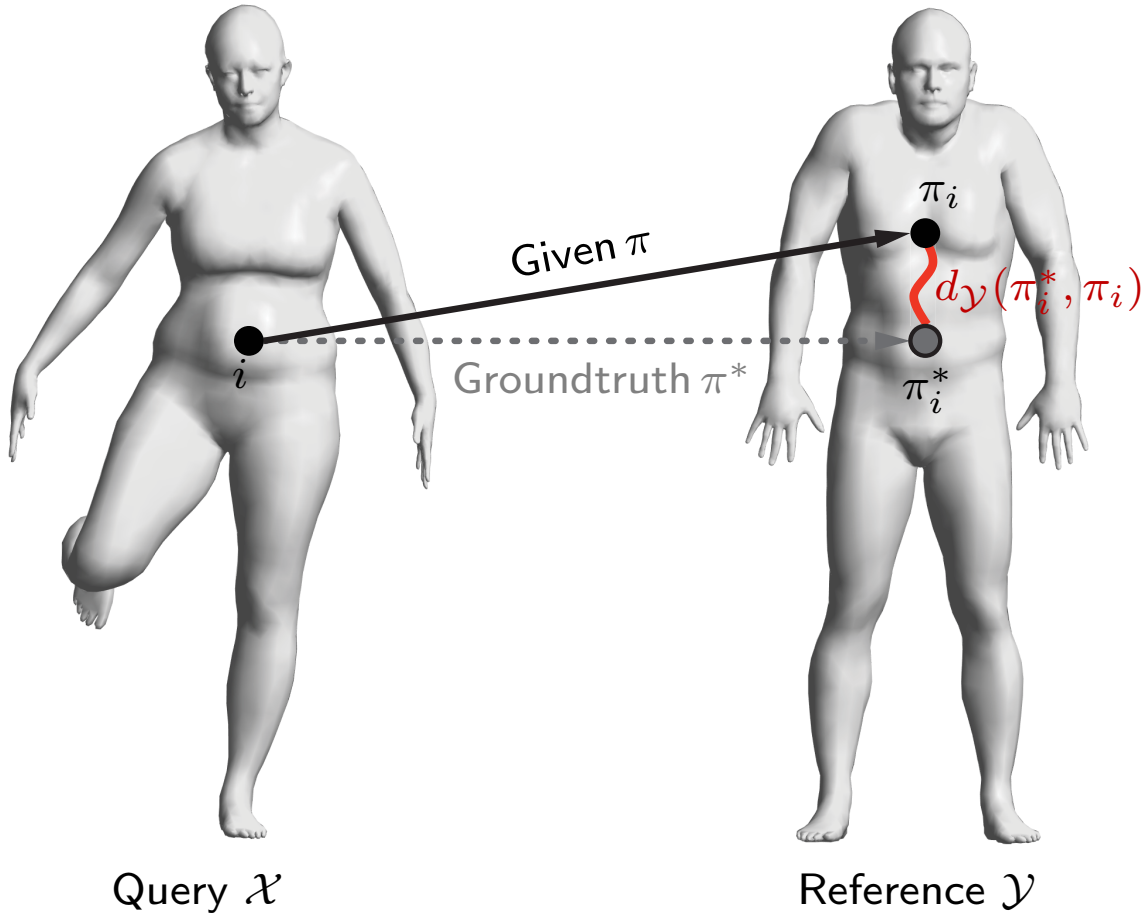


Minimize on training set the cross entropy between groundtruth correspondence and output probability distribution w.r.t. net parameters $\boldsymbol{\Theta}$

$$\min_{\boldsymbol{\Theta}} \sum_{i=1}^{n} H(\boldsymbol{\delta}_{\pi_i^*}, \mathbf{f}_{\boldsymbol{\Theta}}(\mathbf{x}_i))$$

Rodolà et al. 2014; Masci et al. 2015

Correspondence evaluation: Princeton benchmark

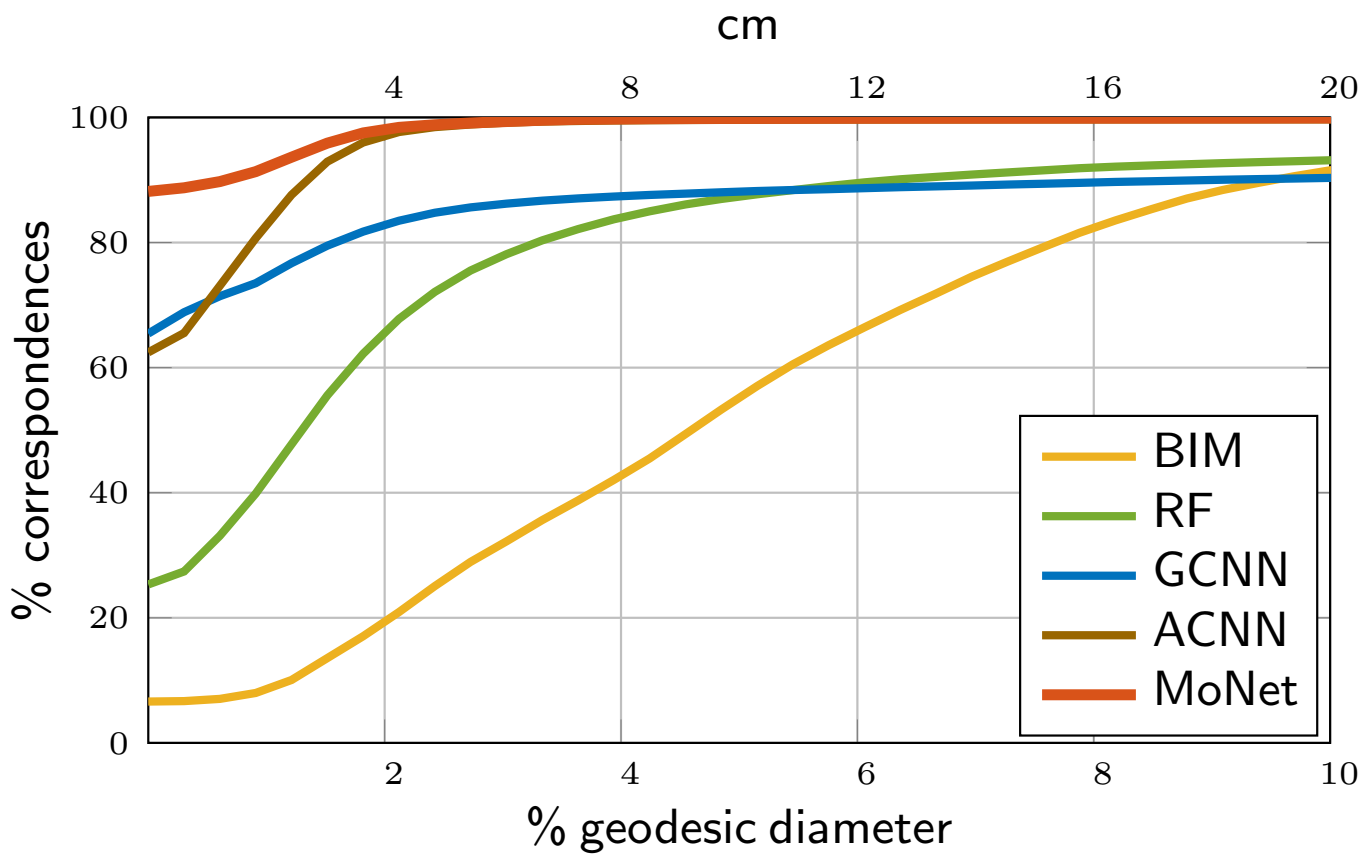


Pointwise correspondence error = geodesic distance from the groundtruth

$$\epsilon_i = d_{\mathcal{Y}}(\pi_i^*, \pi_i)$$

Kim et al. 2011

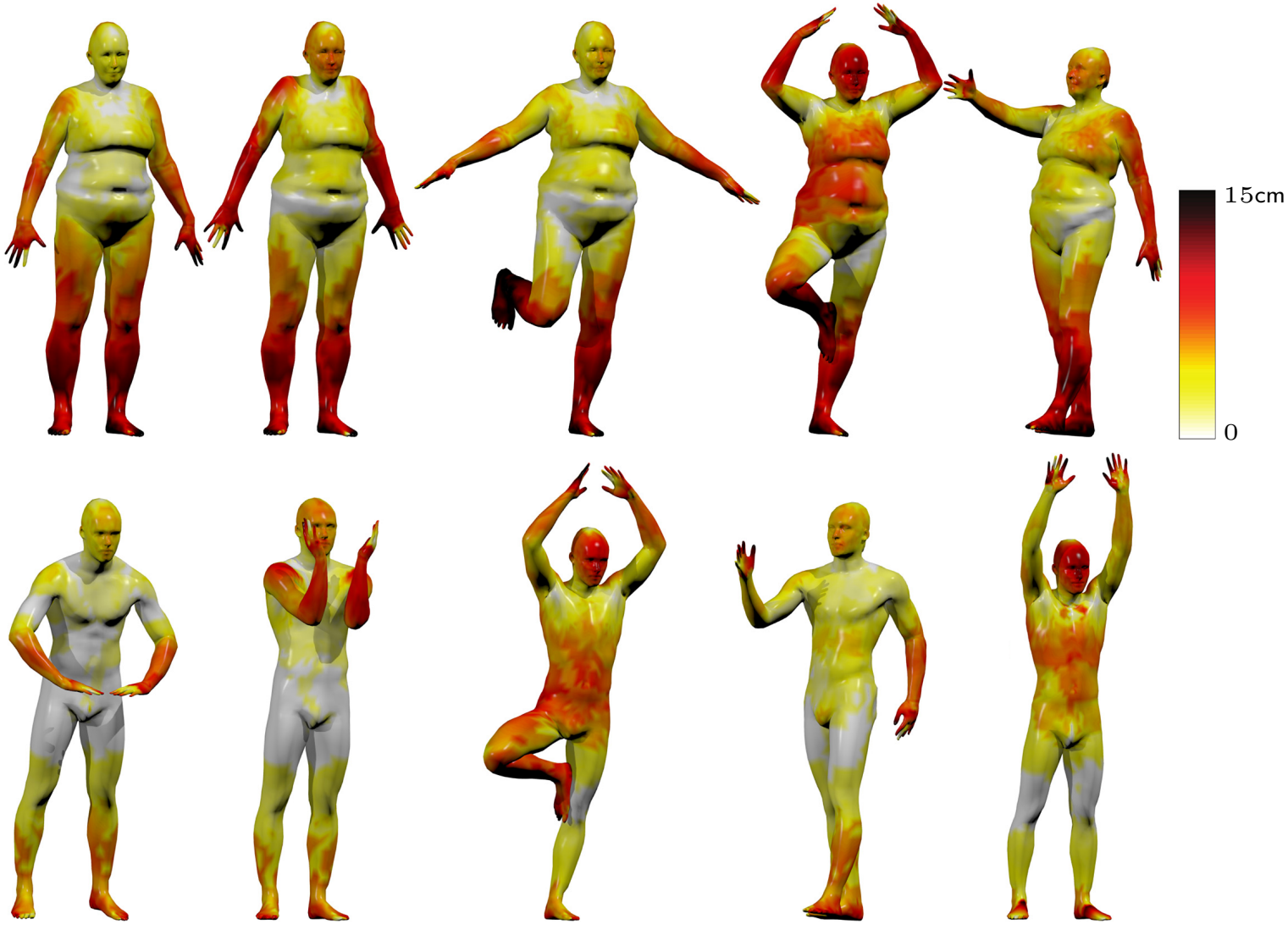
Correspondence quality comparison



Correspondence evaluated using asymmetric Princeton benchmark
(training and testing: disjoint subsets of FAUST)

Methods: Kim et al. 2011 (BIM); Rodolà et al. 2014 (RF); Boscaini et al. 2015 (ADD); Masci et al. 2015 (GCNN); Boscaini et al. 2016 (ACNN); Monti et al. 2016 (MoNet); data: Bogo et al. 2014 (FAUST); benchmark: Kim et al. 2011

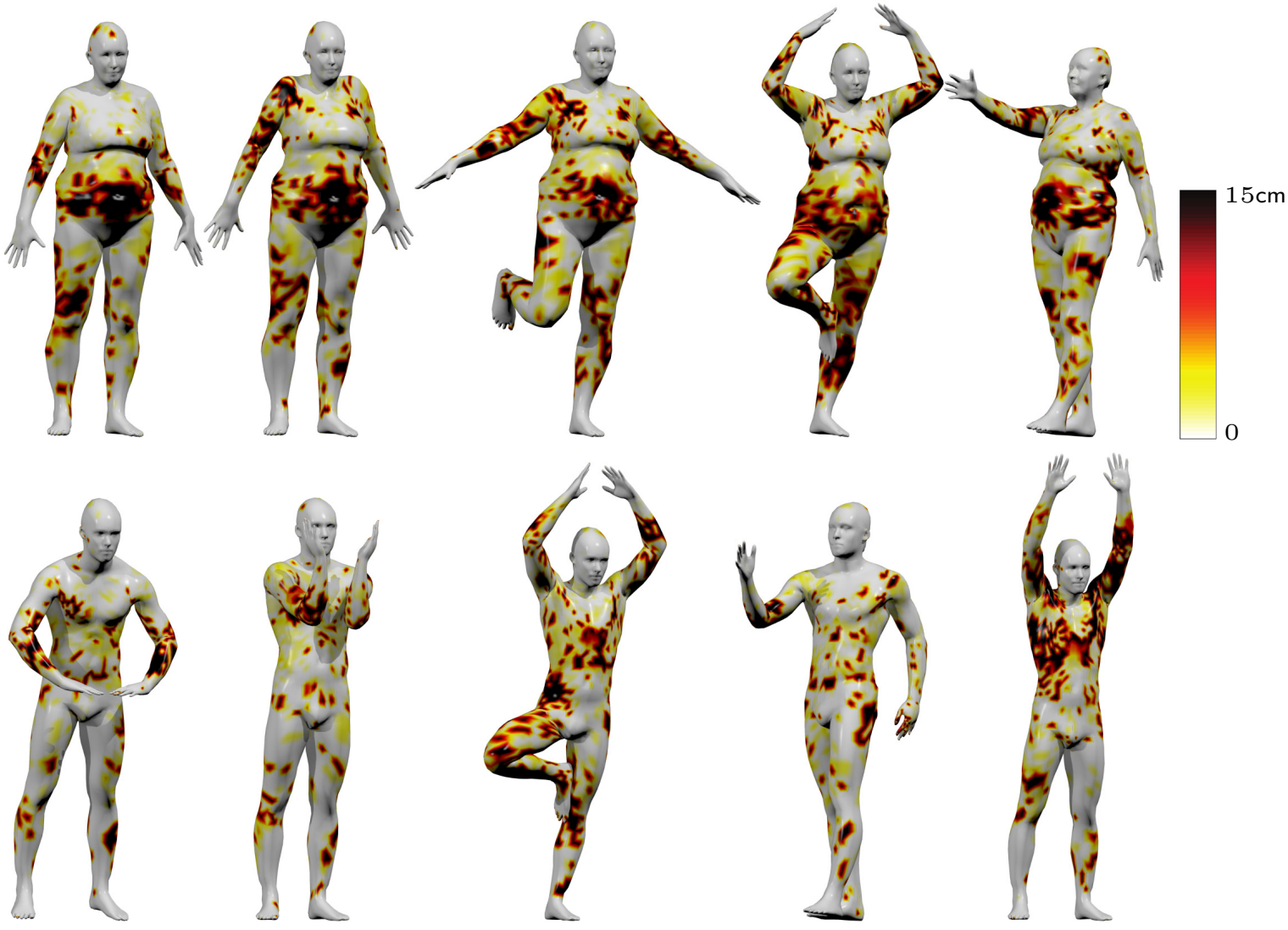
Shape correspondence error: Blended Intrinsic Map



Pointwise correspondence error (geodesic distance from groundtruth)

Kim, Lipman, Funkhouser 2011

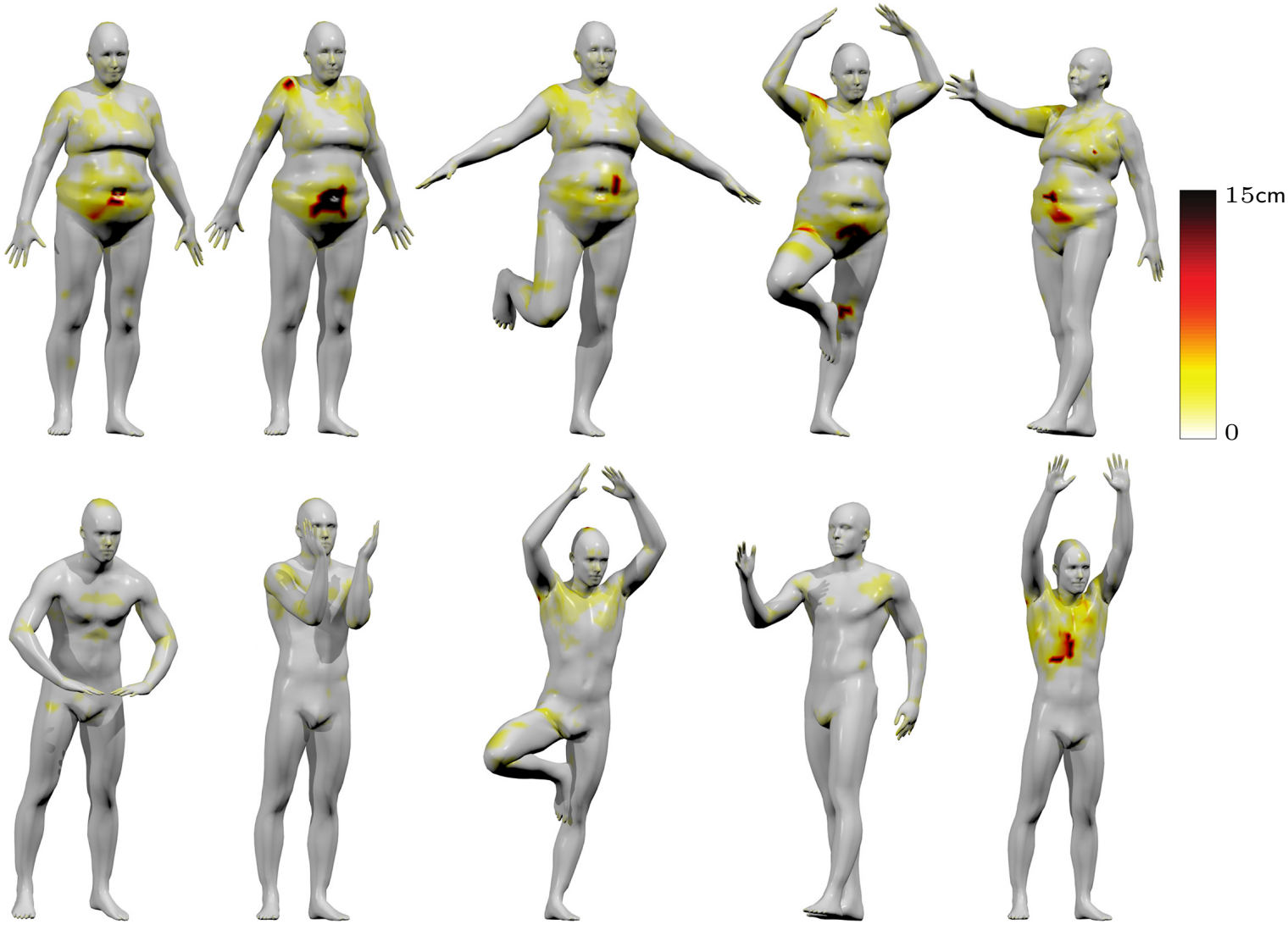
Shape correspondence error: Geodesic CNN



Pointwise correspondence error (geodesic distance from groundtruth)

Masci et al. 2015

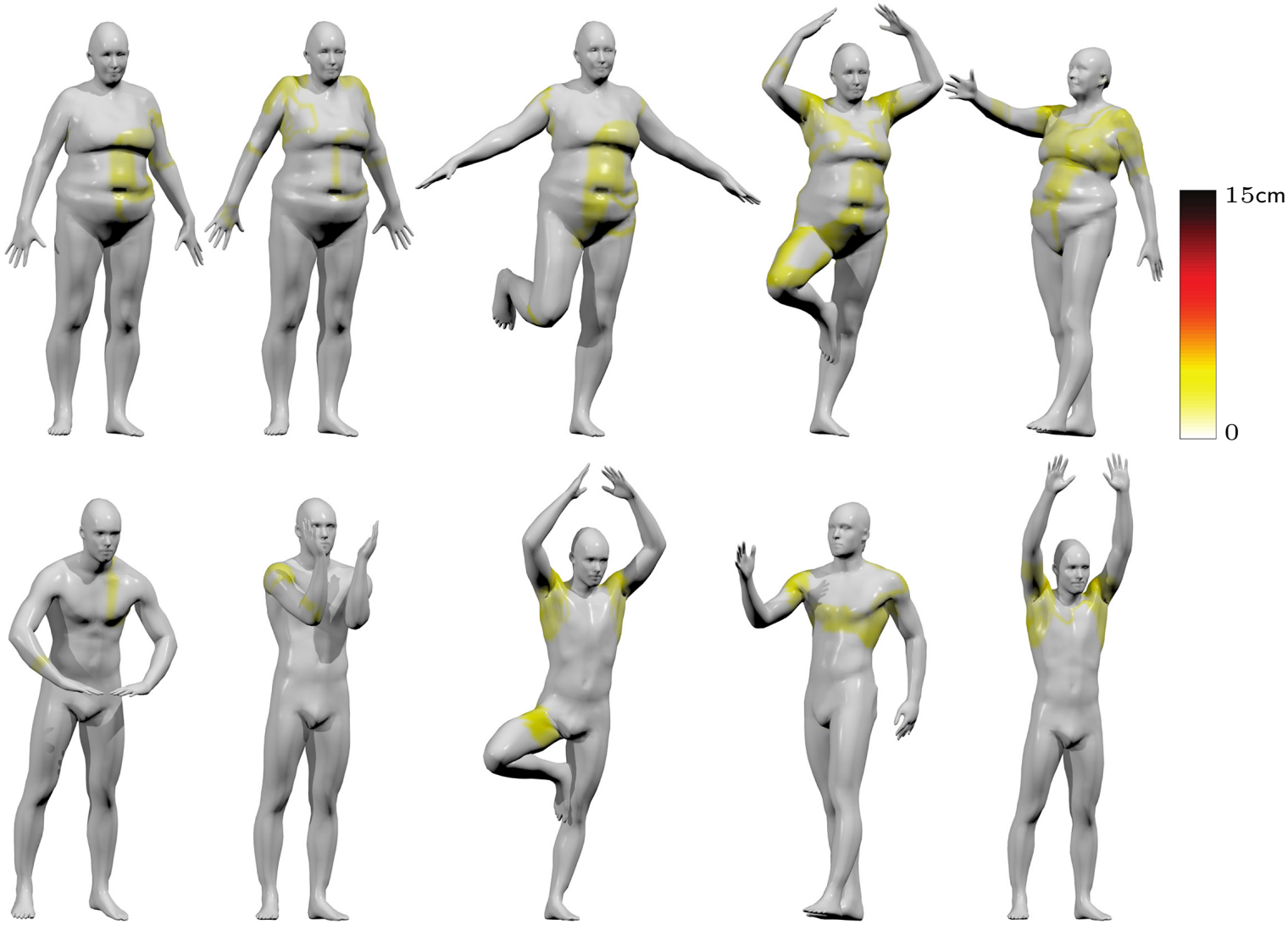
Shape correspondence error: Anisotropic CNN



Pointwise correspondence error (geodesic distance from groundtruth)

Boscaini et al. 2016

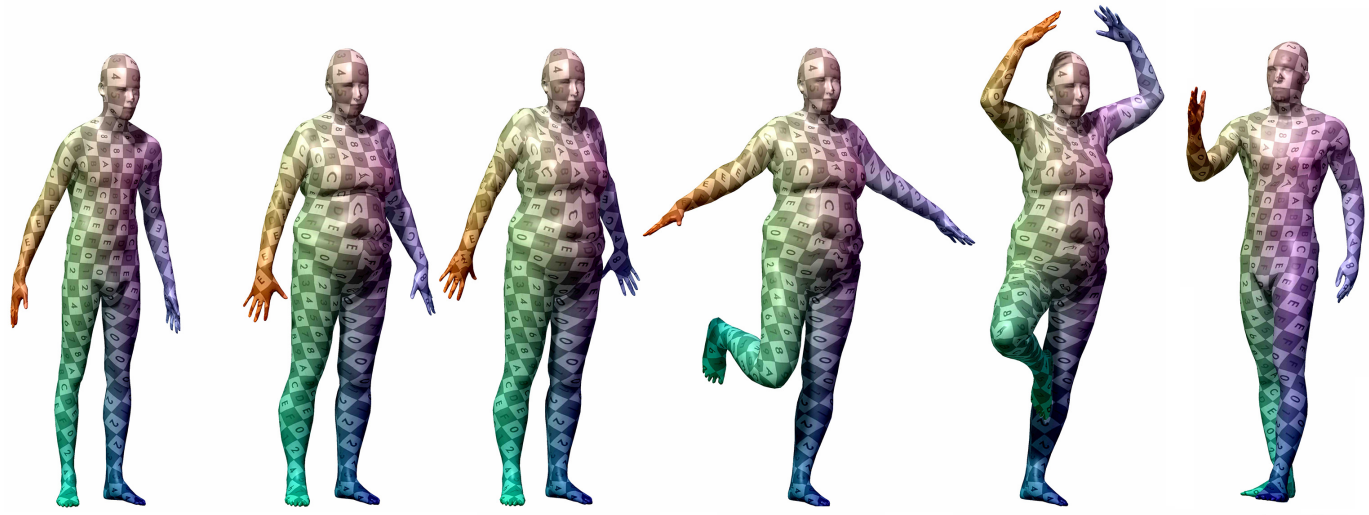
Shape correspondence error: MoNet



Pointwise correspondence error (geodesic distance from groundtruth)

Monti et al. 2016

Shape correspondence visualization: MoNet



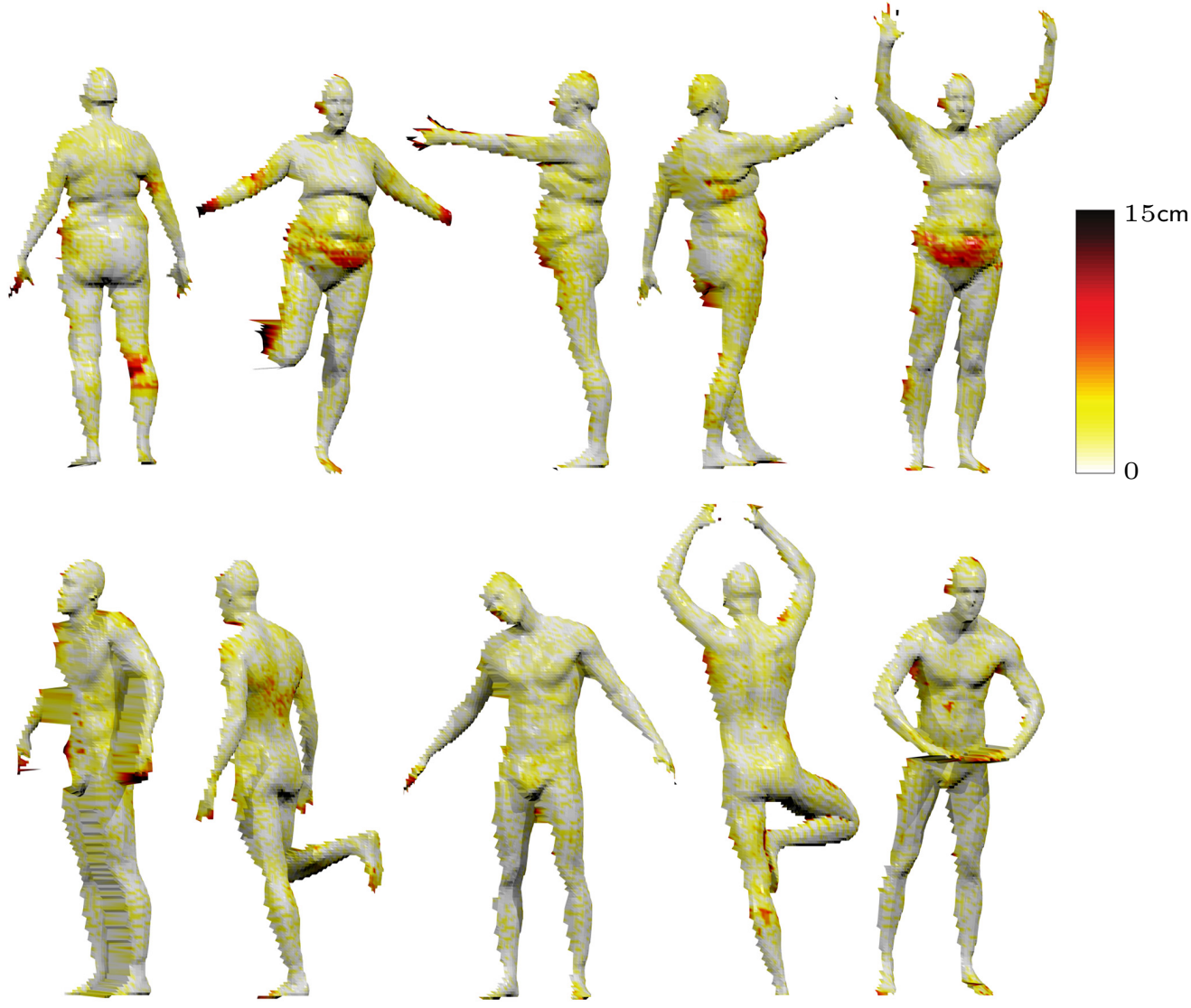
Reference



Texture transferred from reference to query shapes

Monti et al. 2016

Correspondence on range images: MoNet



Pointwise correspondence error (geodesic distance from groundtruth)

Monti et al. 2016

Correspondence with MoNet: Range images



Reference

Correspondence visualization (similar colors encode corresponding points)

