Geometric Deep Learning

Emanuele Rodolà

Sapienza University of Rome
Italy

RegML Workshop, Genova, 01 July 2020
Audio signals

Images
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

2D:
only data

3D:
only structure
Fixed vs different domain

Social network (fixed graph)
Fixed vs different domain

Social network
(fixed graph)

3D shapes
(different manifolds)
Geometric learning ≠ Manifold learning

In manifold learning, we seek for a (possibly high-dimensional) manifold that justifies a given set of data points:
Geometric learning ≠ 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
- \textbf{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
- \( \text{CNN}_1 \): 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

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

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)

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

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

Extrinsic

Intrinsic
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

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

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

Triangular mesh

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

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

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

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?
Non-Euclidean convolution?
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.

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

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

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**

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**

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):

![Diagram showing translation-invariant convolution on surfaces](image-url)
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 $u_{ij}$ 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** $u_{ij}$ around $i$ (e.g. geodesic polar)

- **Local weights** $w(u_{ij})$, e.g. Gaussians with learnable $\mu, \Sigma$:
  $$w = \exp\left(-\left(u_{ij} - \mu\right)^\top \Sigma^{-1} \left(u_{ij} - \mu\right)\right)$$

Monti et al, “Geometric deep learning on graphs and manifolds using mixture model CNNs”, CVPR 2016
Spatial convolution on meshes

- **Local system of coordinates** $u_{ij}$ around $i$ (e.g. geodesic polar)

- **Local weights** $w(u_{ij})$, e.g. Gaussians with learnable $\mu, \Sigma$:
  \[
  w = \exp\left( -(u_{ij} - \mu)^\top \Sigma^{-1} (u_{ij} - \mu) \right)
  \]

- **Spatial convolution of feature $f$ with filter $g$**:
  - Represent the **input** $f$ as above
    \[
    \Rightarrow f
    \]
  - Represent the **learnable** filter $g$
    as above $\Rightarrow g$
  - Sum up the element-wise products $\Rightarrow f^\top g$

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** $\Delta$ acting locally on the neighborhood of $i$:

  $$(\Delta x)_i = \sum_j w_{ij} (x_j - x_i)$$
Spectral convolution on meshes

- **Laplacian operator** $\Delta$ acting locally on the neighborhood of $i$:

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

  $= \text{neighborhood avg} - \text{value at } i$
Spectral convolution on meshes

- **Laplacian operator** $\Delta$ acting locally on the neighborhood of $i$:
  \[
  (\Delta x)_i = \sum_j w_{ij} (x_j - x_i)
  \]

- Eigenvectors of the Laplacian $\Delta = \Phi \Lambda \Phi^T$ are a generalization of the Fourier transform:
  \[
  \hat{x} = \Phi^T x
  \]

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

- **Laplacian operator** $\Delta$ acting locally on the neighborhood of $i$:
  $$ (\Delta x)_i = \sum_j w_{ij} (x_j - x_i) $$

- **Eigenvectors of the Laplacian** $\Delta = \Phi \Lambda \Phi^T$ are a generalization of the **Fourier transform**:
  $$ \hat{x} = \Phi^T x $$

- **Spectral convolution**
  $$ x \star y = \Phi \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{pmatrix} \hat{x} $$

Bruna et al, "Spectral Networks and Locally Connected Networks on Graphs", 2014
First eigenfunctions of 1D Euclidean Laplacian = standard Fourier basis
Laplacian eigenfunctions: manifold

First eigenfunctions of a manifold Laplacian
Laplacian eigenfunctions: graph

First eigenfunctions of a graph Laplacian

$\phi_1$, $\phi_2$, $\phi_3$, $\phi_4$
Fourier analysis: Euclidean space

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

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

$$= \alpha_1 + \alpha_2 + \alpha_3 + \ldots$$
Fourier analysis: Euclidean space

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

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

\[
\hat{f}_k = \langle f, e^{ikx} \rangle_{L^2([-\pi, \pi])}
\]

\[
= \alpha_1 + \alpha_2 + \alpha_3 + \ldots
\]
Fourier analysis: Euclidean space

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

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

$$\hat{f}_k = \langle f, e^{ikx} \rangle_{L^2([-\pi, \pi])}$$

$$= \alpha_1 + \alpha_2 + \alpha_3 + \ldots$$

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

A function \( f : \mathcal{X} \rightarrow \mathbb{R} \) can be written as Fourier series

\[
f(x) = \sum_{k \geq 1} \int_{\mathcal{X}} f(x') \phi_k(x') \, dx' \, \phi_k(x)
\]

\( \hat{f}_k = \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \)

\[
\phi_1 + \phi_2 + \phi_3 + \ldots
\]

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

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

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

\[
(f \ast g)(x) = \int_{-\pi}^{\pi} f(x')g(x-x')dx'
\]

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

\[
\widehat{(f \ast g)} = \hat{f} \cdot \hat{g}
\]
Convolution theorem

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

\[
(f \ast g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'
\]

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

\[
\mathbf{f} \ast \mathbf{g} = \begin{bmatrix}
g_1 & g_2 & \cdots & \cdots & g_n \\
g_n & g_1 & g_2 & \cdots & g_{n-1} \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
g_3 & g_4 & \cdots & g_1 & g_2 \\
g_2 & g_3 & \cdots & \cdots & g_1
\end{bmatrix} \begin{bmatrix}
f_1 \\
f_2 \\
\vdots \\
f_n
\end{bmatrix}
\]
Convolution theorem

