Geometric Deep Learning

...

Sebastian Crites
Statistical Learning Group 2019

Gameplan

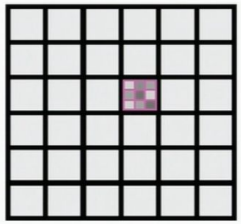
- Review of Euclidean Convolutional Neural Networks
- Laplacian operators in the Euclidean and non-Euclidean domains
- Spectral Analysis
- Constructing non-Euclidean Convolutional Neural Networks

geometricdeeplearning.com

Different formulations of non-Euclidean CNNs

Spectral domain

Spatial domain



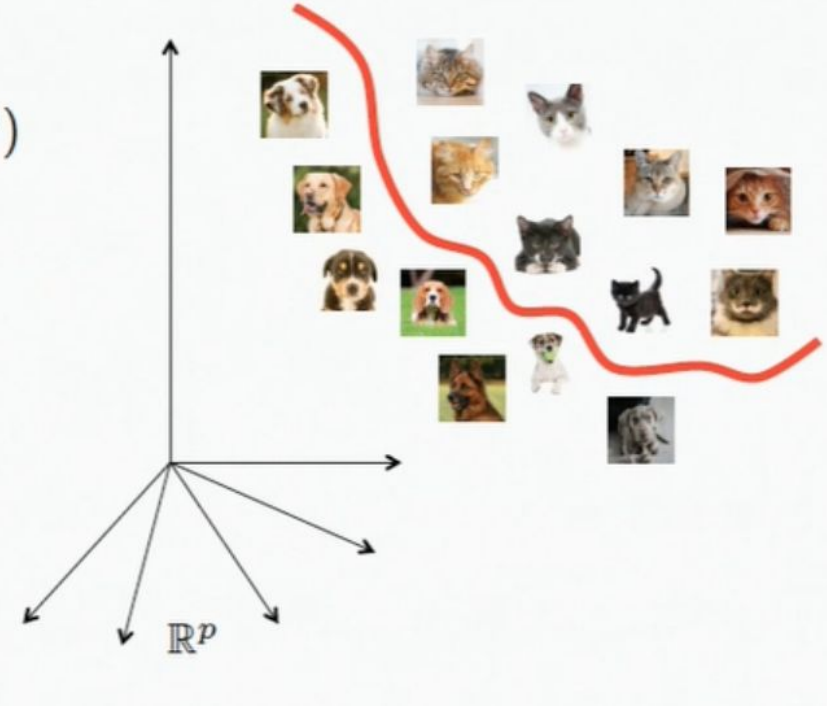
Embedding domain

Euclidean Supervised Learning

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

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

- Parametric model y_{Θ} of y



Euclidean Supervised Learning

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

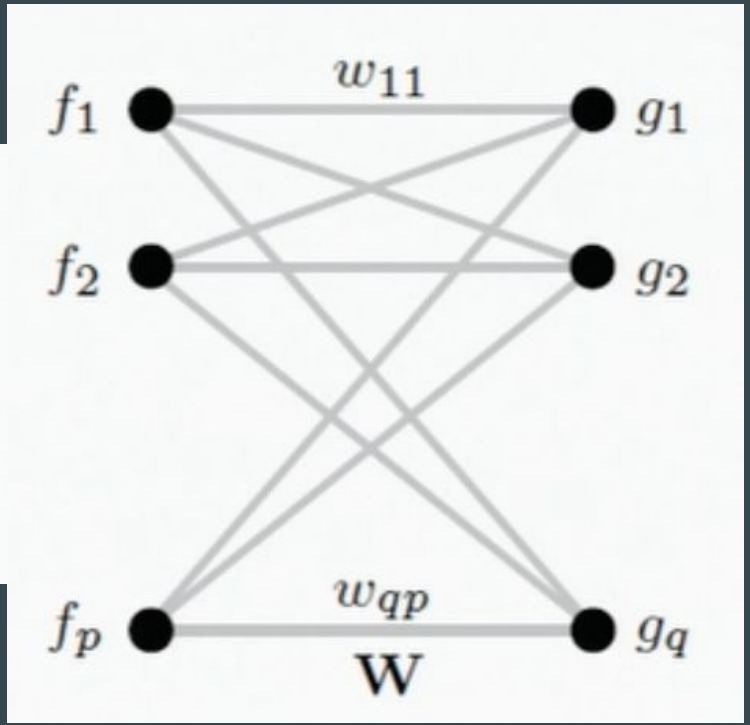
$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^{T} \ell(y_\Theta(\mathbf{f}_i), y_i)$$