Monti et al. 2016

Correspondence with MoNet: Range images

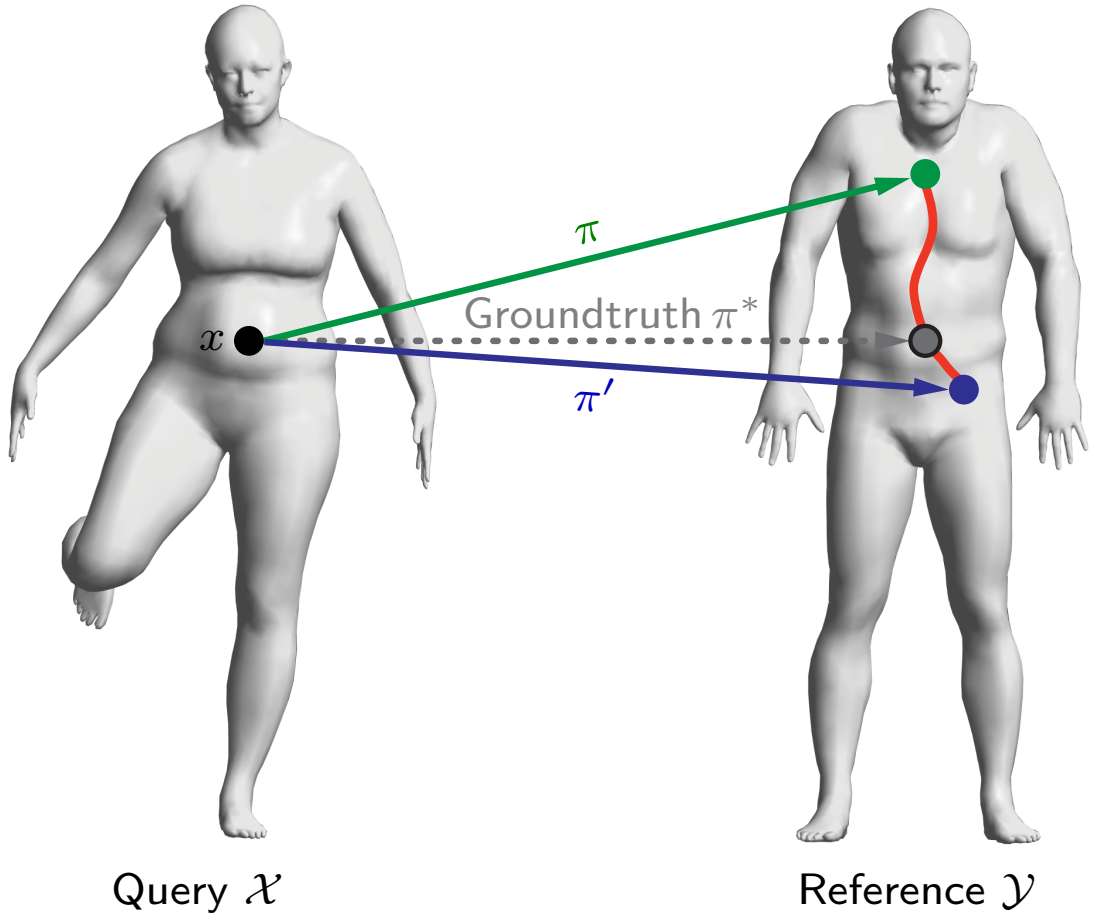


Reference

Correspondence visualization (similar colors encode corresponding points)

Monti et al. 2016

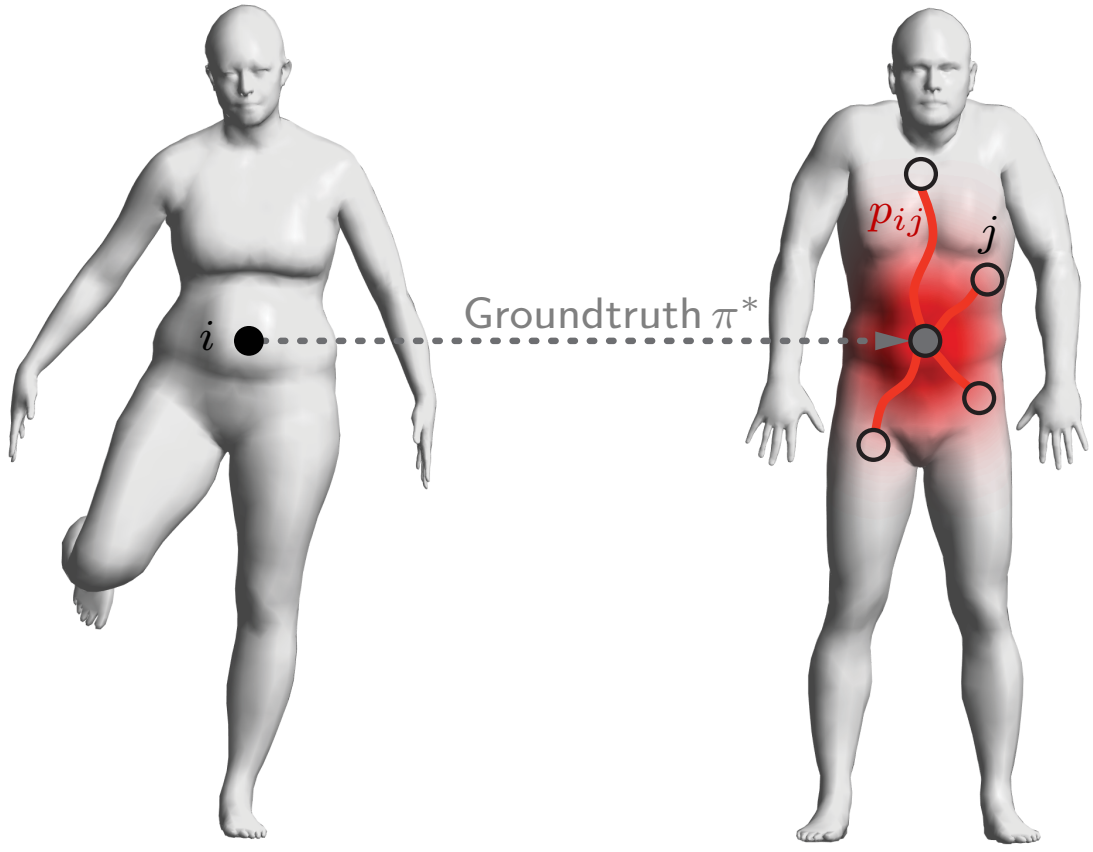
Correspondence as classification problem, revisited



Classification cost considers equally correspondences that deviate from the groundtruth (no matter how far)

Kim et al. 2011

Soft correspondence error

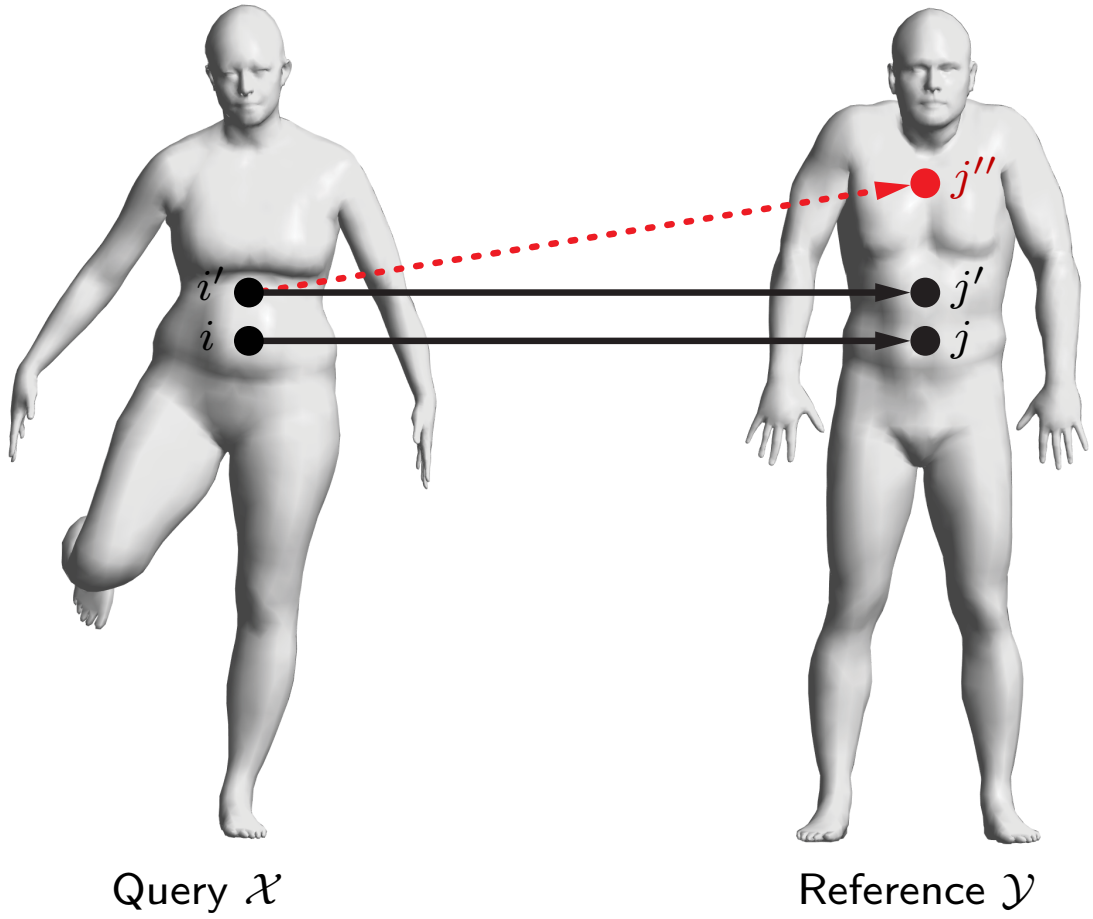


Soft correspondence error = probability-weighted geodesic distance from the groundtruth

$$\bar{\epsilon}_i = \sum_{j=1}^{m} p_{ij} d_{\mathcal{Y}}(\pi_i^*, j)$$

Kovnatsky et al. 2015; Litany et al. 2017

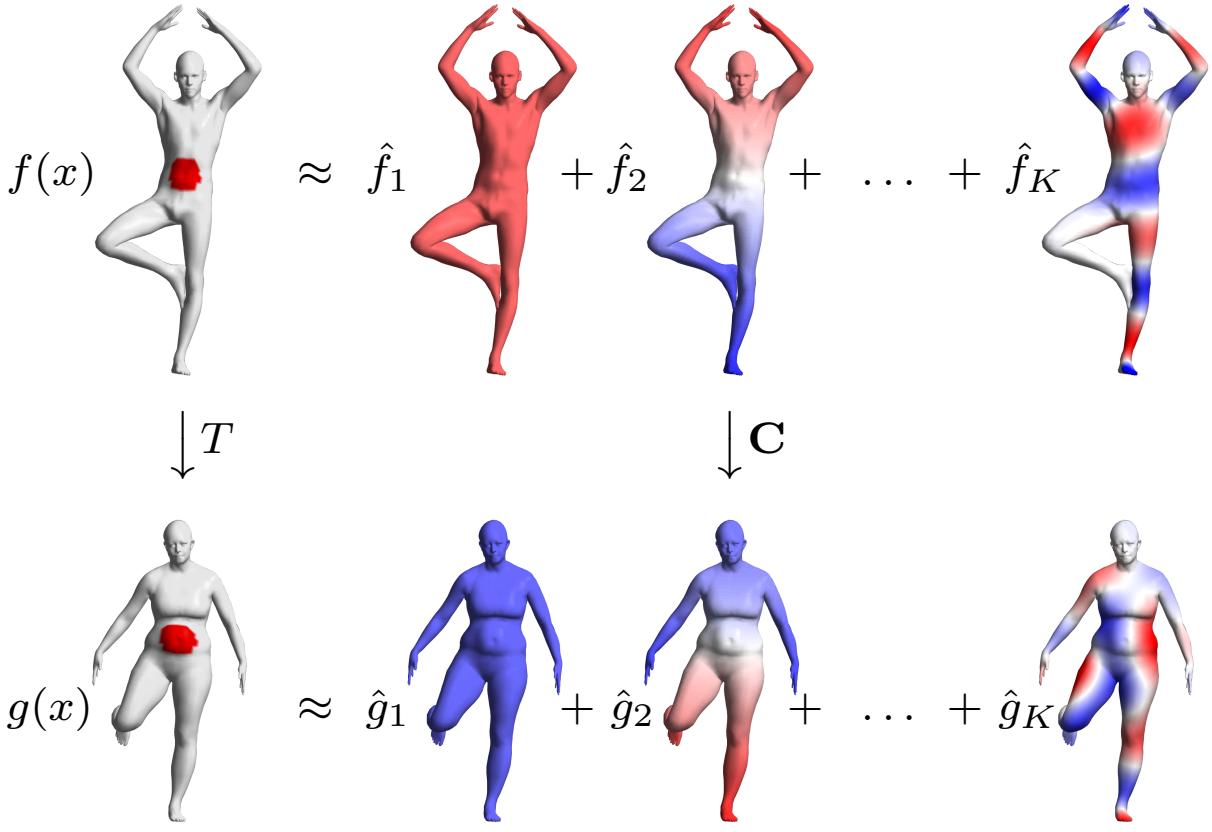
Pointwise vs Structured learning



Nearby points i, i' on query shape are **not guaranteed** to map to nearby points j, j' on reference shape at **test time**

Litany et al. 2017

Functional maps: spectral domain

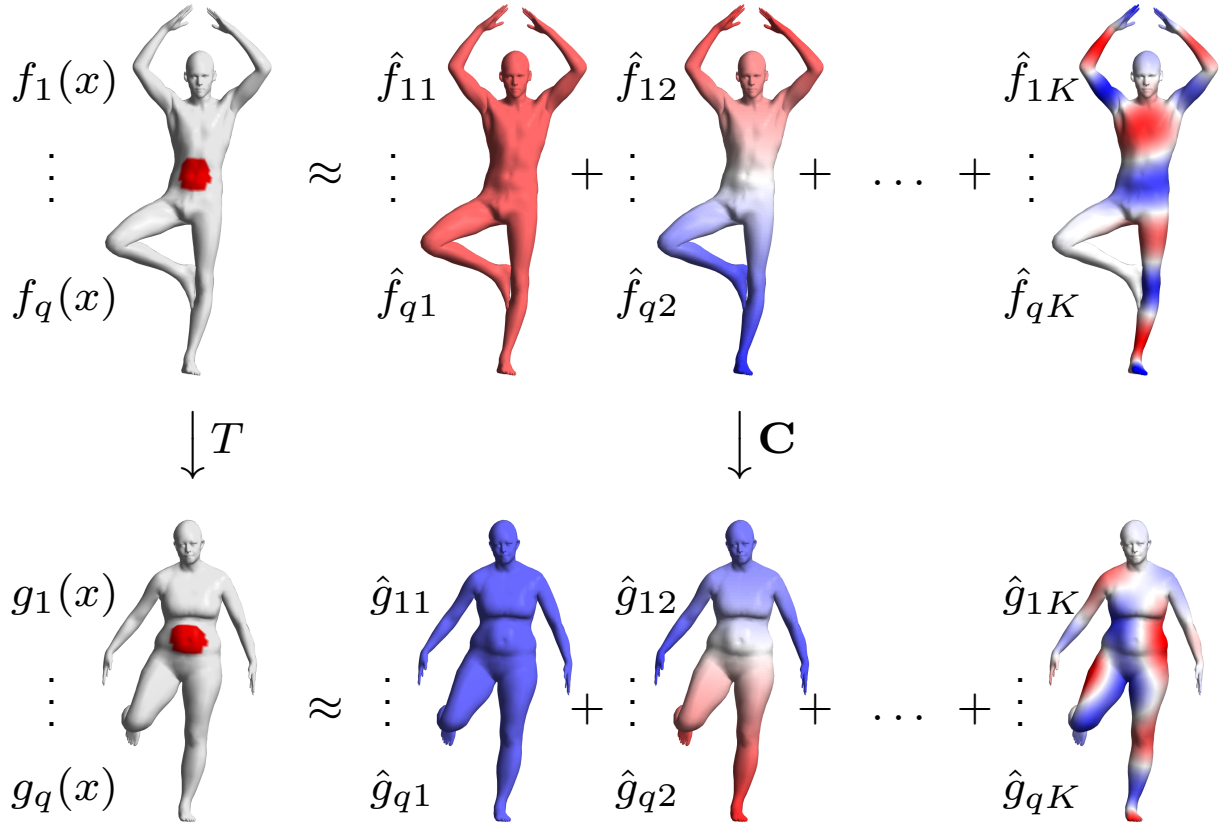


Functional correspondence T = linear map $\mathbf{C}$ between Fourier coefficients

$$\hat{\mathbf{g}}^\top = \hat{\mathbf{f}}^\top \mathbf{C}$$

Ovsjanikov et al. 2012

Functional maps: spectral domain

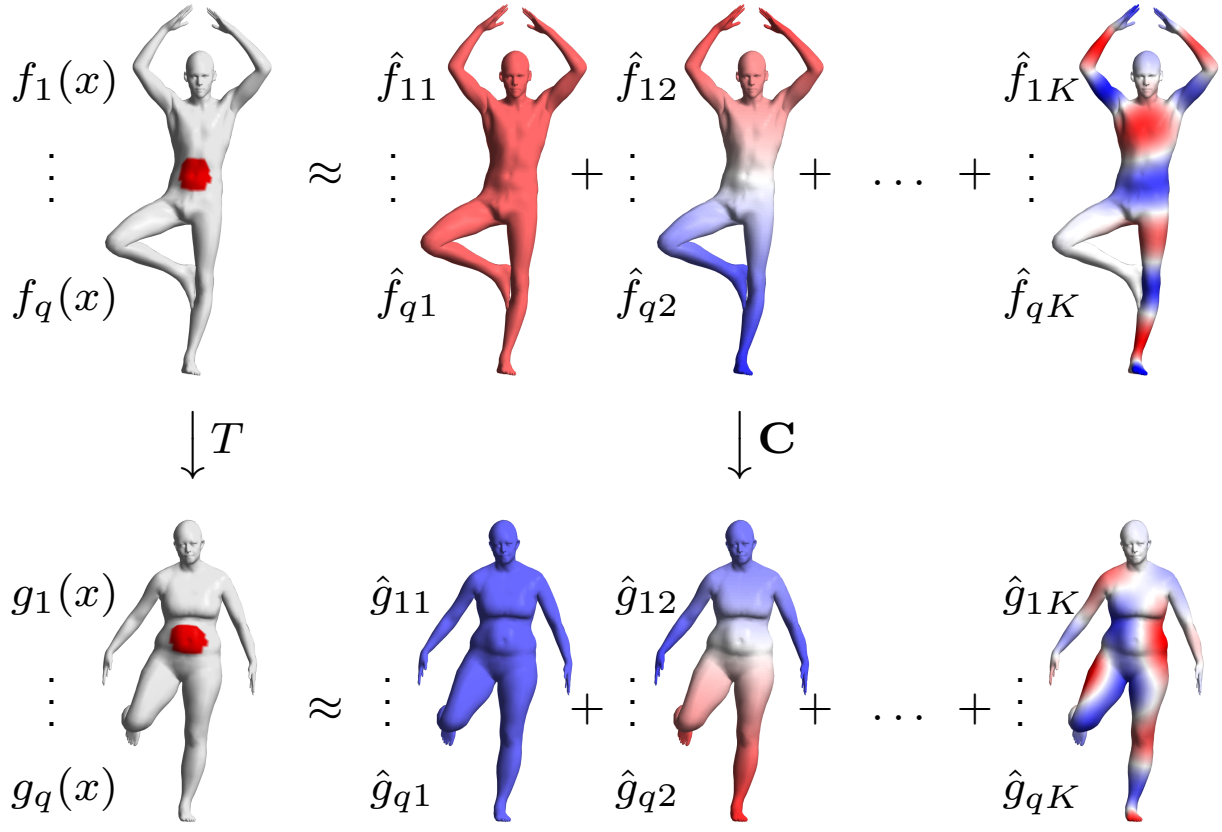


Recover correspondence from $q \geq k$ dimensional pointwise features

$$\begin{pmatrix} \hat{g}_{11} & \hat{g}_{12} & \cdots & \hat{g}_{1K} \\ \vdots & \vdots & & \vdots \\ \hat{g}_{q1} & \hat{g}_{q2} & \cdots & \hat{g}_{qK} \end{pmatrix} = \begin{pmatrix} \hat{f}_{11} & \hat{f}_{12} & \cdots & \hat{f}_{1K} \\ \vdots & \vdots & & \vdots \\ \hat{f}_{q1} & \hat{f}_{q2} & \cdots & \hat{f}_{qK} \end{pmatrix} \mathbf{C}$$

Ovsjanikov et al. 2012

Functional maps: spectral domain

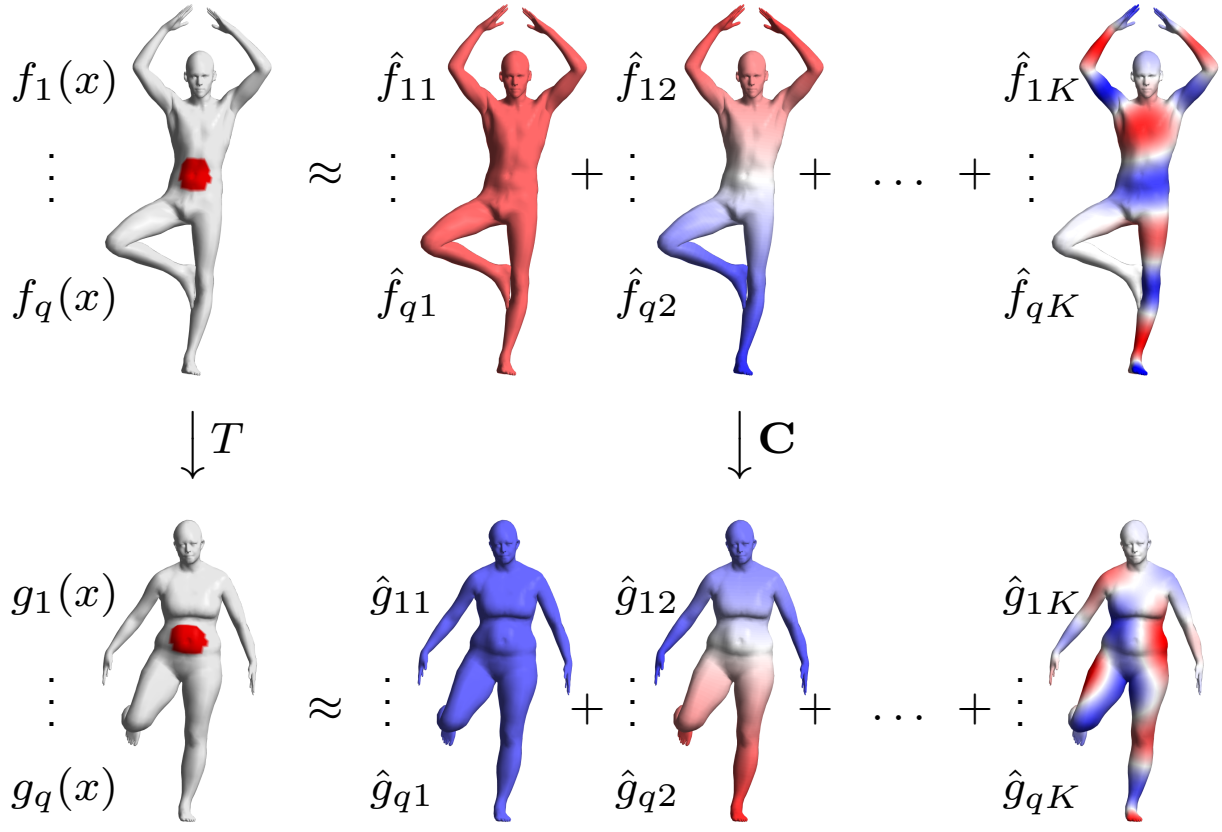


Recover correspondence from $q \geq k$ dimensional pointwise features

$$\hat{\mathbf{G}} = \hat{\mathbf{F}}\mathbf{C}$$

Ovsjanikov et al. 2012

Functional maps: spectral domain

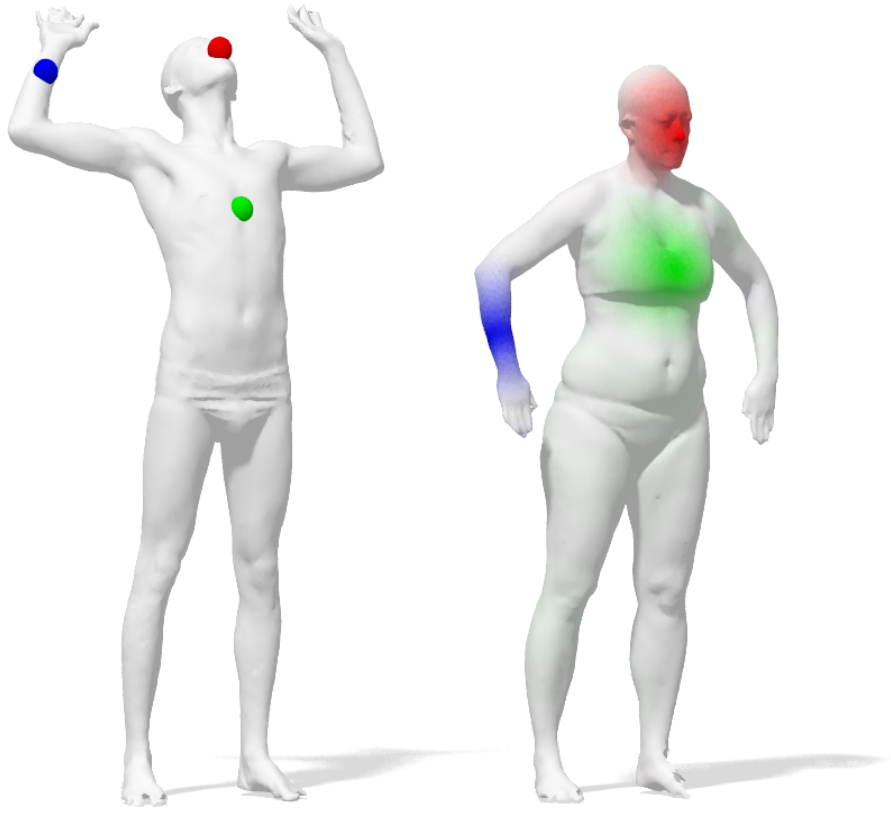


Recover correspondence from $q \geq k$ dimensional pointwise features

$$\mathbf{C}^* = \underset{\mathbf{C}}{\operatorname{argmin}} \|\hat{\mathbf{F}}\mathbf{C} - \hat{\mathbf{G}}\|_{\mathrm{F}}^2$$

Ovsjanikov et al. 2012

Functional maps: spatial domain

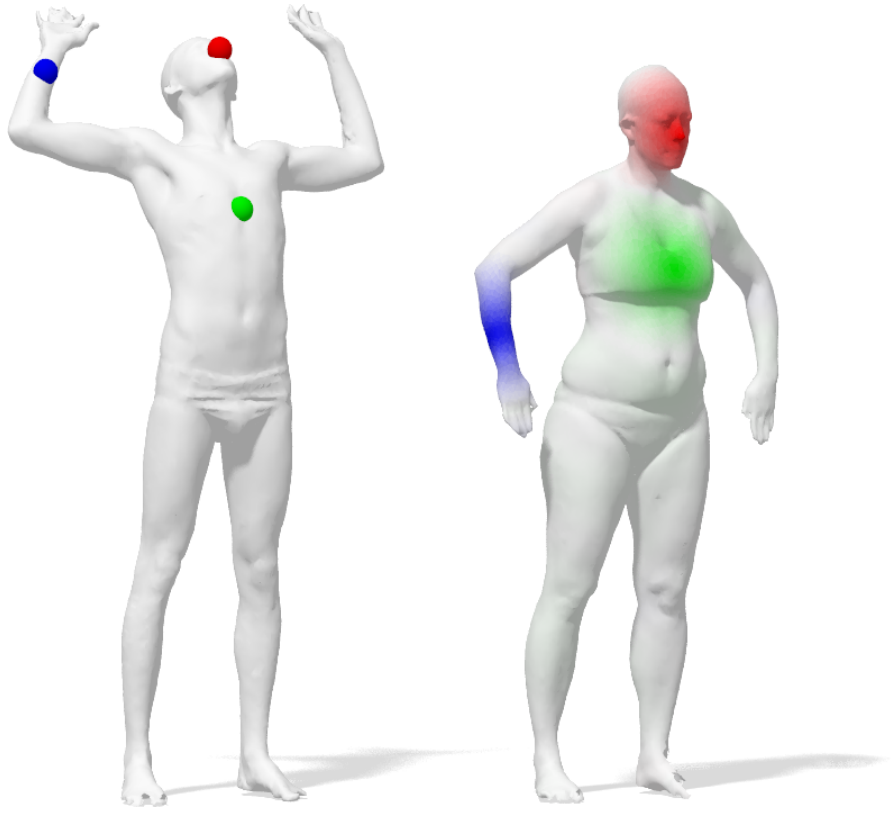


Rank-K approximation of spatial correspondence

$$\mathbf{T} \approx \mathbf{\Psi C \Phi}^\top$$

Ovsjanikov et al. 2012

Functional maps: spatial domain

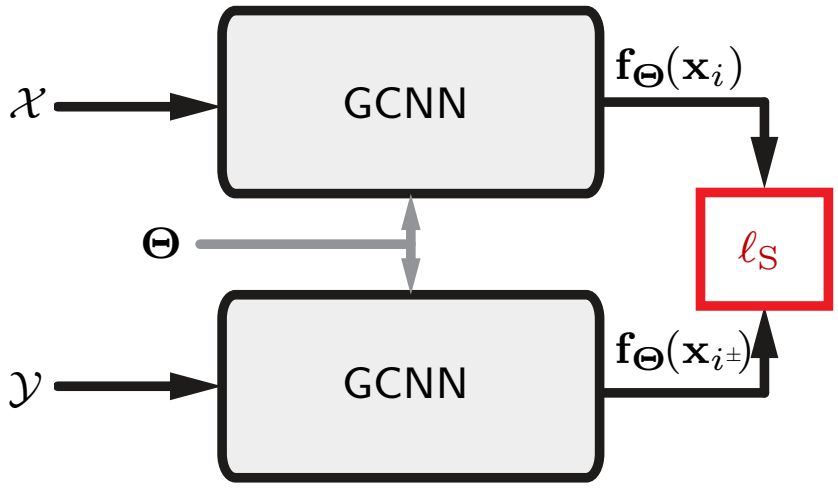


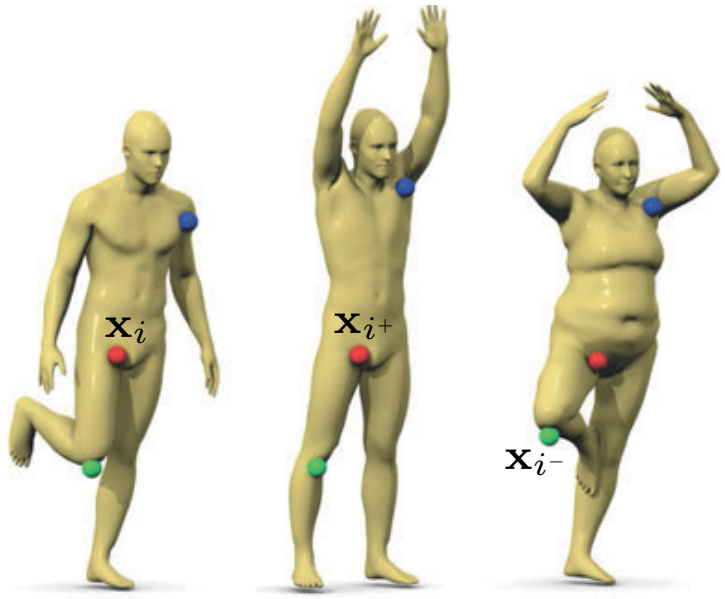
Probability p_{ij} of point j mapping to i

$$\mathbf{P} \approx |\mathbf{\Psi C \Phi}^\top|_{\|\cdot\|}$$

Ovsjanikov et al. 2012

Siamese metric learning

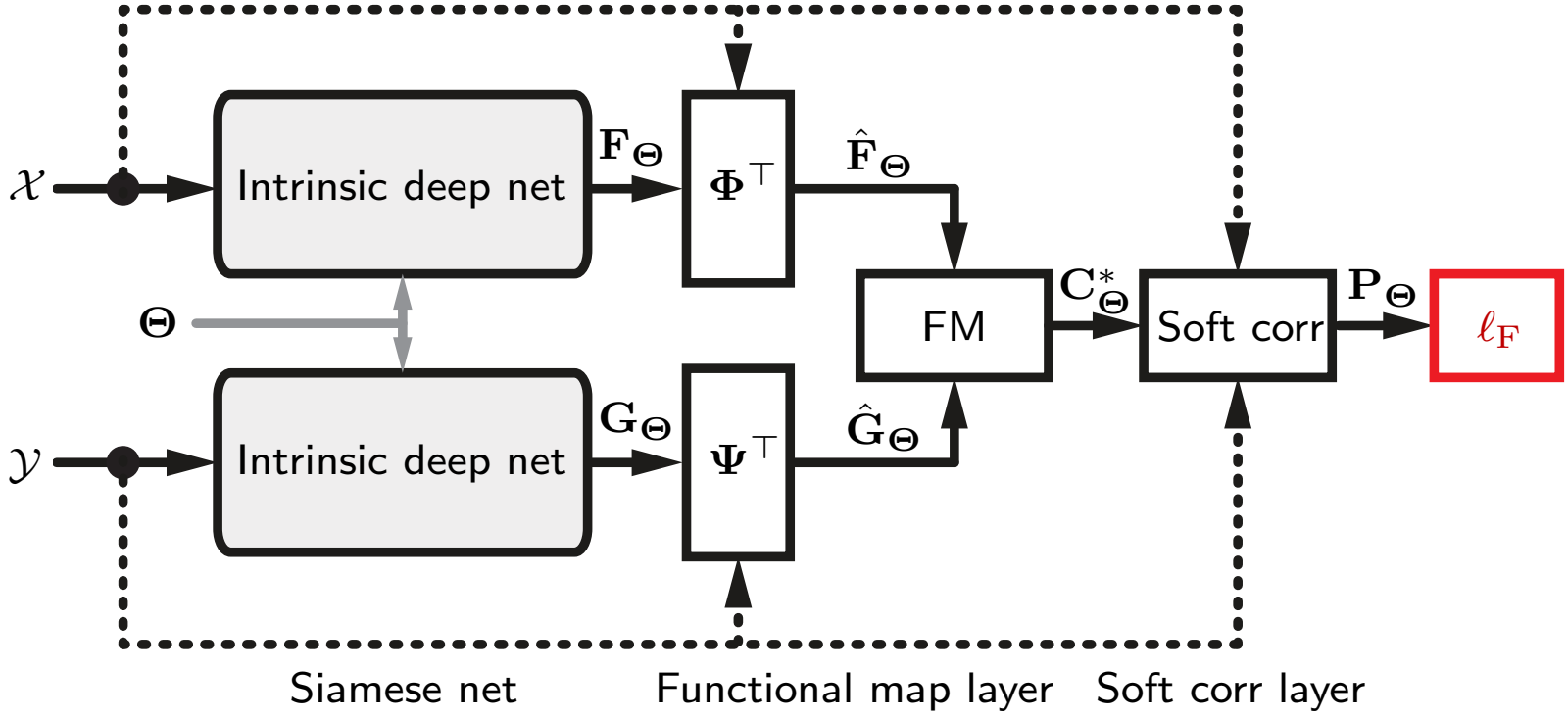




Training set	positive (i, i^+) and negative (i, i^-) pairs of points
Siamese net	two net instances with shared parameters $\mathbf{\Theta}$
Poitwise feature cost	$\ell_S(\mathbf{\Theta}) = \gamma \sum_{i,i^+} \|\mathbf{f}_\Theta(\mathbf{x}_i) - \mathbf{f}_\Theta(\mathbf{x}_{i^+})\|_2^2$
	$+ (1-\gamma) \sum_{i,i^-} \left[\mu - \|\mathbf{f}_\Theta(\mathbf{x}_i) - \mathbf{f}_\Theta(\mathbf{x}_{i^-})\|_2^2\right]_+$

