Agenda

- Deep learning on regular structures
  - Multi-view representation
  - Volumetric Representation

- **Intrinsic deep learning on manifolds**
  - Spectral methods
  - Spatial methods
  - Embedding-based methods

- Deep learning on point cloud and parametric models
Extrinsic vs Intrinsic CNNs

Intrinsic
Riemannian geometry in one minute

- Manifold $\mathcal{X}$ = topological space
- No global Euclidean structure
- **Tangent plane** $T_x \mathcal{X} =$ local Euclidean representation of manifold $\mathcal{X}$ around $x$
Riemannian geometry in one minute

- Manifold $\mathcal{X} = \text{topological space}$
- No global Euclidean structure
- Tangent plane $T_x \mathcal{X} = \text{local Euclidean representation of manifold } \mathcal{X} \text{ around } x$
- Riemannian metric
  \[
  \langle \cdot, \cdot \rangle_{T_x \mathcal{X}} : T_x \mathcal{X} \times T_x \mathcal{X} \to \mathbb{R}
  \]
depending smoothly on $x$
Riemannian geometry in one minute

- Manifold $\mathcal{X} = \text{topological space}$
- No global Euclidean structure
- **Tangent plane** $T_x \mathcal{X} = \text{local Euclidean representation of manifold } \mathcal{X} \text{ around } x$
- **Riemannian metric**
  \[
  \langle \cdot, \cdot \rangle_{T_x \mathcal{X}} : T_x \mathcal{X} \times T_x \mathcal{X} \to \mathbb{R}
  \]
  depending smoothly on $x$
- **Isometry** = metric-preserving shape deformation
**Riemannian geometry in one minute**

- Manifold $\mathcal{X} = $ topological space
- No global Euclidean structure
- **Tangent plane** $T_x\mathcal{X} =$ local Euclidean representation of manifold $\mathcal{X}$ around $x$
- **Riemannian metric**
  \[
  \langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}
  \]
  depending smoothly on $x$
- **Isometry** = metric-preserving shape deformation
- **Intrinsic** = expressed solely in terms of the Riemannian metric
Riemannian geometry in one minute

- **Manifold** $\mathcal{X} = \text{topological space}
- No global Euclidean structure
- **Tangent plane** $T_x \mathcal{X} = \text{local Euclidean representation of manifold } \mathcal{X} \text{ around } x$
- **Riemannian metric**
  \[ \langle \cdot, \cdot \rangle_{T_x \mathcal{X}} : T_x \mathcal{X} \times T_x \mathcal{X} \to \mathbb{R} \]
  depending smoothly on $x$
- **Isometry** = metric-preserving shape deformation
- **Intrinsic** = expressed solely in terms of the Riemannian metric
- **Geodesic** = shortest path on $\mathcal{X}$ between $x$ and $x'$

\[ \Gamma(x, x') \]
Calculus on manifolds: scalar fields

Scalar field $f : \mathcal{X} \rightarrow \mathbb{R}$
Scalar field $f : \mathcal{X} \to \mathbb{R}$

Hilbert space $L^2(\mathcal{X})$ with inner product

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

where $dx = \text{area element induced by the Riemannian metric}$
Calculus on manifolds: Laplacian operator

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

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

“difference between $f(x)$ and average value of $f$ around $x$”
Calculus on manifolds: Laplacian operator

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

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

“difference between $f(x)$ and average value of $f$ around $x$”

- **Intrinsic** (expressed solely in terms of the Riemannian metric)
Calculus on manifolds: Laplacian operator

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

\[ \Delta f = -\text{div}(\nabla f) \]

“difference between \( f(x) \) and average value of \( f \) around \( x \)”

- **Intrinsic** (expressed solely in terms of the Riemannian metric)
- **Isometry-invariant**
Calculus on manifolds: Laplacian operator

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

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

“difference between $f(x)$ and average value of $f$ around $x$”

- Intrinsic (expressed solely in terms of the Riemannian metric)
- Isometry-invariant
- Self-adjoint $\langle \Delta f, g \rangle_{L^2(\mathcal{X})} = \langle f, \Delta g \rangle_{L^2(\mathcal{X})}$
Calculus on manifolds: Laplacian operator

Laplacian \( \Delta : L^2(\mathcal{X}) \rightarrow L^2(\mathcal{X}) \)

\[ \Delta f = -\text{div}(\nabla f) \]

“difference between \( f(x) \) and average value of \( f \) around \( x \)”

- **Intrinsic** (expressed solely in terms of the Riemannian metric)
- **Isometry-invariant**
- **Self-adjoint** \( \langle \Delta f, g \rangle_{L^2(\mathcal{X})} = \langle f, \Delta g \rangle_{L^2(\mathcal{X})} \Rightarrow \) orthogonal eigenfunctions
Calculus on manifolds: Laplacian operator

Laplacian \( \Delta : L^2(\mathcal{M}) \to L^2(\mathcal{M}) \)

\[ \Delta f = -\text{div}(\nabla f) \]

“difference between \( f(x) \) and average value of \( f \) around \( x \)”

- **Intrinsic** (expressed solely in terms of the Riemannian metric)
- **Isometry-invariant**
- **Self-adjoint** \( \langle \Delta f, g \rangle_{L^2(\mathcal{M})} = \langle f, \Delta g \rangle_{L^2(\mathcal{M})} \Rightarrow \) orthogonal eigenfunctions
- **Positive semidefinite**
Calculus on manifolds: Laplacian operator

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

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

"difference between $f(x)$ and average value of $f$ around $x"$

- **Intrinsic** (expressed solely in terms of the Riemannian metric)
- **Isometry-invariant**
- **Self-adjoint** $\langle \Delta f, g \rangle_{L^2(\mathcal{X})} = \langle f, \Delta g \rangle_{L^2(\mathcal{X})} \Rightarrow$ orthogonal eigenfunctions
- **Positive semidefinite** $\Rightarrow$ non-negative eigenvalues
Discrete Laplacian

Undirected graph \((V, E)\)

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

Triangular mesh \((V, E, F)\)

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

\[a_i = \text{local area element}\]

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

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

$a_i = \text{local area element}$

In matrix-vector notation

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

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

Tutte 1963; MacNeal 1949; Duffin 1959; Pinkall, Polthier 1993
Laplacian eigenfunctions and eigenvalues

Laplacian $\Delta$ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta \phi_i(x) = \lambda_i \phi_i(x), \quad i = 1, 2, \ldots$$
Laplacian eigenfunctions and eigenvalues

Laplacian $\Delta$ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta \phi_i(x) = \lambda_i \phi_i(x), \quad i = 1, 2, \ldots$$

- Eigenfunctions are real and orthonormal $\langle \phi_i, \phi_j \rangle_{L^2(\mathcal{X})} = \delta_{ij}$
Laplacian eigenfunctions and eigenvalues

Laplacian $\Delta$ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta \phi_i(x) = \lambda_i \phi_i(x), \quad i = 1, 2, \ldots$$

- Eigenfunctions are **real** and **orthonormal** $\langle \phi_i, \phi_j \rangle_{L^2(\mathcal{X})} = \delta_{ij}$
- Eigenvalues are **non-negative** $0 = \lambda_1 \leq \lambda_2 \leq \ldots$
Laplacian \( \Delta \) of a compact manifold \( \mathcal{X} \) has countably many eigenfunctions

\[
\Delta \phi_i(x) = \lambda_i \phi_i(x), \quad i = 1, 2, \ldots
\]

- Eigenfunctions are real and orthonormal \( \langle \phi_i, \phi_j \rangle_{L^2(\mathcal{X})} = \delta_{ij} \)
- Eigenvalues are non-negative \( 0 = \lambda_1 \leq \lambda_2 \leq \ldots \)

Eigendecomposition of a discrete Laplacian matrix \( \Delta \)

\[
\Delta \Phi = \Phi \Lambda
\]

where \( \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n) \) is a diagonal matrix of eigenvalues and \( \Phi = (\phi_1, \ldots, \phi_n) \) is a matrix of eigenvectors
Laplacian eigenfunctions and eigenvalues

