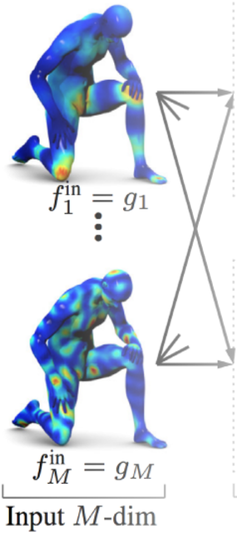
Geometric Deep Learning

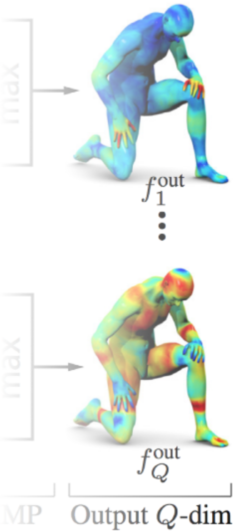
Emanuele Rodolà

Sapienza Università di Roma

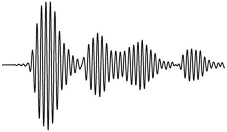
Sapienza University of Rome

Italy

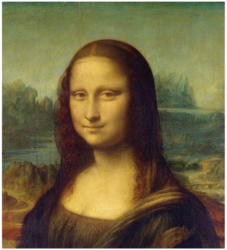




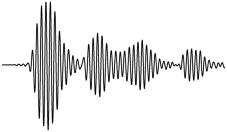
RegML Workshop, Genova, 01 July 2020



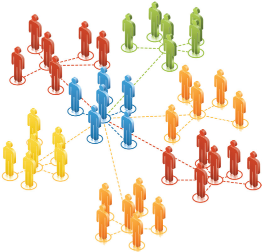
Audio signals



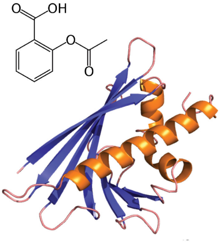
Images



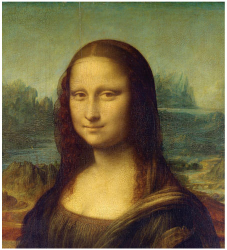
Audio signals



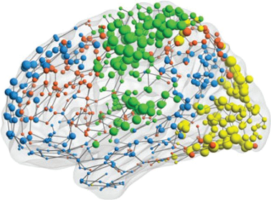
Social networks



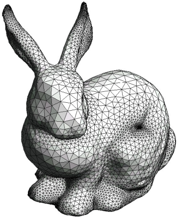
Molecules



Images

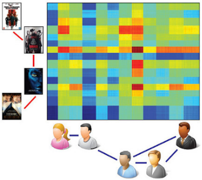


Functional networks

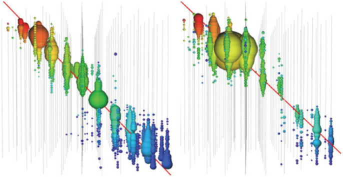


3D shapes

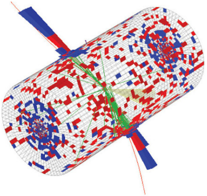
Applications of geometric deep learning



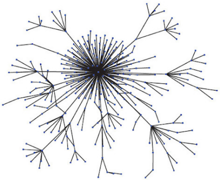
Recommender system



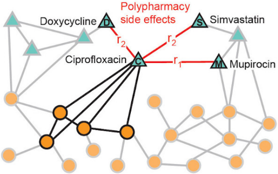
Neutrino detection



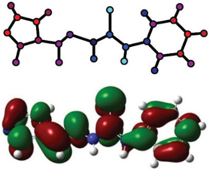
LHC



Fake news detection



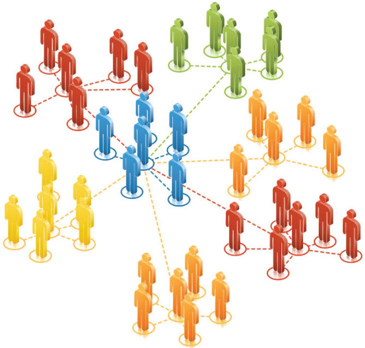
Drug repurposing



Chemistry

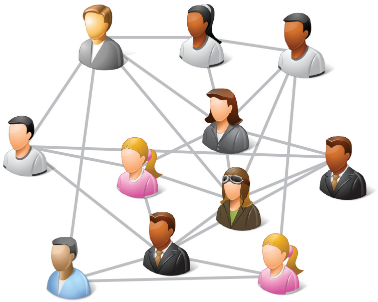
Prototypical non-Euclidean objects

Manifolds

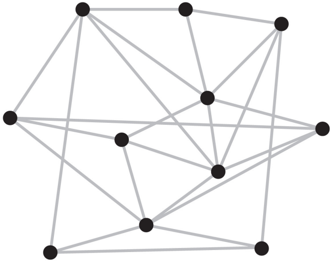


Graphs

Domain structure vs Data on a domain

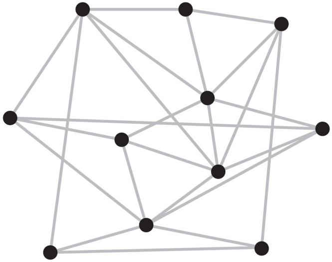


Domain structure vs Data on a domain

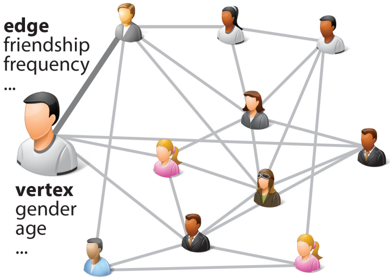


Domain structure

Domain structure vs Data on a domain

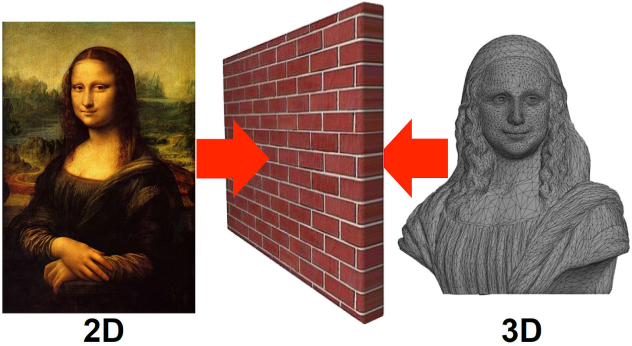


Domain structure



Data on a domain

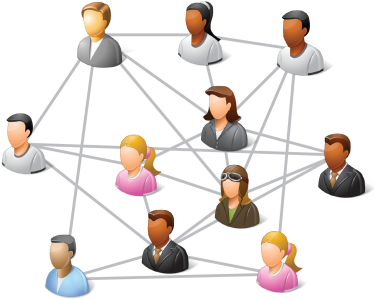
Domain structure vs Data on a domain



only data

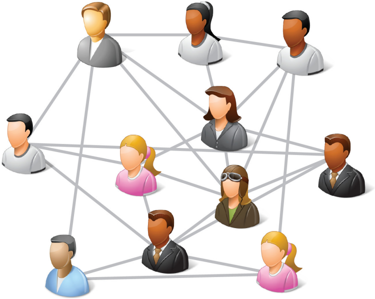
only structure

Fixed vs different domain

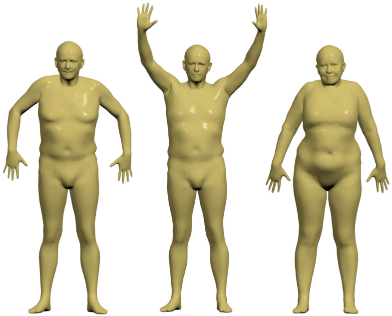


Social network
(fixed graph)