Single Dense Layer

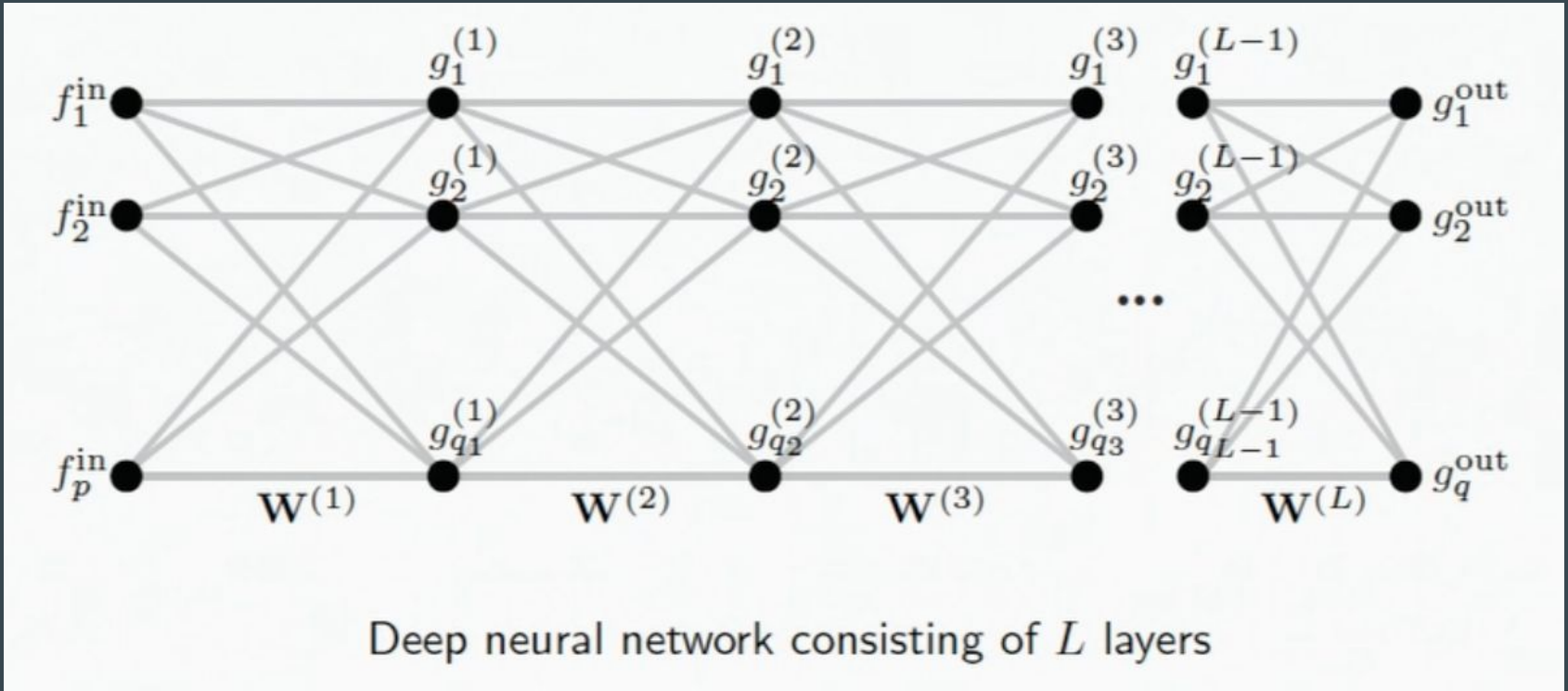
$$g_l = \xi\left(\sum_{l'=1}^{p} f_{l'} w_{l,l'}\right) \qquad \begin{matrix} l = 1, \ldots, q \\ l' = 1, \ldots, p \end{matrix}$$