Laplacian $\Delta$ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta \phi_i(x) = \lambda_i \phi_i(x), \quad i = 1, 2, \ldots$$

- Eigenfunctions are real and orthonormal $\langle \phi_i, \phi_j \rangle_{L^2(\mathcal{X})} = \delta_{ij}$
- Eigenvalues are non-negative $0 = \lambda_1 \leq \lambda_2 \leq \ldots$

Eigendecomposition of a discrete Laplacian matrix $\Delta$

$$\Lambda^{-1}(D - W)\Phi = \Phi \Lambda$$

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix of eigenvalues and $\Phi = (\phi_1, \ldots, \phi_n)$ is a matrix of eigenvectors.
Laplacian eigenfunctions and eigenvalues

Laplacian $\Delta$ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta \phi_i(x) = \lambda_i \phi_i(x), \quad i = 1, 2, \ldots$$

- Eigenfunctions are real and orthonormal $\langle \phi_i, \phi_j \rangle_{L^2(\mathcal{X})} = \delta_{ij}$
- Eigenvalues are non-negative $0 = \lambda_1 \leq \lambda_2 \leq \ldots$

Eigendecomposition of a discrete Laplacian matrix $\Delta$

$$\begin{pmatrix} D - W \end{pmatrix} \Phi = \Lambda \Phi \Lambda$$

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix of eigenvalues and
$\Phi = (\phi_1, \ldots, \phi_n)$ is an $\Lambda$-orthonormal matrix of eigenvectors
$$(\Phi^\top \Lambda \Phi = \mathbf{I})$$
Laplacian eigenvectors \( \phi_i(x) \) of a compact manifold \( \mathcal{X} \) have countably many eigenfunctions

\[
\Delta \phi_i(x) = \lambda_i \phi_i(x), \quad i = 1, 2, \ldots
\]

- Eigenfunctions are real and orthonormal \( \langle \phi_i, \phi_j \rangle_{L^2(\mathcal{X})} = \delta_{ij} \)
- Eigenvalues are non-negative \( 0 = \lambda_1 \leq \lambda_2 \leq \ldots \)

Eigendecomposition of a discrete Laplacian matrix \( \Delta \)

\[
A^{-1/2}(D - W)A^{-1/2}A^{1/2}\Phi = A^{1/2}\Phi\Lambda
\]

where \( \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n) \) is a diagonal matrix of eigenvalues and \( \Phi = (\phi_1, \ldots, \phi_n) \) is an \( A \)-orthonormal matrix of eigenvectors

\( (\Phi^\top A \Phi = I) \)
Laplacian eigenfunctions and eigenvalues

Laplacian $\Delta$ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta \phi_i(x) = \lambda_i \phi_i(x), \quad i = 1, 2, \ldots$$

- Eigenfunctions are real and orthonormal $\langle \phi_i, \phi_j \rangle_{L^2(\mathcal{X})} = \delta_{ij}$
- Eigenvalues are non-negative $0 = \lambda_1 \leq \lambda_2 \leq \ldots$

Eigendecomposition of a discrete Laplacian matrix $\Delta$

$$A^{-1/2}(D - W)A^{-1/2} \Psi = \Psi \Lambda$$

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix of eigenvalues and $\Psi = (\psi_1, \ldots, \psi_n)$ is an orthonormal matrix of eigenvectors ($\Psi^T \Psi = I$)
Laplacian \( \Delta \) of a compact manifold \( \mathcal{X} \) has countably many eigenfunctions
\[
\Delta \phi_i(x) = \lambda_i \phi_i(x), \quad i = 1, 2, \ldots
\]

- Eigenfunctions are real and orthonormal \( \langle \phi_i, \phi_j \rangle_{L^2(\mathcal{X})} = \delta_{i,j} \)
- Eigenvalues are non-negative \( 0 = \lambda_1 \leq \lambda_2 \leq \ldots \)

Eigendecomposition of a discrete Laplacian matrix \( \Delta \)
\[
A^{-1/2} (D - W) A^{-1/2} \Psi = \Psi \Lambda
\]
where \( \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n) \) is a diagonal matrix of eigenvalues and \( \Psi = (\psi_1, \ldots, \psi_n) \) is an orthonormal matrix of eigenvectors \( (\Psi^T \Psi = I) \)

Laplacian eigenvectors = smoothest orthonormal basis
Laplacian eigenfunctions: Euclidean

First eigenfunctions of 1D Euclidean Laplacian = standard Fourier basis
Laplacian eigenfunctions: non-Euclidean

First eigenfunctions of a manifold Laplacian

$\phi_1$, $\phi_2$, $\phi_3$, $\phi_4$
A function $f : [-\pi, \pi] \rightarrow \mathbb{R}$ can be written as \textbf{Fourier series}

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

$$= \hat{f}_0 + \hat{f}_1 + \hat{f}_2 + \ldots$$
Fourier analysis: Euclidean

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

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

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

$$= \hat{f}_0 + \hat{f}_1 + \hat{f}_2 + \ldots$$
A function $f : [−\pi, \pi] \rightarrow \mathbb{R}$ can be written as Fourier series:

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

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

Fourier basis = Laplacian eigenfunctions: $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$
A function $f : \mathcal{X} \to \mathbb{R}$ can be written as Fourier series:

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

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

$$f = \hat{f}_1 + \hat{f}_2 + \hat{f}_3 + \ldots$$

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

\[ c \text{ [m}^2\text{/sec}] = \text{thermal diffusivity constant} \]
Heat diffusion on manifolds

\[
\begin{align*}
    f_t(x, t) &= -\Delta f(x, t) \\
    f(x, 0) &= f_0(x)
\end{align*}
\]

- \( f(x, t) \) = amount of heat at point \( x \) at time \( t \)
- \( f_0(x) \) = initial heat distribution
Heat diffusion on manifolds

\[ \begin{aligned}
  f_t(x, t) &= -\Delta f(x, t) \\
  f(x, 0) &= f_0(x)
\end{aligned} \]

- \( f(x, t) \) = 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

\[ f(x, t) = e^{-t\Delta} f_0(x) \]
Heat diffusion on manifolds

\[
\begin{aligned}
  &f_t(x, t) = -\Delta f(x, t) \\
  &f(x, 0) = f_0(x)
\end{aligned}
\]

- \( f(x, t) \) = 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

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

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

\[
= \int_{\mathcal{X}} f_0(x') \sum_{k \geq 1} e^{-t\lambda_k} \phi_k(x) \phi_k(x') \, dx'
\]
Heat diffusion on manifolds

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

- \( f(x, t) \) = 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

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

\[
= \int_{\mathcal{M}} f_0(x') \sum_{k \geq 1} e^{-t\lambda_k} \phi_k(x) \phi_k(x') \, dx'
\]

heat kernel \( h_t(x, x') \)
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 $\Delta \phi_k(x) = \lambda_k \phi_k(x)$

\[
f(x, t) = \int_{\mathcal{X}} f_0(x') \sum_{k \geq 1} e^{-t\lambda_k} \phi_k(x) \phi_k(x') \, dx'
\]

Heat kernel $h_t(x, x')$
Interpretation of the heat kernel

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

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

heat kernel $h_t(x,x')$
Interpretation of the heat kernel

Solution of the heat equation on a Euclidean space $[-\pi, \pi]$ expressed in the Laplacian eigenbasis

$-\frac{d^2}{dx^2}e^{ikx} = k^2 e^{ikx}$

\[
f(x, t) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f_0(x') \sum_{k \geq 0} e^{-tk^2} e^{ik(x-x')} \, dx'
\]