Fixed vs different domain



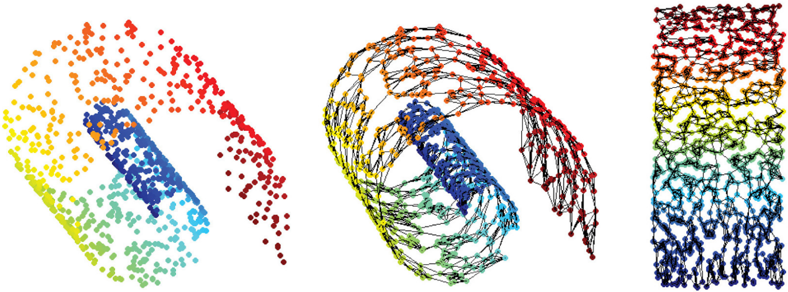
Social network
(fixed graph)



3D shapes
(different manifolds)

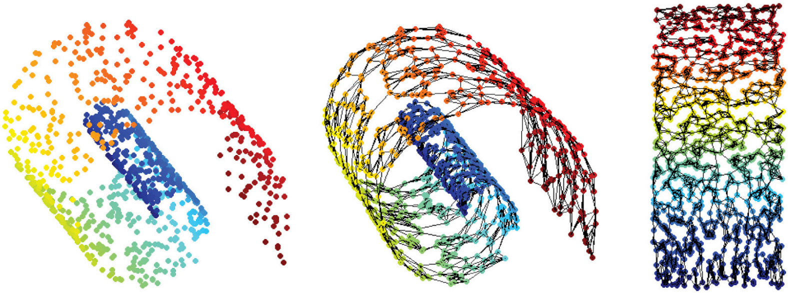
Geometric learning $\neq$ Manifold learning

In manifold learning, we seek for a (possibly high-dimensional) manifold that justifies a given set of data points:



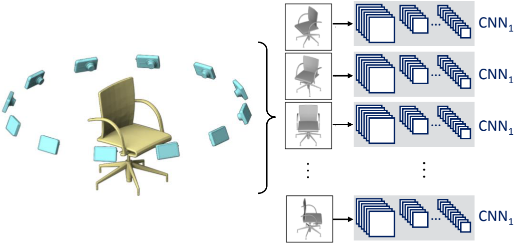
Geometric learning $\neq$ Manifold learning

In manifold learning, we seek for a (possibly high-dimensional) manifold that justifies a given set of data points:



In geometric deep learning, each data point has a known geometric structure.

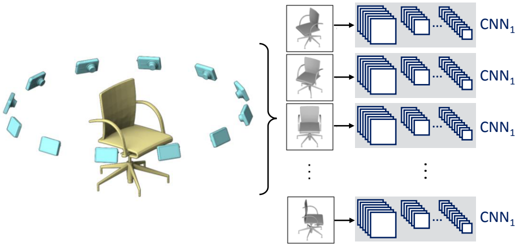
Multi-view CNNs



- Represent 3D object as a collection of range images

Su et al, "Multi-view Convolutional Neural Networks for 3D Shape Recognition", 2015

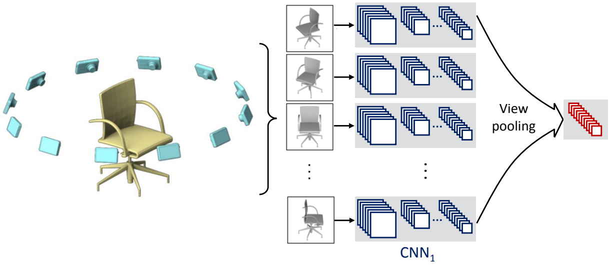
Multi-view CNNs



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

Su et al, "Multi-view Convolutional Neural Networks for 3D Shape Recognition", 2015

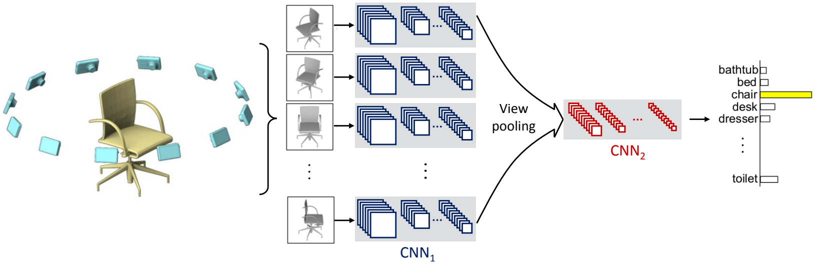
Multi-view CNNs



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

Su et al, "Multi-view Convolutional Neural Networks for 3D Shape Recognition", 2015

Multi-view CNNs



- Represent 3D object as a collection of range images
- CNN$_1$: Extract image features (parameters are shared across views)
- Element-wise max pooling across all views
- CNN$_2$: Produce shape descriptors + final prediction

Su et al, "Multi-view Convolutional Neural Networks for 3D Shape Recognition", 2015

Applications of Multi-view CNNs

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

$\xrightarrow{\text{classify}}$ **"chair"**

Su et al, "Multi-view Convolutional Neural Networks for 3D Shape Recognition", 2015

Applications of Multi-view CNNs

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

$\overset{\text{classify}}{\Longrightarrow}$ **"chair"**

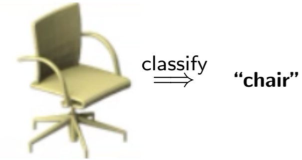
- Sketch classification
  - Mimic views by jittering

$\overset{\text{classify}}{\Longrightarrow}$ **"chair"**

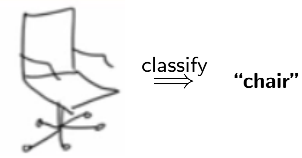
Su et al, "Multi-view Convolutional Neural Networks for 3D Shape Recognition", 2015

Applications of Multi-view CNNs

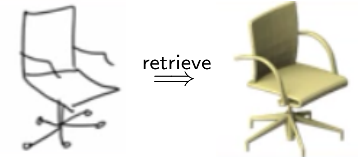
- 3D shape classification and retrieval
  - Pre-trained on ImageNet
  - Fine-tuned on 2D views



- Sketch classification
  - Mimic views by jittering



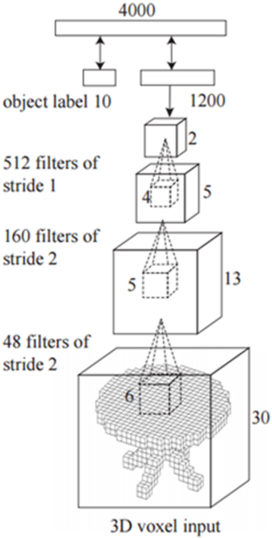
- Sketch-based shape retrieval
  - Render views with hand-drawn style (edge maps)



Su et al, "Multi-view Convolutional Neural Networks for 3D Shape Recognition", 2015

3D ShapeNets

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

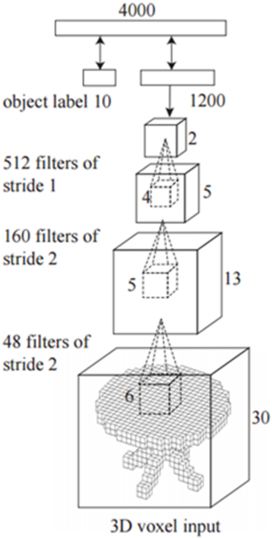


Convolutional deep belief network

Wu et al, "3D ShapeNets: A Deep Representation for Volumetric Shapes" 2015

3D ShapeNets

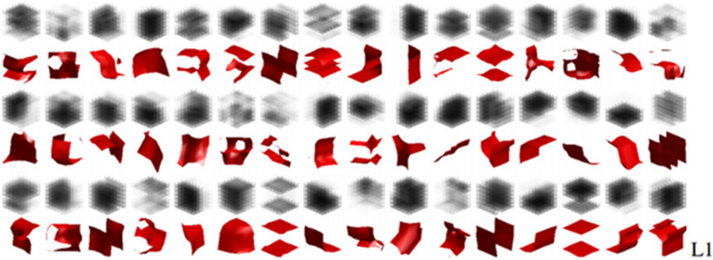
- Volumetric representation (shape = binary voxels on 3D grid)
- 3D convolutional network