Masci et al. 2015

Structured correspondence with FMNet



Siamese net — two net instances with shared parameters Θ

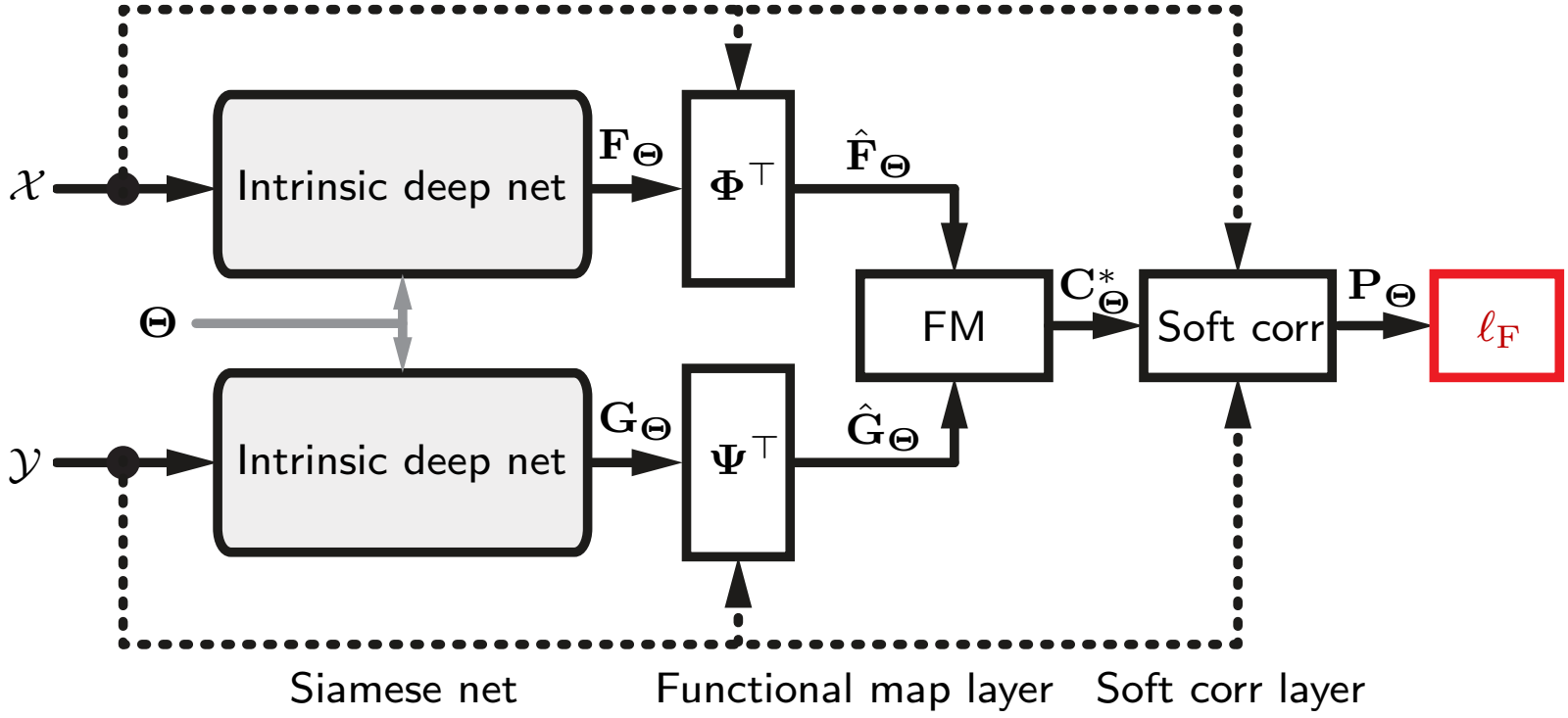
Functional map layer — $\mathbf{C}^*_\Theta = \underset{\mathbf{C}}{\operatorname{argmin}} \|\hat{\mathbf{F}}_\Theta \mathbf{C} - \hat{\mathbf{G}}_\Theta\|^2_{\mathrm{F}}$

Soft correspondence layer — $\mathbf{P}_\Theta = |\Psi \mathbf{C}_\Theta \Phi^\top|_{\|\cdot\|}$

Soft error cost — $\ell_{\mathrm{F}}(\Theta) = \sum_{i=1}^{n} \sum_{j=1}^{m} p_{\Theta,ij} d_{\mathcal{Y}}(\pi^*_i, j)$

Litany et al. 2017

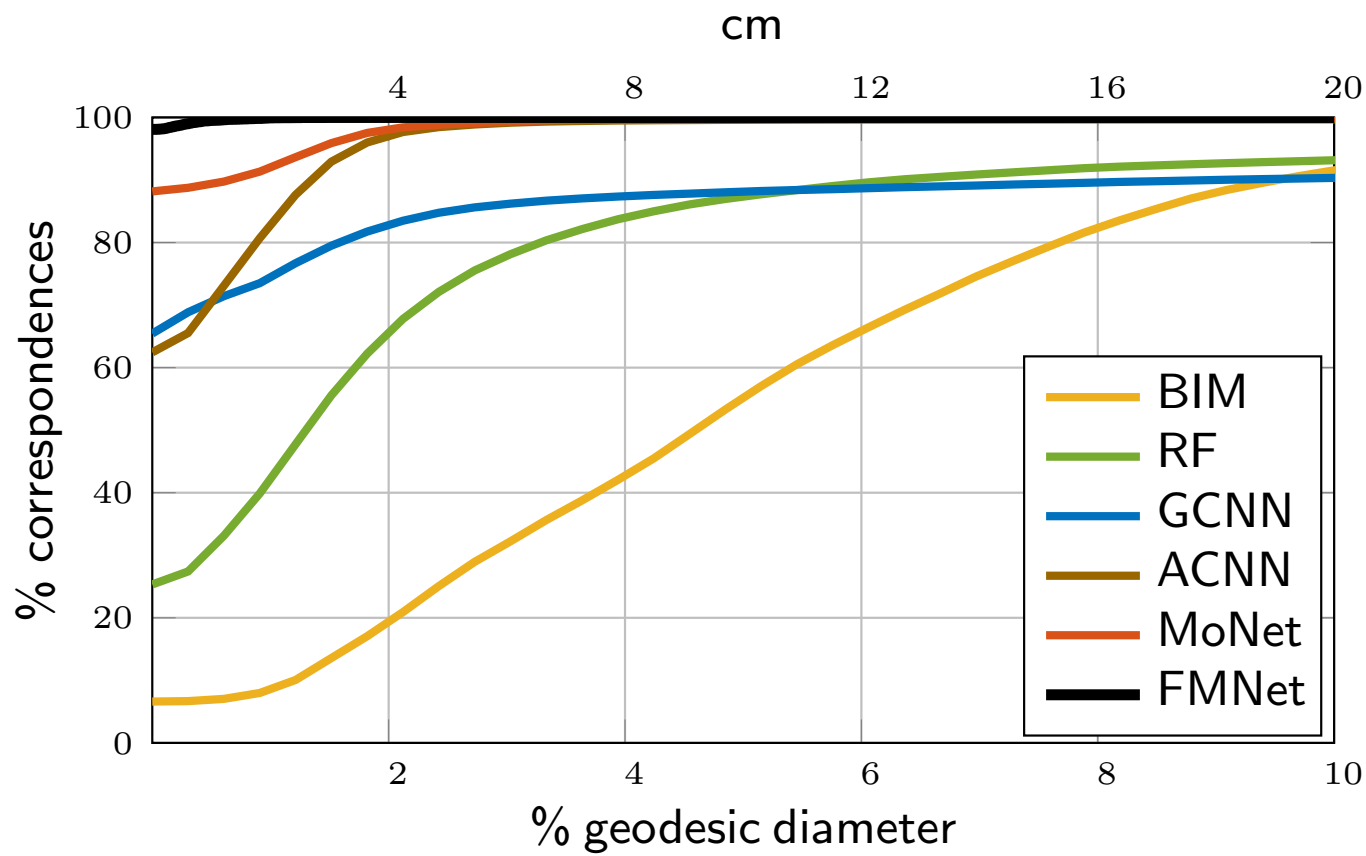
Structured correspondence with FMNet



Siamese net	two net instances with shared parameters Θ
Functional map layer	$\mathbf{C}^*_\Theta = \hat{\mathbf{F}}^\dagger_\Theta \hat{\mathbf{G}}_\Theta$
Soft correspondence layer	$\mathbf{P}_\Theta = \lvert \Psi \mathbf{C}_\Theta \Phi^\top \rvert_{\lVert\cdot\rVert}$
Soft error cost	$\ell_F(\Theta) = \lVert \mathbf{P}_\Theta \circ \mathbf{D}_\mathcal{Y} \rVert$

Litany et al. 2017

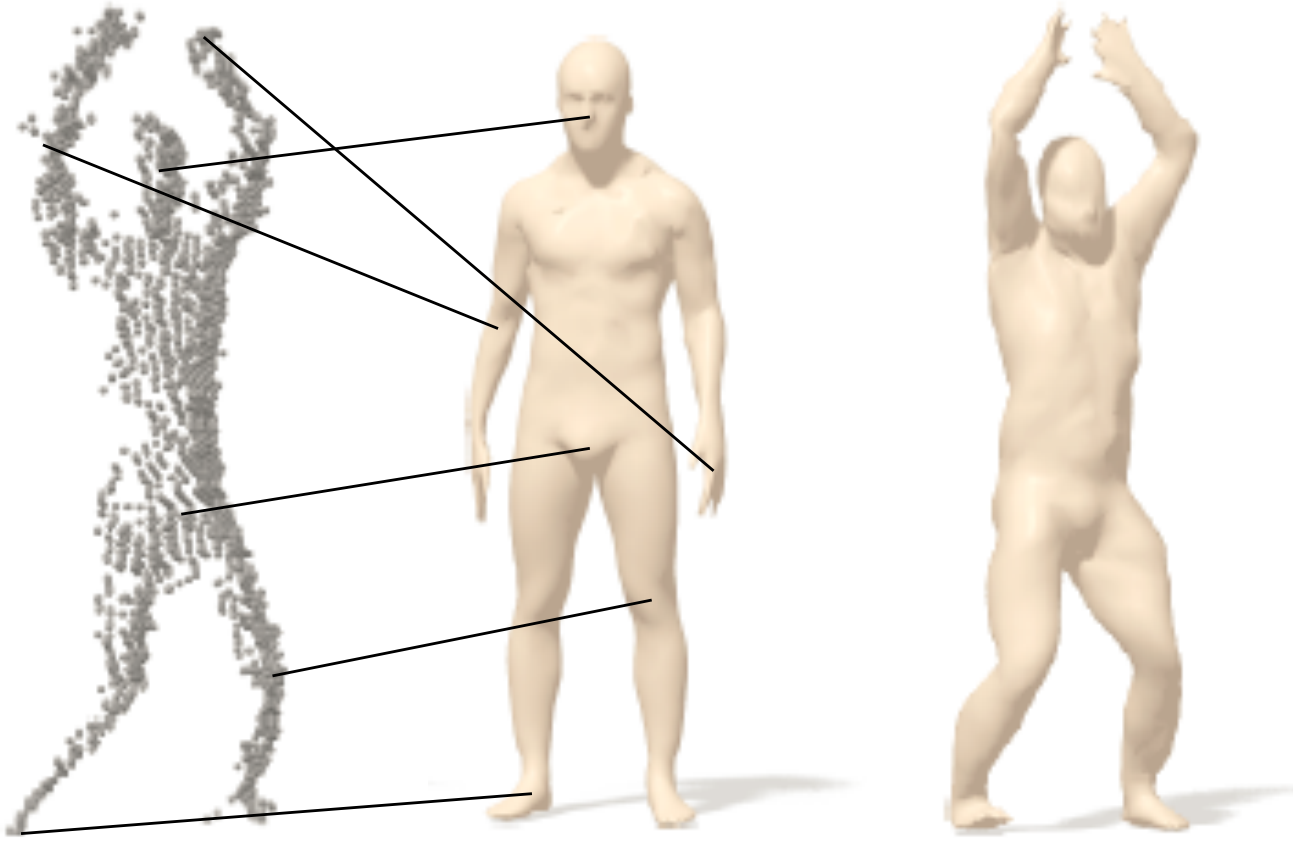
Correspondence quality comparison



Correspondence evaluated using asymmetric Princeton benchmark
(training and testing: disjoint subsets of FAUST)

Methods: Kim et al. 2011 (BIM); Rodolà et al. 2014 (RF); Boscaini et al. 2015 (ADD); Masci et al. 2015 (GCNN); Boscaini et al. 2016 (ACNN); Monti et al. 2016 (MoNet); Litany et al. 2017 (FMNet); data: Bogo et al. 2014 (FAUST); benchmark: Kim et al. 2011

3D shape analysis and synthesis



Input
3D scan

Correspondence to
Reference shape

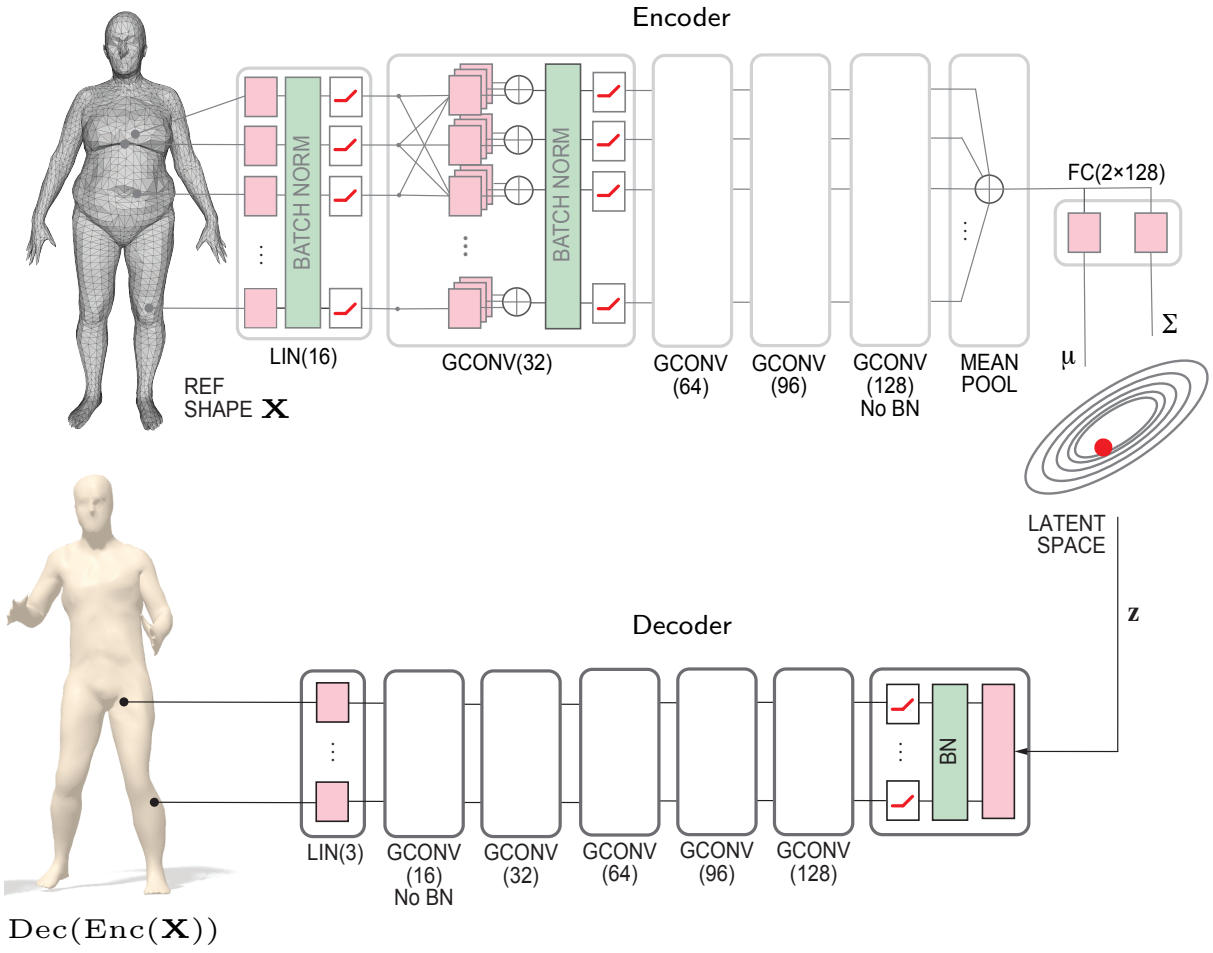
Deformation of
reference shape

Intrinsic Variational Autoencoder (VAE)

Minimize

$$||\mathrm{Dec}(\mathrm{Enc}(\mathbf{X})) - \mathbf{X}||_{\mathrm{F}}$$
$$+\lambda D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{X})||p(\mathbf{z}))$$

w.r.t. net parameters



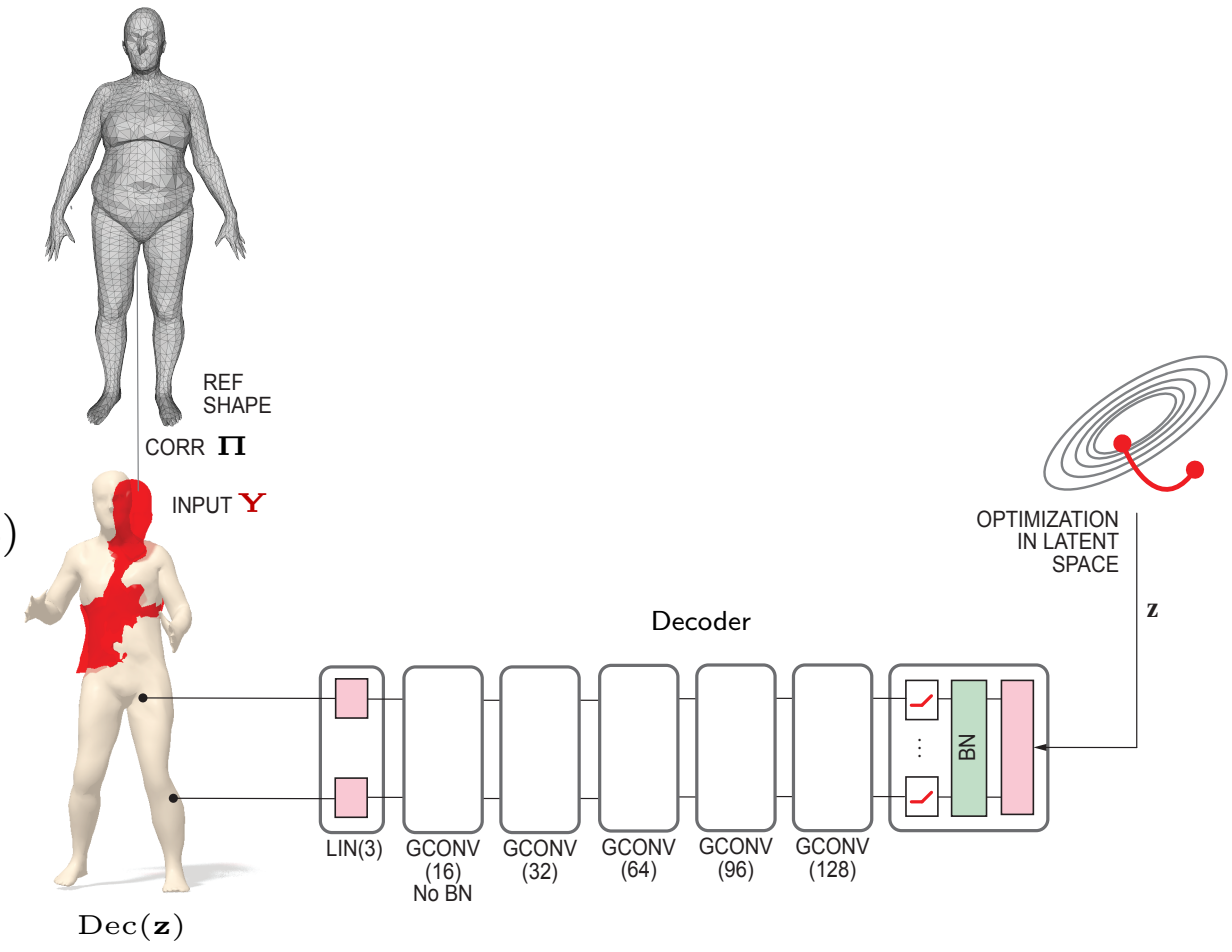
Litany et al. 2017; training on Dynamic FAUST (Bogo et al. 2017)

Shape completion

Minimize

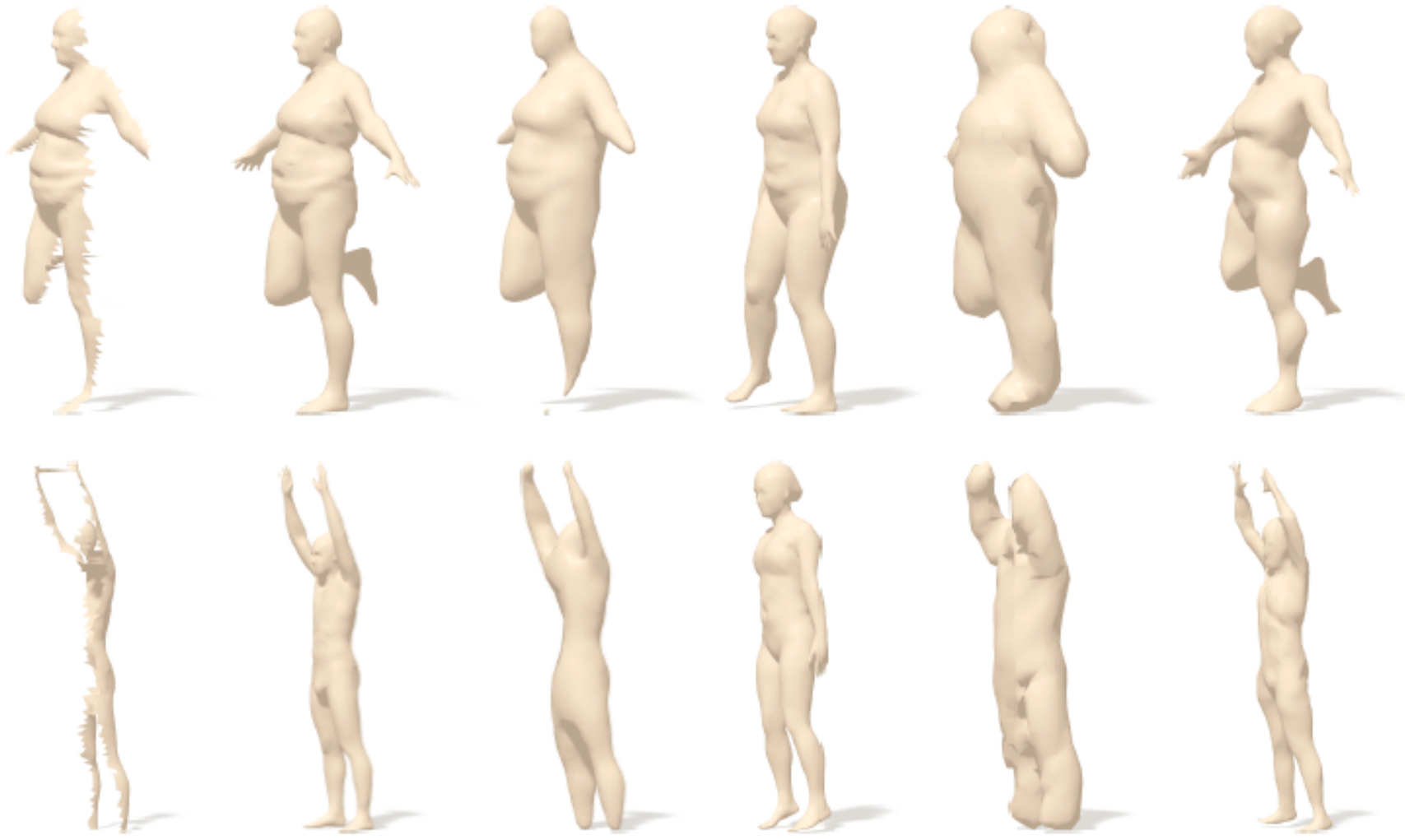
$$\|\mathrm{Dec}(\mathbf{z})\mathbf{\Pi} - \mathbf{T}\mathbf{Y}\|_{\mathrm{F}}$$