$\{\text{heat kernel } h_t(x-x')\}$
Interpretation of the heat kernel

Solution of the heat equation on a Euclidean space \([-\pi, \pi]\) expressed in the Laplacian eigenbasis

\[- \frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}\]

\[
f(x, t) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f_0(x') \sum_{k \geq 0} e^{-tk^2} e^{ik(x-x')} \, dx'
\]

\[
= (f_0 \ast h_t)(x)
\]

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

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

- **Shift-invariance:** $f(x - x_0) \ast g(x) = (f \ast g)(x - x_0)$
Convolution: Euclidean space

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

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

- **Shift-invariance:** $f(x - x_0) \ast g(x) = (f \ast g)(x - x_0)$
- Convolution operator commutes with Laplacian: $(\Delta f) \ast g = \Delta (f \ast g)$
Convolution: Euclidean space

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

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

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

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

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

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

\[
(\hat{f} \ast \hat{g}) = \hat{f} \cdot \hat{g}
\]
Convolution: Euclidean space

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

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

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

\[
(\hat{f} \cdot \hat{g}) = \hat{f} \cdot \hat{g}
\]

- **Efficient computation using FFT**
Convolution Theorem

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

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

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

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

circulant matrix
### Convolution Theorem

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

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

diagonalized by Fourier basis
Convolution Theorem

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

\[
\begin{align*}
\mathbf{f} \ast \mathbf{g} &= \\
&= \begin{bmatrix}
g_1 & g_2 & \ldots & \ldots & g_n \\
g_n & g_1 & g_2 & \ldots & g_{n-1} \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
g_3 & g_4 & \ldots & g_1 & g_2 \\
g_2 & g_3 & \ldots & \ldots & g_1 \\
\end{bmatrix}
\begin{bmatrix}
f_1 \\
\vdots \\
f_n \\
\end{bmatrix} \\
&= \Phi 
\begin{bmatrix}
\hat{g}_1 \\
\vdots \\
\hat{g}_n \\
\end{bmatrix}
\Phi^\top \mathbf{f}
\end{align*}
\]
Convolution Theorem

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

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

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

diagonalized by Fourier basis
Convolution Theorem

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

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

\[
= \Phi \begin{bmatrix}
  \hat{f}_1 \cdot \hat{g}_1 \\
  \vdots \\
  \hat{f}_n \cdot \hat{g}_n
\end{bmatrix}
\]
Spectral convolution of \( f, g \in L^2(X) \) can be defined by analogy

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

Not shift-invariant! (\( G \) has no circulant structure)

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

Not shift-invariant! (G has no circulant structure)
Spectral 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

\[
f \ast g = \Phi (\Phi^\top g) \circ (\Phi^\top f)
\]
Spectral 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

$$f \star g = \Phi \text{diag}(\hat{g}_1, \ldots, \hat{g}_n) \Phi^T f$$
Spectral convolution

Spectral 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)
Spectral convolution

Spectral 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

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

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

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

In matrix-vector notation

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

- Not shift-invariant! ($G$ has no circulant structure)
- Filter coefficients depend on basis $\phi_1, \ldots, \phi_n$
Different formulations of non-Euclidean CNNs

Spectral domain

Spatial domain

Embedding domain
Spectral CNN

Convolutional layer expressed in the **spectral domain**

\[
g_l = \xi \left( \sum_{l' = 1}^{p} \Phi \hat{W}_{l,l'} \Phi^\top f_{l'} \right) \quad l = 1, \ldots, q
\]

\[
l' = 1, \ldots, p
\]

where \( \hat{W}_{l,l} = n \times n \) diagonal matrix of filter coefficients
Spectral CNN

Convolutional layer expressed in the spectral domain

\[ g_l = \xi \left( \sum_{l'=1}^{p} \Phi \hat{W}_{l,l'} \Phi^\top f_{l'} \right) \]

where \( \hat{W}_{l,l} = n \times n \) diagonal matrix of filter coefficients

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

Bruna et al. 2014
Spectral CNN

Convolutional layer expressed in the spectral domain

\[
g_l = \xi \left( \sum_{l'=1}^{p} \Phi \hat{W}_{l,l'} \Phi^\top f_{l'} \right) \quad l = 1, \ldots, q
\]

where \( \hat{W}_{l,l} = n \times n \) diagonal matrix of filter coefficients

- \( \mathcal{O}(n) \) parameters per layer
- \( \mathcal{O}(n^2) \) computation of forward and inverse Fourier transforms \( \Phi^\top, \Phi \) (no FFT on manifolds or graphs)

Bruna et al. 2014
Spectral CNN

Convolutional layer expressed in the spectral domain

\[ g_l = \xi \left( \sum_{l'=1}^{p} \Phi \hat{W}_{l,l'} \Phi^\top f_{l'} \right) \quad l = 1, \ldots, q \]

where \( \hat{W}_{l,l} = n \times n \) diagonal matrix of filter coefficients

- \( O(n) \) parameters per layer
- \( O(n^2) \) computation of forward and inverse Fourier transforms \( \Phi^\top, \Phi \) (no FFT on manifolds or graphs)
- No guarantee of spatial localization of filters

Bruna et al. 2014
Spectral CNN

Convolutional layer expressed in the spectral domain

\[ g_l = \xi \left( \sum_{l' = 1}^{p} \Phi \hat{W}_{l,l'} \Phi^\top f_{l'} \right) \quad l = 1, \ldots, q \]

where \( \hat{W}_{l,l} = n \times n \) diagonal matrix of filter coefficients

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

Bruna et al. 2014
Basis dependence

Function $f$
Basis dependence

‘Edge detecting’ spectral filter $\Phi \hat{W} \Phi^T f$
Basis dependence

\[ \hat{W} \Psi \hat{\Psi}^T f \]
Basis dependence
Localization and Smoothness

In the Euclidean setting (by Parseval’s identity)

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

Bruna et al. 2014; Henaff, Bruna, LeCun 2015
Localization and Smoothness

In the Euclidean setting (by Parseval’s identity)

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

⇒ Localization in space = smoothness in frequency domain

Bruna et al. 2014; Henaff, Bruna, LeCun 2015
Localization and Smoothness

In the Euclidean setting (by Parseval’s identity)

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

⇒ Localization in space = smoothness in frequency domain

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

In the Euclidean setting (by Parseval’s identity)

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

⇒ Localization in space = smoothness in frequency domain

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

Application of the filter

\[
\tau(\Delta)f = \Phi \tau(\Lambda) \Phi^\top f
\]

Bruna et al. 2014; Henaff, Bruna, LeCun 2015
Localization and Smoothness

In the Euclidean setting (by Parseval’s identity)

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

⇒ Localization in space = smoothness in frequency domain

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

Application of the filter

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

Bruna et al. 2014; Henaff, Bruna, LeCun 2015
Localization and Smoothness

In the Euclidean setting (by Parseval’s identity)

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

\Rightarrow \text{Localization in space} = \text{smoothness in frequency domain}

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

Application of the parametric filter with learnable parameters \( \alpha \)

\[ \tau_\alpha(\Delta) f = \Phi \begin{pmatrix} \tau_\alpha(\lambda_1) \\ \vdots \\ \tau_\alpha(\lambda_n) \end{pmatrix} \Phi^T f \]
Spectral CNN with polynomial filters

Represent spectral transfer function as a polynomial or order $r$

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

where $\alpha = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters
Spectral CNN with polynomial filters

Represent spectral transfer function as a polynomial of order $r$

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

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

腘 $O(1)$ parameters per layer

Defferrard, Bresson, Vandergheynst 2016
Spectral CNN with polynomial filters

Represent spectral transfer function as a polynomial or order \( r \)

\[
\tau_{\alpha}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j
\]

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

- \( \mathcal{O}(1) \) parameters per layer
- Filters have guaranteed \( r \)-hops support
Represent spectral transfer function as a polynomial or order \( r \)

\[
\tau_{\alpha}(\Delta) = \sum_{j=0}^{r} \alpha_j \Delta^j
\]

where \( \alpha = (\alpha_0, \ldots, \alpha_r)^T \) is the vector of filter parameters