Convolutional deep belief network

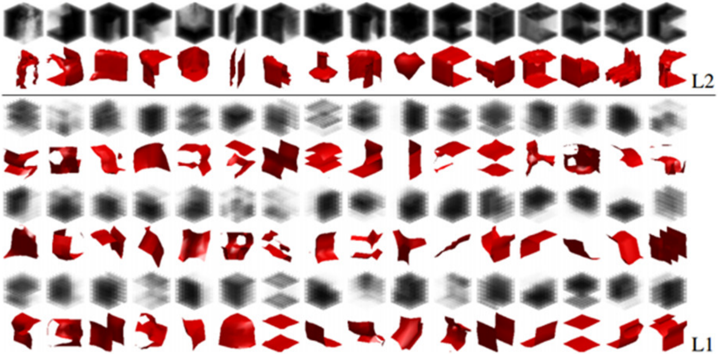
Wu et al, "3D ShapeNets: A Deep Representation for Volumetric Shapes" 2015

Learned features: 3D primitives



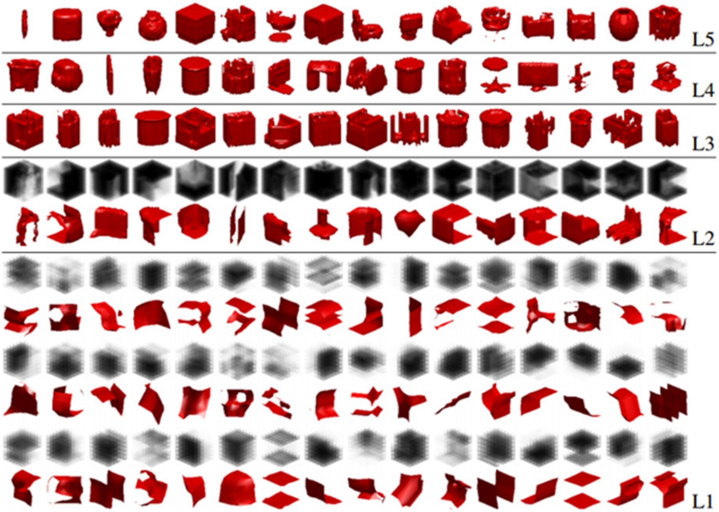
Wu et al, "3D ShapeNets: A Deep Representation for Volumetric Shapes" 2015

Learned features: 3D primitives



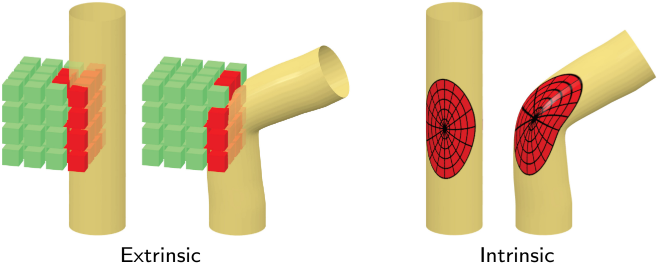
Wu et al, "3D ShapeNets: A Deep Representation for Volumetric Shapes" 2015

Learned features: 3D primitives

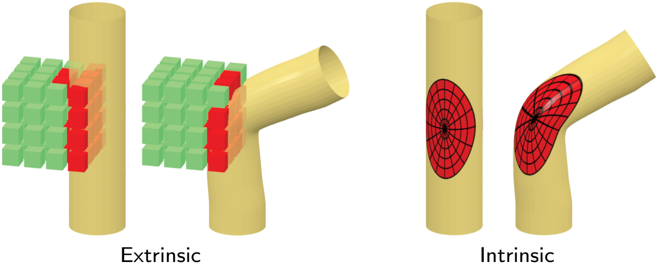


Wu et al, “3D ShapeNets: A Deep Representation for Volumetric Shapes” 2015

Challenges of geometric deep learning

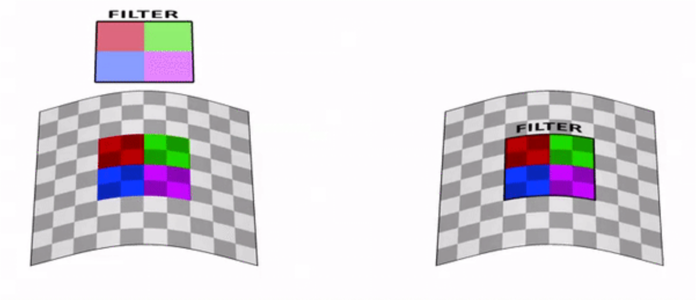


Challenges of geometric deep learning



- How to define convolution?
- How to do pooling?
- How to work fast?

Extrinsic vs Intrinsic



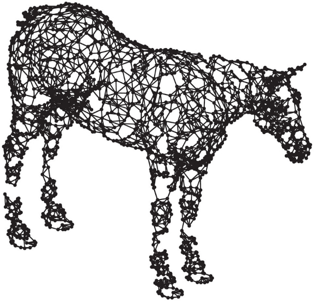
Extrinsic **Intrinsic**

Prototypical non-Euclidean objects

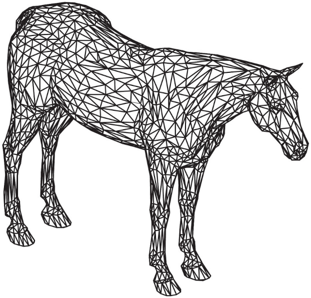
Manifolds

Graphs

Discrete manifolds



Nearest neighbor graph



Triangular mesh

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

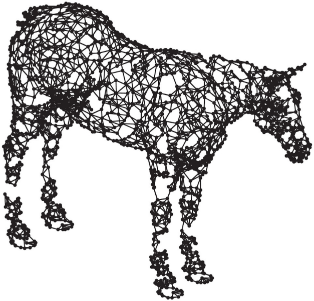
Edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$

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

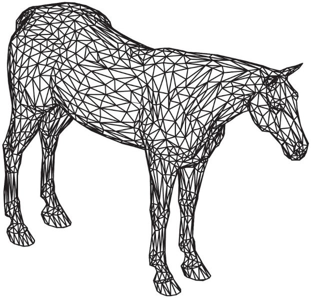
Edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$

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

Discrete manifolds



Nearest neighbor graph



Triangular mesh

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

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

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

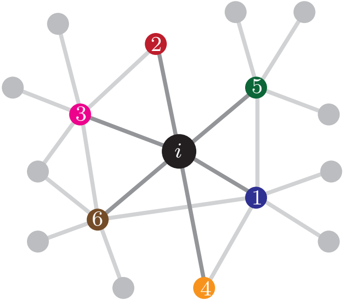
Edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$

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

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

Local ambiguity

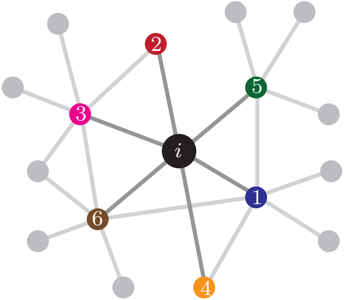
Unlike images, there is **no** canonical ordering of the domain points.



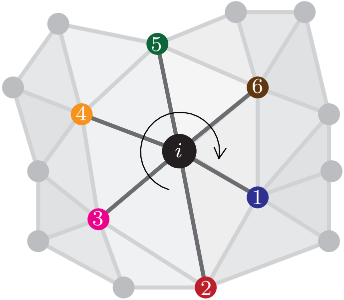
Graph (permutation)

Local ambiguity

Unlike images, there is **no** canonical ordering of the domain points.