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

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



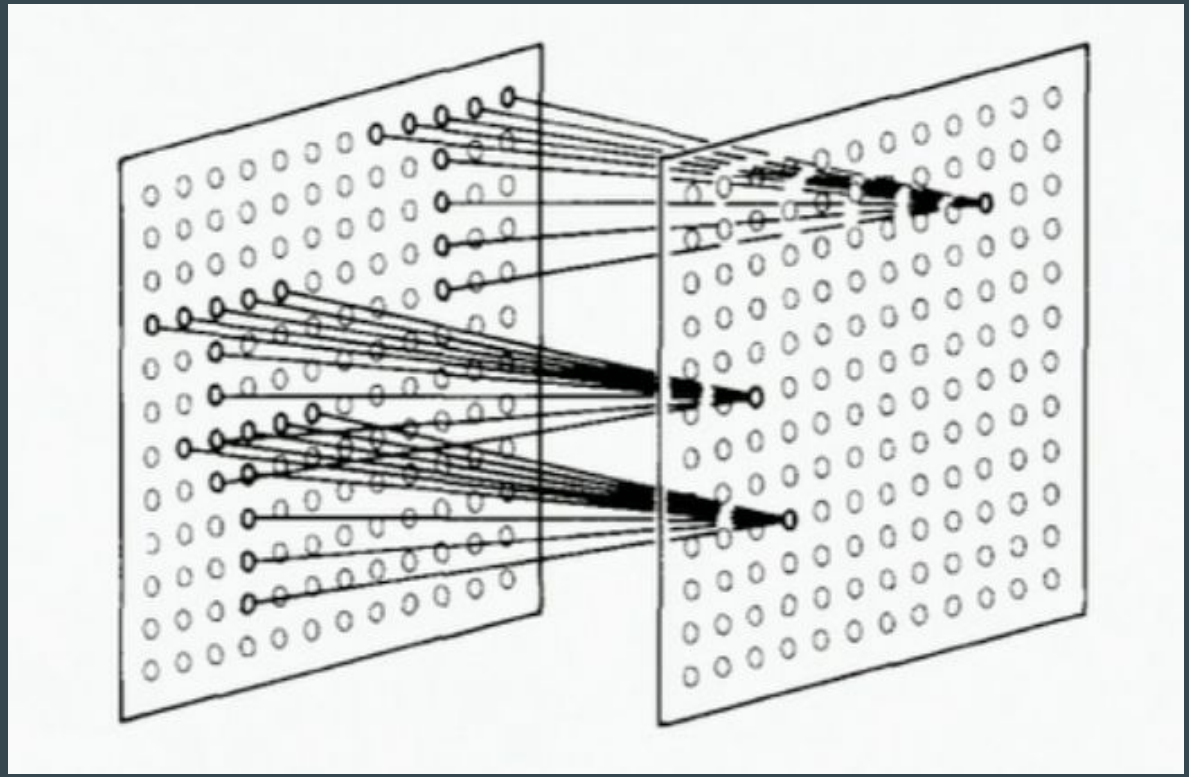
Composing Dense Layers



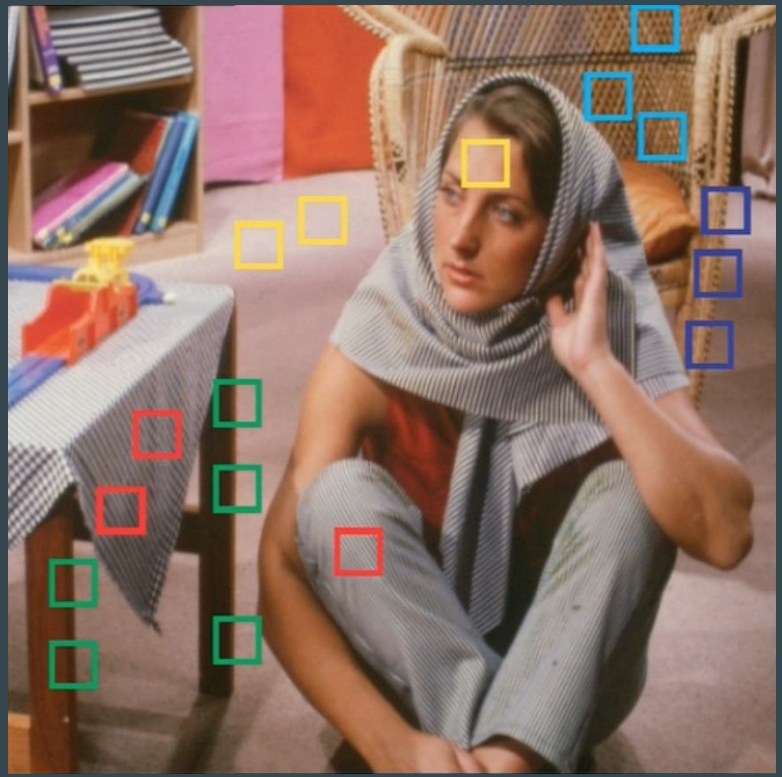
Deep neural network consisting of L layers