- \( \mathcal{O}(1) \) parameters per layer
- Filters have guaranteed \( r \)-hops support
- No explicit computation of \( \Phi^T, \Phi \Rightarrow \mathcal{O}(nr) \) complexity
Spectral CNN with polynomial filters

Represent spectral transfer function as a polynomial of order $r$

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

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

- $\bigO(1)$ parameters per layer
- Filters have guaranteed $r$-hops support
- No explicit computation of $\Phi^\top$, $\Phi \Rightarrow \bigO(nr)$ complexity
- Does not generalize across domains

Defferrard, Bresson, Vandergheynst 2016
Laplacian eigenbases on non-isometric domains
Functional maps

\[ y \approx f_1 + f_2 + \ldots + f_K \]
\[ x \approx g_1 + g_2 + \ldots + g_K \]

Functional maps

\[ f \approx \hat{f}_1 + \hat{f}_2 + \ldots + \hat{f}_K \]

\[ g \approx \hat{g}_1 + \hat{g}_2 + \ldots + \hat{g}_K \]

\[ T : L^2(\mathcal{X}) \to L^2(\mathcal{Y}) \]

Functional maps

\[ y = f \approx f_1 + f_2 + \ldots + f_K \]

\[ x = g \approx g_1 + g_2 + \ldots + g_K \]

Functional maps

\[ f \approx \hat{f}_1 + \hat{f}_2 + \ldots + \hat{f}_K \]

\[ g \approx \hat{g}_1 + \hat{g}_2 + \ldots + \hat{g}_K \]

Functional maps

\[ y \approx \hat{f}_1 + \hat{f}_2 + \ldots + \hat{f}_K \]

\[ x \approx \hat{g}_1 + \hat{g}_2 + \ldots + \hat{g}_K \]

Basis synchronization with functional maps

\[ f \approx \hat{f}_1 + \hat{f}_2 + \ldots + \hat{f}_K \]

\[ g \approx \hat{g}_1 + \hat{g}_2 + \ldots + \hat{g}_K \]

\[ T \approx \Psi \Phi^T \]

Basis synchronization with functional maps

\[ f \approx \hat{f}_1 + \hat{f}_2 + \ldots + \hat{f}_K \]

\[ g \approx \hat{g}_1 + \hat{g}_2 + \ldots + \hat{g}_K \]

\[ T \approx \Psi(\Phi C)^T \]

Basis synchronization with functional maps

\[ f \approx \tilde{f}_1 + \tilde{f}_2 + \ldots + \tilde{f}_K \]

\[ g \approx \hat{g}_1 + \hat{g}_2 + \ldots + \hat{g}_K \]

\[ T \approx \Psi I \Phi^T \]

Filtering in different bases

\[ \Phi \tau(\Lambda_\Phi) \Phi^\top \delta_0 \]

\[ \Psi \tau(\Lambda_\Psi) \Psi^\top \delta_0 \]

Apply spectral filter \( \tau(\lambda) \) in different bases \( \Phi \) and \( \Psi \)

\( \Rightarrow \) different results!
Filtering in different bases

\[
\Phi \tau(\Lambda_\Phi) \Phi^\top \delta_0 \quad \text{Canonical shape with basis } \Sigma, \Lambda \\
\Psi \tau(\Lambda_\Psi) \Psi^\top \delta_0
\]

Apply spectral filter \( \tau(\lambda) \) in different bases \( \Phi \) and \( \Psi \)

\[\Rightarrow \text{different results!}\]
Filtering in synchronized bases

Apply spectral filter $\tau(\lambda)$ in synchronized bases $\Phi C_\Phi$ and $\Psi C_\Psi$  
$\Rightarrow$ similar results!
Spectral CNN

Convolutional filter of a Spectral CNN

- Fixed basis $\Rightarrow$ Does not generalize across domains
- Possible $O(n)$ complexity avoiding explicit FT and IFT

Bruna et al. 2014
Basis synchronization allows generalization across domains

Explicit FT and IFT

(Jaderberg et al. 2015); Yi et al. 2017
Example: normal prediction with SpecTN

Groundtruth

Predicted

Yi et al. 2017
Example: shape segmentation with SpecTN

Predicted

Groundtruth

Yi et al. 2017
Different formulations of non-Euclidean CNNs

Spectral domain

Spatial domain

Embedding domain
**Convolution**

**Euclidean**

**Spatial domain**

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

**Spectral domain**

\[\hat{(f \ast g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)\]

‘Convolution Theorem’

**Non-Euclidean**

\[\left(f \ast g\right)_k = \langle f, \phi_k \rangle_{L^2(X)} \langle g, \phi_k \rangle_{L^2(X)}\]
Patch operator

Boscaini et al. 2015
Patch operator

- Image

- Manifold

\[ \Delta \mathcal{B} \ni B : \mathcal{B} \mapsto \mathcal{Z}(x) \]

- Local system of coordinates: bijection

Boscaini et al. 2015
Patch operator

- Local system of coordinates: bijection $\varsigma_x : B_{\rho_0}(x) \rightarrow [0, 1]^2$
Patch operator

- Local system of coordinates: bijection $\varsigma_x : B_{\rho_0}(x) \to [0, 1]^2$
- Patch operator $\mathcal{D} : L^2(\mathcal{X}) \to L^2([0, 1]^2)$ mapping $f$ around $x$

$$(\mathcal{D}(x)f)(\mathbf{u}) = (f \circ \varsigma_x^{-1})(\mathbf{u})$$

Boscaini et al. 2015
**Patch operator**

- **Local system of coordinates:** bijection $\varsigma_x : B_{\rho_0}(x) \to [0, 1]^2$
- **Patch operator** $\mathcal{D} : L^2(\mathcal{X}) \to L^2([0, 1]^2)$ mapping $f$ around $x$
  \[(\mathcal{D}(x)f)(\mathbf{u}) = (f \circ \varsigma_x^{-1})(\mathbf{u})\]
• Local system of coordinates: bijection $\varsigma_x : B_{\rho_0}(x) \to [0, 1]^2$

• **Patch operator** applying weighting function $w_u(x, x') = \delta_{\varsigma_x^{-1}(u)}(x')$

$$ (\mathcal{D}(x)f)(u) = \langle f, w_u(x, x') \rangle_{L^2(X)} $$
Patch operator

- Local system of coordinates: bijection $\varsigma_x : B_{\rho_0}(x) \rightarrow [0, 1]^2$
- **Patch operator** applying weighting function $w_u(x, x') = \delta_{\varsigma_x^{-1}(u)}(x')$

$$ (\mathcal{D}(x)f)(u) = \langle f, w_u(x, x') \rangle_{L^2(X)} $$

In Euclidean case, $w(x, x')$ is shift-invariant

Boscaini et al. 2015
Spatial convolution of $f \in L^2(\mathcal{X})$ with continuous filter $g \in L^2([0, 1]^2)$

$$(f \ast g)(x) = \int_{[0, 1]^2} g(u) (\mathcal{D}(x)f)(u) \, du$$
Spatial convolution of \( f \in L^2(\mathcal{X}) \) with discrete filter \( g = (g_1, \ldots, g_J) \)

\[
(f \ast g)(x) = \sum_{j=1}^{J} g_j(D(x)f)_j
\]
Geodesic polar patch operator

Patch expressed in local geodesic polar coordinates

\[(D(x)f)(\rho, \theta) = \int_{\mathcal{X}} \underbrace{w_\rho(x, x') w_\theta(x, x') f(x')}_{w_{\rho\theta}(x, x')} dx'\]

Radial weight

\[w_\rho(x, x') \propto e^{-(d_{\mathcal{X}}(x, x') - \rho)^2 / \sigma_\rho^2}\]

Angular weight

\[w_\theta(x, x') \propto e^{-d_{\mathcal{X}}(\Gamma_\theta(x), x') / \sigma_\theta^2}\]

Kokkinos et al. 2012; Boscaini et al. 2015
Geodesic polar patch operator construction

Weighting functions of the geodesic polar patch operator shown in $(\rho, \theta)$ coordinates (contours mark the $\frac{1}{2}$-level set)
(\mathcal{D}(x)f)(\rho, \theta) g(\theta, \rho) \, d\rho d\theta
Geodesic convolution

\[(f \ast g)(x) = \int_0^{\rho_0} \int_0^{2\pi} (\mathcal{D}(x)f)(\rho, \theta) g(\theta, \rho) \, d\rho \, d\theta\]
Geodesic convolution

\[(f * g)(x) = \int_0^{\rho_0} \int_0^{2\pi} (D(x)f)(\rho, \theta) \ g(\theta + \Delta \theta, \rho) \ d\rho \ d\theta\]

Angular coordinate origin is arbitrary = rotation ambiguity!