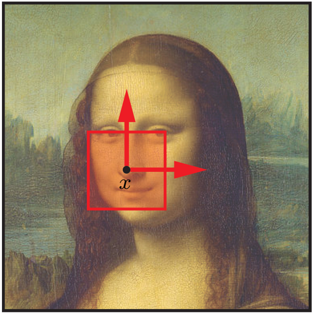


Graph (permutation)

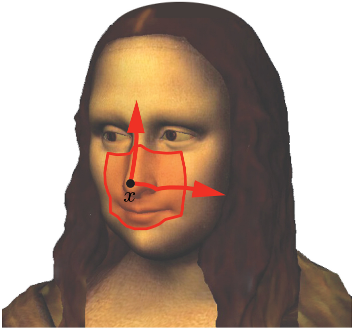


Mesh (rotation)

Non-Euclidean convolution?

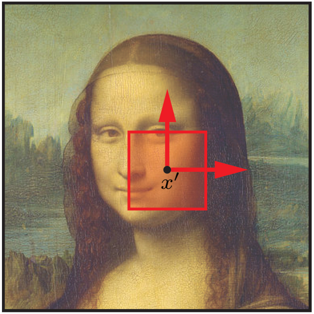


Euclidean

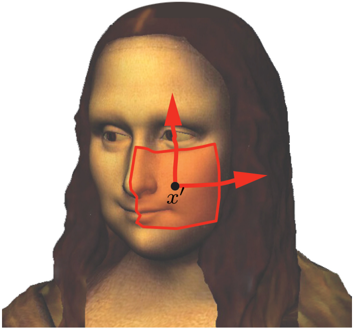


Non-Euclidean

Non-Euclidean convolution?

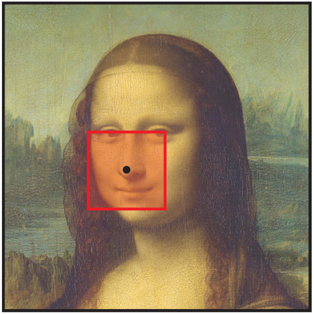


Euclidean

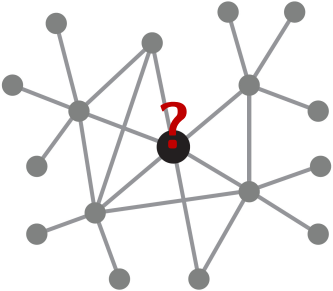


Non-Euclidean

Non-Euclidean convolution?



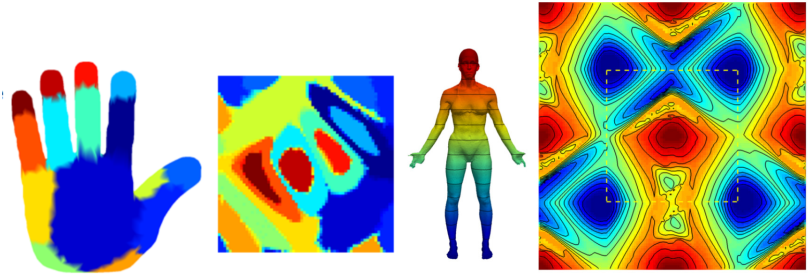
Image



Graph

Global parametrization

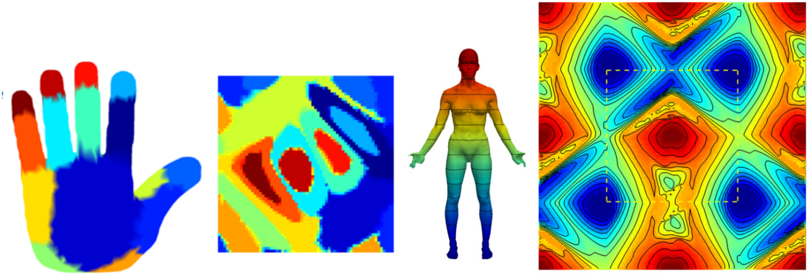
Map the input mesh to some parametric domain (e.g. 2D plane) where operations can be defined more easily.



Sinha et al, "Deep learning 3d shape surfaces using geometry images", 2016

Global parametrization

Map the input mesh to some parametric domain (e.g. 2D plane) where operations can be defined more easily.

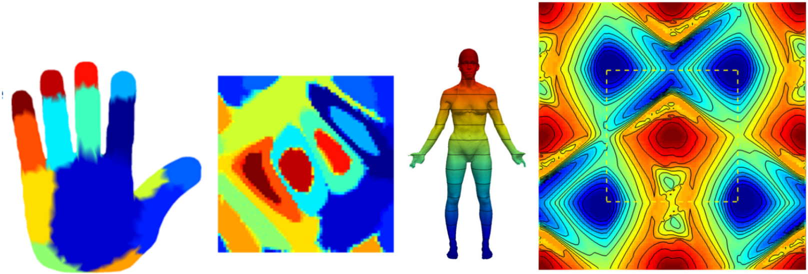


- Can use Euclidean techniques in the embedding space

Sinha et al, "Deep learning 3d shape surfaces using geometry images", 2016

Global parametrization

Map the input mesh to some parametric domain (e.g. 2D plane) where operations can be defined more easily.

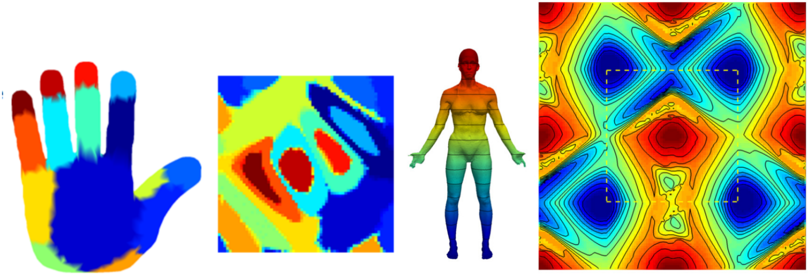


- Can use Euclidean techniques in the embedding space
- Provides invariance to certain transformations

Sinha et al, "Deep learning 3d shape surfaces using geometry images", 2016

Global parametrization

Map the input mesh to some parametric domain (e.g. 2D plane) where operations can be defined more easily.

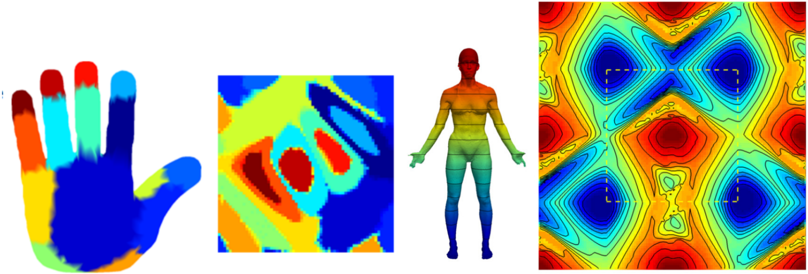


- Can use Euclidean techniques in the embedding space
- Provides invariance to certain transformations
- Parametrization may be non-unique

Sinha et al, “Deep learning 3d shape surfaces using geometry images”, 2016

Global parametrization

Map the input mesh to some parametric domain (e.g. 2D plane) where operations can be defined more easily.



- Can use Euclidean techniques in the embedding space
- Provides invariance to certain transformations
- Parametrization may be non-unique
- The map can introduce distortion

Sinha et al, "Deep learning 3d shape surfaces using geometry images", 2016

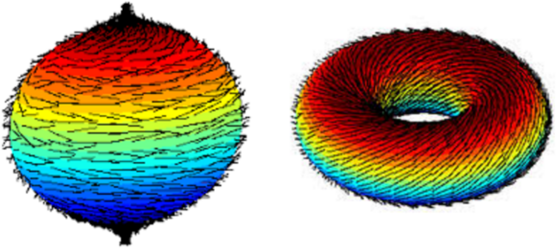
Convolution on surfaces

Is translation-invariant convolution on surfaces possible?

Convolution on surfaces

Is translation-invariant convolution on surfaces possible?