in alternating manner
w.r.t. $\mathbf{z}$ and $\mathbf{T} \in \mathrm{SO}(3)$



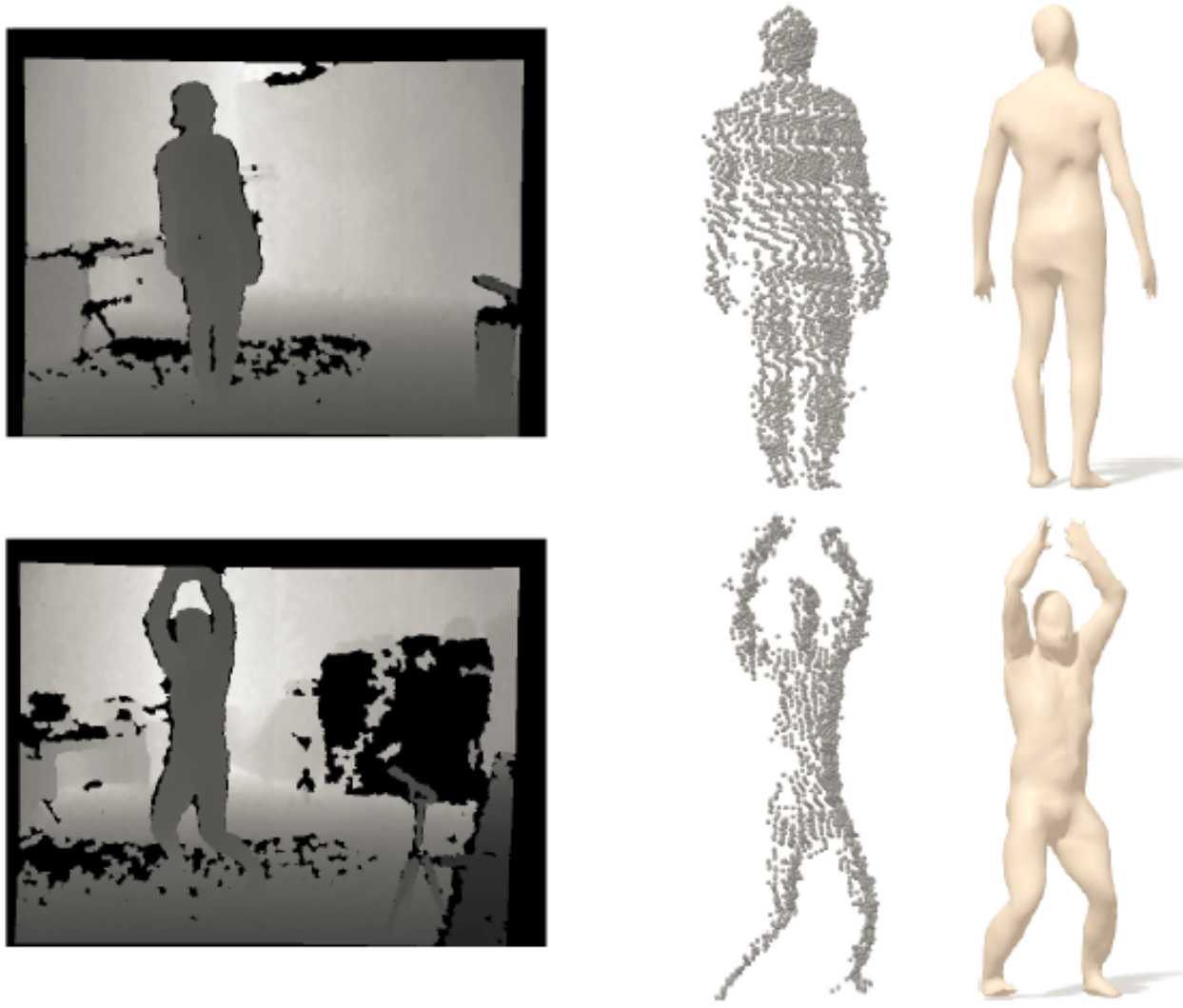
Litany et al. 2017

Shape completion comparison



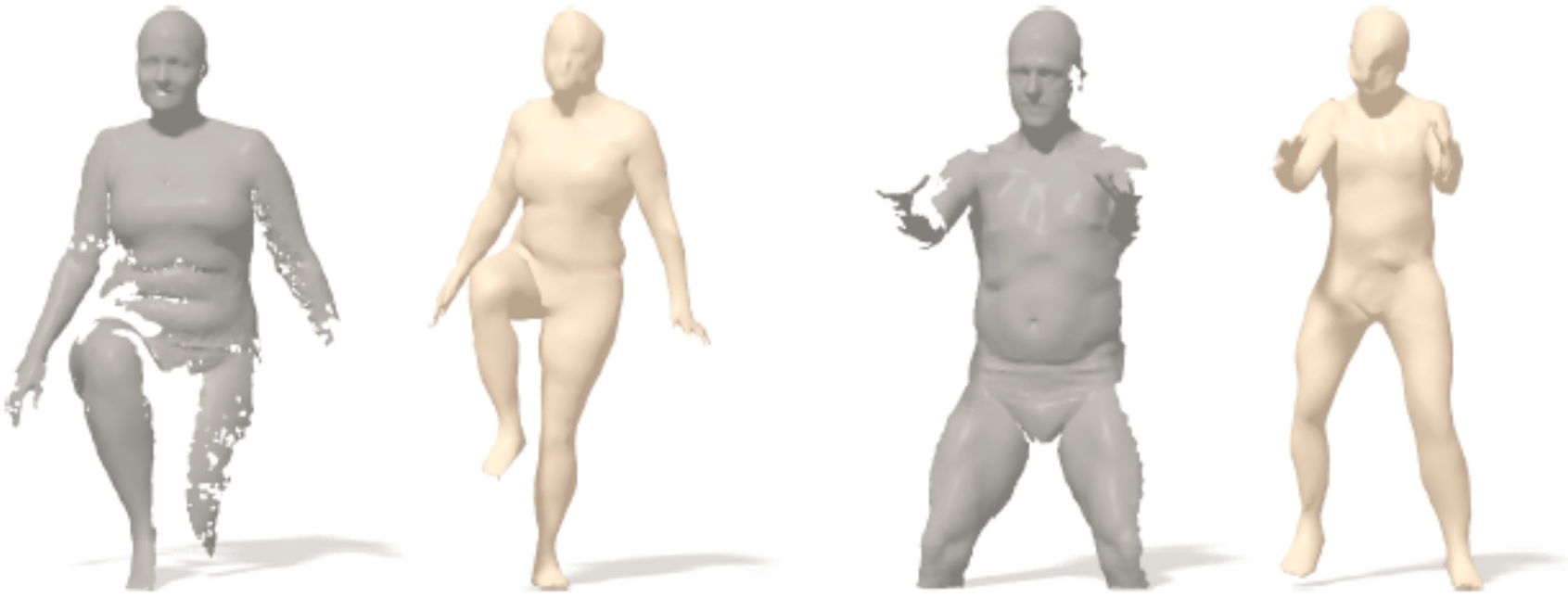
Methods: Litany et al. 2017; Dai et al. 2016 (3D-EPN); Kazhdan et al. 2013 (Poisson)

Shape completion examples



Litany et al. 2017; data: Ofli et al. 2014 (MHAD)

Shape completion examples



Litany et al. 2017; data: Bogo et al. 2014 (FAUST)

Application in Recommendation Systems



Delivery Man
2013



12 Years a Slave
2013



Gravity
2013



Frozen
2013



American Hustle
2013



Thor: The Dar...
2013



The Hunger Ga...
2013



Captain Phillips
2013



Dallas Buyers Club
2013



Escape Plan
2013



Saving Mr. Banks
2013



Reasonable Doubt
2014



Last Vegas
2013



Ender's Game
2013



Jackass Prese...
2013



Rush
2013



The Wolverine...
2013



About Time
2013



The Bag Man
2014



Nebraska
2013



Despicable Me 2
2013



World War Z UNR...
2013



The Counselor
2013



We're the Mil...
2013



Man of Steel
2013



Now You See M...
2013



Cloudy with a...
2013



This Is the End
2013



The Returned
2013



Homefront
2013



August Osage C...



Elysium



Pacific Rim



The Great Gatsby



The Lone Ranger



Blue Jasmine



2 Guns



Romeo and Juliet

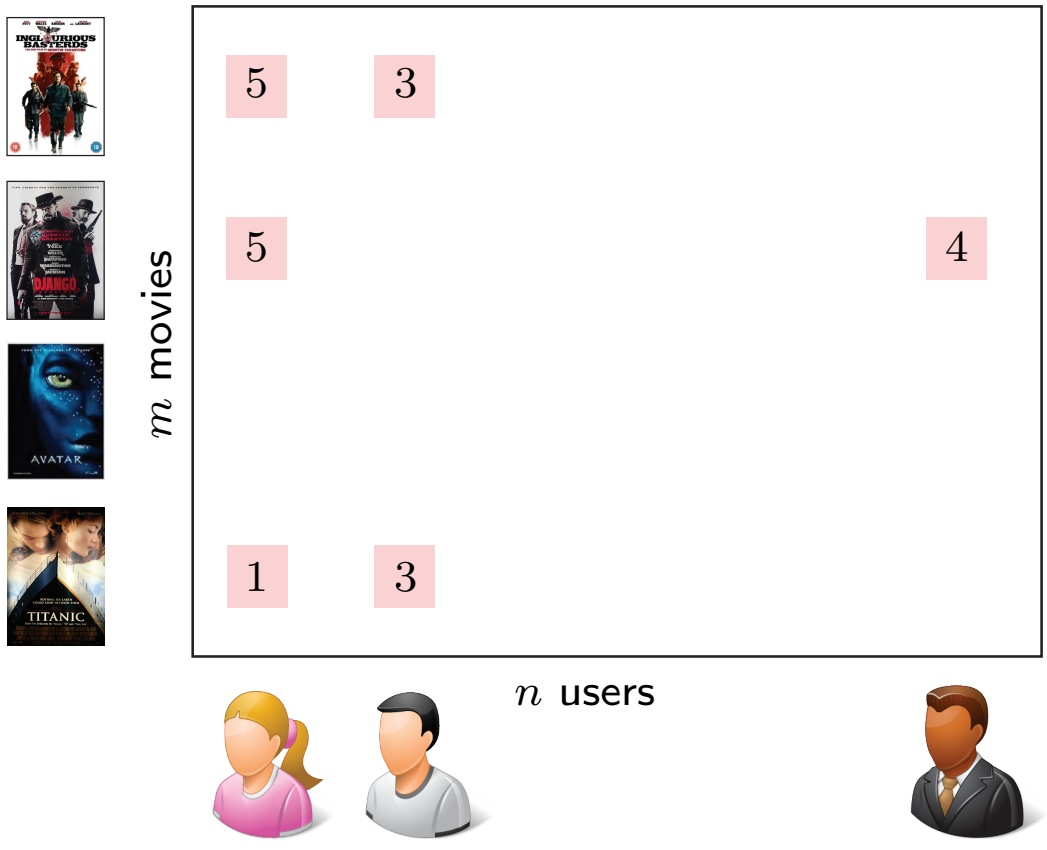


Monsters Univer...



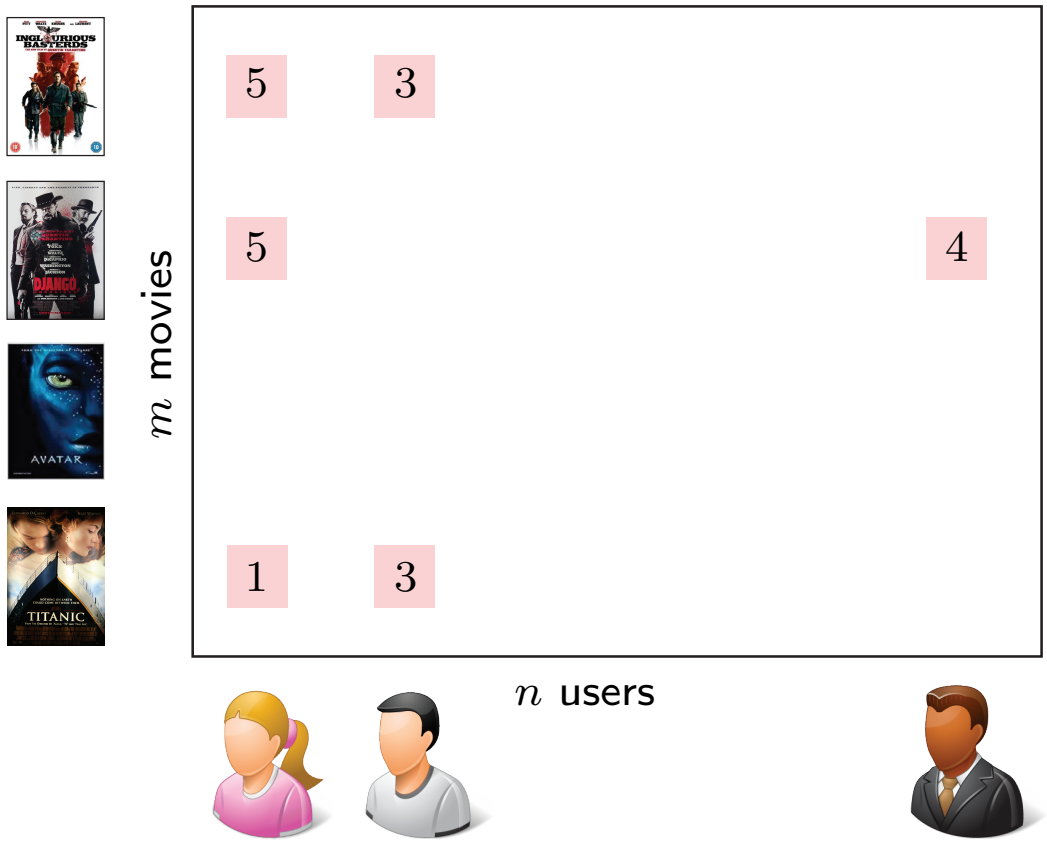
The Hobbit: A...

Matrix completion: 'Netflix challenge'



$$\min_{\mathbf{X} \in \mathbb{R}^{m \times n}} \quad \text{rank}(\mathbf{X}) \quad \text{s.t.} \quad x_{ij} = a_{ij} \;\; \forall ij \in \Omega$$

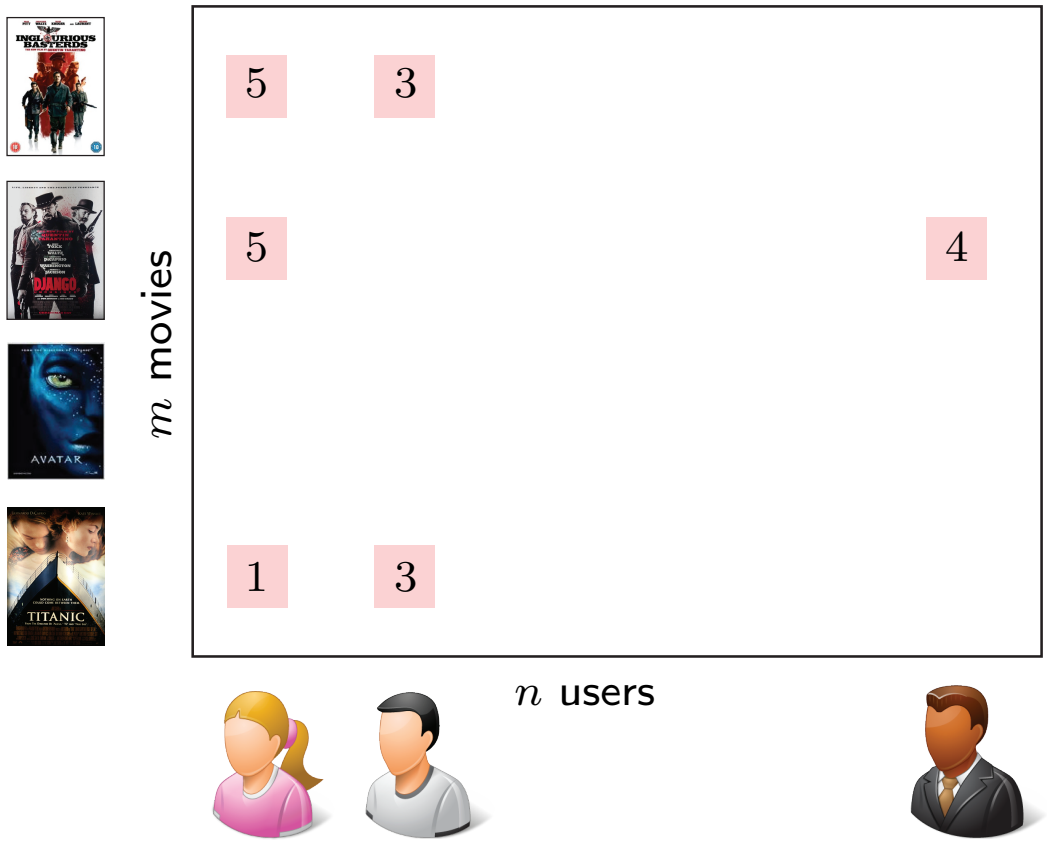
Matrix completion: 'Netflix challenge'



$$\min_{\mathbf{X} \in \mathbb{R}^{m \times n}} \quad \|\mathbf{X}\|_* \quad \text{s.t.} \quad x_{ij} = a_{ij} \;\; \forall ij \in \Omega$$

Candès 2008

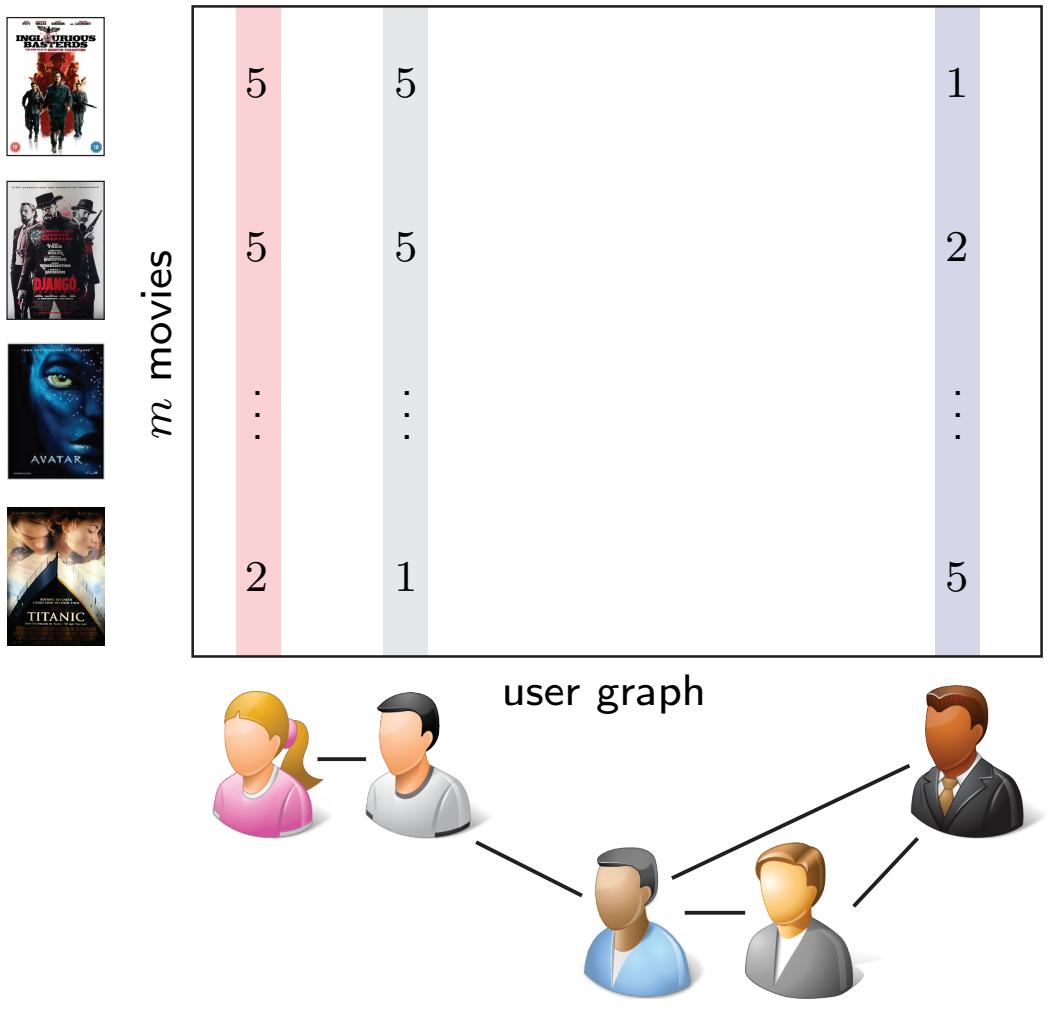
Matrix completion: 'Netflix challenge'



$$\min_{\mathbf{X} \in \mathbb{R}^{m \times n}} \quad \|\mathbf{X}\|_* + \mu \|\mathbf{\Omega} \circ (\mathbf{X} - \mathbf{A})\|_{\mathrm{F}}^2$$

Candès 2008

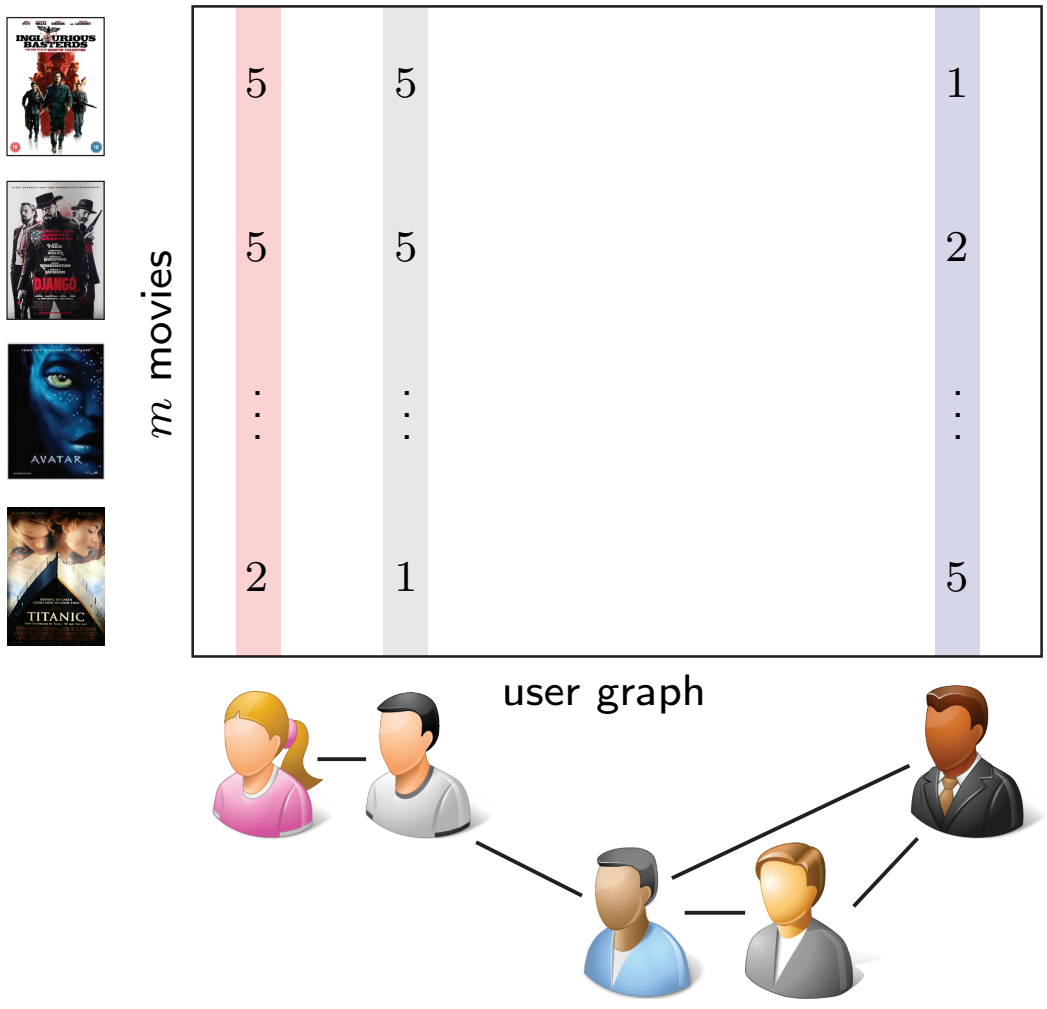
Geometric matrix completion



$$\min_{\mathbf{X}\in\mathbb{R}^{m\times n}} \quad \mu\|\mathbf{\Omega}\circ(\mathbf{X}-\mathbf{A})\|_{\mathrm{F}}^2+\mu_{\mathrm{c}}\,\mathrm{tr}(\mathbf{X}\mathbf{\Delta}_{\mathrm{c}}\mathbf{X}^\top)$$

Coifman, Gavish 2011; Kalofolias et al. 2014

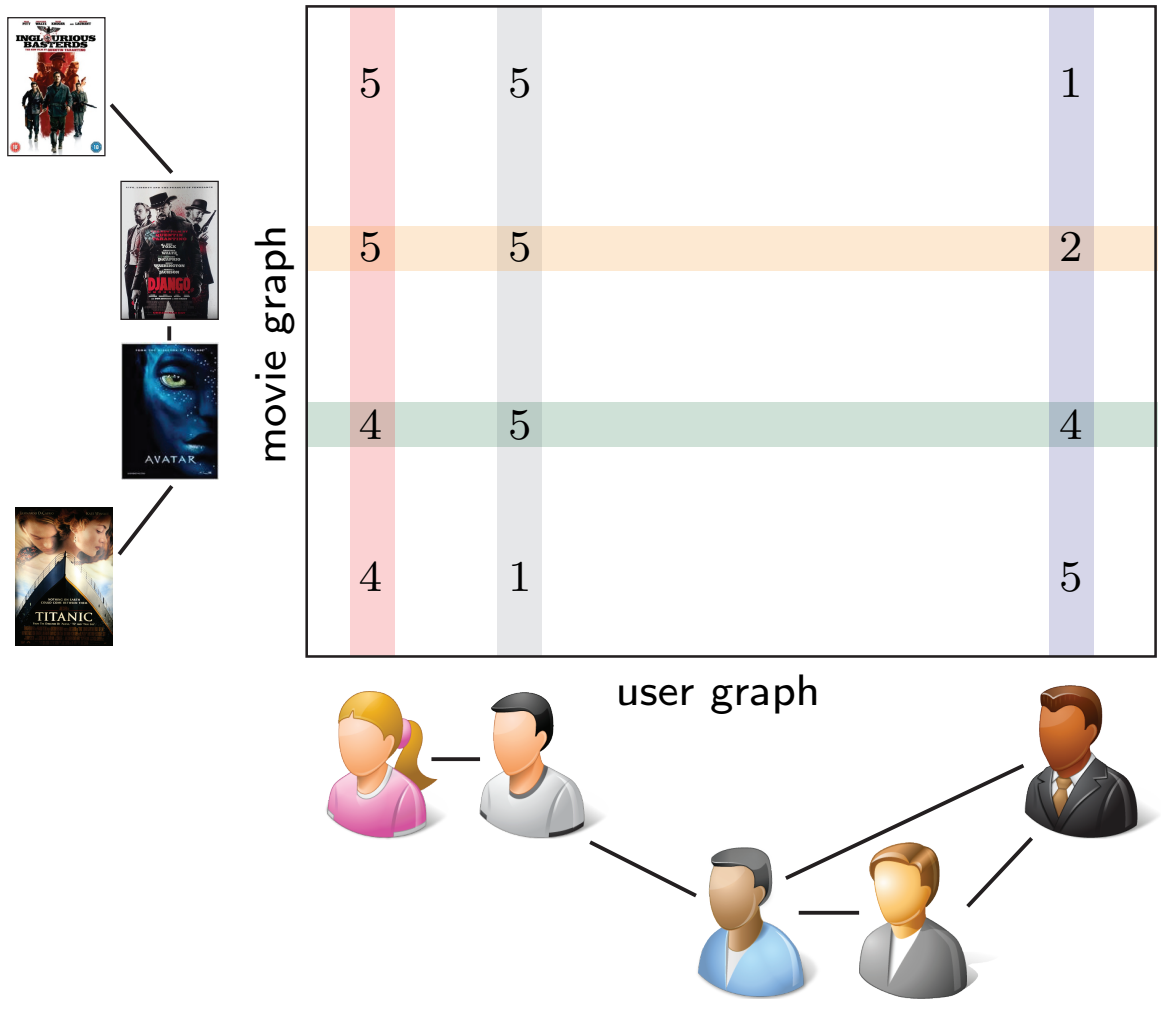
Geometric matrix completion



$$\min_{\mathbf{X}\in\mathbb{R}^{m\times n}} \quad \mu\|\mathbf{\Omega}\circ(\mathbf{X}-\mathbf{A})\|_{\mathrm{F}}^2 + \mu_{\mathrm{c}}\underbrace{\mathrm{tr}(\mathbf{X}\mathbf{\Delta}_{\mathrm{c}}\mathbf{X}^\top)}_{\|\mathbf{X}\|_{\mathcal{G}_{\mathrm{c}}}^2}$$

Coifman, Gavish 2011; Kalofolias et al. 2014

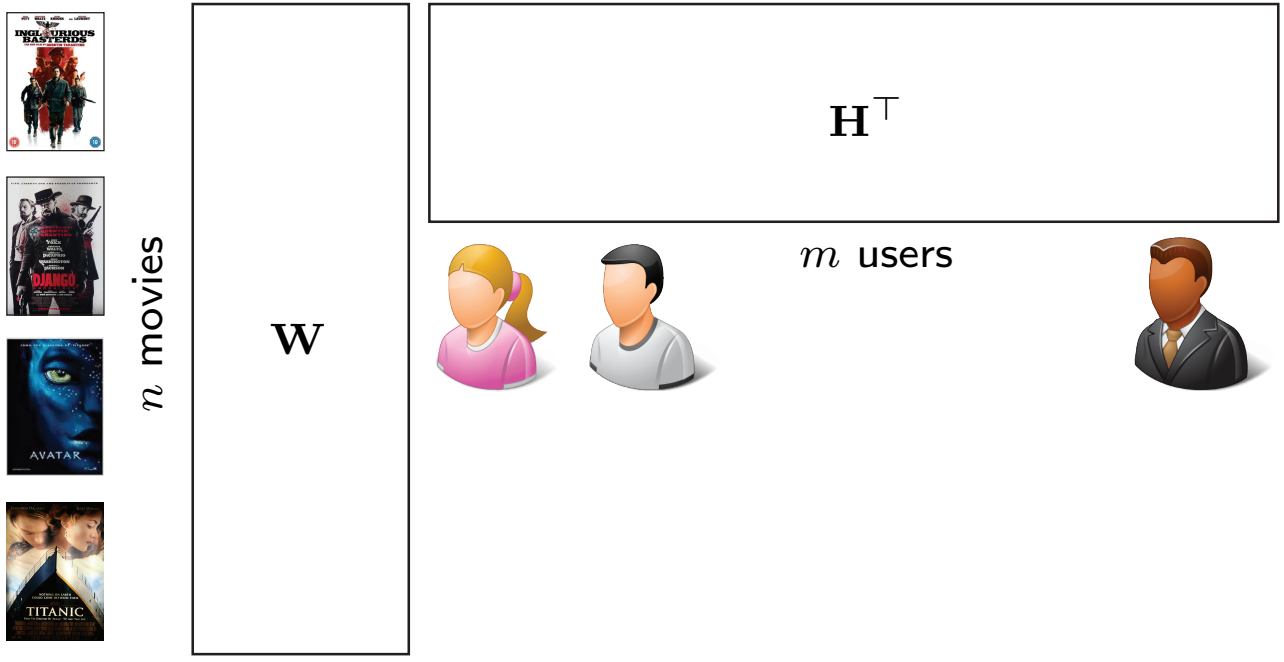
Geometric matrix completion



$$\min_{\mathbf{X}\in\mathbb{R}^{m\times n}} \quad \mu\|\mathbf{\Omega}\circ(\mathbf{X}-\mathbf{A})\|_{\mathrm{F}}^2 + \mu_{\mathrm{c}}\underbrace{\mathrm{tr}(\mathbf{X}\mathbf{\Delta}_{\mathrm{c}}\mathbf{X}^\top)}_{\|\mathbf{X}\|_{\mathcal{G}_{\mathrm{c}}}^2} + \mu_{\mathrm{r}}\underbrace{\mathrm{tr}(\mathbf{X}^\top\mathbf{\Delta}_{\mathrm{r}}\mathbf{X})}_{\|\mathbf{X}\|_{\mathcal{G}_{\mathrm{r}}}^2}$$