Masci et al. 2015
Geodesic convolution

\[(f \ast g)(x) = \int_0^{\rho_0} \int_0^{2\pi} (D(x)f)(\rho, \theta) \ g(\theta+\Delta\theta, \rho) \ d\rho d\theta\]

Angular coordinate origin is arbitrary = rotation ambiguity!

- Select reference direction, e.g. maximum curvature vector
Geodesic convolution

\[(f \ast g)(x) = \int_0^{\rho_0} \int_0^{2\pi} (\mathcal{D}(x)f)(\rho, \theta) \cdot g(\theta + \Delta \theta, \rho) \, d\rho d\theta\]

Angular coordinate origin is arbitrary = rotation ambiguity!

- Select reference direction, e.g. maximum curvature vector
- Take Fourier transform w.r.t. \(\theta\)

Masci et al. 2015
Geodesic convolution

\[(f \ast g)(x) = \int_0^{\rho_0} \int_0^{2\pi} \{ \mathcal{D}(x)f(\rho, \theta) \} g(\theta + \Delta \theta, \rho) \, d\rho \, d\theta \]

Angular coordinate origin is arbitrary = rotation ambiguity!

- Select reference direction, e.g. maximum curvature vector
- Take Fourier transform w.r.t. \(\theta\)

\[ (\mathcal{D}(x)f)(\rho, \theta) \xrightarrow{\mathcal{F}_\theta} (\mathcal{D}(x)f)(\rho, \omega) \]

Masci et al. 2015
Geodesic convolution

\[(f \ast g)(x) = \int_0^\rho \int_0^{2\pi} (\mathcal{D}(x)f)(\rho, \theta) g(\theta + \Delta\theta, \rho) \, d\rho \, d\theta\]

Angular coordinate origin is arbitrary = rotation ambiguity!

- Select reference direction, e.g. maximum curvature vector
- Take Fourier transform w.r.t. \(\theta\)

\[(\mathcal{D}(x)f)(\rho, \theta + \Delta\theta) \xrightarrow{\mathcal{F}_\theta} e^{-i\omega\theta}(\mathcal{D}(x)f)(\rho, \omega)\]

Masci et al. 2015
Geodesic convolution

\[(f \ast g)(x) = \int_0^{\rho_0} \int_0^{2\pi} (\mathcal{D}(x)f)(\rho, \theta) g(\theta + \Delta \theta, \rho) \, d\rho \, d\theta\]

Angular coordinate origin is arbitrary = rotation ambiguity!

- Select reference direction, e.g. maximum curvature vector
- Take Fourier transform magnitude w.r.t. \( \theta \)

\[ (\mathcal{D}(x)f)(\rho, \theta + \Delta \theta) \xleftarrow{\mathcal{F}_{\theta}} \left| e^{-i\omega \theta} (\mathcal{D}(x)f)(\rho, \omega) \right| \]

Masci et al. 2015
Geodesic convolution

\[(f \ast g)(x) = \int_0^{\rho_0} \int_0^{2\pi} (\mathcal{D}(x)f)(\rho, \theta) g(\theta + \Delta \theta, \rho) \; d\rho d\theta\]

Angular coordinate origin is arbitrary = rotation ambiguity!

- Select reference direction, e.g. maximum curvature vector
- Take Fourier transform magnitude w.r.t. \( \theta \)
- Keep all possible rotations

Masci et al. 2015
Geodesic convolution layer

Conv. layer \[ g_{\Delta \theta, l}(x) = \xi \left( \sum_{l'=1}^{p} (f_{l'} \ast w_{\Delta \theta, l, l'})(x) \right) \]

where \( l = 1, \ldots, q \), \( l' = 1, \ldots, p \), and \( \Delta \theta = \frac{2\pi}{N_\theta}, \ldots, 2\pi \)

Masci et al. 2015
Geodesic convolution layer

$g_{\Delta \theta, l}(x) = \xi \left( \sum_{l' = 1}^{p} (f_{l'} \ast w_{\Delta \theta, l, l'}) (x) \right)$

$g_{l}(x) = \max_{\Delta \theta} g_{\Delta \theta, l}(x)$

Masci et al. 2015
Geodesic CNN (GCNN)

Convolutional layer expressed in the spatial domain using geodesic polar patch operator + angular max pooling to solve rotational ambiguity

\[ g_l(x) = \max_{\Delta \theta} \xi \left( \sum_{l' = 1}^{p} \int_{0}^{\rho_0} \int_{0}^{2\pi} w_{l', l'}(\rho, \theta + \Delta \theta) (D(x)f_{l'}) \left( \rho, \theta \right) d\rho d\theta \right) \]

\[ (f_{l'} \ast w_{\Delta \theta, l, l'})(x) \]

\[ l = 1, \ldots, q \]
\[ l' = 1, \ldots, p \]

where \( w_{l, l'} = (w_{l, l', 1}, \ldots, w_{l, l', J}) \) are spatial filter coefficients
Geodesic CNN (GCNN)

Convolutional layer expressed in the \textit{spatial domain} using geodesic polar patch operator + angular max pooling to solve rotational ambiguity

\begin{equation}
    g_l(x) = \max_{\Delta \theta} \xi \left( \sum_{l' = 1}^{p} \int_{0}^{\rho_0} \int_{0}^{2\pi} w_{l', l'}(\rho, \theta + \Delta \theta)(D(x)f_{l'})(\rho, \theta) \, d\rho \, d\theta \right)
\end{equation}

\begin{align*}
    l &= 1, \ldots, q \\
    l' &= 1, \ldots, p
\end{align*}

where \( w_{l, l'} = (w_{l, l', 1}, \ldots, w_{l, l', J}) \) are spatial filter coefficients

\text{😊 Directional filters}
Geodesic CNN (GCNN)

Convolutional layer expressed in the spatial domain using geodesic polar patch operator + angular max pooling to solve rotational ambiguity

\[ g_l(x) = \max_{\Delta \theta} \xi \left( \sum_{l'}^{p} \int_{0}^{\rho_0} \int_{0}^{2\pi} w_{l',l'}(\rho, \theta + \Delta \theta)(D(x)f_{l'})(\rho, \theta) d\rho d\theta \right) \]

\[ (f_{l'} \ast w_{\Delta \theta, l', l'})(x) \]

where \( w_{l,l'} = (w_{l,l',1}, \ldots, w_{l,l',J}) \) are spatial filter coefficients

😊 Directional filters
😊 Spatially-localized filters

Boscaini et al. 2015
Convolutional layer expressed in the spatial domain using geodesic polar patch operator + angular max pooling to solve rotational ambiguity

\[
g_l(x) = \max_{\Delta \theta} \xi \left( \sum_{l'=1}^{p} \int_{0}^{\rho_0} \int_{0}^{2\pi} w_{l',l'}(\rho, \theta + \Delta \theta) (D(x) f_{l'})(\rho, \theta) d\rho d\theta \right)
\]

\[
= (f_{l'} * w_{l',l'})(x)
\]

\[
l = 1, \ldots, q
\]

\[
l' = 1, \ldots, p
\]

where \( w_{l,l'} = (w_{l,l',1}, \ldots, w_{l,l',J}) \) are spatial filter coefficients.