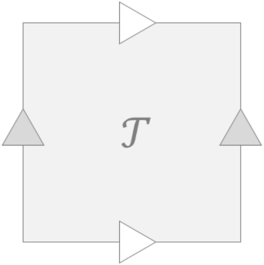
Not in general due to singularities in the translation field (Poincaré-Hopf or "hairy ball" theorem):

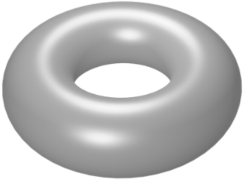


Convolution on surfaces

Is translation-invariant convolution on surfaces possible?

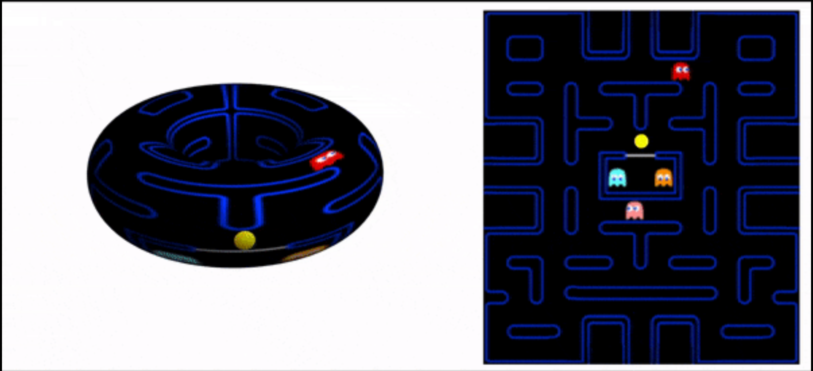
The torus is the only closed orientable surface admitting a translational group.





Maron et al, "Convolutional Neural Networks on Surfaces via Seamless Toric Covers", SIGGRAPH 2017

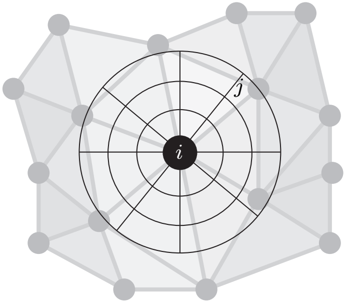
Convolution on surfaces



Video by Ajeet Gary, 2019

Spatial convolution on meshes

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

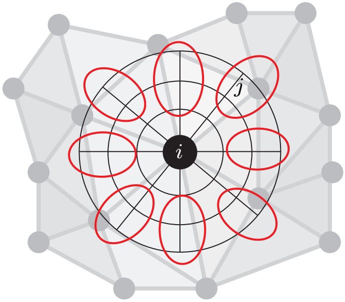


Monti et al, "Geometric deep learning on graphs and manifolds using mixture model CNNs", CVPR 2016

Spatial convolution on meshes

- Local system of coordinates $\mathbf{u}_{ij}$ around i (e.g. geodesic polar)
- Local weights $w(\mathbf{u}_{ij})$, e.g. Gaussians with learnable $\boldsymbol{\mu}, \boldsymbol{\Sigma}$:

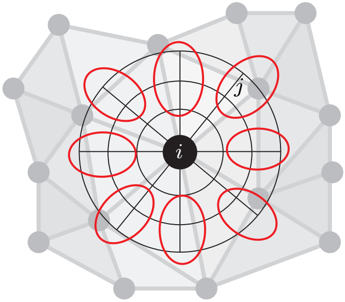
  $w = \exp\left(-(\mathbf{u}_{ij} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{u}_{ij} - \boldsymbol{\mu})\right)$



Monti et al, "Geometric deep learning on graphs and manifolds using mixture model CNNs", CVPR 2016

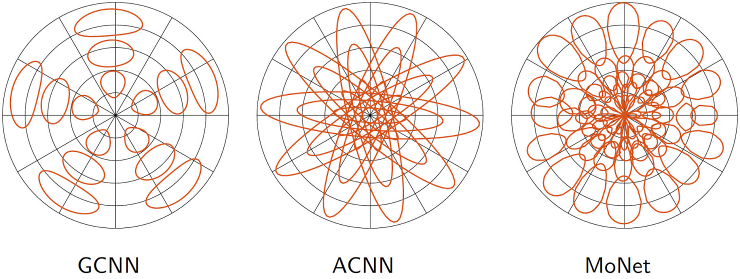
Spatial convolution on meshes

- Local system of coordinates $\mathbf{u}_{ij}$ around i (e.g. geodesic polar)
- Local weights $w(\mathbf{u}_{ij})$, e.g. Gaussians with learnable $\boldsymbol{\mu}, \boldsymbol{\Sigma}$:
  $w = \exp\left(-(\mathbf{u}_{ij} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{u}_{ij} - \boldsymbol{\mu})\right)$
- Spatial convolution of feature f with filter g:
  - Represent the input f as above $\Rightarrow \mathbf{f}$
  - Represent the learnable filter g as above $\Rightarrow \mathbf{g}$
  - Sum up the element-wise products $\Rightarrow \mathbf{f}^\top \mathbf{g}$



Monti et al, "Geometric deep learning on graphs and manifolds using mixture model CNNs", CVPR 2016

Local weighting kernels



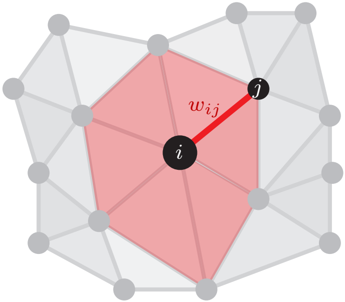
Monti et al, "Geometric deep learning on graphs and manifolds using mixture model CNNs", CVPR 2016

Coffee break (10min?)

Spectral convolution on meshes

- Laplacian operator $\mathbf{\Delta}$ acting locally on the neighborhood of i:

$$(\mathbf{\Delta x})_i = \sum_j w_{ij}(\mathbf{x}_j - \mathbf{x}_i)$$

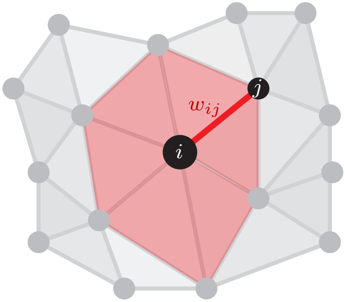


Spectral convolution on meshes

- Laplacian operator $\boldsymbol{\Delta}$ acting locally on the neighborhood of i:

$$(\boldsymbol{\Delta}\mathbf{x})_i = \sum_j w_{ij}(\mathbf{x}_j - \mathbf{x}_i)$$

$=$ neighborhood avg – value at i



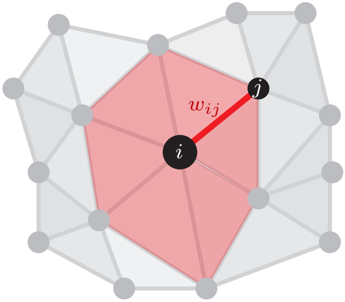
Spectral convolution on meshes

- Laplacian operator $\boldsymbol{\Delta}$ acting locally on the neighborhood of i:

$$(\boldsymbol{\Delta}\mathbf{x})_i = \sum_j w_{ij}(\mathbf{x}_j - \mathbf{x}_i)$$

- Eigenvectors of the Laplacian $\boldsymbol{\Delta} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^\top$ are a generalization of the Fourier transform:

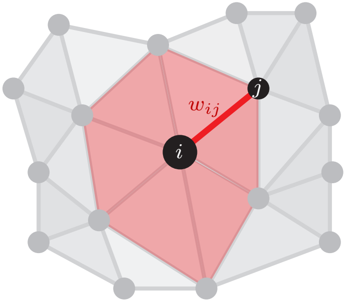
$$\hat{\mathbf{x}} = \boldsymbol{\Phi}^\top \mathbf{x}$$