Coifman, Gavish 2011; Kalofolias et al. 2014

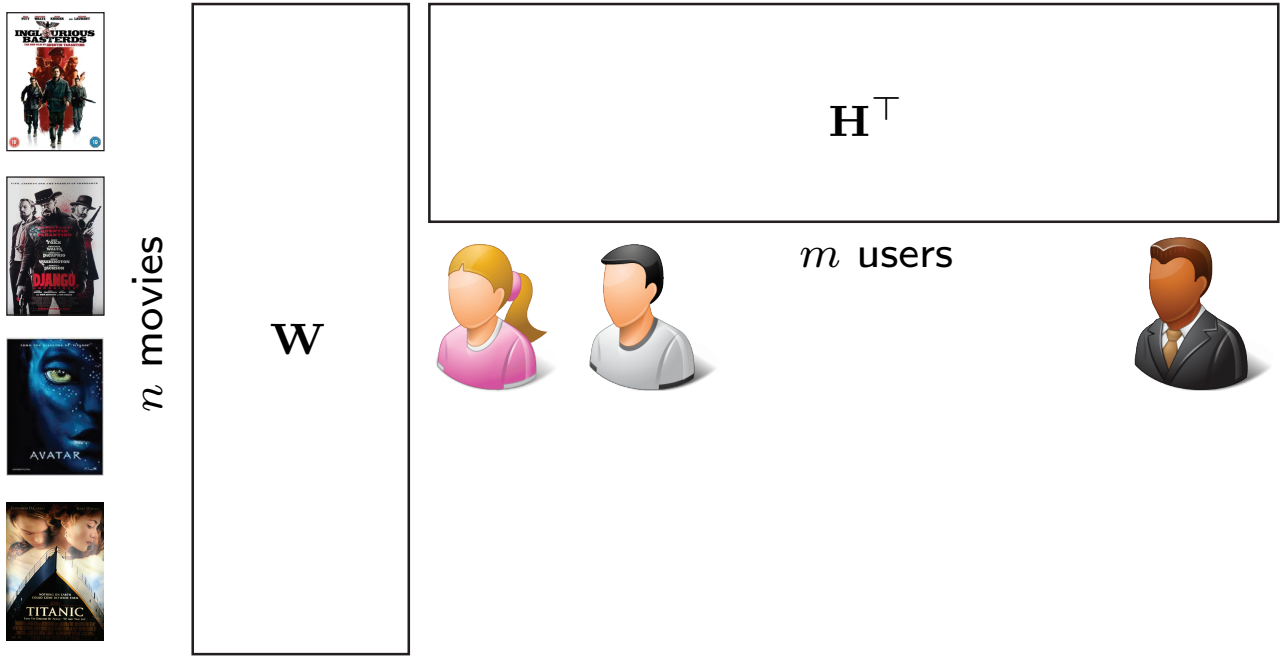
Factorized matrix completion models



$$\min_{\substack{\mathbf{W}\in\mathbb{R}^{m\times s}\\ \mathbf{H}\in\mathbb{R}^{n\times s}}} \quad \mu\|\mathbf{\Omega}\circ(\mathbf{X}-\mathbf{A})\|_{\mathrm{F}}^2+\mu_{\mathrm{c}}\|\mathbf{W}\|_{\mathrm{F}}^2+\mu_{\mathrm{r}}\|\mathbf{H}\|_{\mathrm{F}}^2$$

Srebro, Rennie, Jaakkola 2004

Factorized matrix completion models

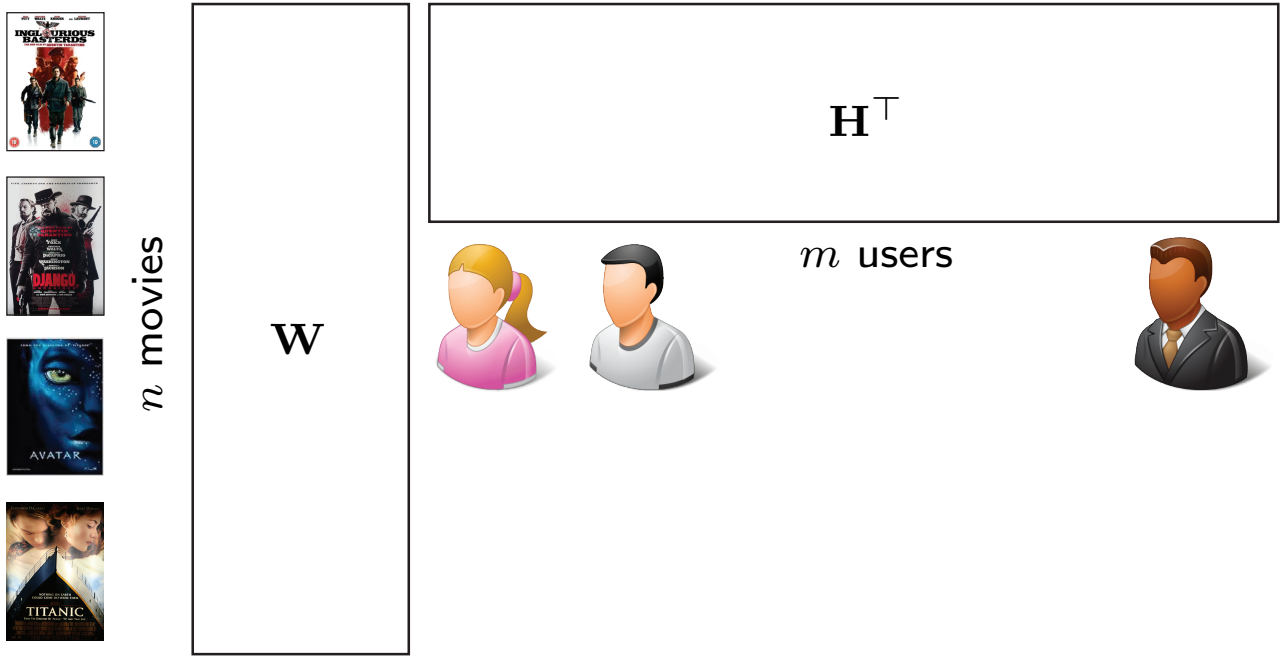


$$\min_{\substack{\mathbf{W}\in\mathbb{R}^{m\times s}\\ \mathbf{H}\in\mathbb{R}^{n\times s}}} \quad \mu\|\mathbf{\Omega}\circ(\mathbf{X}-\mathbf{A})\|_{\mathrm{F}}^2+\mu_{\mathrm{c}}\|\mathbf{W}\|_{\mathrm{F}}^2+\mu_{\mathrm{r}}\|\mathbf{H}\|_{\mathrm{F}}^2$$

☺ $\mathcal{O}(n+m)$ variables instead of $\mathcal{O}(nm)$

Srebro, Rennie, Jaakkola 2004

Factorized matrix completion models



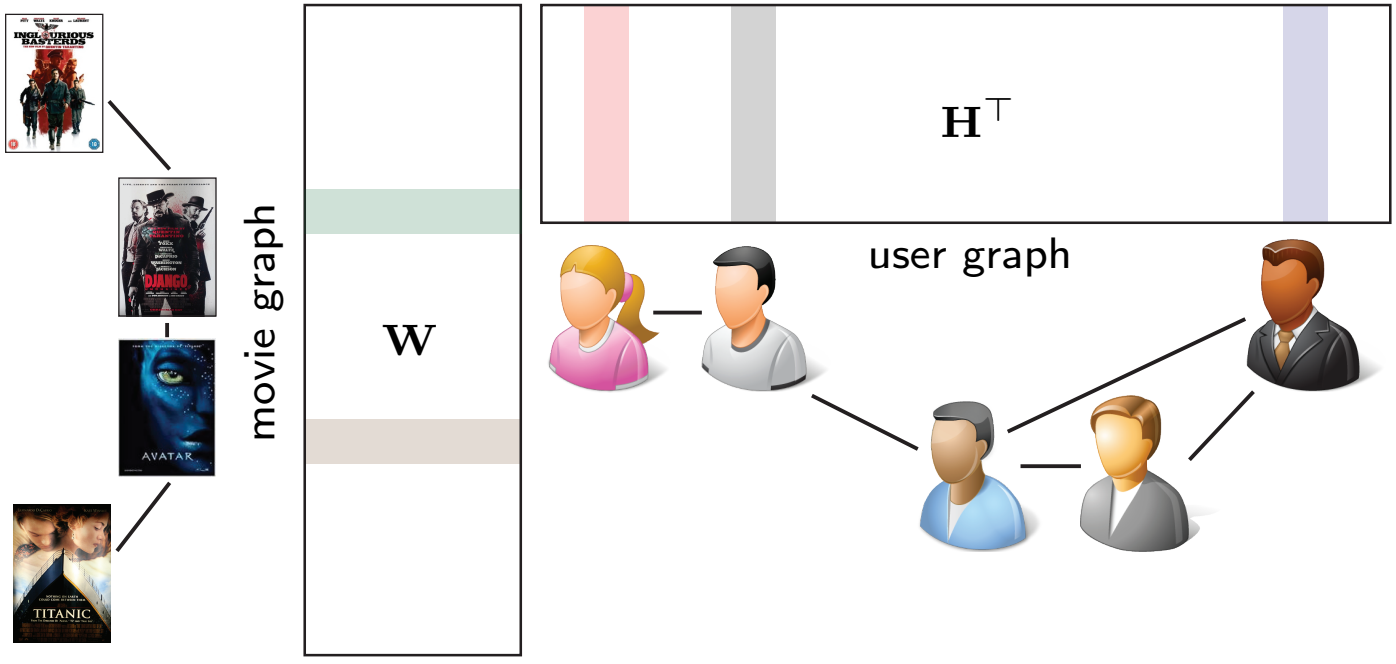
$$\min_{\substack{\mathbf{W}\in\mathbb{R}^{m\times s}\\ \mathbf{H}\in\mathbb{R}^{n\times s}}} \quad \mu\|\mathbf{\Omega}\circ(\mathbf{X}-\mathbf{A})\|_{\mathrm{F}}^2+\mu_{\mathrm{c}}\|\mathbf{W}\|_{\mathrm{F}}^2+\mu_{\mathrm{r}}\|\mathbf{H}\|_{\mathrm{F}}^2$$

☺ $\mathcal{O}(n+m)$ variables instead of $\mathcal{O}(nm)$

☺ $\mathrm{rank}(\mathbf{X})=\mathrm{rank}(\mathbf{W}\mathbf{H}^\top)\leq s$ by construction

Srebro, Rennie, Jaakkola 2004

Factorized geometric matrix completion models



$$\min_{\substack{\mathbf{W}\in\mathbb{R}^{m\times s}\\ \mathbf{H}\in\mathbb{R}^{n\times s}}} \quad \mu\|\mathbf{\Omega}\circ(\mathbf{X}-\mathbf{A})\|_{\mathrm{F}}^2+\mu_{\mathrm{c}}\mathrm{tr}(\mathbf{H}^\top\mathbf{\Delta}_{\mathrm{c}}\mathbf{H})+\mu_{\mathrm{r}}\mathrm{tr}(\mathbf{W}^\top\mathbf{\Delta}_{\mathrm{r}}\mathbf{W})$$

☺ $\mathcal{O}(n+m)$ variables instead of $\mathcal{O}(nm)$

☺ $\mathrm{rank}(\mathbf{X})=\mathrm{rank}(\mathbf{W}\mathbf{H}^\top)\leq s$ by construction

Rao et al. 2015

How to learn filters on multiple graphs?

2D Fourier transform



$\mathbf{X}$

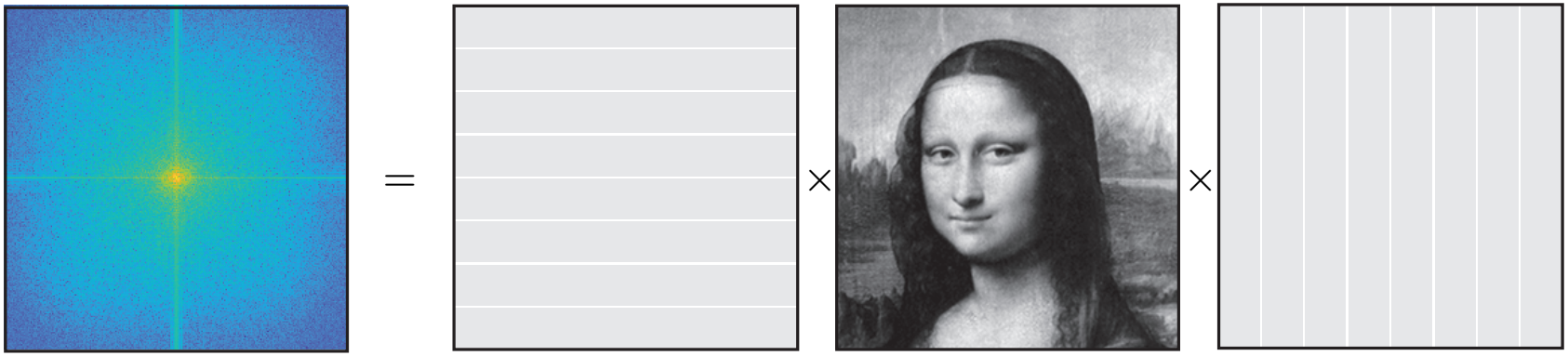
2D Fourier transform



$(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)$ $\quad = \quad$ $\boldsymbol{\Phi}^\top$ $\quad \times \quad$ $\mathbf{X}$

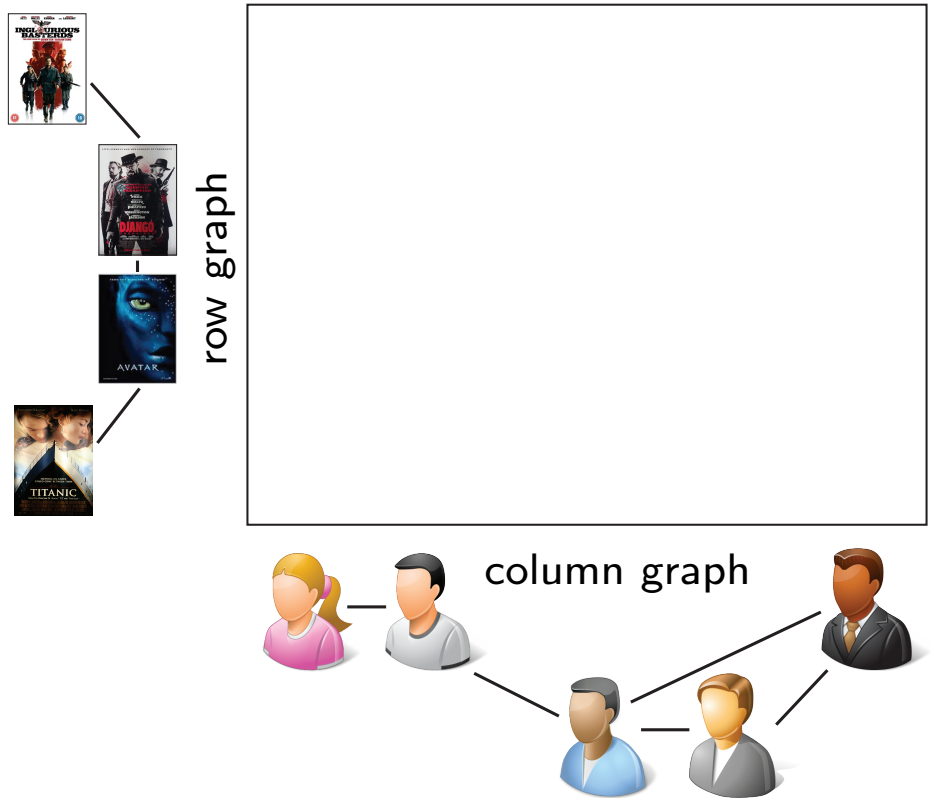
Column-wise trasform

2D Fourier transform



Column-wise trasform + Row-wise transform = 2D transform

Multi-graph Fourier transform



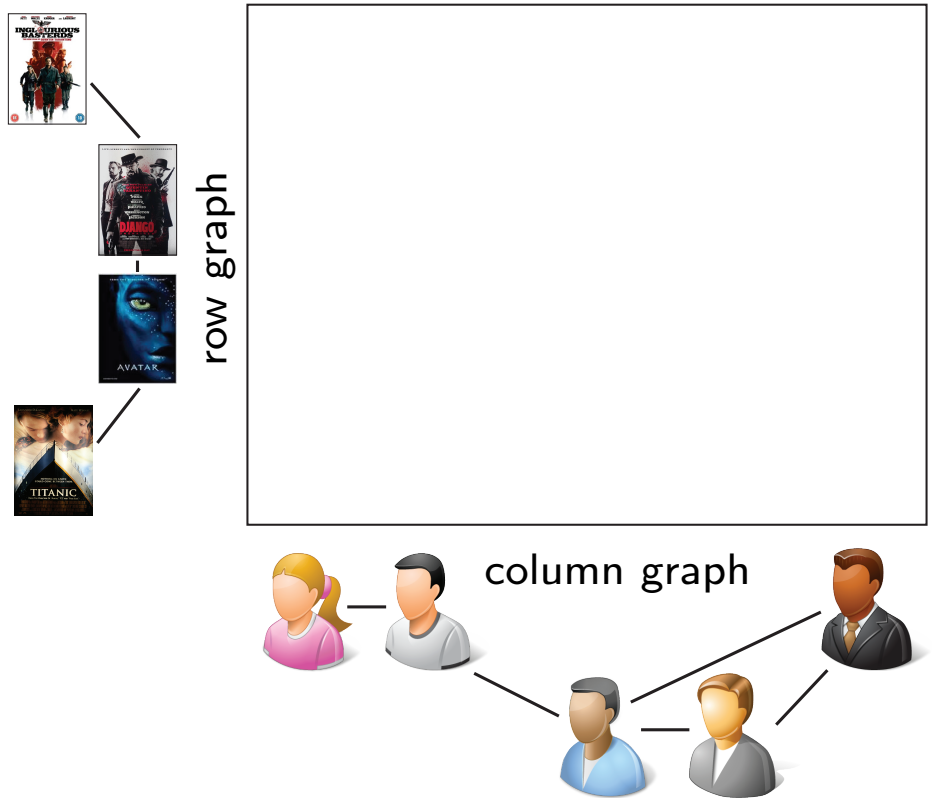
Multi-graph Fourier transform

$$\hat{\mathbf{X}} = \mathbf{\Phi}_{\mathrm{r}}^{\top} \mathbf{X} \mathbf{\Phi}_{\mathrm{c}}$$

where $\mathbf{\Phi}_{\mathrm{c}}$ and $\mathbf{\Phi}_{\mathrm{r}}$ are the eigenvectors of the column- and row-graph Laplacians $\mathbf{\Delta}_{\mathrm{c}}$ and $\mathbf{\Delta}_{\mathrm{r}}$, respectively

Sandryhaila, Moura 2014; Monti, Bresson, Bronstein 2017

Multi-graph convolution

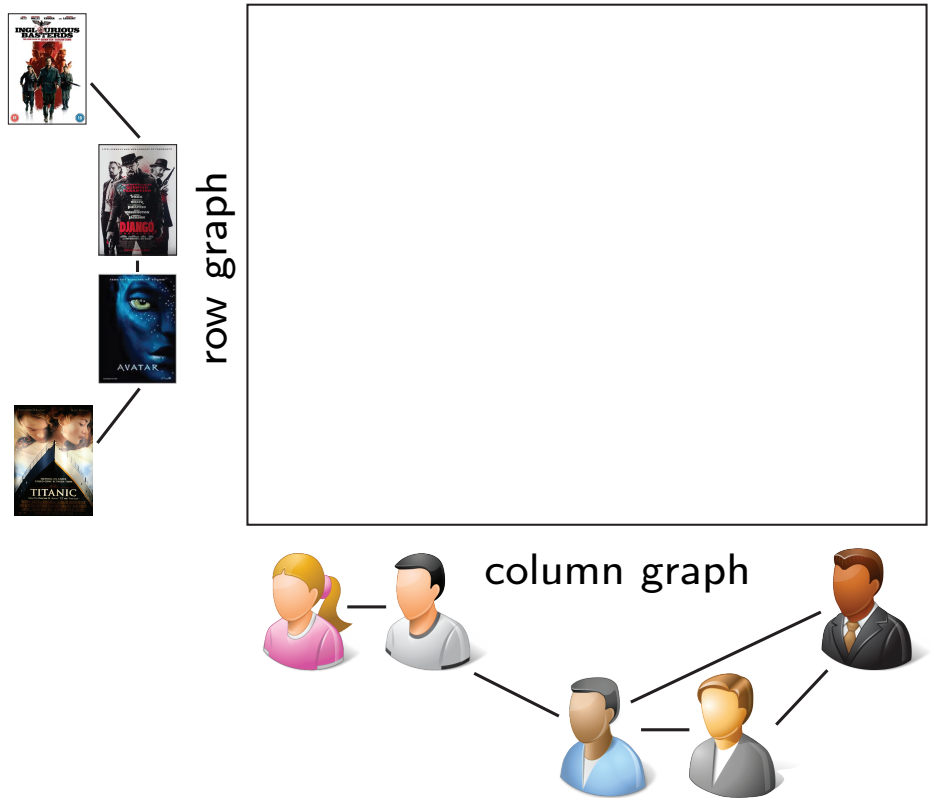


Multi-graph spectral convolution

$$\mathbf{X} \star \mathbf{G} = \mathbf{\Phi}_{\mathrm{r}}((\mathbf{\Phi}_{\mathrm{r}}^{\top}\mathbf{X}\mathbf{\Phi}_{\mathrm{c}}) \circ (\mathbf{\Phi}_{\mathrm{r}}^{\top}\mathbf{G}\mathbf{\Phi}_{\mathrm{c}}))\mathbf{\Phi}_{\mathrm{c}}^{\top}$$

Sandryhaila, Moura 2014; Monti, Bresson, Bronstein 2017

Multi-graph convolution

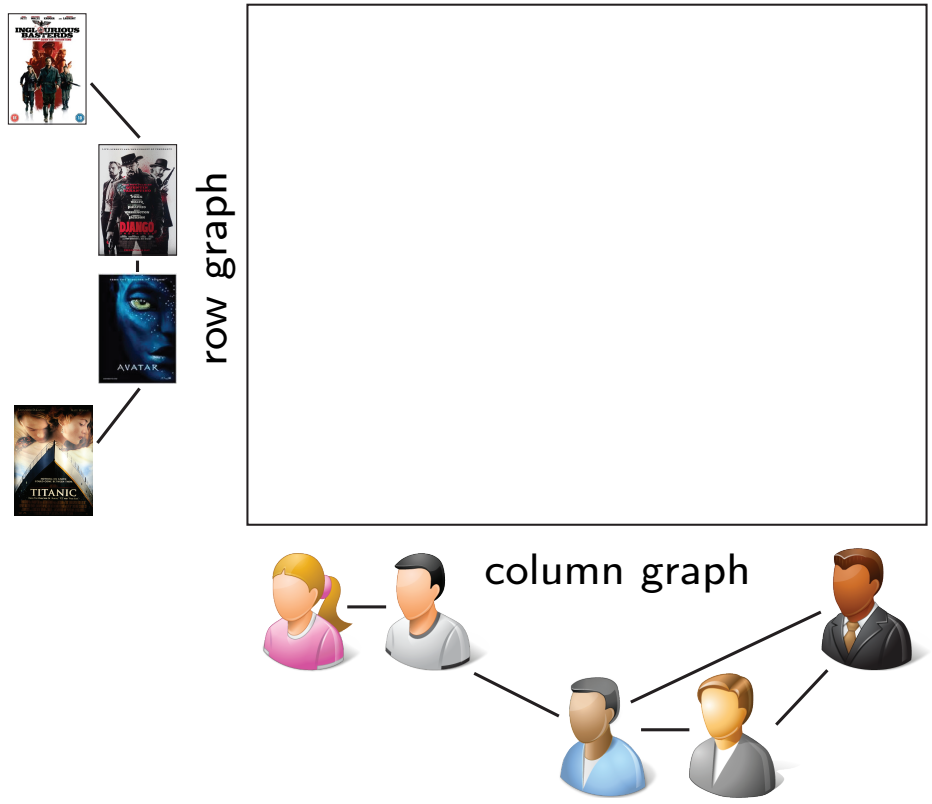


Multi-graph spectral convolution

$$\mathbf{X} \star \mathbf{G} = \mathbf{\Phi}_{\mathrm{r}}(\hat{\mathbf{X}} \circ \hat{\mathbf{G}})\mathbf{\Phi}_{\mathrm{c}}^{\top}$$

Sandryhaila, Moura 2014; Monti, Bresson, Bronstein 2017

Multi-graph spectral filters



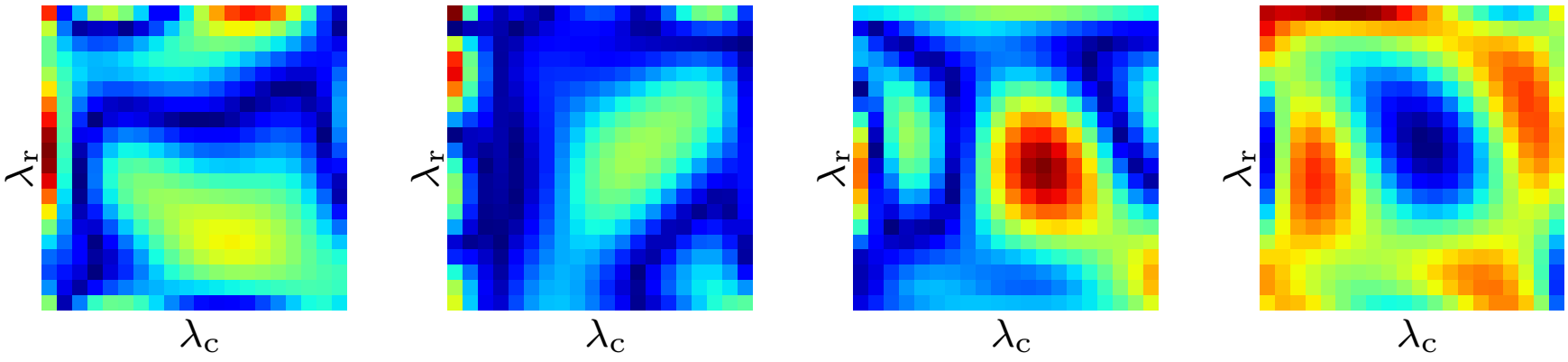
Multi-graph bi-variate polynomial filter

$$\mathbf{Y} = \tau_{\mathbf{\Theta}}(\mathbf{X}) = \sum_{j,j'=1}^{r} \theta_{jj'} \mathbf{\Delta}_{\mathrm{r}}^{j} \mathbf{X} \mathbf{\Delta}_{\mathrm{c}}^{j'}$$

where $\mathbf{\Theta} = (\theta_{jj'})$ is the $r \times r$ matrix of filter parameters

Sandryhaila, Moura 2014; Monti, Bresson, Bronstein 2017

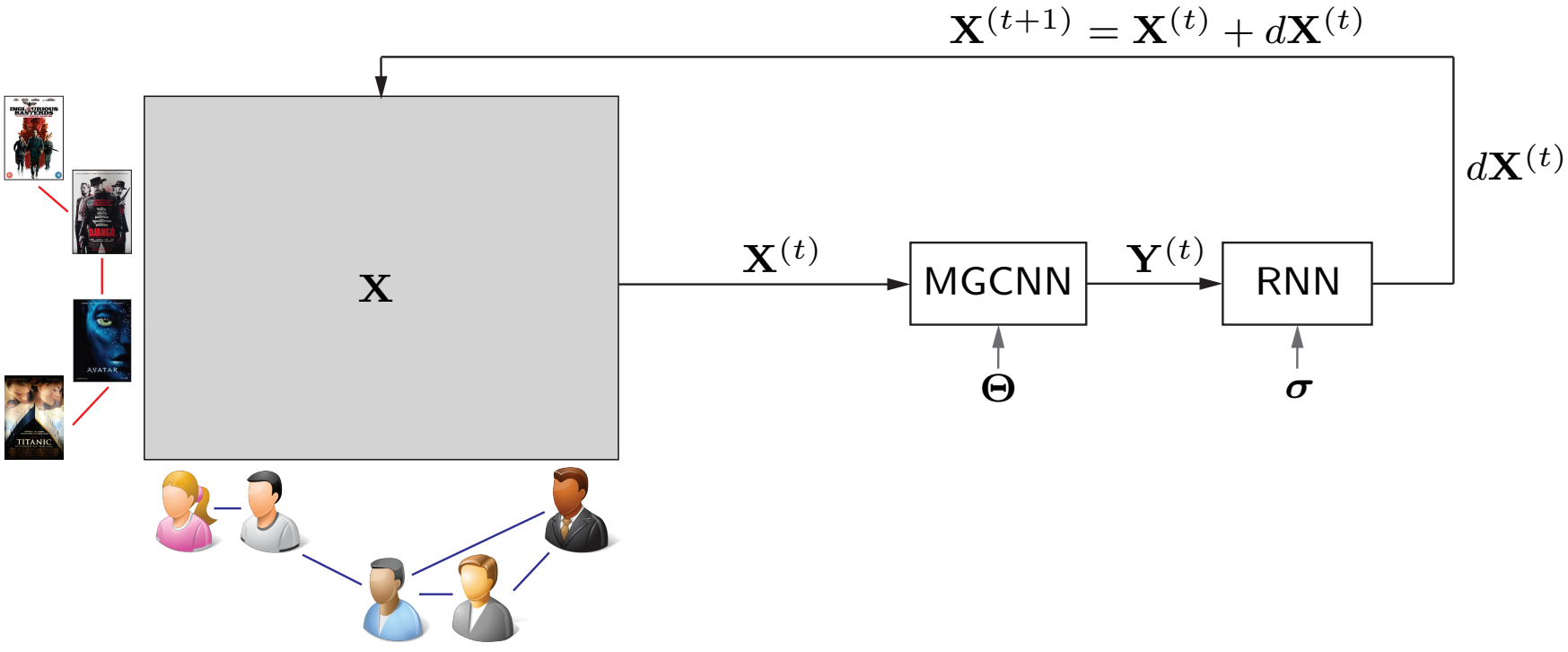
Multi-graph spectral filters



Examples of multi-graph spectral filters (shown is $|\tau(\lambda_{\mathrm{c}}, \lambda_{\mathrm{r}})|$)