- Directional filters
- Spatially-localized filters
- \( \mathcal{O}(1) \) parameters per layer

Geodesic CNN (GCNN)
Convolutional layer expressed in the spatial domain using geodesic polar patch operator + angular max pooling to solve rotational ambiguity

\[ g_l(x) = \max_{\Delta \theta} \xi \left( \sum_{l'=1}^{p} \int_{\rho_0}^{\rho_0} \int_{0}^{2\pi} w_{l,l'}(\rho, \theta + \Delta \theta)(D(x)f_{l'})(\rho, \theta) \, d\rho \, d\theta \right) \]

\[ (f_{l'} \ast w_{\Delta \theta, l, l'})(x) \]

\[ l = 1, \ldots, q \]
\[ l' = 1, \ldots, p \]

where \( w_{l,l'} = (w_{l,l',1}, \ldots, w_{l,l',J}) \) are spatial filter coefficients

- ☺ Directional filters
- ☺ Spatially-localized filters
- ☺ \( O(1) \) parameters per layer
- ☺ All operations are local \( \Rightarrow O(n) \) computational complexity

Boscaini et al. 2015
Convolutional layer expressed in the *spatial domain* using geodesic polar patch operator + angular max pooling to solve rotational ambiguity

$$g_l(x) = \max_{\Delta \theta} \xi \left( \sum_{l'=1}^{p} \int_0^{\rho_0} \int_0^{2\pi} w_{l,l'}(\rho, \theta + \Delta \theta)(D(x)f_{l'})(\rho, \theta) \, d\rho \, d\theta \right)$$

$$ (f_{l'} * w_{\Delta \theta, l, l'})(x) $$

$l = 1, \ldots, q$

$l' = 1, \ldots, p$

where $w_{l,l'} = (w_{l,l',1}, \ldots, w_{l,l',J})$ are spatial filter coefficients

- ☺ Directional filters
- ☺ Spatially-localized filters
- ☺ $O(1)$ parameters per layer
- ☺ All operations are local $\Rightarrow O(n)$ computational complexity
- ☹ Angular max pooling potentially reduces discriminativity

Boscaini et al. 2015
Example: Learning local descriptors with GCNN

**Training set**
- positive \((x, x^+)\) and negative \((x, x^-)\) pairs of points

**Siamese net**
- two net instances with shared parameters \(\Theta\)

**Pointwise feature cost**

\[
\ell_S(\Theta) = \gamma \sum_{x, x^+} \|f_\Theta(x) - f_\Theta(x^+)\|_2^2 \\
+ (1 - \gamma) \sum_{x, x^-} \left[\mu - \|f_\Theta(x) - f_\Theta(x^-)\|_2^2\right]_+
\]

Boscaini et al. 2015
Example: 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 et al. 2008 (TOSCA); Anguelov et al. 2005 (SCAPE); Bogo et al. 2014 (FAUST)
Example: 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 et al. 2008 (TOSCA); Anguelov et al. 2005 (SCAPE);
Bogo et al. 2014 (FAUST)
Example: descriptor learning 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 et al. 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, Bronstein 2014 (OSD); Masci et al. 2015 (GCNN);
data: Bogo et al. 2014 (FAUST); benchmark: Kim et al. 2011
Homogeneous diffusion

\[ f_t(x) = -c \Delta f(x) \]

\( c = \text{thermal diffusivity constant} \) describing heat conduction properties of the material (diffusion speed is equal everywhere)
Anisotropic diffusion

\[ f_t(x) = -\text{div}(c \nabla f(x)) \]

\( c = \text{thermal diffusivity constant} \) describing heat conduction properties of the material (diffusion speed is equal everywhere)
\[ f_t(x) = -\text{div}(\mathbf{A}(x) \nabla f(x)) \]

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

\[ f_t(x) = - \text{div} \left( R_\theta \begin{pmatrix} \alpha \\ 1 \end{pmatrix} R_\theta^T \nabla f(x) \right) \]

Andreux et al. 2014; Boscaini et al. 2016
Anisotropic diffusion on manifolds

\[ f_t(x) = -\text{div}\left( R_\theta \begin{pmatrix} \alpha \\ 1 \end{pmatrix} R_\theta^T \nabla f(x) \right) \]

- Anisotropic Laplacian \( \Delta_{\alpha\theta} f(x) = \text{div} (D_{\alpha\theta}(x) \nabla f(x)) \)
- \( \theta \) = orientation w.r.t. max curvature direction
- \( \alpha \) = ‘elongation’

Andreux et al. 2014; Boscaini et al. 2016
Anisotropic heat kernels

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

Orientation \( \theta \)  Elongation \( \alpha \)
Anisotropic CNN (ACNN)

Use anisotropic heat kernels as weighting functions of the patch operator

$$(\mathcal{D}(x)f)_j = \int_{\chi} f(x') h_{\alpha_i, \theta_i, t_i}(x, x') dx' \quad j = 1, \ldots, J$$

for a discrete set of angles/scales/anisotropic constants

Boscaini et al. 2016
Anisotropic CNN (ACNN)

Use anisotropic heat kernels as weighting functions of the patch operator

\[(D(x)f)_j = \int f(x') h_{\alpha_i,\theta_i,t_i}(x,x')dx' \quad j = 1, \ldots, J\]

for a discrete set of angles/scales/anisotropic constants

Boscaini et al. 2016
Anisotropic CNN (ACNN)

Use anisotropic heat kernels as weighting functions of the patch operator

\[(D(x)f)_j = \int_{\chi} f(x') h_{\alpha_i, \theta_i, t_i}(x, x') dx' \quad j = 1, \ldots, J\]

for a discrete set of angles/scales/anisotropic constants

Convolutional layer expressed in the \textit{spatial domain}

\[g_l(x) = \xi \left( \sum_{l'=1}^{p} w_{l, l'}^\top D(x)f \right) \quad l = 1, \ldots, q\]

\[l' = 1, \ldots, p\]

where \(w_{l, l'} = (w_{l, l', 1}, \ldots, w_{l, l', J})\) are spatial filter coefficients

Boscaini et al. 2016
Anisotropic CNN (ACNN)

Use anisotropic heat kernels as weighting functions of the patch operator

\[
(D(x)f)_j = \int_{\mathcal{X}} f(x') h_{\alpha_i, \theta_i, t_i}(x, x') dx' \quad j = 1, \ldots, J
\]

for a discrete set of angles/scales/anisotropic constants

Convolutional layer expressed in the spatial domain

\[
g_l(x) = \xi \left( \sum_{l' = 1}^{p} w_{l, l'}^T D(x)f \right) \quad l = 1, \ldots, q
\]
\[
l' = 1, \ldots, p
\]

where \( w_{l, l'} = (w_{l, l', 1}, \ldots, w_{l, l', J}) \) are spatial filter coefficients

😊 Directional filters
😊 Spatially-localized filters
😊 \( O(1) \) parameters per layer
Anisotropic CNN (ACNN)

Use anisotropic heat kernels as weighting functions of the patch operator

\[
(D(x)f)_j = \int_{\chi} f(x') h_{\alpha_i,\theta_i,t_i}(x,x') dx' \quad j = 1, \ldots, J
\]

for a discrete set of angles/scales/anisotropic constants

Convolutional layer expressed in the spatial domain

\[
g_l(x) = \xi \left( \sum_{l'=1}^{p} w_{l,l'}^T D(x)f \right) \quad l = 1, \ldots, q
\]

where \( w_{l,l'} = (w_{l,l',1}, \ldots, w_{l,l',J}) \) are spatial filter coefficients

😊 Directional filters
😊 Spatially-localized filters
😊 \( O(1) \) parameters per layer
😊 Expensive computation of heat kernels for many orientations