Properties of Image Data

Locality



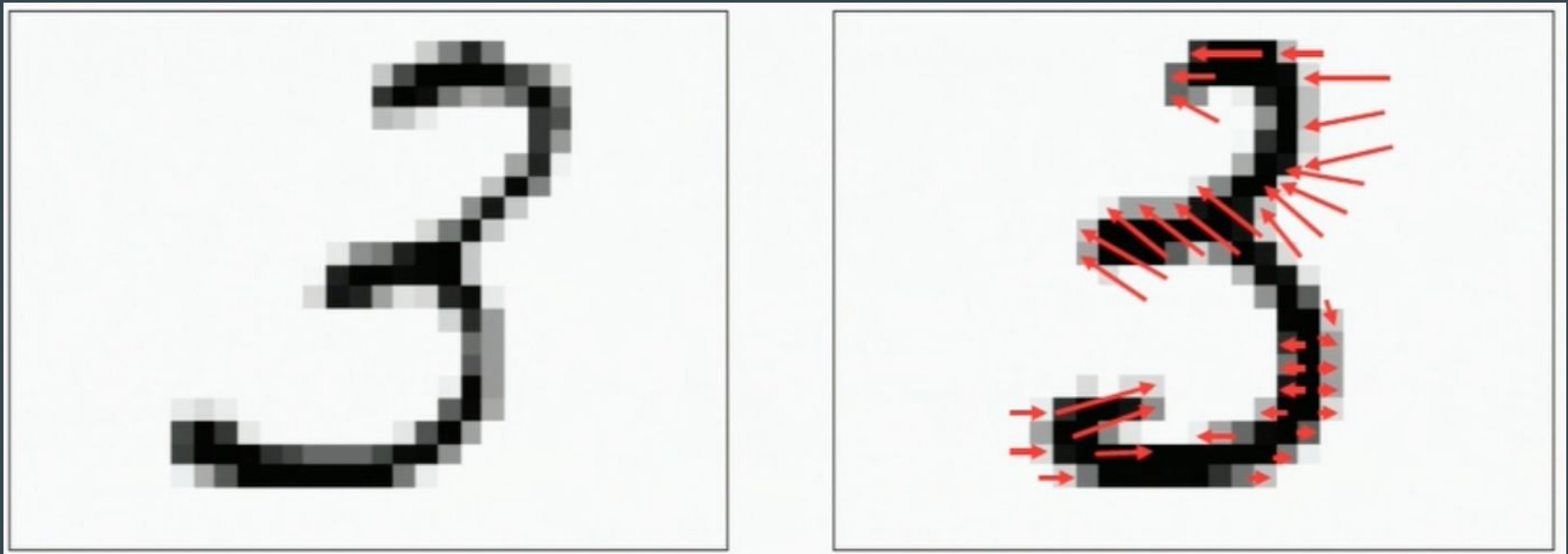
Self-Similarity



Translation Invariance

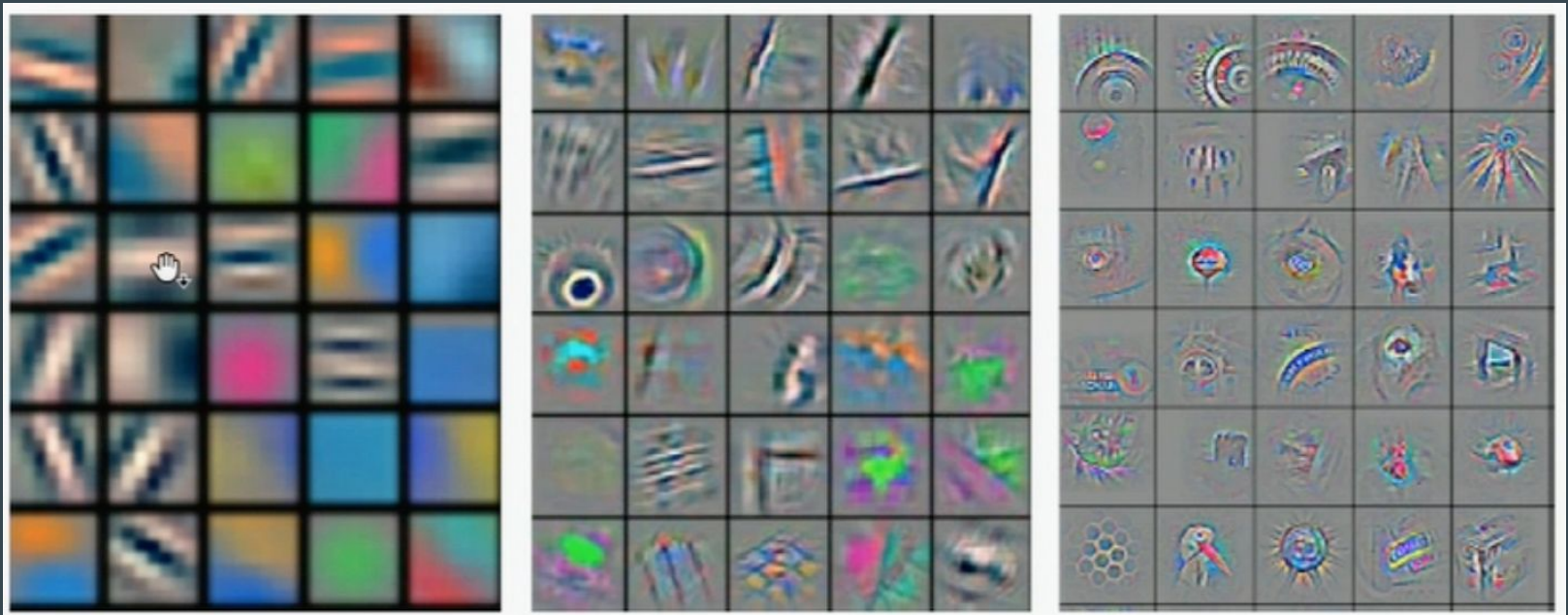
$$y(\mathcal{T}_v f) = y(f) \quad \forall f, v$$

Deformation Tolerance