Spectral convolution on meshes

- Laplacian operator $\mathbf{\Delta}$ acting locally on the neighborhood of i:

$$(\mathbf{\Delta x})_i = \sum_j w_{ij}(\mathbf{x}_j - \mathbf{x}_i)$$



- Eigenvectors of the Laplacian $\mathbf{\Delta} = \mathbf{\Phi \Lambda \Phi}^\top$ are a generalization of the Fourier transform:

$$\hat{\mathbf{x}} = \mathbf{\Phi}^\top \mathbf{x}$$

- Spectral convolution

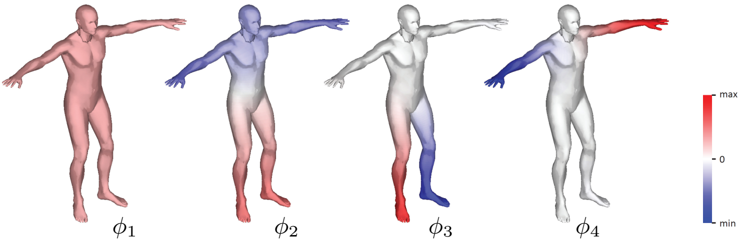
$$\mathbf{x} \star \mathbf{y} = \mathbf{\Phi} \underbrace{\begin{pmatrix} \hat{y}_1 & & \\ & \ddots & \\ & & \hat{y}_n \end{pmatrix}}_{\hat{\mathbf{Y}}} \hat{\mathbf{x}}$$

Bruna et al, "Spectral Networks and Locally Connected Networks on Graphs", 2014

Laplacian eigenfunctions: Euclidean

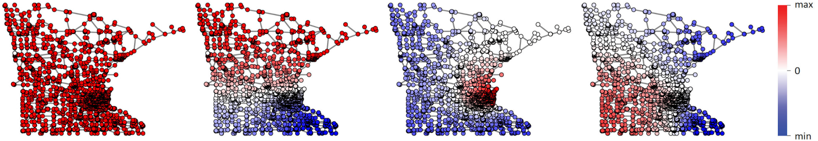
First eigenfunctions of 1D Euclidean Laplacian = standard Fourier basis

Laplacian eigenfunctions: manifold



First eigenfunctions of a manifold Laplacian

Laplacian eigenfunctions: graph



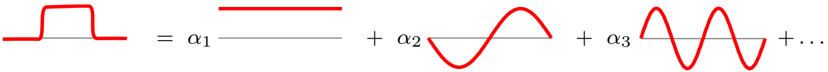
ϕ_1 ϕ_2 ϕ_3 ϕ_4

First eigenfunctions of a graph Laplacian

Fourier analysis: Euclidean space

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

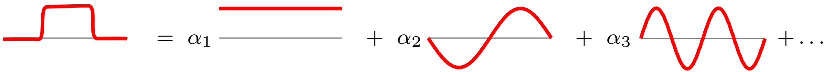
$$f(x) = \sum_{k \geq 0} \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} dx' \, e^{ikx}$$



Fourier analysis: Euclidean space

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

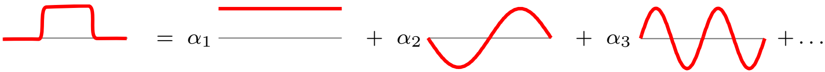
$$f(x) = \sum_{k \geq 0} \underbrace{\frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} dx'}_{\hat{f}_k = \langle f, e^{ikx} \rangle_{L^2([-\pi,\pi])}} e^{ikx}$$



Fourier analysis: Euclidean space

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

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

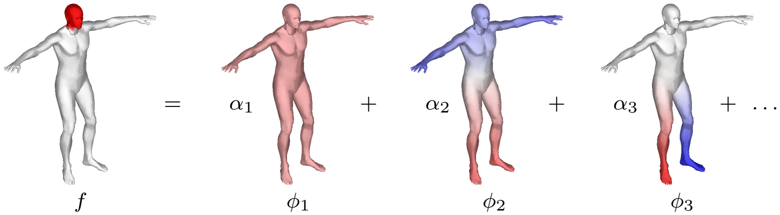


Fourier basis = Laplacian eigenfunctions: $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

Fourier analysis: non-Euclidean space

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

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



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

Convolution theorem

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 theorem

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 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 theorem

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

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

Convolution theorem

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

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

Convolution theorem

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

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

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

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

Convolution theorem

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

$$\begin{aligned}
\mathbf{f} \star \mathbf{g} &= \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \\
&= \mathbf{\Phi} \begin{bmatrix} \hat{g}_1 & & \\ & \ddots & \\ & & \hat{g}_n \end{bmatrix} \mathbf{\Phi}^\top \mathbf{f}
\end{aligned}$$

Convolution theorem

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

$$\begin{aligned}
\mathbf{f} \star \mathbf{g} \quad &= \quad \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \\
&= \quad \mathbf{\Phi} \begin{bmatrix} \hat{g}_1 & & \\ & \ddots & \\ & & \hat{g}_n \end{bmatrix} \begin{bmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_n \end{bmatrix}
\end{aligned}$$

Convolution theorem

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

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

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

Spectral convolution

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

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

Spectral convolution

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

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

Spectral convolution

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

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

Spectral convolution

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

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

In matrix-vector notation

$$\mathbf{f} \star \mathbf{g} = \boldsymbol{\Phi} \, (\boldsymbol{\Phi}^\top \mathbf{g}) \circ (\boldsymbol{\Phi}^\top \mathbf{f})$$

Spectral convolution

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

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

In matrix-vector notation

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

Spectral convolution

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

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

In matrix-vector notation

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

Spectral convolution

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

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

In matrix-vector notation

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

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

Spectral convolution

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

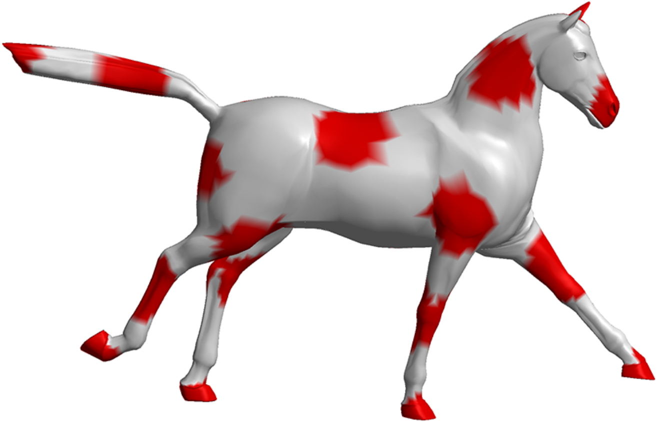
$$f \star g \quad = \quad \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \, \phi_k$$

In matrix-vector notation

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

- Not shift-invariant! ($\mathbf{G}$ has no circulant structure)
- Filter coefficients depend on basis $\phi_1, \ldots, \phi_n$

Basis dependence

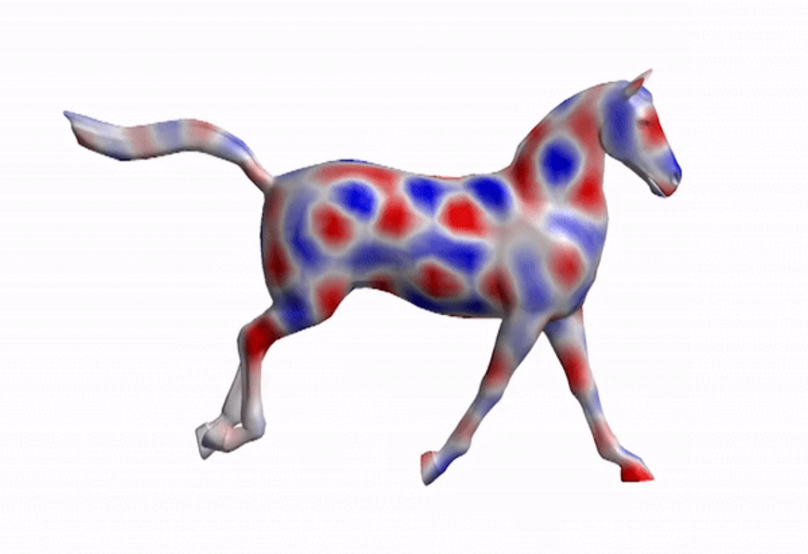


Function $\mathbf{x}$

Basis dependence

'Edge detecting' spectral filter $\mathbf{\Phi}\hat{\mathbf{Y}}\mathbf{\Phi}^{\top}\mathbf{x}$