Boscaini et al. 2016
Example: descriptor learning with ACNN

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

Boscaini et al. 2015; Boscaini et al. 2016
Learnable patch operator

- Local system of coordinates $\mathbf{u}(x, x')$ around point $x$

Monti et al. 2016
Learnable patch operator

- Local system of coordinates $\mathbf{u}(x, x')$ around point $x$
- Parametric weighting functions $\mathbf{w}_\Theta(\mathbf{u})$
Learnable patch operator

- Local system of coordinates $\mathbf{u}(x, x')$ around point $x$
- Parametric weighting functions $w_\Theta(u)$
- Parametric patch operator applying $J$ such weighting functions

\[
(D_{\Theta_1, \ldots, \Theta_J} f)_j = \int_{\mathcal{X}} f(x') w_{\Theta_j}(\mathbf{u}(x, x')) dx' \quad j = 1, \ldots, J
\]

Monti et al. 2016
Learnable patch operator

- Local system of coordinates $u(x, x')$ around point $x$
- Parametric weighting functions, e.g. $w_{\mu, \Sigma}(u) = e^{-\frac{1}{2}(u-\mu)^\top \Sigma^{-1}(u-\mu)}$
- Parametric patch operator applying $J$ such weighting functions

$$
(D_{\mu_1, \Sigma_1, \ldots, \mu_J, \Sigma_J}(x)f)_j = \int_{\mathcal{X}} f(x')w_{\mu_j, \Sigma_j}(u(x, x'))dx' \quad j = 1, \ldots, J
$$

Monti et al. 2016
Learnable patches on manifolds

- Geodesic polar coordinates
  \[ \mathbf{u}(x, y) = (\rho(x, y), \theta(x, y)) \]

\[ w_{\mu, \nu}(\mathbf{u}) = \exp \left( \frac{1}{2} (\mathbf{u} - \mu)^T \Sigma (\mathbf{u} - \mu) \right) \]

with learnable covariance \( \Sigma \) and mean \( \mu \).

Spatial convolution
\[ (f \ast g)(x) = \int_{X} g_j w_{\mu_j, \nu_j}(\mathbf{u}(x, x_0)) \, \text{d}x_0 \]

Gaussian mixture
\[ f(x_0) \, \text{d}x_0 \]

Monti et al. 2016
Learnable patches on manifolds

- Geodesic polar coordinates
  \[ \mathbf{u}(x, y) = (\rho(x, y), \theta(x, y)) \]

- Gaussian weighting functions
  \[ w_{\mu, \Sigma}(\mathbf{u}) = \exp \left( -\frac{1}{2} (\mathbf{u} - \mu)^T \Sigma^{-1} (\mathbf{u} - \mu) \right) \]
  with learnable covariance \( \Sigma \) and mean \( \mu \)
Learnable patches on manifolds

- Geodesic polar coordinates
  \[ \mathbf{u}(x, y) = (\rho(x, y), \theta(x, y)) \]

- Gaussian weighting functions
  \[ w_{\mu, \Sigma}(u) = \exp\left(-\frac{1}{2}(u - \mu)^\top \Sigma^{-1}(u - \mu)\right) \]
  with learnable covariance \( \Sigma \) and mean \( \mu \)

Spatial convolution

\[ (f \ast g)(x) = \sum_{j=1}^{J} g_j \int_X w_{\mu_j, \Sigma_j}(\mathbf{u}(x, x')) f(x') \, dx' \]

Monti et al. 2016
Learnable patches on manifolds

- Geodesic polar coordinates
  \( u(x, y) = (\rho(x, y), \theta(x, y)) \)

- Gaussian weighting functions
  \[
  w_{\mu, \Sigma}(u) = \exp\left( -\frac{1}{2}(u - \mu)^\top \Sigma^{-1}(u - \mu) \right)
  \]
  with learnable covariance \( \Sigma \) and mean \( \mu \)

Spatial convolution

\[
(f \star g)(x) = \int_{\mathcal{X}} \sum_{j=1}^{J} g_j \, w_{\mu_j, \Sigma_j}(u(x, x')) \, f(x') \, dx'
\]
Learnable patches on manifolds

- Geodesic polar coordinates
  \[ u(x, y) = (\rho(x, y), \theta(x, y)) \]

- Gaussian weighting functions
  \[ w_{\mu, \Sigma}(u) = \exp\left(-\frac{1}{2}(u - \mu)^{\top}\Sigma^{-1}(u - \mu)\right) \]

  with learnable covariance \( \Sigma \) and mean \( \mu \)

Spatial convolution

\[
(f \ast g)(x) = \int_{\mathcal{X}} \sum_{j=1}^{J} g_j w_{\mu_j, \Sigma_j}(u(x, x')) f(x') \, dx'
\]

\[ \underbrace{\sum_{j=1}^{J} g_j w_{\mu_j, \Sigma_j}(u(x, x')) f(x')}_{\text{Gaussian mixture } g(u(x, x'))} \]
Convolutional layer expressed in the spatial domain using a learnable patch operator

\[ g_l(x) = \xi \left( \sum_{l' = 1}^{p} w_{l,l'}^\top D_\Theta(x)f \right) \]  
\[ l = 1, \ldots, q \]  
\[ l' = 1, \ldots, p \]

where \( w_{l,l'} = (w_{l,l',1}, \ldots, w_{l,l',J}) \) are spatial filter coefficients and \( \Theta = (\mu_1, \Sigma_1, \ldots, \mu_J, \Sigma_J) \) are the patch parameters
Mixture Model Networks (MoNet)

Convolutional layer expressed in the spatial domain using a learnable patch operator

\[ g_l(x) = \xi \left( \sum_{l'=1}^{p} w_{l,l'}^\top D_\Theta(x) f^{l'} \right) \quad l = 1, \ldots, q \]
\[ l' = 1, \ldots, p \]

where \( w_{l,l'} = (w_{l,l',1}, \ldots, w_{l,l',J}) \) are spatial filter coefficients and \( \Theta = (\mu_1, \Sigma_1, \ldots, \mu_J, \Sigma_J) \) are the patch parameters

- 😊 Directional filters
- 😊 Spatially-localized filters
- 😊 Learnable patch operator
- 😊 \( \mathcal{O}(1) \) parameters per layer

Monti et al. 2016
Patch operator weight functions

Masci et al. 2015 (GCNN); Boscaini et al. 2016 (ACNN); Monti et al. 2016 (MoNet)
MoNet as generalization of previous methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Coordinates $u(x, x')$</th>
<th>Weight function $w_{\Theta}(u)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN$^1$</td>
<td>$u(x') - u(x)$</td>
<td>$\delta(u - v)$ fixed parameters $\Theta = v$</td>
</tr>
<tr>
<td>GCNN$^2$</td>
<td>$\rho(x, x'), \theta(x, x')$</td>
<td>$\exp\left(-\frac{1}{2}(u - v)^{\top}\begin{pmatrix} \sigma_p^2 &amp; 0 \ 0 &amp; \sigma_\theta^2 \end{pmatrix}^{-1}(u - v)\right)$ fixed parameters $\Theta = (v, \sigma_p, \sigma_\theta)$</td>
</tr>
<tr>
<td>ACNN$^3$</td>
<td>$\rho(x, x'), \theta(x, x')$</td>
<td>$\exp\left(-tu^{\top}R_\varphi\begin{pmatrix} \alpha_1 \ 1 \end{pmatrix}R_\varphi^{\top}u\right)$ fixed parameters $\Theta = (\alpha, \varphi, t)$</td>
</tr>
<tr>
<td>MoNet$^4$</td>
<td>$\rho(x, x'), \theta(x, x')$</td>
<td>$\exp\left(-\frac{1}{2}(u - \mu)^{\top}\Sigma^{-1}(u - \mu)\right)$ learnable parameters $\Theta = (\mu, \Sigma)$</td>
</tr>
</tbody>
</table>