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

\[
(f \ast g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'
\]

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

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

\( \text{circulant matrix} \)
Convolution theorem

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

\[
(f \ast g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'
\]

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

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

diagonalized by Fourier basis
Convolution theorem

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

\[
(f \ast g)(x) = \int_{-\pi}^{\pi} f(x')g(x-x')dx'
\]

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

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

\[
= \Phi \begin{bmatrix}
\hat{g}_1 \\
\vdots \\
\hat{g}_n
\end{bmatrix} \Phi^\top \mathbf{f}
\]
**Convolution theorem**

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

\[
(f \ast g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'
\]

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

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

\[
= \Phi \begin{bmatrix}
\hat{g}_1 \\
\vdots \\
\hat{g}_n
\end{bmatrix}
\begin{bmatrix}
\hat{f}_1 \\
\vdots \\
\hat{f}_n
\end{bmatrix}
\]
Convolution theorem

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

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

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

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

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

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

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

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

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

product in the Fourier domain
Spectral convolution

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

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

- Product in the Fourier domain
- Inverse Fourier transform
Spectral convolution

**Generalized convolution** of \( f, g \in L^2(X) \) can be defined by analogy

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

In matrix-vector notation

\[
f \star g = \Phi (\Phi^T g) \circ (\Phi^T f)
\]
Spectral convolution

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

\[
f \ast g = \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \phi_k
\]

In matrix-vector notation

\[
f \ast g = \Phi \text{diag}(\hat{g}_1, \ldots, \hat{g}_n) \Phi^\top f
\]
Spectral convolution

**Generalized convolution** of \( f, g \in L^2(X) \) can be defined by analogy

\[
f \ast g = \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(X)} \langle g, \phi_k \rangle_{L^2(X)} \phi_k
\]

In matrix-vector notation

\[
f \ast g = \Phi \text{diag}(\hat{g}_1, \ldots, \hat{g}_n) \Phi^\top f
\]

Not shift-invariant! (\( G \) has no circulant structure)
Filter coefficients depend on basis \( \phi_1, \ldots, \phi_n \)
Spectral convolution

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

\[
f \ast g = \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \phi_k
\]

In matrix-vector notation

\[
f \ast g = \Phi \text{diag}(\hat{g}_1, \ldots, \hat{g}_n) \Phi^\top f
\]

Not shift-invariant! (\( G \) has no circulant structure)
Generalized convolution of $f, g \in L^2(\mathcal{X})$ can be defined by analogy

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

In matrix-vector notation

$$f \ast g = \Phi \text{diag}(\hat{g}_1, \ldots, \hat{g}_n) \Phi^\top f$$

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

Function $x$
Basis dependence

‘Edge detecting’ spectral filter $\tilde{\Phi} \hat{Y} \Phi^T x$
Basis dependence

Same spectral filter, different basis $\Psi \hat{Y} \Psi^T x$
Spectral convolution on meshes

- **Laplacian operator** $\Delta$ acting locally on the neighborhood of $i$:
  $$(\Delta x)_i = \sum_j w_{ij} (x_j - x_i)$$

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

- **Spectral convolution**:
  $$x \ast y = \Phi \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{pmatrix} \hat{x}$$
Spectral convolution on meshes

- **Laplacian operator** $\Delta$ acting locally on the neighborhood of $i$:

$$ (\Delta x)_i = \sum_j w_{ij} (x_j - x_i) $$

- Eigenvectors of the Laplacian $\Delta = \Phi \Lambda \Phi^T$ are a generalization of the Fourier transform:

$$ \hat{x} = \Phi^T x $$

- **Spectral convolution** defined as a filter applied on the Laplacian:

$$ X' = \Phi \tau(\Lambda) \Phi^T X $$
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

$$\tau(\Delta)f = \Phi \tau(\Lambda) \Phi^T f$$

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

\[
\tau(\Delta) f = \Phi \begin{pmatrix} \tau(\lambda_1) \\ \vdots \\ \tau(\lambda_n) \end{pmatrix} \Phi^\top f
\]

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 $\alpha$

$$\tau_\alpha(\Delta)f = \Phi \begin{pmatrix} \tau_\alpha(\lambda_1) \\ \vdots \\ \tau_\alpha(\lambda_n) \end{pmatrix} \Phi^T f$$

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

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

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

Henaff et al, “Deep Convolutional Networks on Graph-Structured Data”, 2015
Consider a **linear combination of smooth kernel functions** $\beta_1(\lambda), \ldots, \beta_r(\lambda)$:

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

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

Henaff et al, “Deep Convolutional Networks on Graph-Structured Data”, 2015
Consider a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$:

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

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

Henaff et al, “Deep Convolutional Networks on Graph-Structured Data”, 2015
Consider a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$:

$$W = \text{Diag}(B\alpha)$$

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

Henaff et al, “Deep Convolutional Networks on Graph-Structured Data”, 2015
Consider a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$:

$$W = \text{Diag}(B\alpha)$$

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

$O(1)$ 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

Molecule property prediction

Generative models

Generative models can be used to predict properties of molecules, such as toxicity, solubility, efficiency, and binding affinity. The process involves encoding the molecular structure using a graph CNN, followed by a property predictor that uses another graph CNN to predict the desired property. The generated structure is then decoded using another graph CNN to obtain a new molecular structure.
Molecule generation

Molecules generated with a graph VAE

Face from DNA

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