Monti, Bresson, Bronstein 2017

Matrix completion with Recurrent Multi-Graph CNN

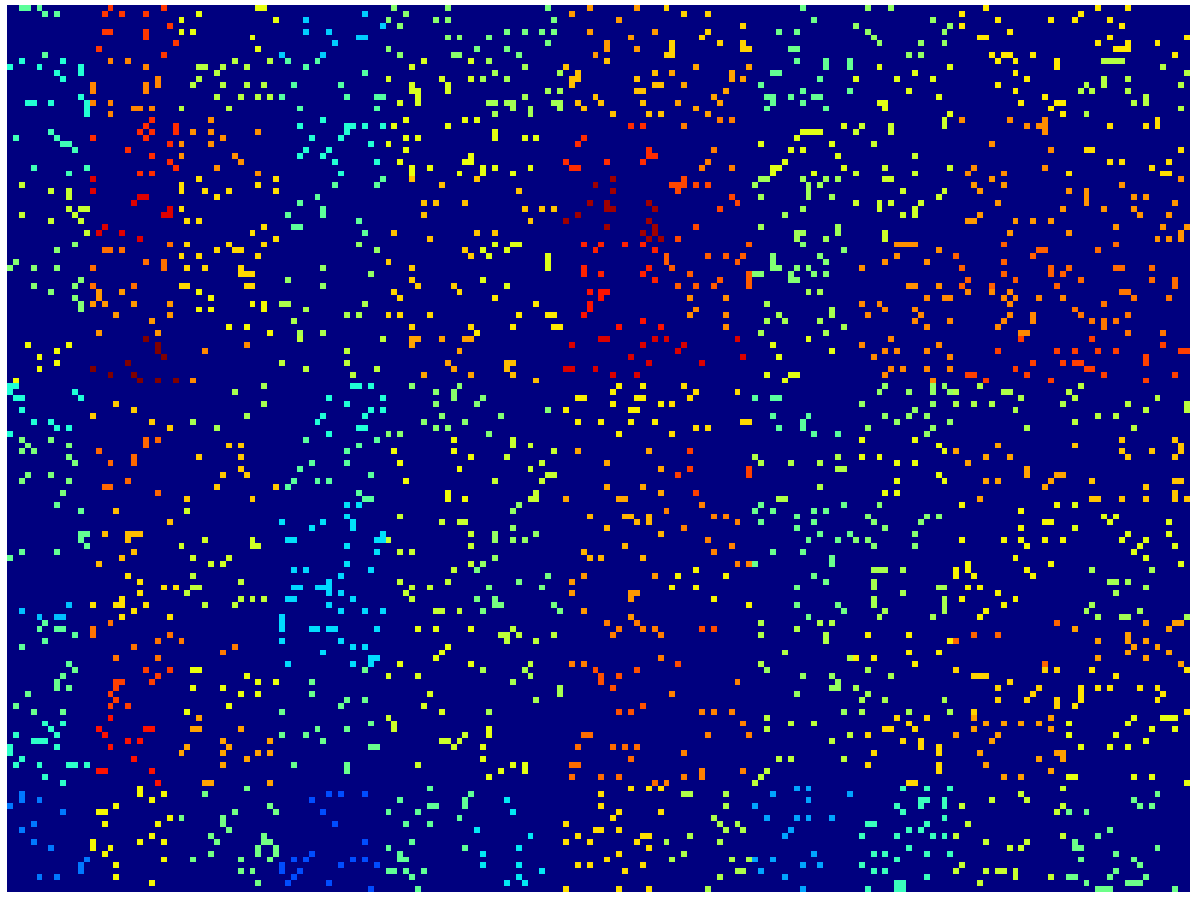


Recurrent multigraph CNN (RMCNN) architecture for matrix completion

$$\min_{\boldsymbol{\Theta},\boldsymbol{\sigma}} \|\mathbf{X}^{(T)}_{\boldsymbol{\Theta},\boldsymbol{\sigma}}\|^2_{\mathcal{G}_\mathrm{r}} + \|\mathbf{X}^{(T)}_{\boldsymbol{\Theta},\boldsymbol{\sigma}}\|^2_{\mathcal{G}_\mathrm{c}} + \frac{\mu}{2}\|\boldsymbol{\Omega} \circ (\mathbf{X}^{(T)}_{\boldsymbol{\Theta},\boldsymbol{\sigma}} - \mathbf{A})\|^2_\mathrm{F}$$

Monti, Bresson, Bronstein 2017

Incremental updates with RNN

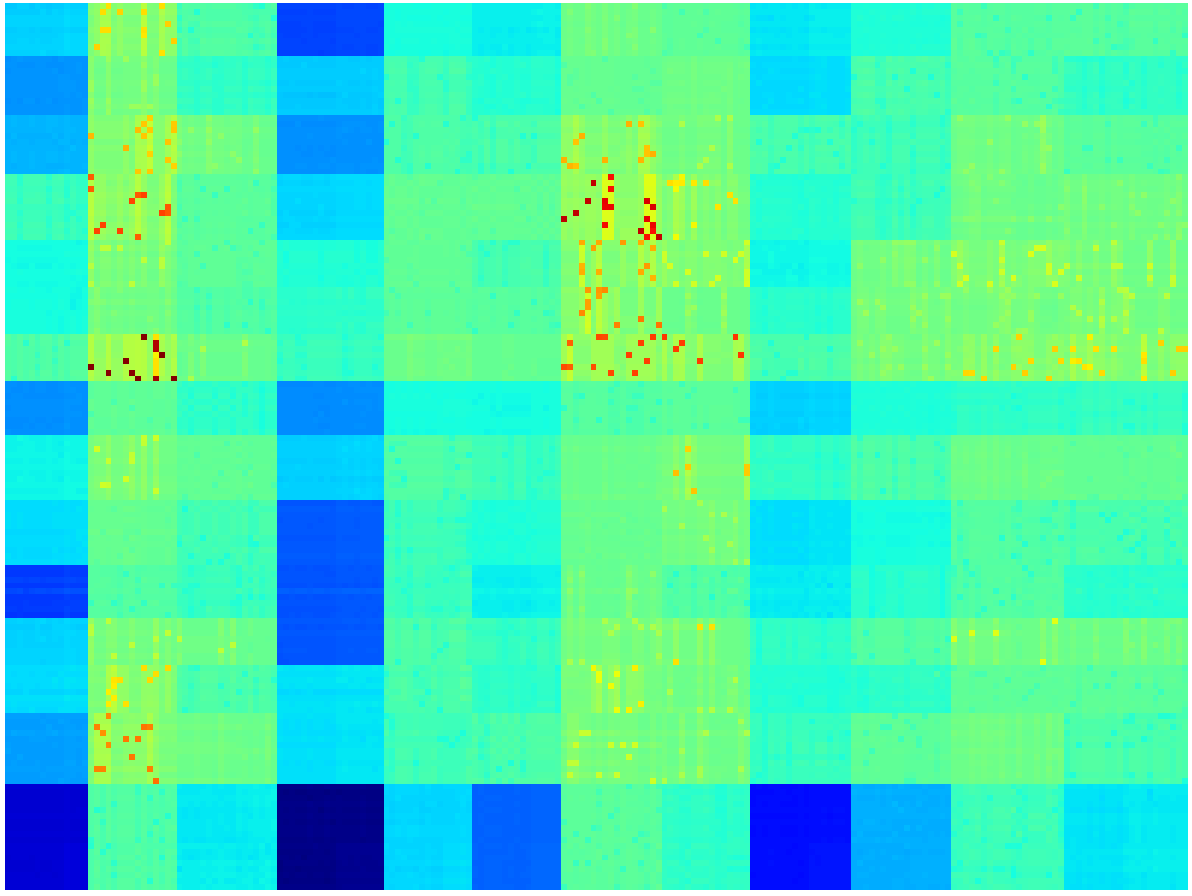


RMSE=2.26

Matrix completion results on a synthetic dataset (initialization)

Monti, Bresson, Bronstein 2017

Incremental updates with RNN

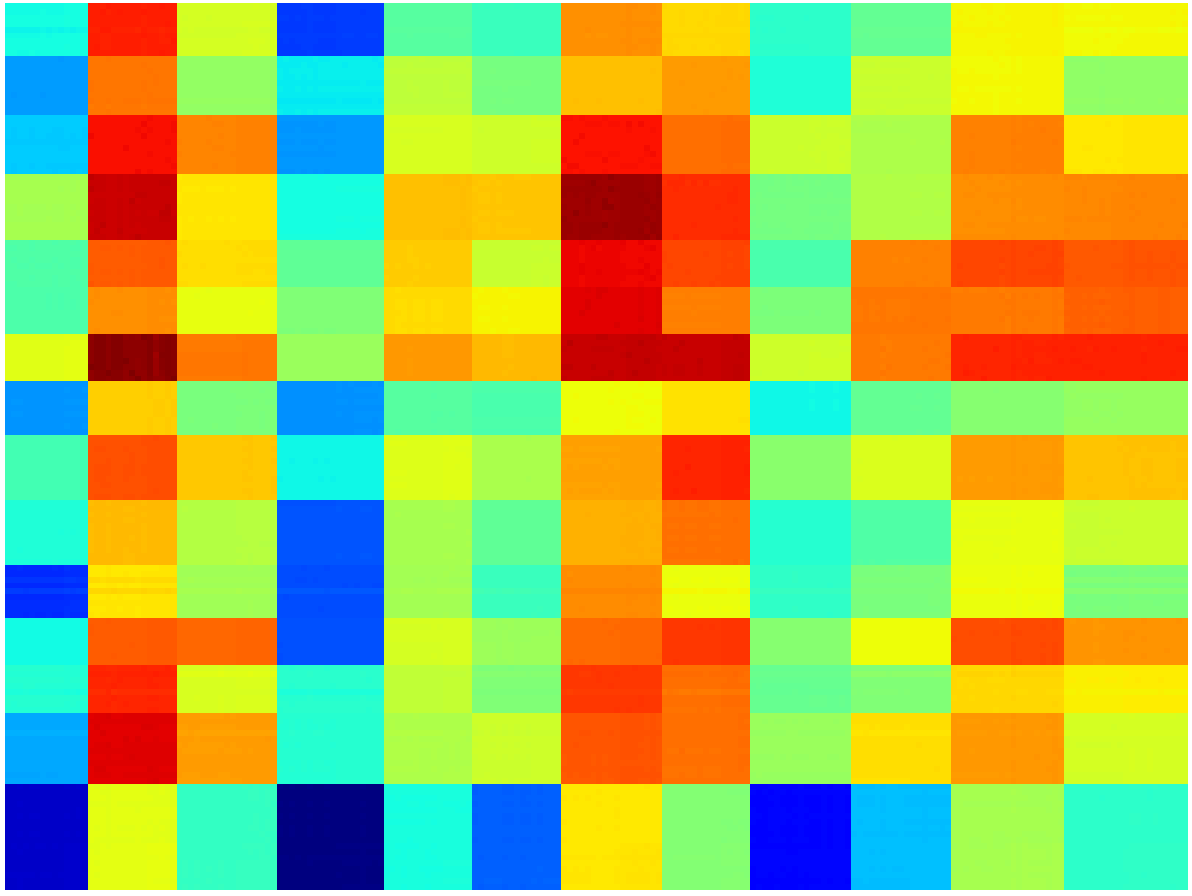


RMSE=0.52

Matrix completion results on a synthetic dataset after $t = 5$ instances

Monti, Bresson, Bronstein 2017

Incremental updates with RNN

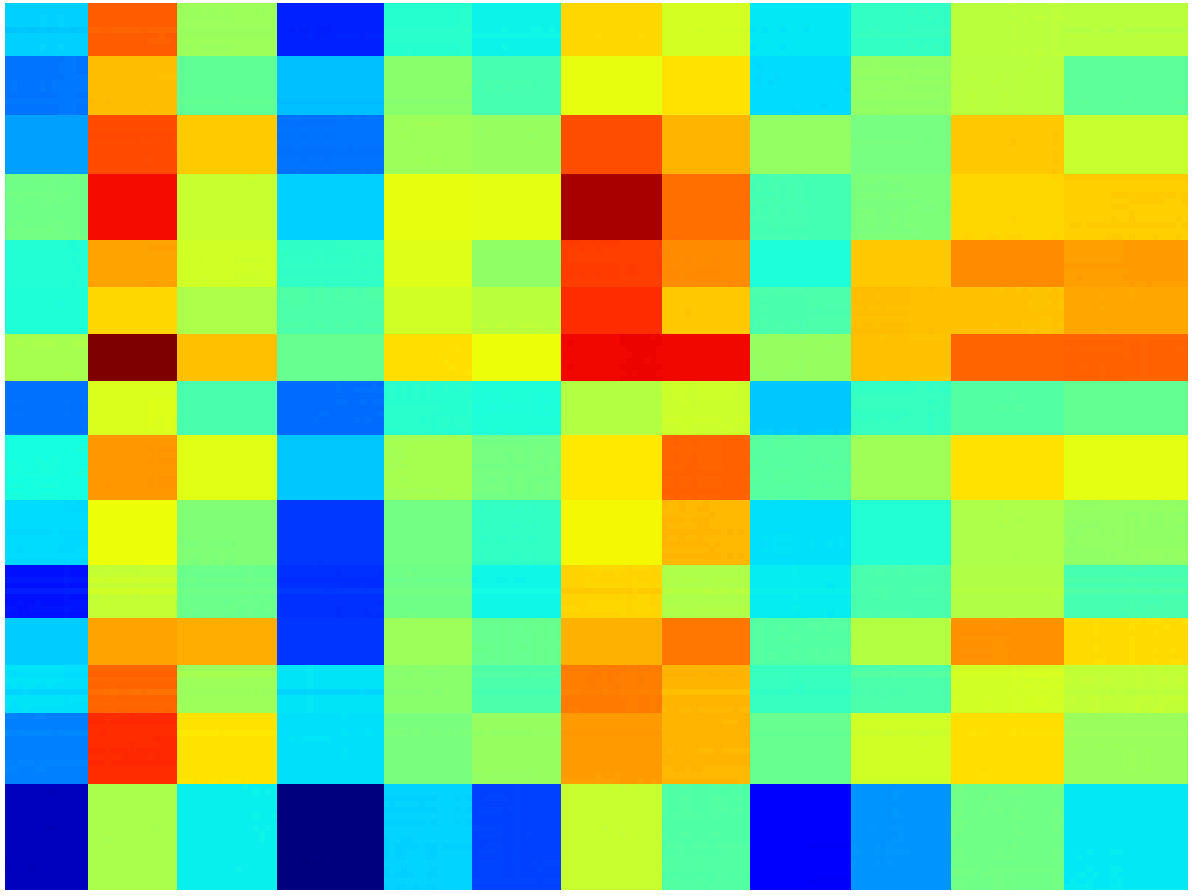


RMSE=0.38

Matrix completion results on a synthetic dataset after $t = 8$ instances

Monti, Bresson, Bronstein 2017

Incremental updates with RNN



RMSE=0.01

Matrix completion results on a synthetic dataset after $t = 10$ instances

Monti, Bresson, Bronstein 2017

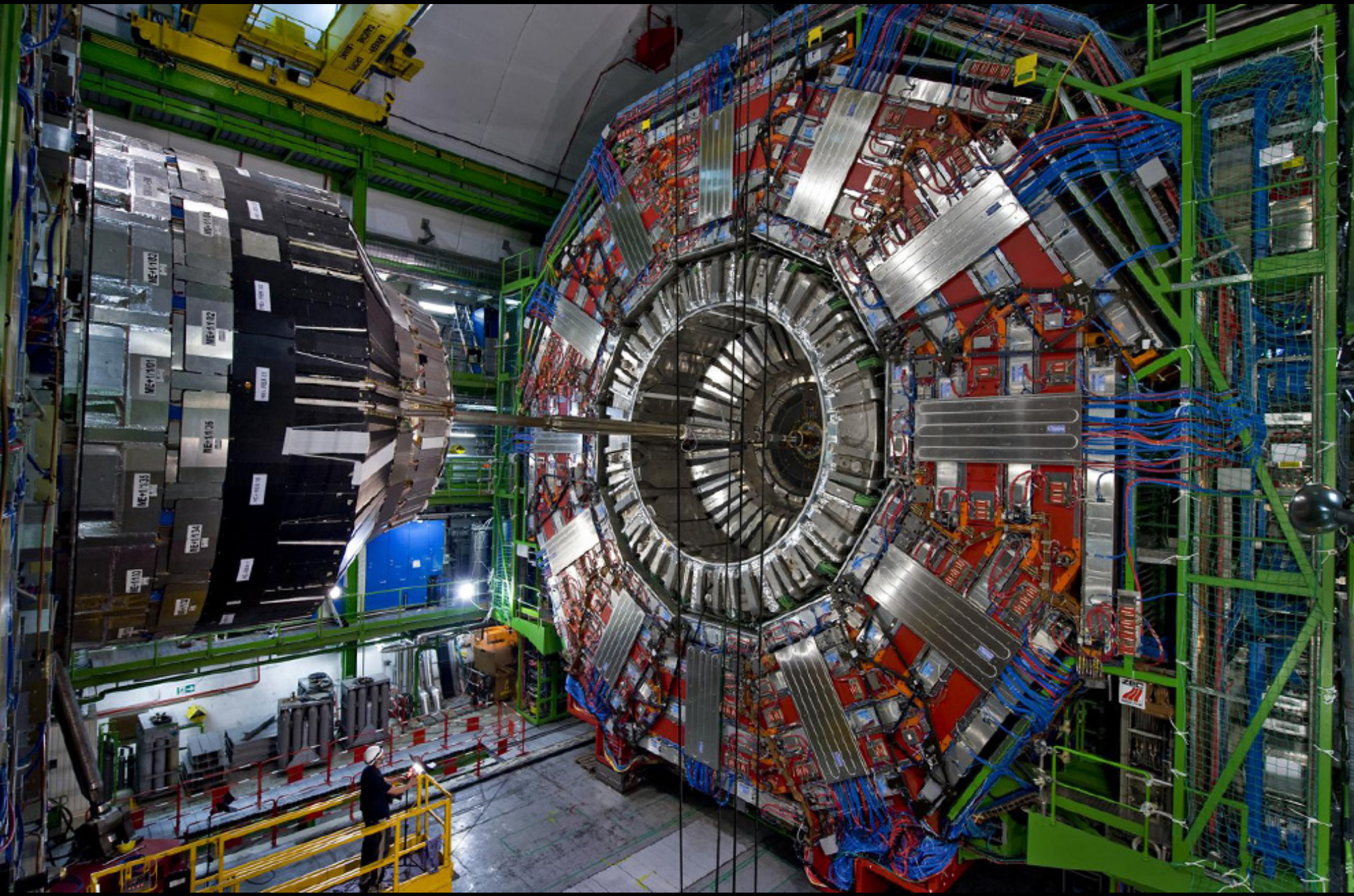
Matrix completion methods comparison

Method	**MovieLens**[1]	**Flixster**[2]	**Douban**[3]	**Yahoo**[4]
IMC[5]	1.653	–	–	–
GMC[6]	0.996	–	–	–
MC[7]	0.973	–	–	–
GRALS[8]	0.945	1.245	0.833	38.042
sRGCNN (Cheb)[9]	**0.929**	**0.926**	**0.801**	**22.415**
sRGCNN (Cayley)[10]	**0.922**	–	–	–
sRGCNN (Primal/Dual)[11]	**0.915**	**0.902**	**0.789**	**21.970**

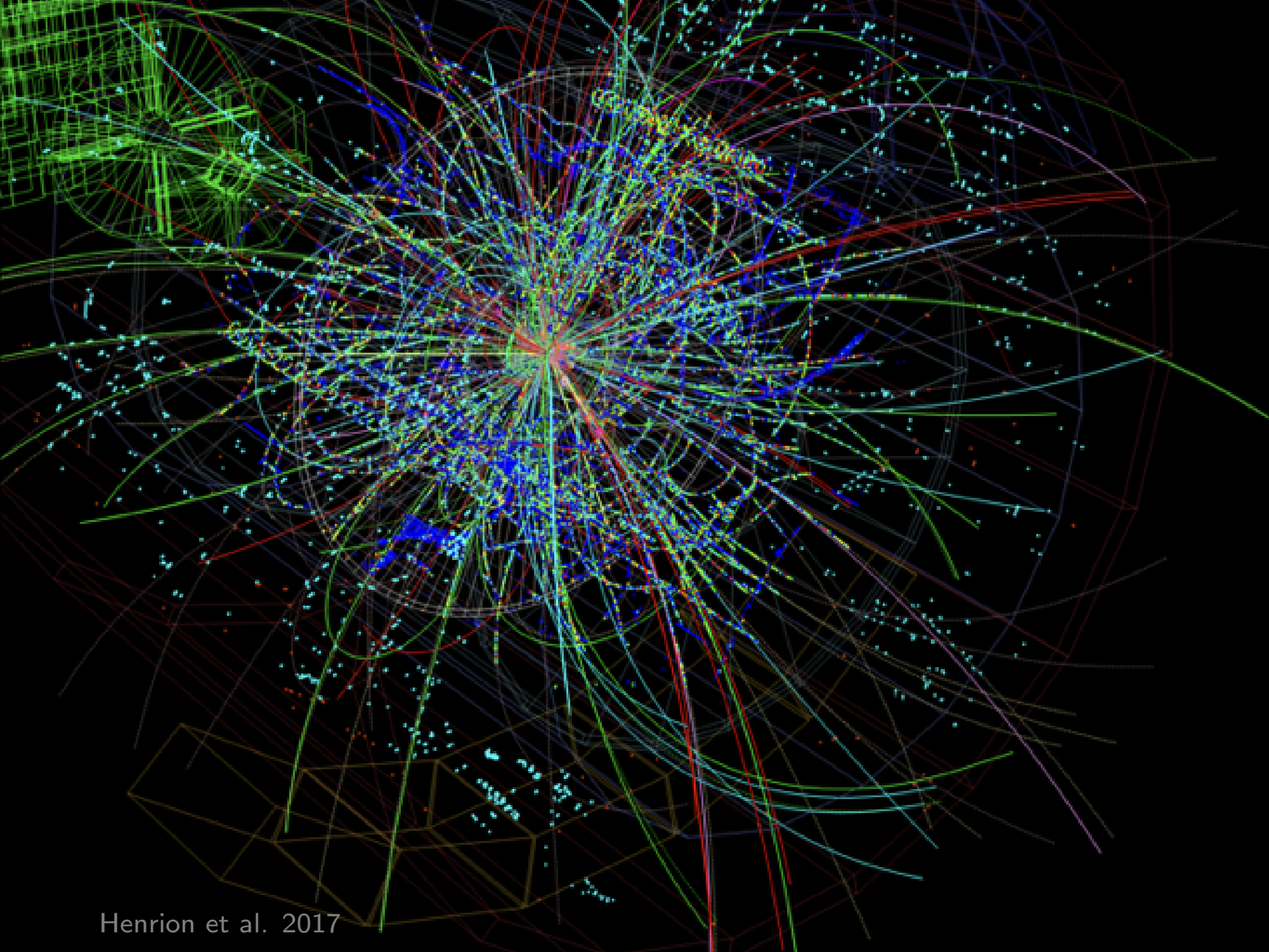
Accuracy (RMS error) of matrix completion methods on real data

Data: [1]Miller et al. 2003; [2]Jamali, Ester 2010; [3]Ma et al. 2011; [4]Dror et al. 2012
Methods: [5]Jain, Dhillon 2013; [6]Kalofolias et al. 2014; [7]Candès, Recht 2012; [8]Rao et al. 2015; [9]Monti, Bresson, Bronstein 2017; [10]Levie et al. 2017; [11]Monti et al. 2018

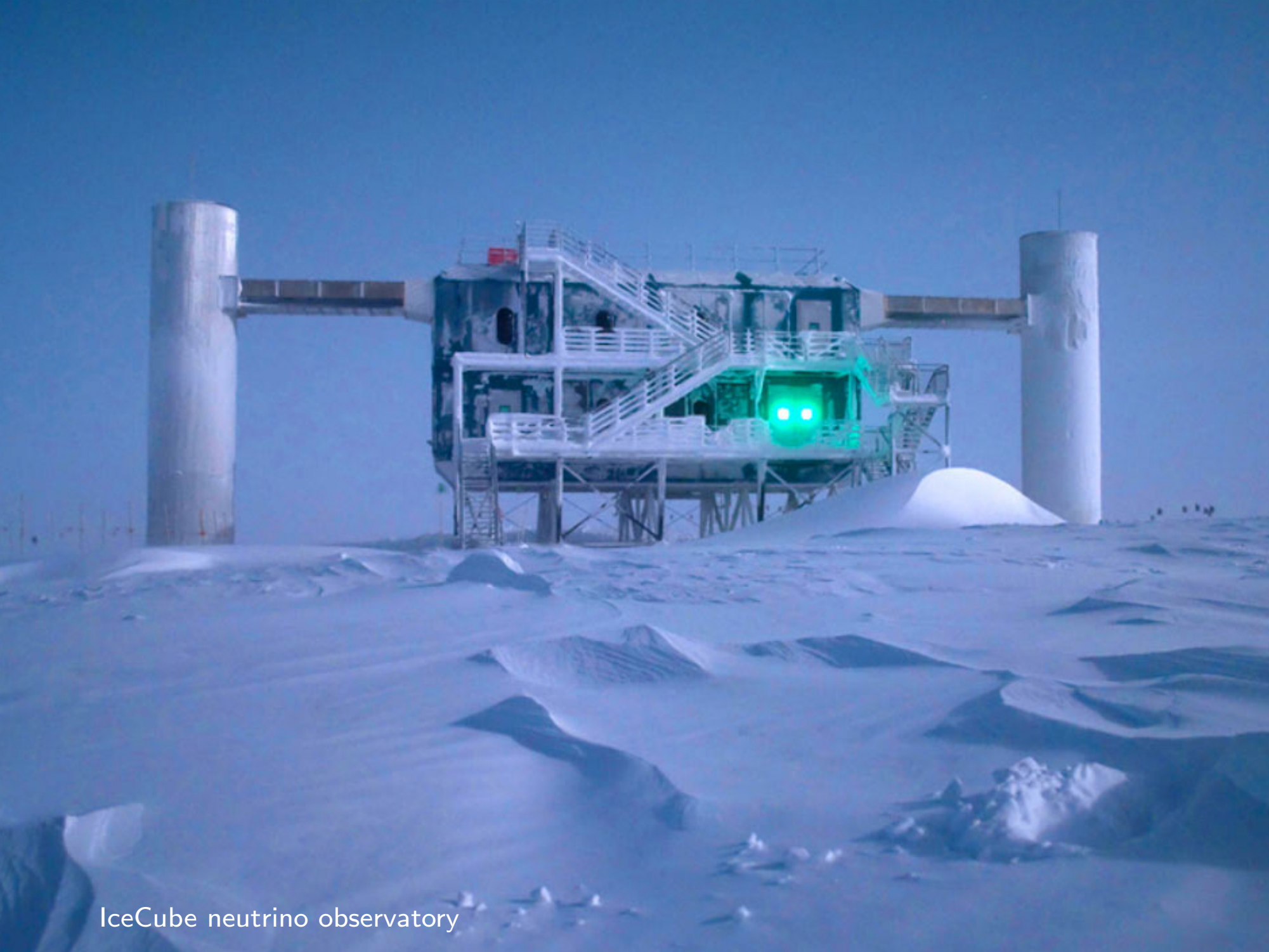
Application in High-Energy Physics



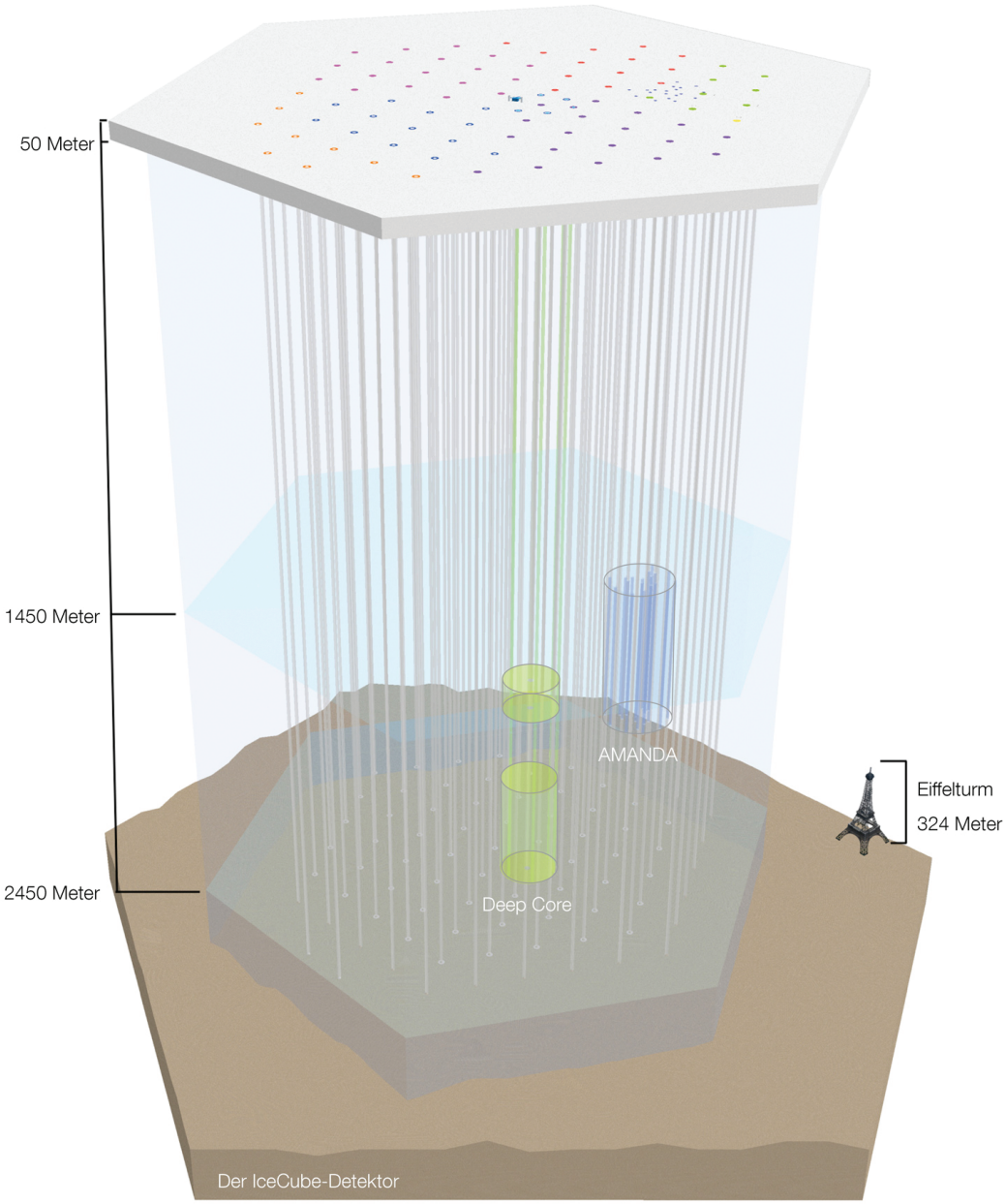
ATLAS detector in the Large Hadron Collider (CERN)