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

Learning deformation-invariant correspondence

- Groundtruth correspondence  $\pi^*: \mathcal{X} \rightarrow \mathcal{Y}$ from query shape $\mathcal{X}$ to some reference shape $\mathcal{Y}$ (discretized with $n$ vertices)

- Correspondence = label each query vertex $x$ as reference vertex $y$

Rodolà et al. 2014; Masci et al. 2015
Learning deformation-invariant correspondence

- Groundtruth correspondence $\pi^*: \mathcal{X} \rightarrow \mathcal{Y}$ from query shape $\mathcal{X}$ to some reference shape $\mathcal{Y}$ (discretized with $n$ vertices)
- Correspondence = label each query vertex $x$ as reference vertex $y$
- Net output at $x$ after softmax layer $f_\Theta(x) = (f_{\Theta,1}(x), \ldots, f_{\Theta,n}(x))$
  $= \text{probability distribution on } \mathcal{Y}$

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 \( \mathcal{Y} \) (discretized with \( n \) vertices)
- Correspondence = label each query vertex \( x \) as reference vertex \( y \)
- Net output at \( x \) after softmax layer
  \[
  f_{\Theta}(x) = (f_{\Theta,1}(x), \ldots, f_{\Theta,n}(x))
  \]
  = probability distribution on \( \mathcal{Y} \)

Minimize on training set the cross entropy between groundtruth correspondence and output probability distribution w.r.t. net parameters \( \Theta \)

\[
\min_{\Theta} \sum_x H(\delta_{\pi^*(x)}, f_{\Theta}(x))
\]

Rodolà et al. 2014; Masci et al. 2015
**Correspondence evaluation: Princeton benchmark**

**Query** $\mathcal{X}$  

**Reference** $\mathcal{Y}$

**Pointwise correspondence error** = geodesic distance from the groundtruth

$$\epsilon(x) = d_\mathcal{Y}(\pi^*(x), \pi(x))$$

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

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

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
Partial correspondence with ACNN

Correspondence

Correspondence error

Boscaini et al. 2016
Partial correspondence with ACNN
Classification cost considers equally correspondences that deviate from the groundtruth (no matter how far).

Kim et al. 2011
Soft correspondence error

\[ \bar{\epsilon}(x) = \int_{\mathcal{Y}} p(x, y) d_{\mathcal{Y}}(\pi^*(x), y) dy \]

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

Kovnatsky et al. 2015; Litany et al. 2017
Pointwise vs Structured learning

Nearby points $x, x'$ on query shape are not guaranteed to map to nearby points $y, y'$ on reference shape at test time

Litany et al. 2017
Functional maps: spectral domain

\[ f(x) \approx \hat{f}_1 + \hat{f}_2 + \ldots + \hat{f}_K \]

\[ g(x) \approx \hat{g}_1 + \hat{g}_2 + \ldots + \hat{g}_K \]


Functional correspondence \( T = \text{linear map} \ C \) between Fourier coefficients

\[ \hat{g}^\top = \hat{f}^\top C \]
**Functional maps: spectral domain**

\[ f_1(x) \approx \frac{\hat{f}_{11}}{T} + \frac{\hat{f}_{12}}{C} + \ldots + \frac{\hat{f}_{1K}}{C} \]

\[ f_q(x) \approx \frac{\hat{f}_{q1}}{T} + \frac{\hat{f}_{q2}}{C} + \ldots + \frac{\hat{f}_{qK}}{C} \]

\[ g_1(x) \approx \frac{\hat{g}_{11}}{T} + \frac{\hat{g}_{12}}{C} + \ldots + \frac{\hat{g}_{1K}}{C} \]

\[ g_q(x) \approx \frac{\hat{g}_{q1}}{T} + \frac{\hat{g}_{q2}}{C} + \ldots + \frac{\hat{g}_{qK}}{C} \]

Recover correspondence from \( q \geq k \) dimensional pointwise features

\[
\begin{pmatrix}
\hat{g}_{11} & \hat{g}_{12} & \ldots & \hat{g}_{1K} \\
\vdots & \vdots & \ddots & \vdots \\
\hat{g}_{q1} & \hat{g}_{q2} & \ldots & \hat{g}_{qK}
\end{pmatrix}
= 
\begin{pmatrix}
\hat{f}_{11} & \hat{f}_{12} & \ldots & \hat{f}_{1K} \\
\vdots & \vdots & \ddots & \vdots \\
\hat{f}_{q1} & \hat{f}_{q2} & \ldots & \hat{f}_{qK}
\end{pmatrix}
\begin{pmatrix}
\hat{c}_{11} \\
\hat{c}_{12} \\
\vdots \\
\hat{c}_{q1} \\
\hat{c}_{q2} \\
\vdots \\
\hat{c}_{1K} \\
\hat{c}_{2K}
\end{pmatrix}
\]

Ovsjanikov et al. 2012
Functional maps: spectral domain

\[
\begin{align*}
\hat{f}_1(x) & \approx \hat{f}_{11} + \hat{f}_{12} + \cdots + \hat{f}_{1K} \\
\vdots & \\
f_q(x) & \approx \hat{f}_{q1} + \hat{f}_{q2} + \cdots + \hat{f}_{qK} \\
\downarrow T & \\
\hat{g}_1(x) & \approx \hat{g}_{11} + \hat{g}_{12} + \cdots + \hat{g}_{1K} \\
\vdots & \\
g_q(x) & \approx \hat{g}_{q1} + \hat{g}_{q2} + \cdots + \hat{g}_{qK} \\
\downarrow C & \\
\hat{G} & = \hat{F}C
\end{align*}
\]

Recover correspondence from \( q \geq k \) dimensional pointwise features

Ovsjanikov et al. 2012
Functional maps: spectral domain

\[ f_1(x) \approx \hat{f}_{q1} + \hat{f}_{q2} + \ldots + \hat{f}_{qK} \]

\[ g_1(x) \approx \hat{g}_{q1} + \hat{g}_{q2} + \ldots + \hat{g}_{qK} \]

Recover correspondence from \( q \geq k \) dimensional pointwise features

\[ C^* = \arg\min_C \| \hat{F}C - \hat{G} \|^2_F \]
Functional maps: spatial domain

Rank-$K$ approximation of spatial correspondence

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

Ovsjanikov et al. 2012
Functional maps: spatial domain

Probability $p(x, y)$ of point $x$ mapping to $y$

\[ P \approx |\Psi C \Phi^T|_{\| \cdot \|} \]

Ovsjanikov et al. 2012
Siamese metric learning

\[ f(x) \rightarrow \text{Intrinsic deep net} \]

\[ g(y) \rightarrow \text{Intrinsic deep net} \]

\[ \ell_S \]

\[ x \]

\[ y \]

\[ \Theta \]

Siamese net

two net instances with shared parameters \( \Theta \)

Poitwise feature cost

\[
\ell_S(\Theta) = \gamma \sum_{x,x^+} \| f_\Theta(x) - f_\Theta(x^+) \|^2_2 \\
+ (1 - \gamma) \sum_{x,x^-} [\mu - \| f_\Theta(x) - f_\Theta(x^-) \|^2_2]_+ 
\]

Boscaini et al. 2015; Cosmo et al. 2016
Structured correspondence with FMNet

Siamese net

Functional map layer

Soft correspondence layer

Soft error cost

Litany et al. 2017
Structured correspondence with FMNet

Siamese net
- two net instances with shared parameters $\Theta$

Functional map layer
- $C^*_\Theta = \hat{F}^\dagger_\Theta \hat{G}_\Theta$

Soft correspondence layer
- $P_\Theta = |\Psi C_\Theta \Phi^T|_{|| \cdot ||}$

Soft error cost
- $\ell_F(\Theta) = ||P_\Theta \circ D_\gamma||$

Litany et al. 2017
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
Different formulations of non-Euclidean CNNs

Spectral domain

Spatial domain

Embedding domain
Global parametrization

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

Sinha et al. 2016; Maron 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
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
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
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
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

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

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 oriented 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.
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

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