Basis dependence

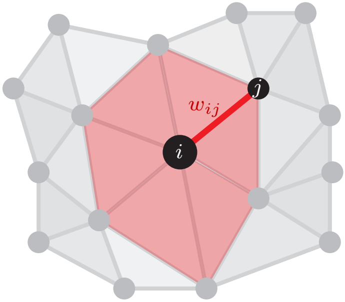


Same spectral filter, different basis $\mathbf{\Psi}\hat{\mathbf{Y}}\mathbf{\Psi}^{\top}\mathbf{x}$

Spectral convolution on meshes

- Laplacian operator $\boldsymbol{\Delta}$ acting locally on the neighborhood of i:

$$(\boldsymbol{\Delta}\mathbf{x})_i = \sum_j w_{ij}(\mathbf{x}_j - \mathbf{x}_i)$$



- Eigenvectors of the Laplacian $\boldsymbol{\Delta} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^\top$ are a generalization of the Fourier transform:

$$\hat{\mathbf{x}} = \boldsymbol{\Phi}^\top \mathbf{x}$$

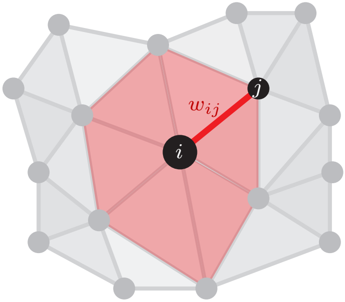
- Spectral convolution:

$$\mathbf{x} \star \mathbf{y} = \boldsymbol{\Phi} \underbrace{\begin{pmatrix} \hat{y}_1 & & \\ & \ddots & \\ & & \hat{y}_n \end{pmatrix}}_{\hat{\mathbf{Y}}} \hat{\mathbf{x}}$$

Spectral convolution on meshes

- Laplacian operator $\boldsymbol{\Delta}$ acting locally on the neighborhood of i:

$$(\boldsymbol{\Delta}\mathbf{x})_i = \sum_j w_{ij}(\mathbf{x}_j - \mathbf{x}_i)$$



- Eigenvectors of the Laplacian $\boldsymbol{\Delta} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^\top$ are a generalization of the Fourier transform:

$$\hat{\mathbf{x}} = \boldsymbol{\Phi}^\top \mathbf{x}$$

- Spectral convolution defined as a filter applied on the Lapacian:

$$\mathbf{X}' = \boldsymbol{\Phi}\ \tau(\boldsymbol{\Lambda})\ \boldsymbol{\Phi}^\top \mathbf{X}$$

Locality and smoothness

In the Euclidean setting (by Parseval's identity), the following holds:

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

Localization in space = smoothness in frequency domain

Henaff et al, "Deep Convolutional Networks on Graph-Structured Data", 2015

Locality and smoothness

In the Euclidean setting (by Parseval's identity), the following holds:

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

Localization in space = smoothness in frequency domain

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

Henaff et al, "Deep Convolutional Networks on Graph-Structured Data", 2015

Locality and smoothness

In the Euclidean setting (by Parseval's identity), the following holds:

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

Localization in space = smoothness in frequency domain

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

Application of the filter

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

Henaff et al, "Deep Convolutional Networks on Graph-Structured Data", 2015

Locality and smoothness

In the Euclidean setting (by Parseval's identity), the following holds:

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

Localization in space = smoothness in frequency domain

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

Application of the filter

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

Henaff et al, "Deep Convolutional Networks on Graph-Structured Data", 2015

Locality and smoothness

In the Euclidean setting (by Parseval's identity), the following holds:

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

Localization in space = smoothness in frequency domain

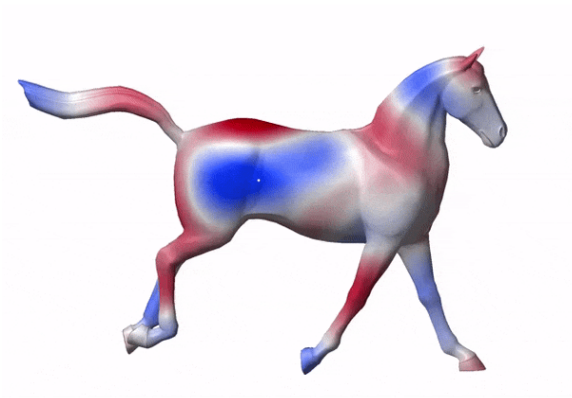
Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$.

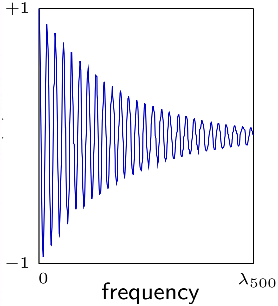
Application of the parametric filter with learnable parameters $\boldsymbol{\alpha}$

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

Henaff et al, "Deep Convolutional Networks on Graph-Structured Data", 2015

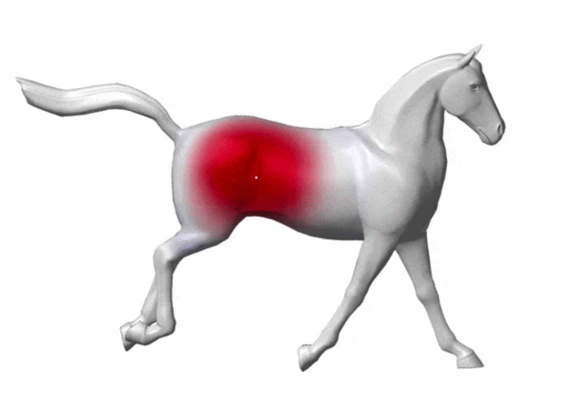
Locality and smoothness

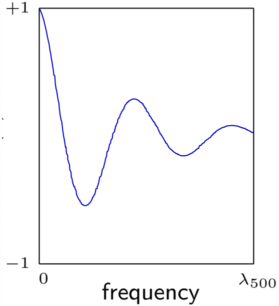




Non-smooth spectral filter (delocalized in space)

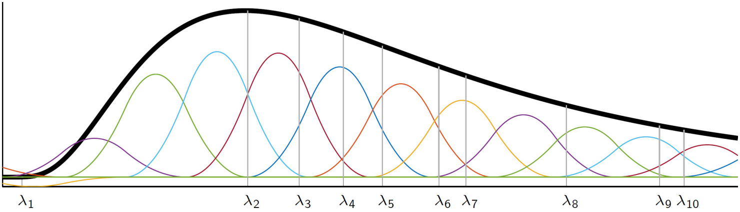
Locality and smoothness





Smooth spectral filter (localized in space)

Spectral graph CNN with smooth spectral filters



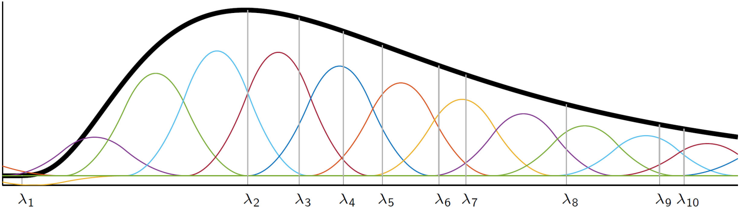
Consider a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$:

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

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

Henaff et al, "Deep Convolutional Networks on Graph-Structured Data", 2015

Spectral graph CNN with smooth spectral filters



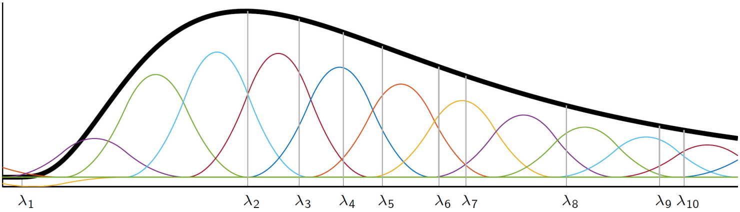
Consider a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$:

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

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

Henaff et al, "Deep Convolutional Networks on Graph-Structured Data", 2015

Spectral graph CNN with smooth spectral filters



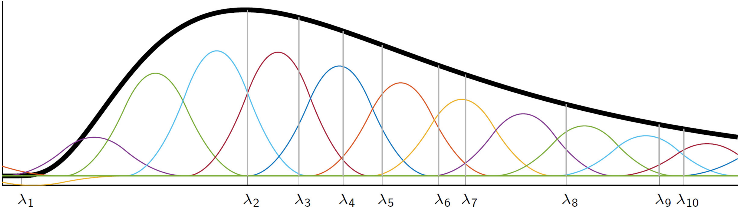
Consider a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$:

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

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

Henaff et al, "Deep Convolutional Networks on Graph-Structured Data", 2015

Spectral graph CNN with smooth spectral filters



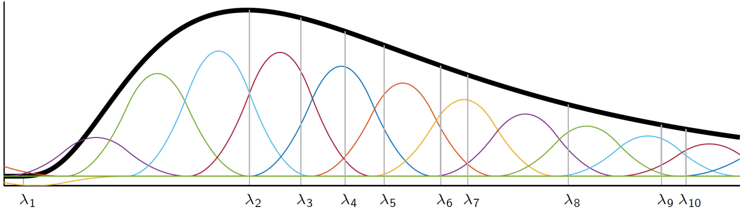
Consider a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$:

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

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

Henaff et al, "Deep Convolutional Networks on Graph-Structured Data", 2015

Spectral graph CNN with smooth spectral filters



Consider a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$:

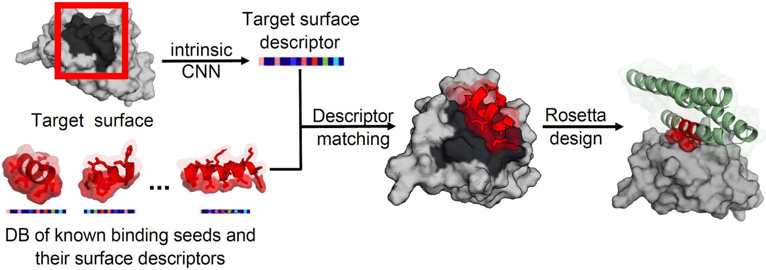
$$\mathbf{W} = \mathrm{Diag}(\mathbf{B}\boldsymbol{\alpha})$$

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

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

Henaff et al, "Deep Convolutional Networks on Graph-Structured Data", 2015

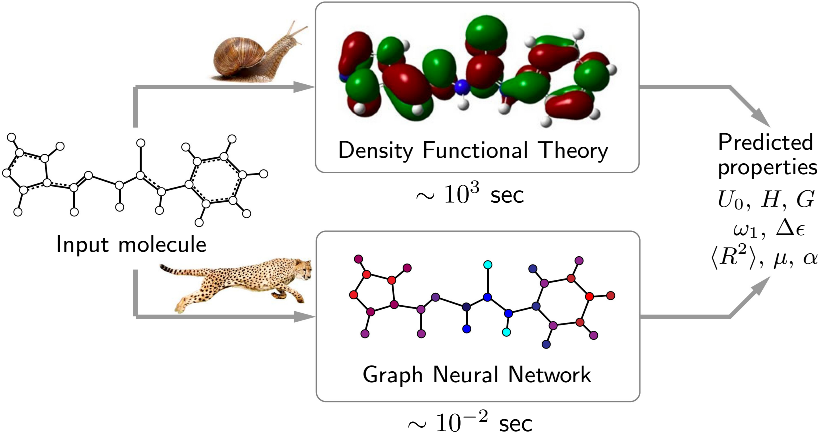
Application: Protein-Protein Interaction



Designing protein binder for the PD-L1 protein target

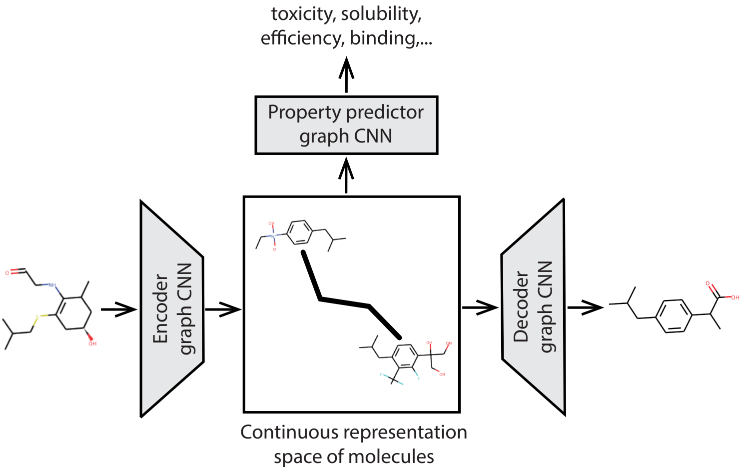
Gainza et al, "Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning", Nature Methods 2020

Molecule property prediction

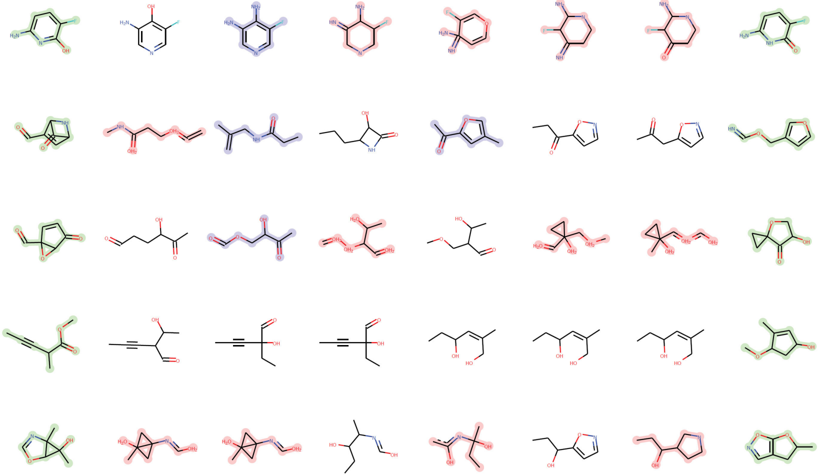


Duvenaud et al, "Convolutional Networks on Graphs for Learning Molecular Fingerprints", NIPS 2015; Gomez-Bombarelli et al, "Automatic chemical design using a data-driven continuous representation of molecules", ACS Cent. Sci. 2018

Generative models



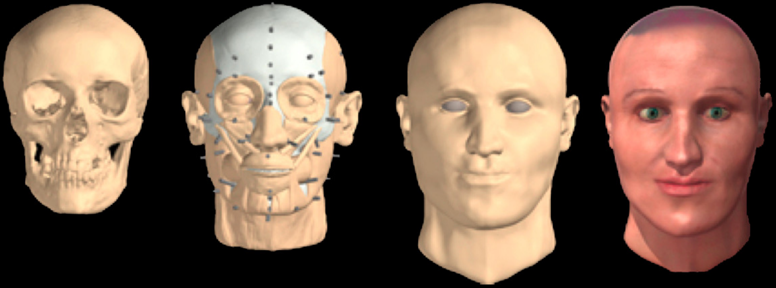
Molecule generation



Molecules generated with a graph VAE

Simonovsky and Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders", 2017; De Cao and Kipf, "MolGAN: An implicit generative model for small molecular graphs", 2018

Face from DNA



Claes et al, “Facial recognition from DNA using face-to-DNA classifiers”, Nature Communications 2019

Thank you!