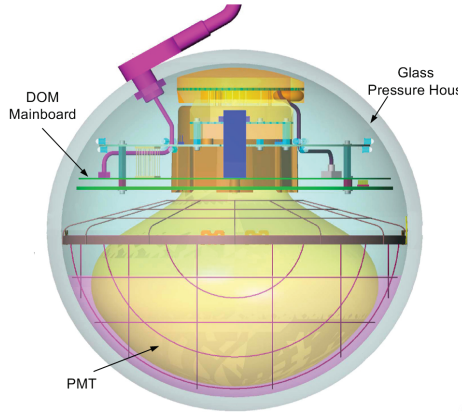
Henrion et al. 2017



IceCube neutrino observatory



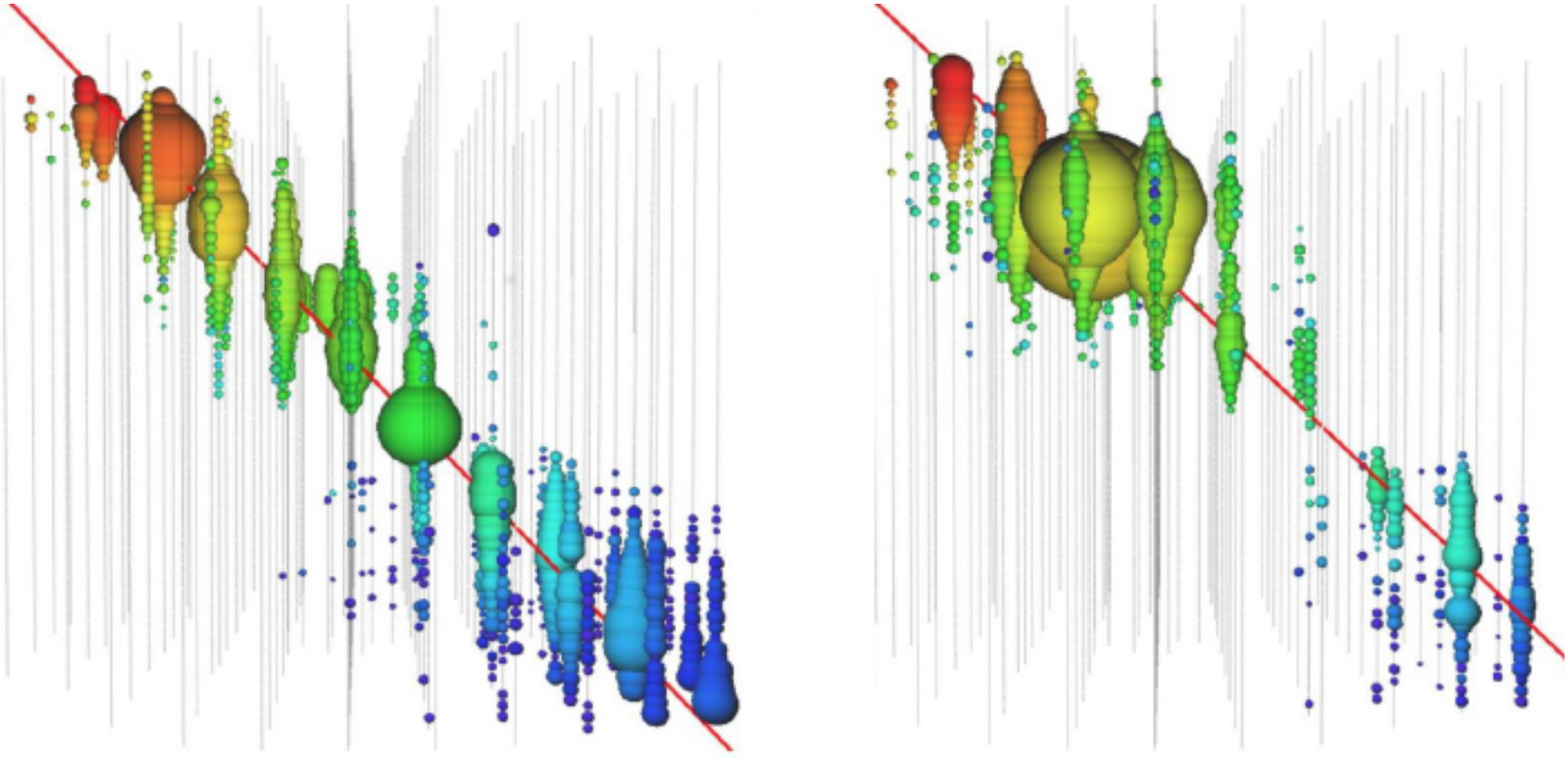






IceCube neutrino observatory

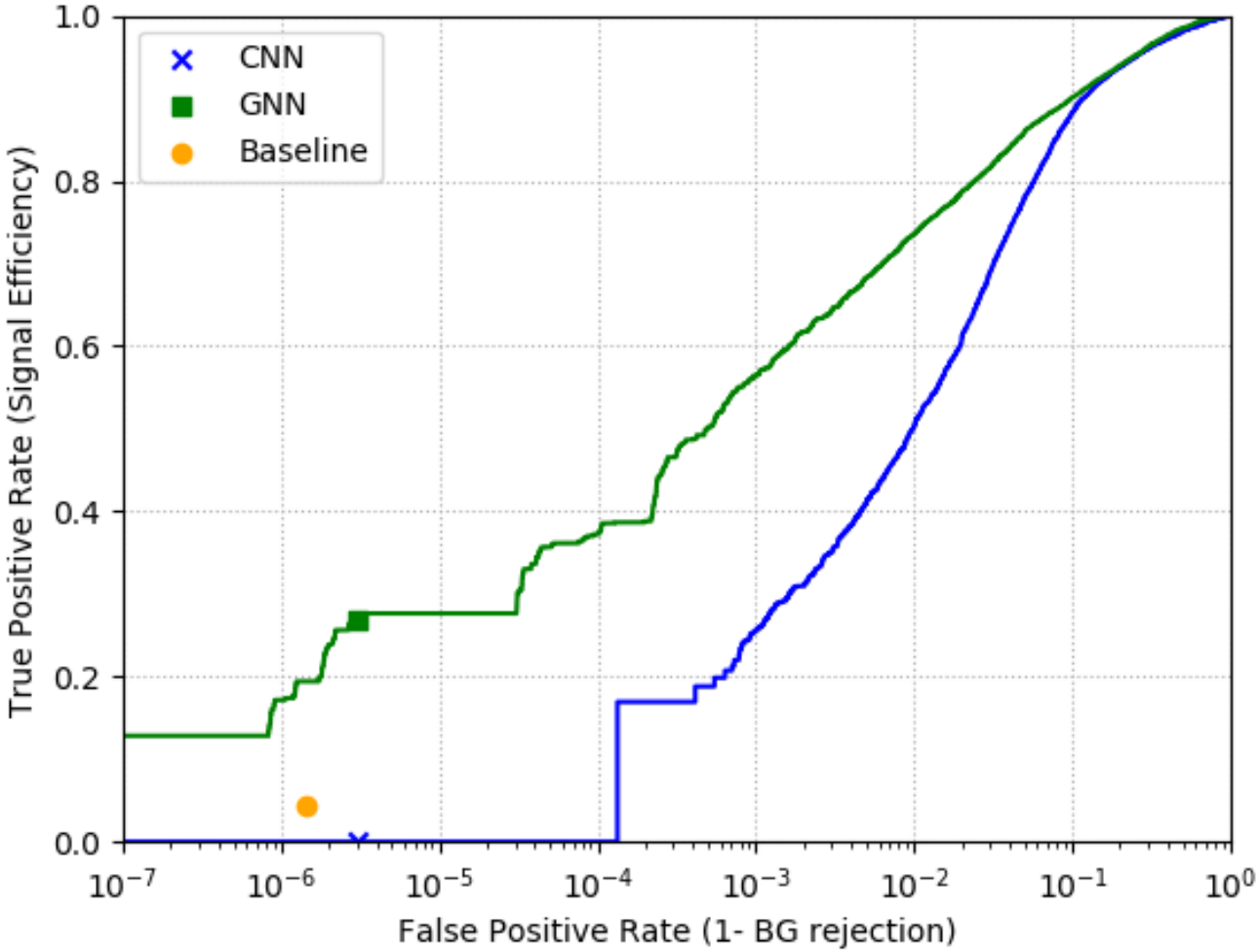
Neutrino detection in IceCube



Light deposition for muon bundles (left) and a high-energy single muon (right) in IceCube detector. Arrival time is coded in color; Number of observed photons is the bubble size.

Choma, Monti et al. 2018 for the IceCube collaboration
(joint work with NERSC+LBL, Berkeley)

Neutrino detection in IceCube

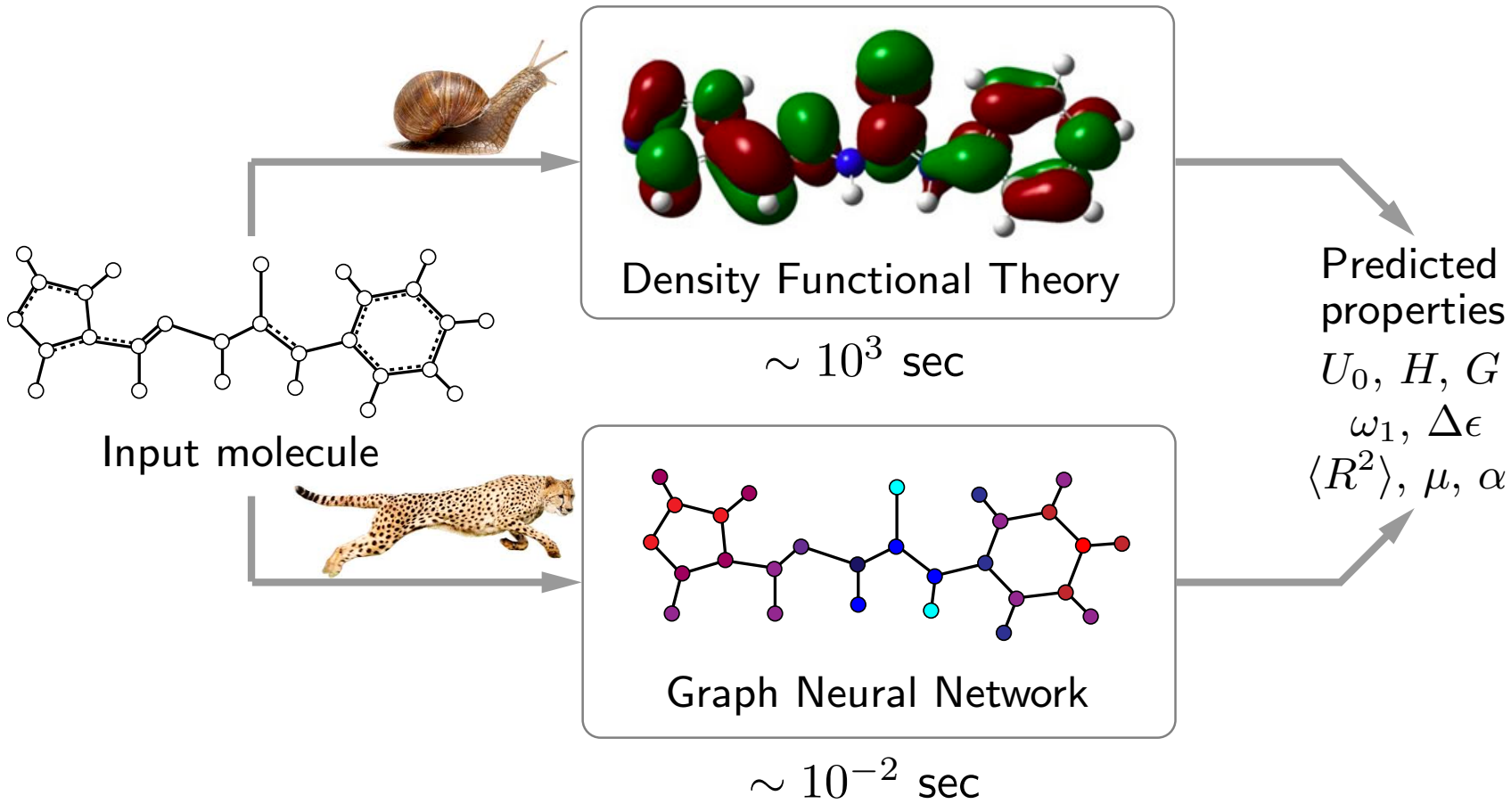


ROC curve comparing different methods for neutrino detection

Choma, Monti et al. 2018 for the IceCube collaboration
(joint work with NERSC+LBL, Berkeley)

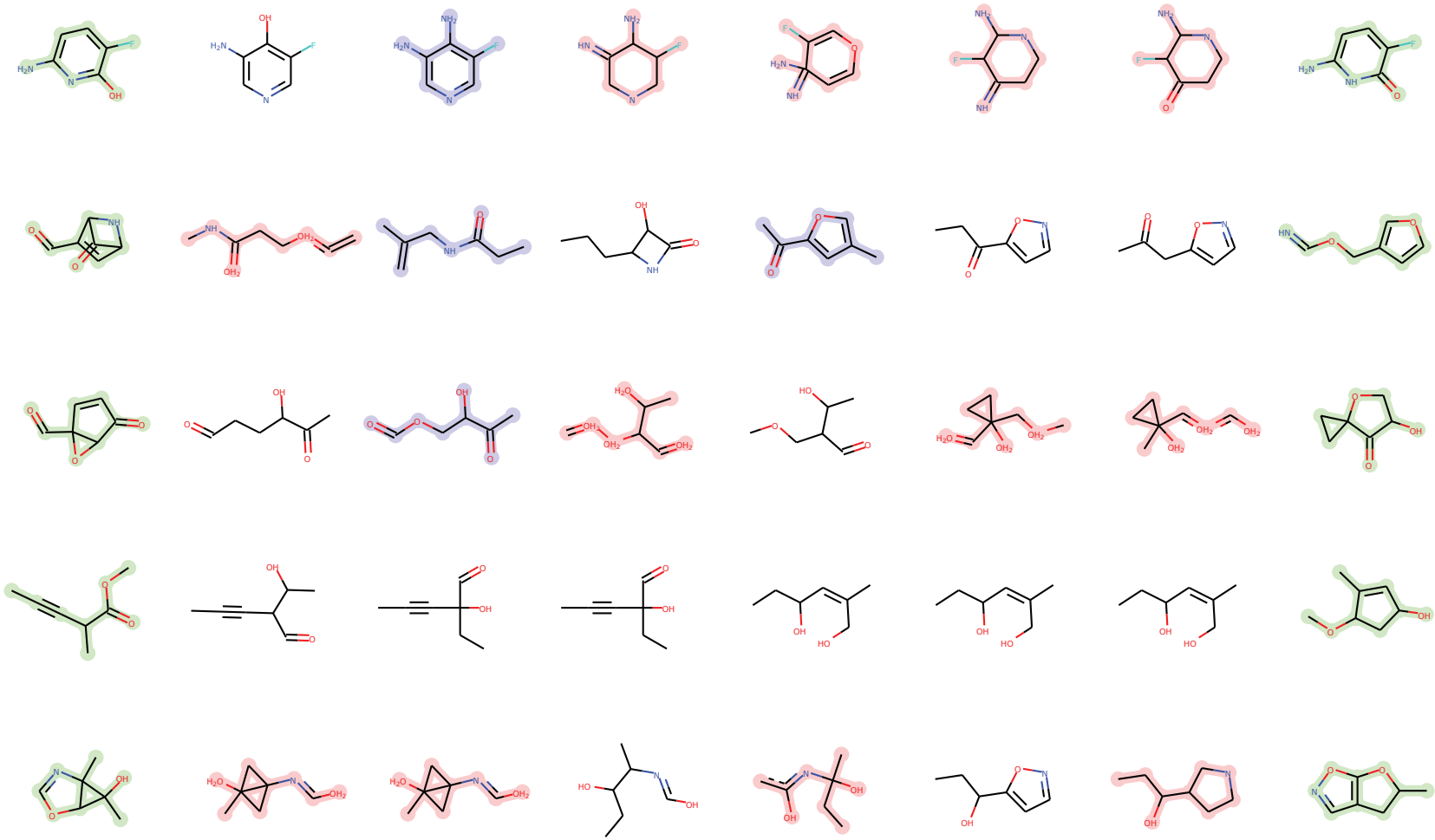
Applications in Chemistry and Drug Design

Molecule property prediction



Duvenaud et al. 2015; Gomez-Bombarelli et al. 2016; Gilmer et al. 2017

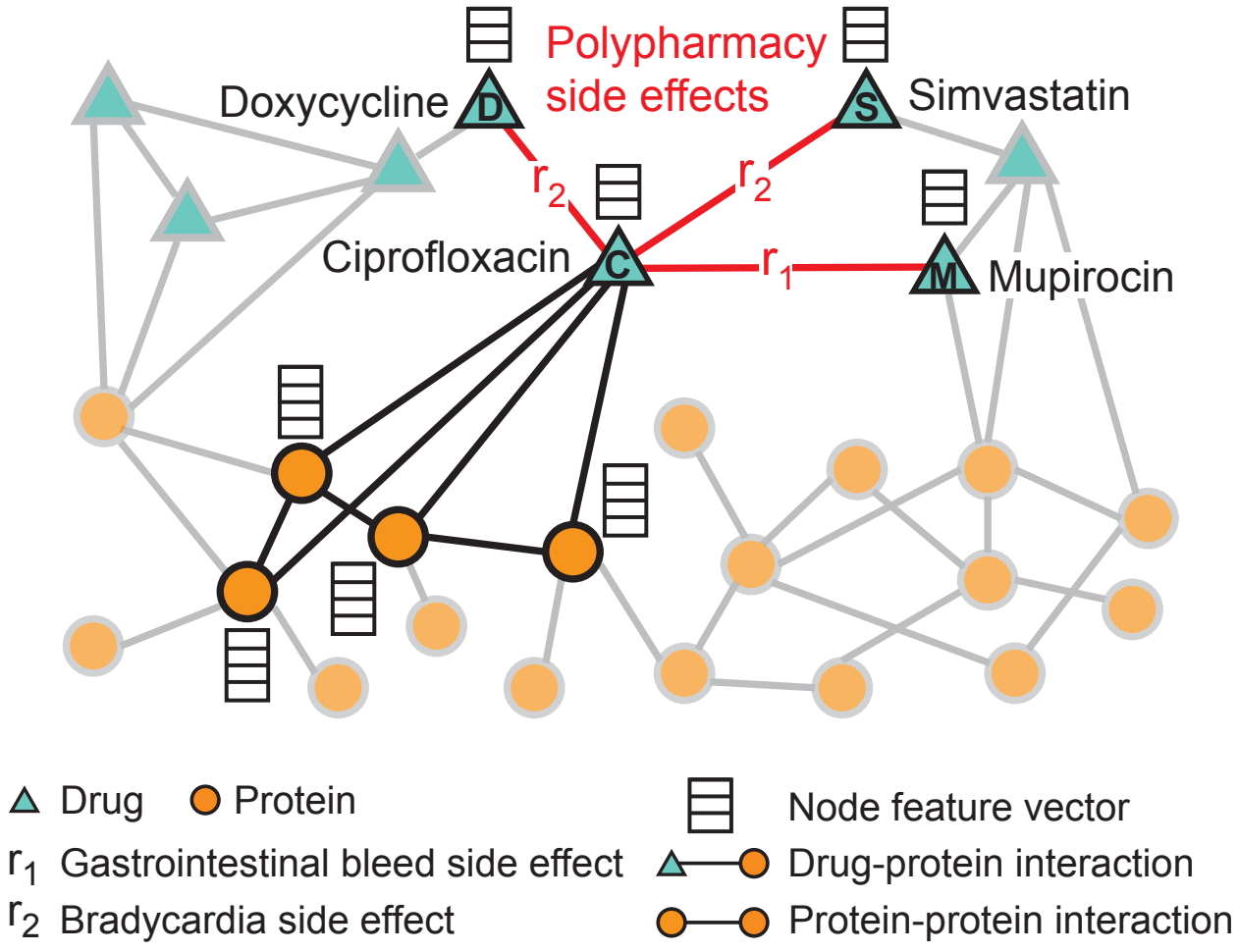
Molecule generation



Molecules generated with a graph VAE

Simonovsky, Komodakis 2017; You et al. 2018; De Cao, Kipf 2018 (MolGAN)

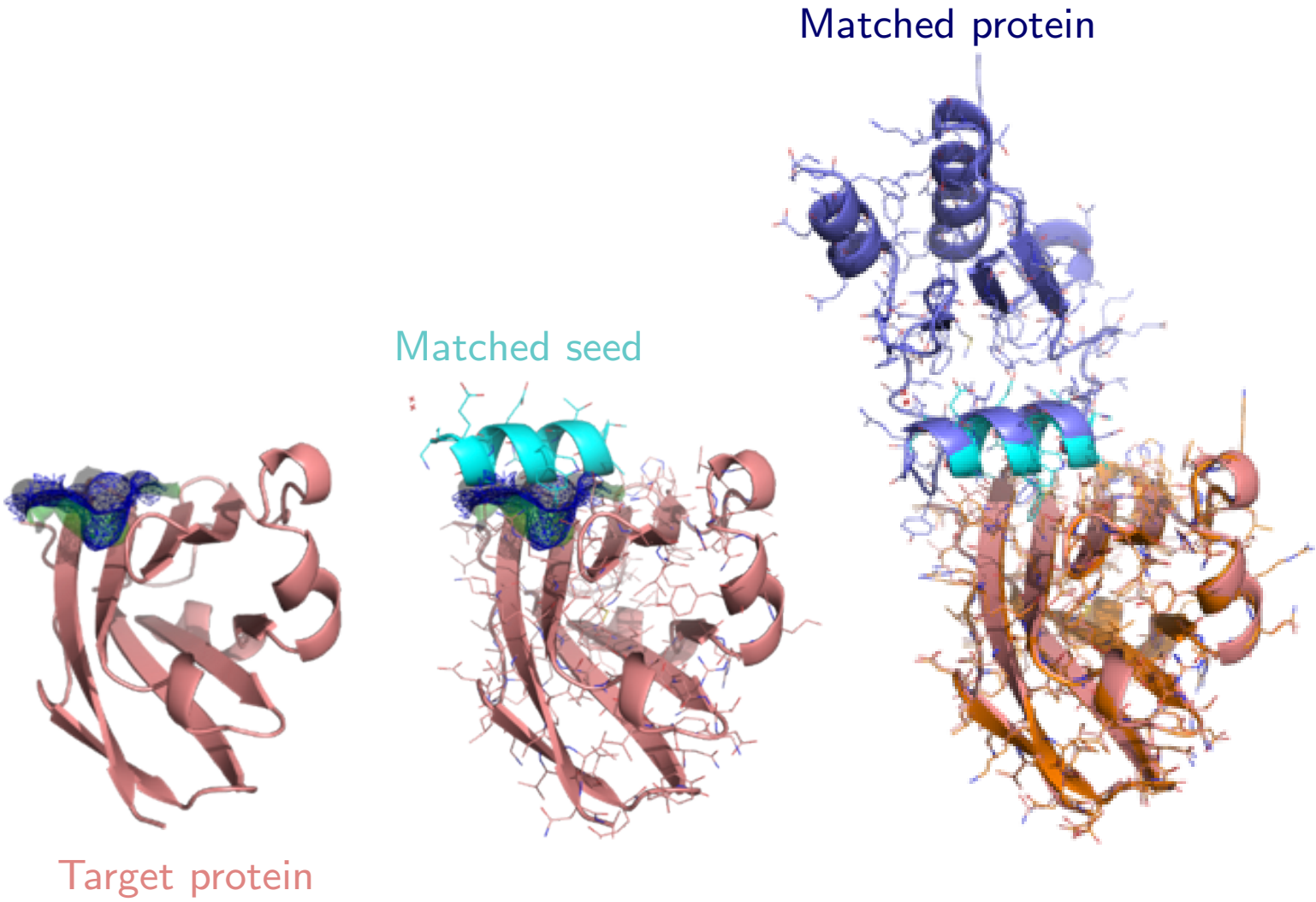
Polypharmacy and Drug repurposing



Prediction of side effects of drug combinations using deep learning on multimodal graph

Zitnik, Agrawal, Leskovec 2018

Protein-Protein Interaction

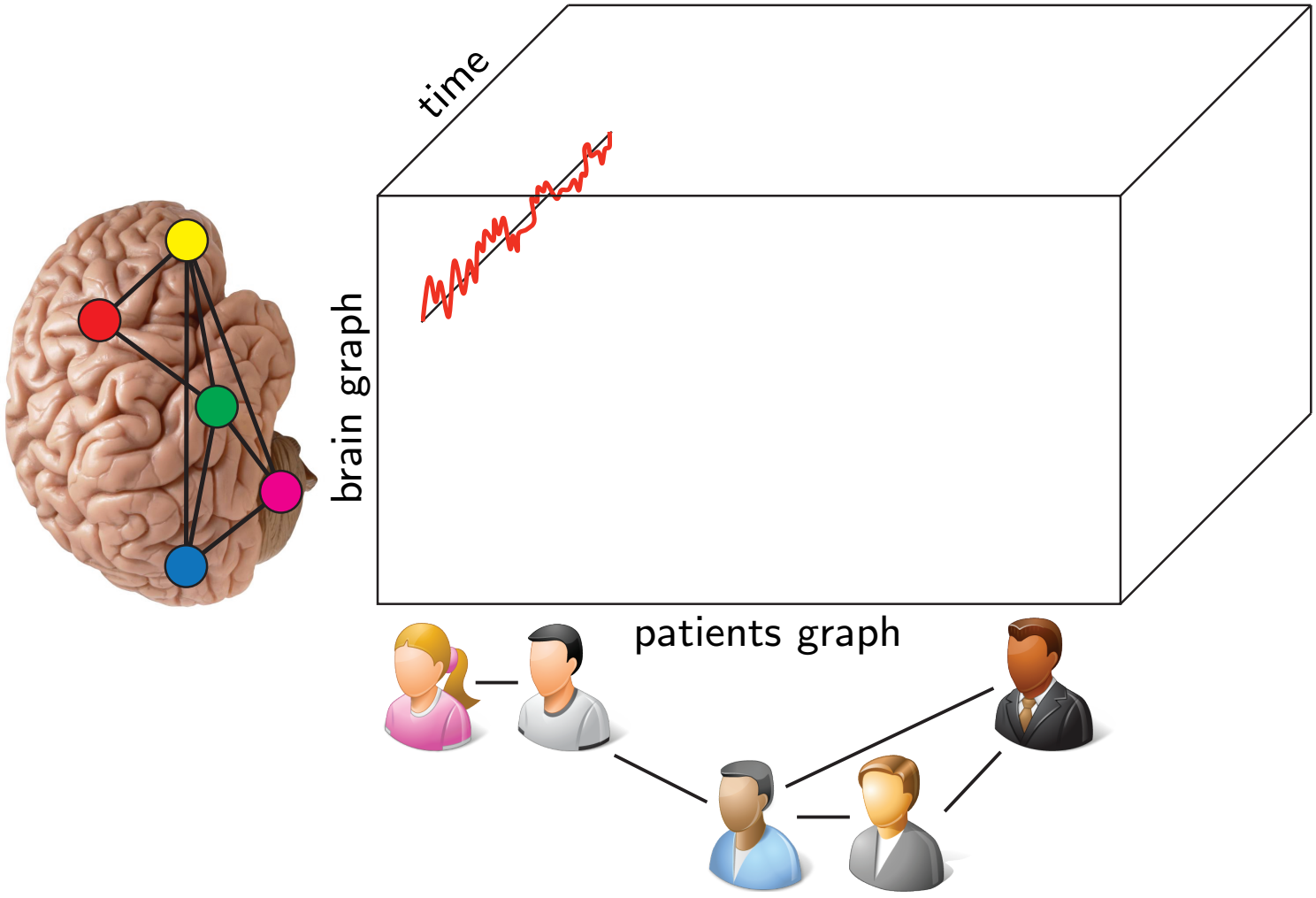


Designing protein binder for the PD-L1 protein target

Collaboration with B. Correia and P. Gainza-Cirauqui (EPFL)

Other Applications

Brain imaging



Parisot et al. 2017

Adversarial noise

Targeted





stop speed limit 45

Eykholt et al. 2017

One pixel



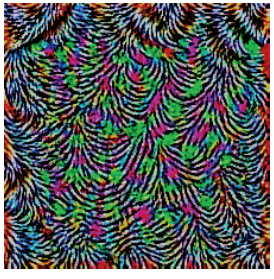


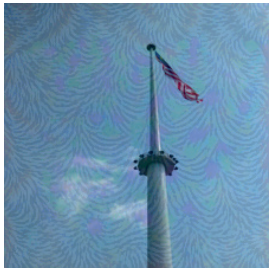
cat bird

Su, Vargas, Sakurai 2018

Universal



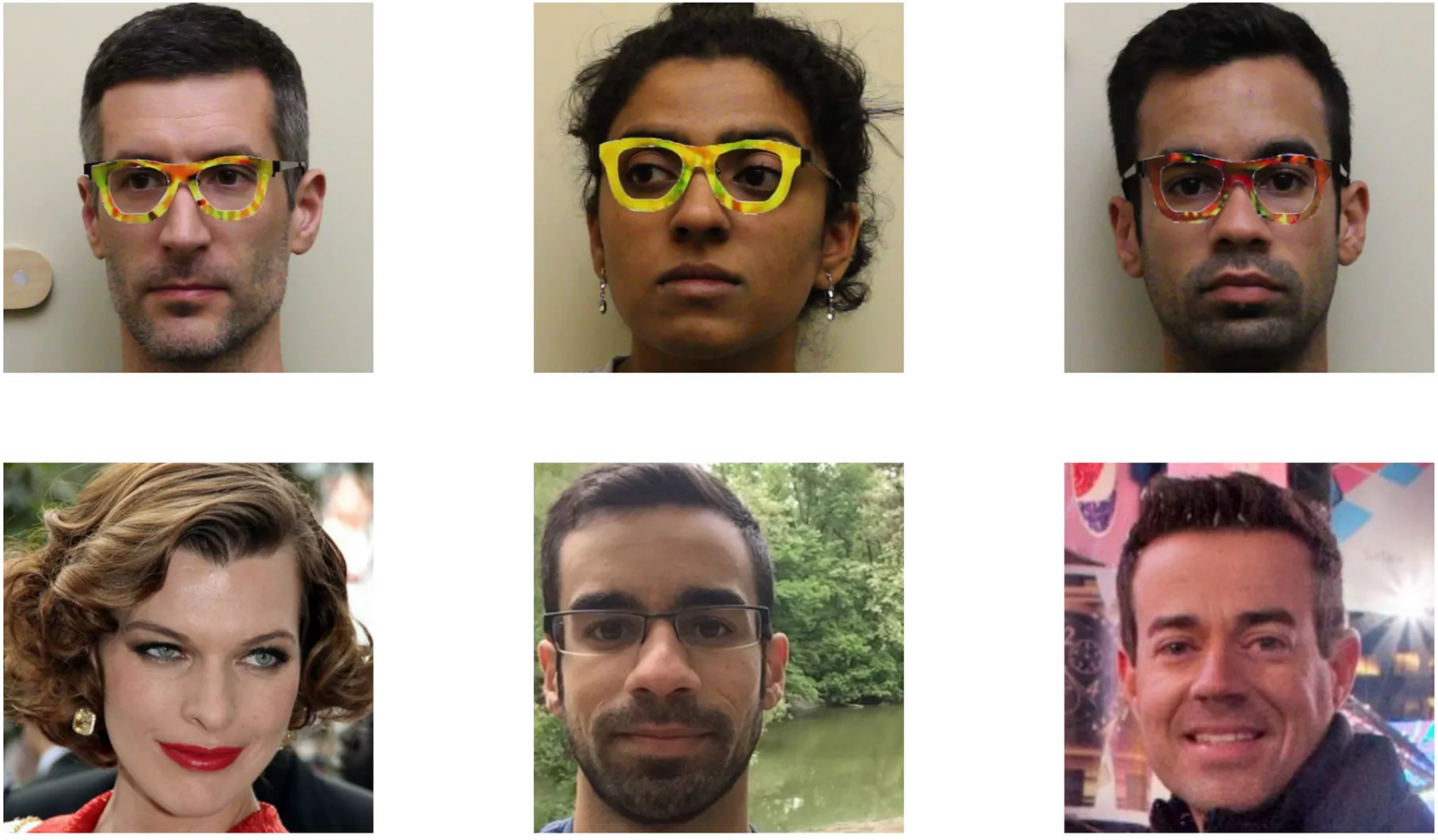




flagpole dog

Moosavi-Dezfooli et al. 2017

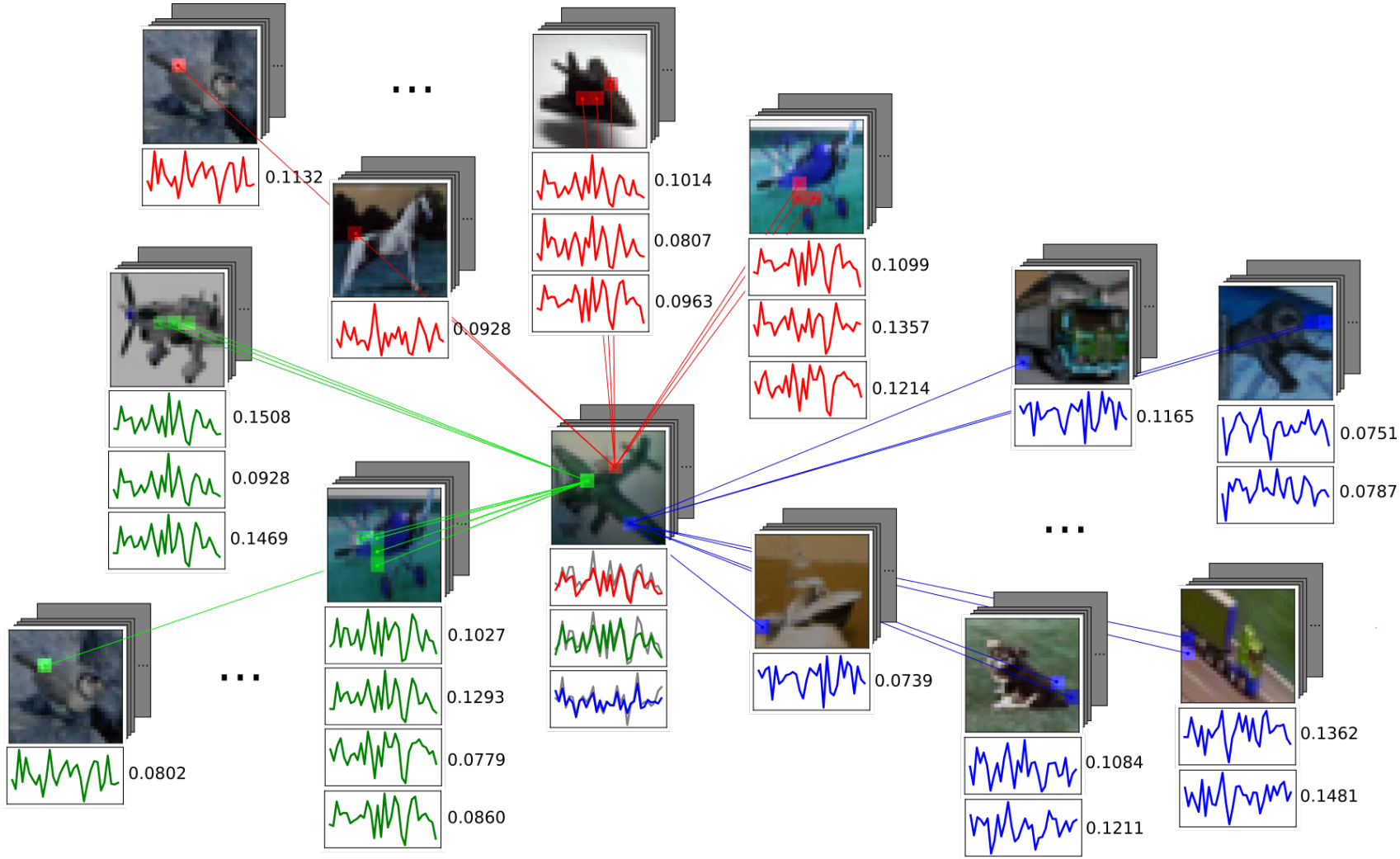
Adversarial noise



Adversarial attack on a face recognition system makes it mistake faces with specially crafted glasses (top) for other people (bottom)

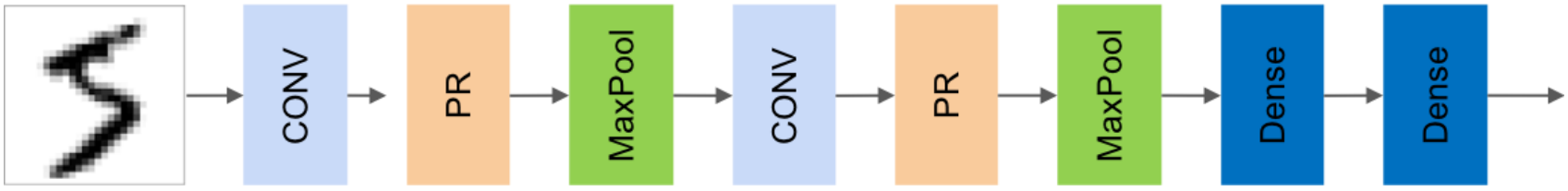
Sharif et al. 2016

Peer regularization

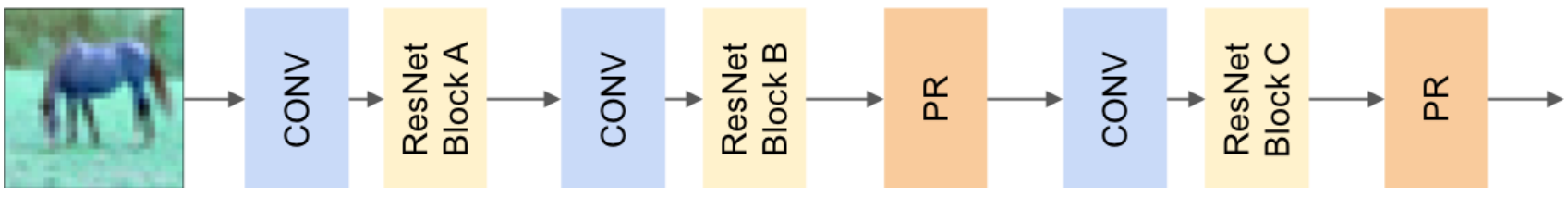


Svoboda et al. 2018

Peer regularization



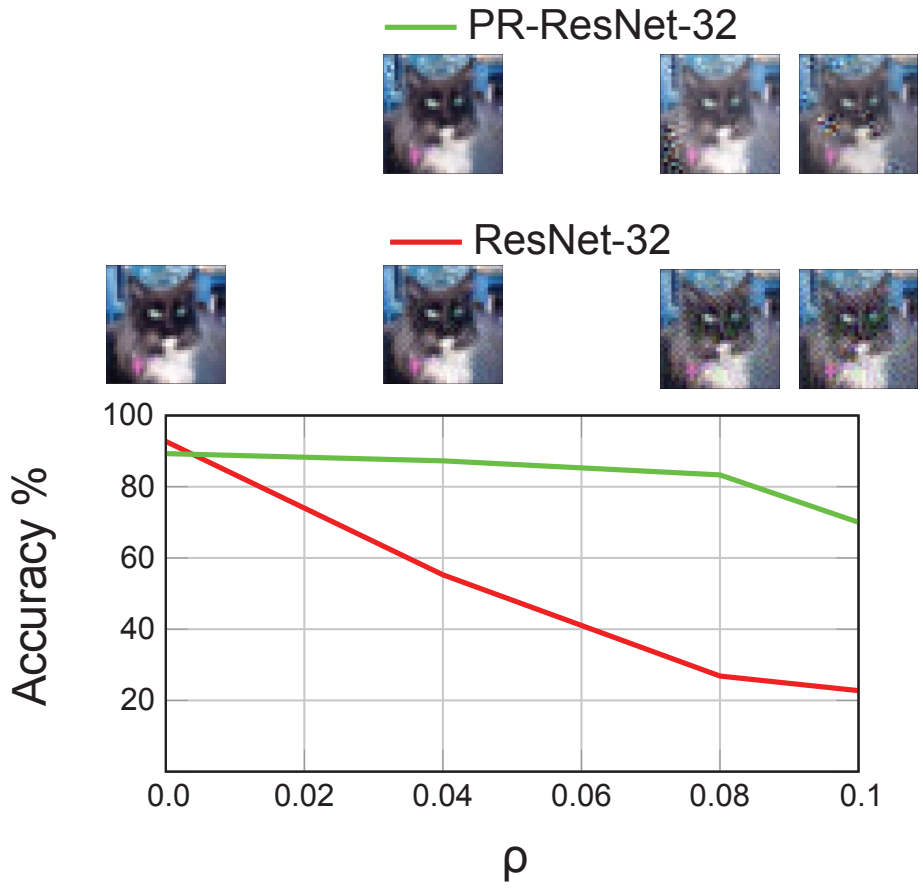
Peer-Regularized LeNet-5



Peer-Regularized ResNet-32

Svoboda et al. 2018

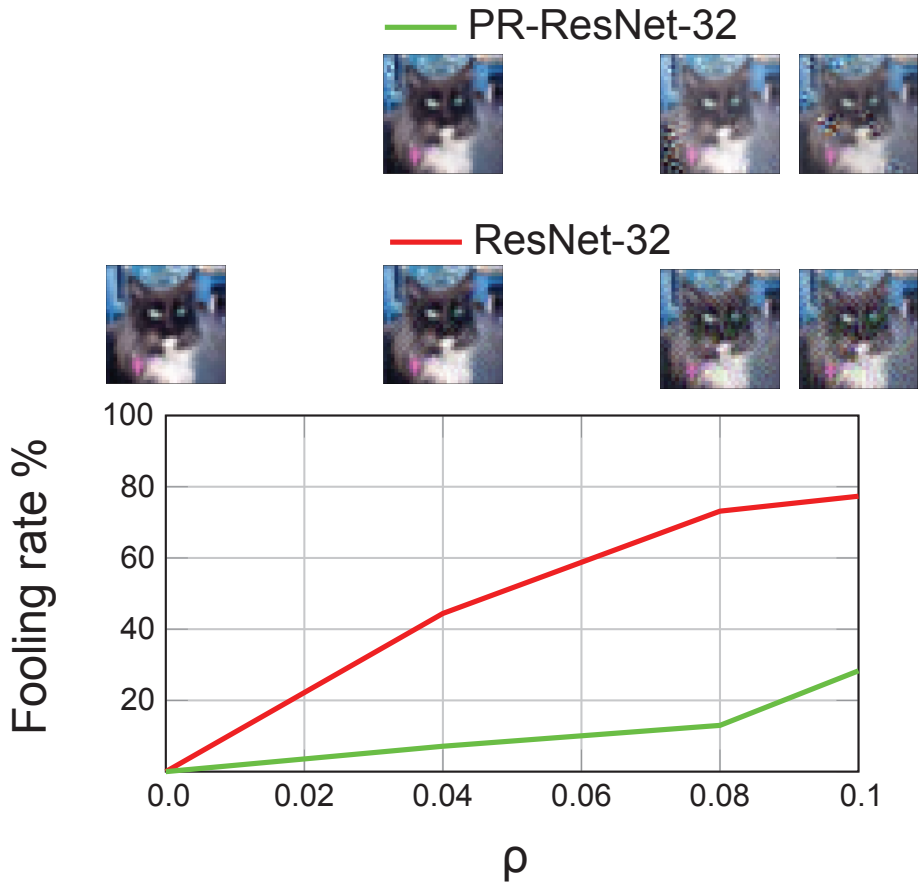
Universal perturbation



Classification accuracy on CIFAR-10 for universal adversarial perturbation of different strength ρ

Svoboda et al. 2018; data: Krizhevsky 2009 (CIFAR-10)

Universal perturbation

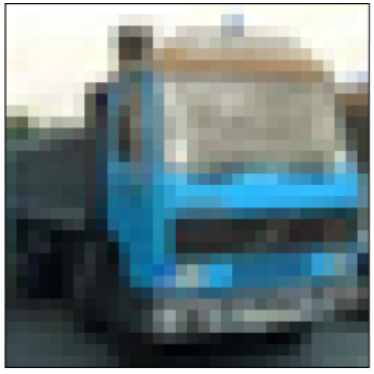


Fooling rate on CIFAR-10 for universal adversarial perturbation of different strength ρ

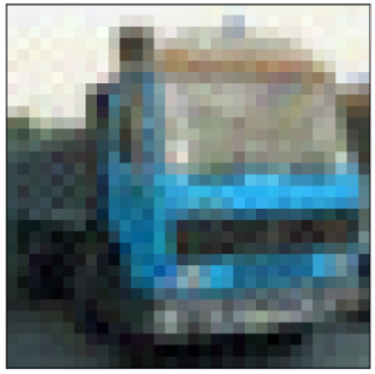
Svoboda et al. 2018; data: Krizhevsky 2009 (CIFAR-10)

Universal perturbation

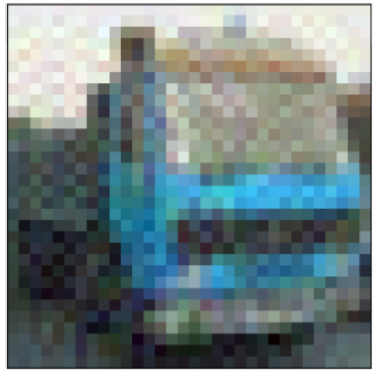
Original



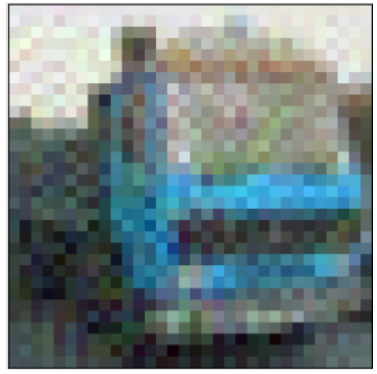
$\rho = 0.04$ truck



$\rho = 0.08$ bird



$\rho = 0.1$ frog



Standard ResNet-32

$\rho = 0.04$ truck



$\rho = 0.08$ truck



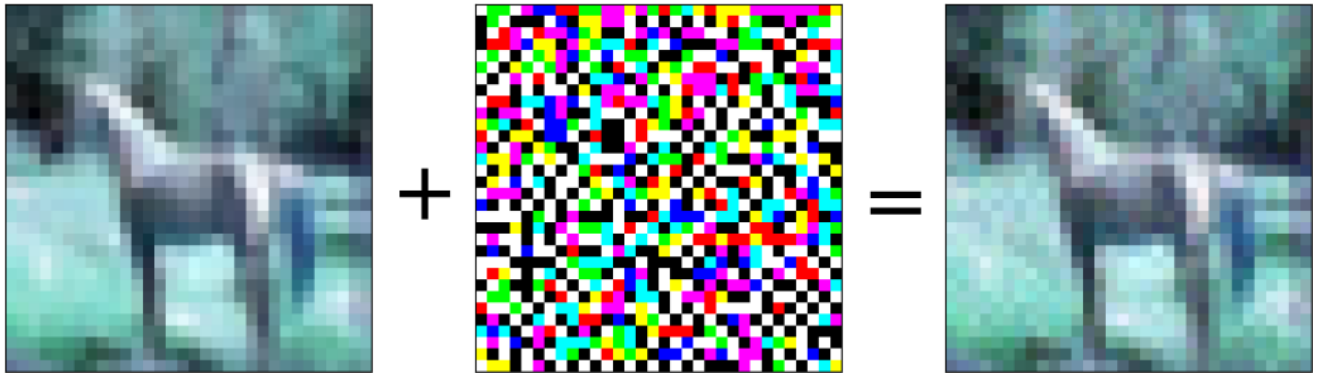
$\rho = 0.1$ truck



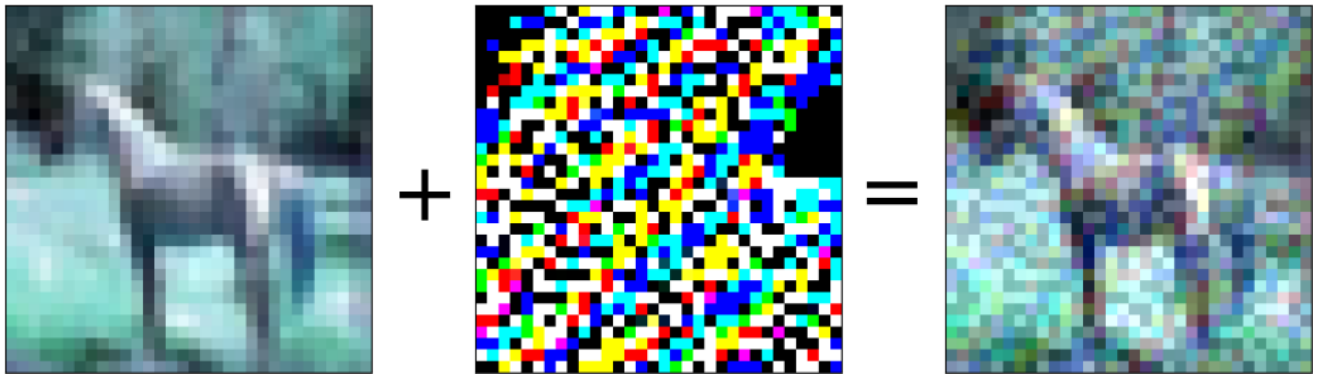
Peer-Regularized ResNet-32

Svoboda et al. 2018; Method: Moosavi-Dezfooli et al. 2016

Non-targeted perturbation (Fast Gradient Sign Method)



Standard ResNet-32

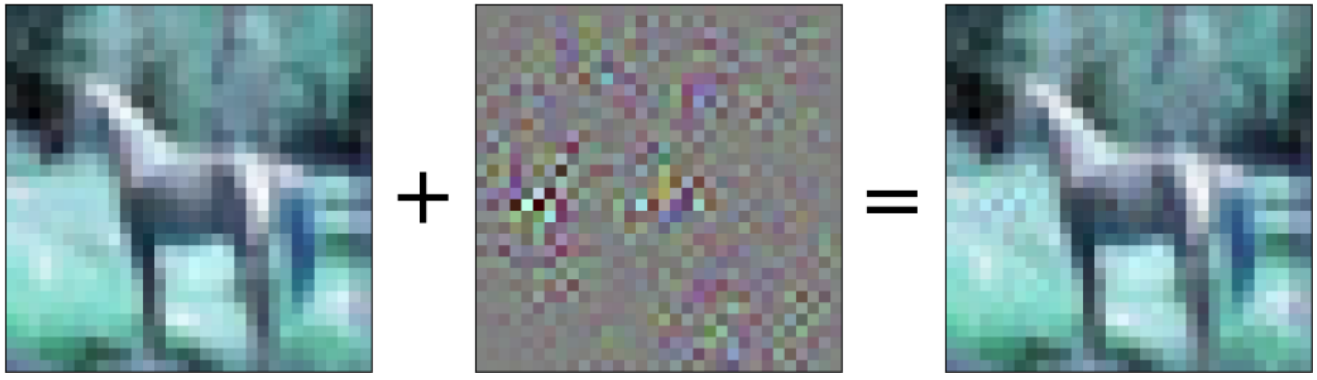


Peer-Regularized ResNet-32

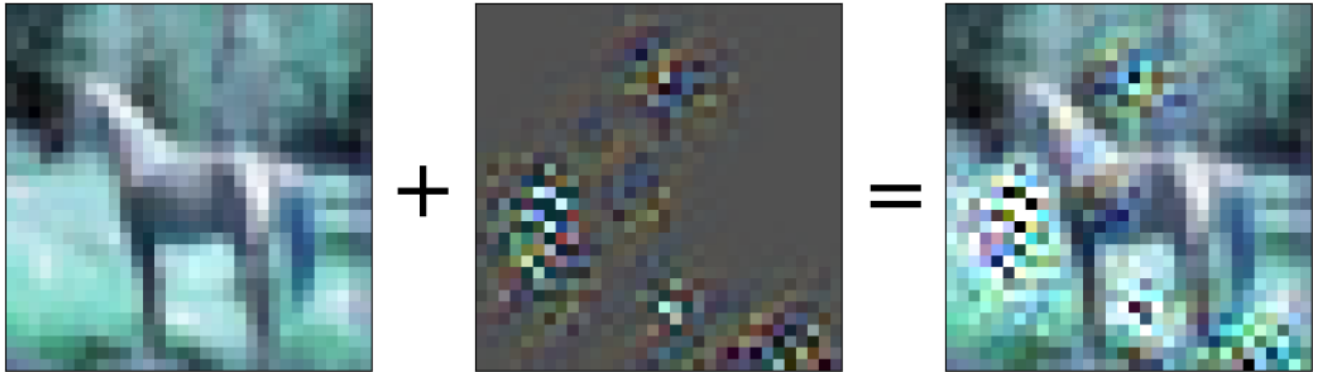
Much stronger noise is needed to fool PeerNet!

Svoboda et al. 2018; FGSM method: Goodfellow et al. 2015

Non-targeted perturbation (gradient descent)



Standard ResNet-32



Peer-Regularized ResNet-32

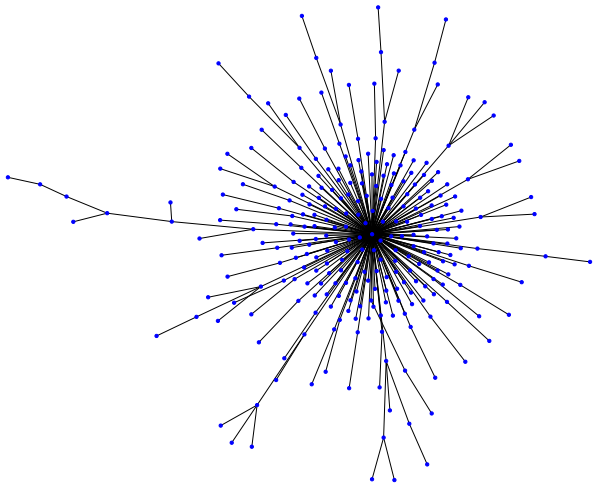
Much stronger noise is needed to fool PeerNet!

Svoboda et al. 2018; Method: Rauber et al. 2017

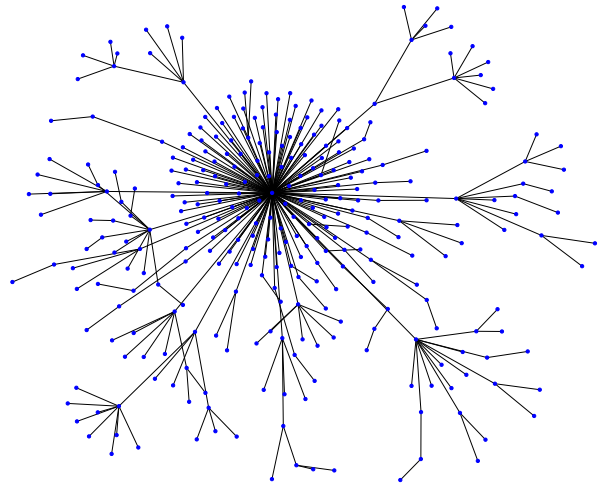


Vosoghi et al. 2018

Fake news detection



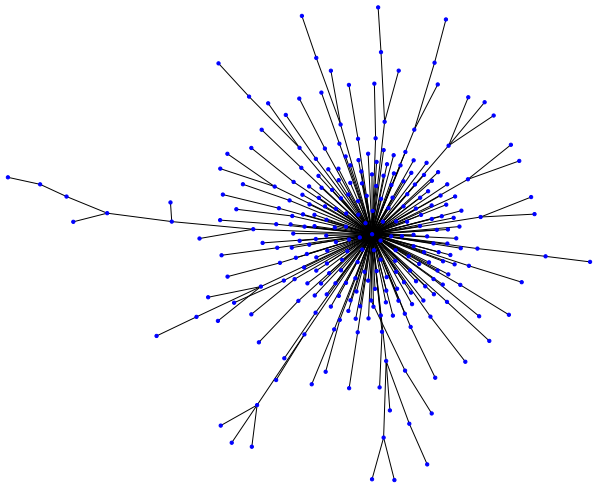
Fake news



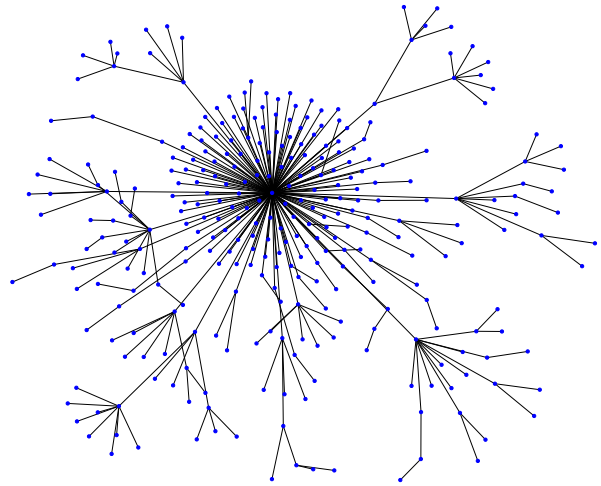
True news

Fabula AI proprietary technology. Multiple patents pending

Fake news detection



Fake news



True news



We are hiring in Switzerland and UK!

Fabula AI proprietary technology. Multiple patents pending

Conclusions

- Geometric Priors exist beyond Euclidean domains
- They can be exploited by modeling locality and weight sharing
- Spectral and Spatial models obtain locality and sharing through different routes
- Resulting neural models related to attention mechanisms
- Challenge: Scaling up models to millions of nodes
- Large areas of application: physical sciences, machine translation, graphics, relational learning

geometricdeeplearning.com

Bibliography

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, "Geometric deep learning: going beyond Euclidean data", arXiv:1611.08097, 2016. **First review paper of geometric deep learning**

M. Gori, G. Monfardini, F. Scarselli, "A new model for learning in graph domains", *Proc. IJCNN* 2005. **First neural net on graphs (GNN framework)**

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, "The graph neural network model", *IEEE Trans. Neural Networks* 20(1):61–80, 2009.

J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, "Spectral networks and locally connected networks on graphs", *Proc. ICML* 2014. **First Spectral CNN on graphs**

M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data", arXiv:1506.05163, 2015. **Spectral CNN with smooth multipliers**

J. Masci, D. Boscaini, M. M. Bronstein, P. Vandergheynst, "Geodesic convolutional neural networks on Riemannian manifolds", arXiv:1501.06297v2, 2015. **First spatial CNN on manifolds (Geodesic CNN framework)**

Bibliography

M. Defferrard, X. Bresson, P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering", *Proc. NIPS* 2016. **Spectral CNN with Chebychev polynomial filters (ChebNet)**

T. N. Kipf, M. Welling, "Semi-supervised classification with graph convolutional networks", arXiv:1609.02907, 2016. **Graph convolutional network (GCN) framework, a simplification of ChebNet**

R. Levie, F. Monti, X. Bresson, M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters", arXiv:1705.07664, 2017. **Spectral CNN with complex rational filters (CayleyNet)**

J. Atwood and D. Towsley, "Diffusion-convolutional neural networks", arXiv:1511.02136v2, 2016. **Diffusion CNN framework**

F. Monti, K. Otness, M. M. Bronstein, "MotifNet: a motif-based Graph Convolutional Network for directed graphs", arXiv:1802.01572, 2018. **MotifNet**

Bibliography

F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs", *Proc. CVPR* 2017. **Mixture Model network (MoNet) framework**

D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, P. Vandergheynst, "Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks", *Computer Graphics Forum* 34(5):13-23, 2015. **Localized Spectral CNN framework**

D. Boscaini, J. Masci, E. Rodolà, M. M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks", *Proc. NIPS* 2016. **Anisotropic CNN framework**

F. Monti, M. M. Bronstein, X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks", Proc. NIPS 2017. **CNNs on multiple graphs and application to matrix completion**

Bibliography

A. Sinha, J. Bai, K. Ramani, "Deep learning 3D shape surfaces using geometry images", *Proc. ECCV* 2016. **CNN on spherical authalic parametrization**

H. Maron, M. Galun, N. Aigerman, M. Trope, N. Dym, E. Yumer, V. G. Kim, Y. Lipman, "Convolutional neural networks on surfaces via seamless toric covers", *Trans. Graphics* 36(4), 2017. **CNN on planar flat-torus parametrization**

L. Yi, H. Su, X. Guo, L. Guibas, "SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation", *Proc. CVPR* 2017. **Spectral transformer networks**

C. R. Qi, H. Su, K. Mo, L. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation", *Proc. CVPR* 2017. **PointNet**

Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, J. M. Solomon, "Dynamic Graph CNN for learning on point clouds", arXiv:1712.00268, 2018. **Edge convolution**

Bibliography

O. Litany, T. Remez, E. Rodolà, A. M. Bronstein, M. M. Bronstein, "Deep functional maps: Structured prediction for dense shape correspondence", *Proc. ICCV* 2017. **Intrinsic structured prediction (FMNet)**

O. Litany, A. M. Bronstein, M. M. Bronstein, A. Makadia, "Deformable shape completion with graph convolutional autoencoders", *Proc. CVPR* 2018. **Shape completion with intrinsic variational autoencoder**

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, "Graph attention networks", *Proc. ICLR* 2018. **GAT model**

F. Monti, O. Shchur, A. Bojchevski, O. Litany, S. Günnemann, M. M. Bronstein, "Dual-Primal Graph Convolutional Networks", arXiv:1806.00770, 2018. **DPGCNN framework**

J. Svoboda, J. Masci, F. Monti, M. M. Bronstein, L. J. Guibas, "PeerNets: Exploiting peer wisdom against adversarial attacks", arXiv:1806.00088, 2018. **PeerNets against adversarial noise**

Bibliography

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, "Neural message passing for quantum chemistry", *Proc. ICML* 2017.

I. Henrion, J. Brehmer, J. Bruna, K. Cho, K. Cranmer, G. Louppe, G. Rochette, "Neural message passing for jet physics", *Proc. NIPS Workshop on Deep Learning for Physical Sciences*, 2017.

Y. LeCun, B. Boser, J. S. Denker, R. E. Howard, W. Hubbard, L. D. Jackel, "Backpropagation applied to handwritten ZIP code recognition", *Neural Computation* 1(4):541–551, 1989. **Classical Euclidean CNN**

J. Bruna, S. Mallat, "Invariant scattering convolution networks", *Trans. PAMI* 35(8):1872–1886, 2013. **Theoretical motivation for CNNs**