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

Hierarchical & Composososable



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

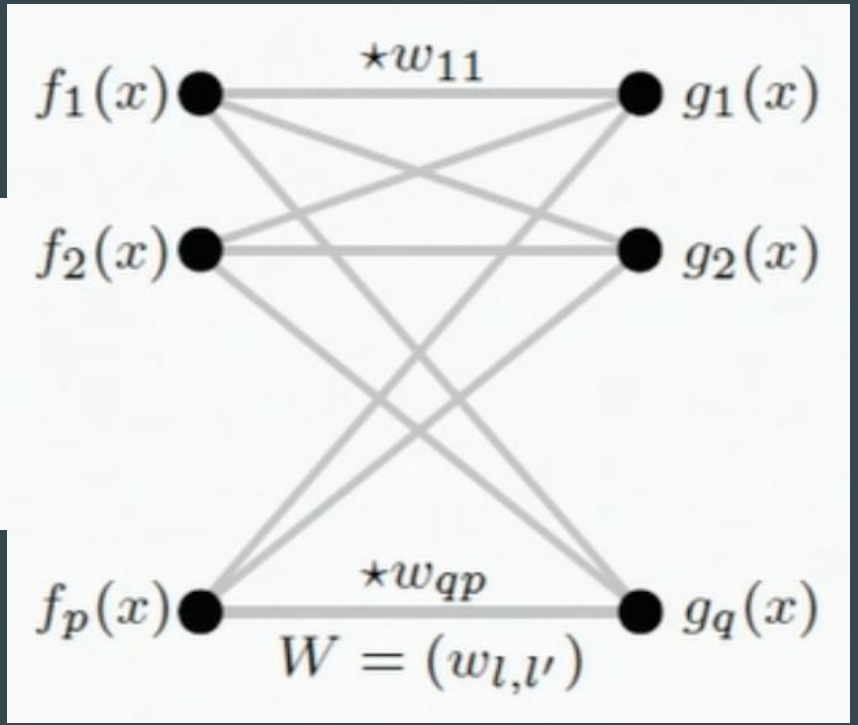
Convolutional Neural Networks

Convolutional Neural Network (CNN) Layer

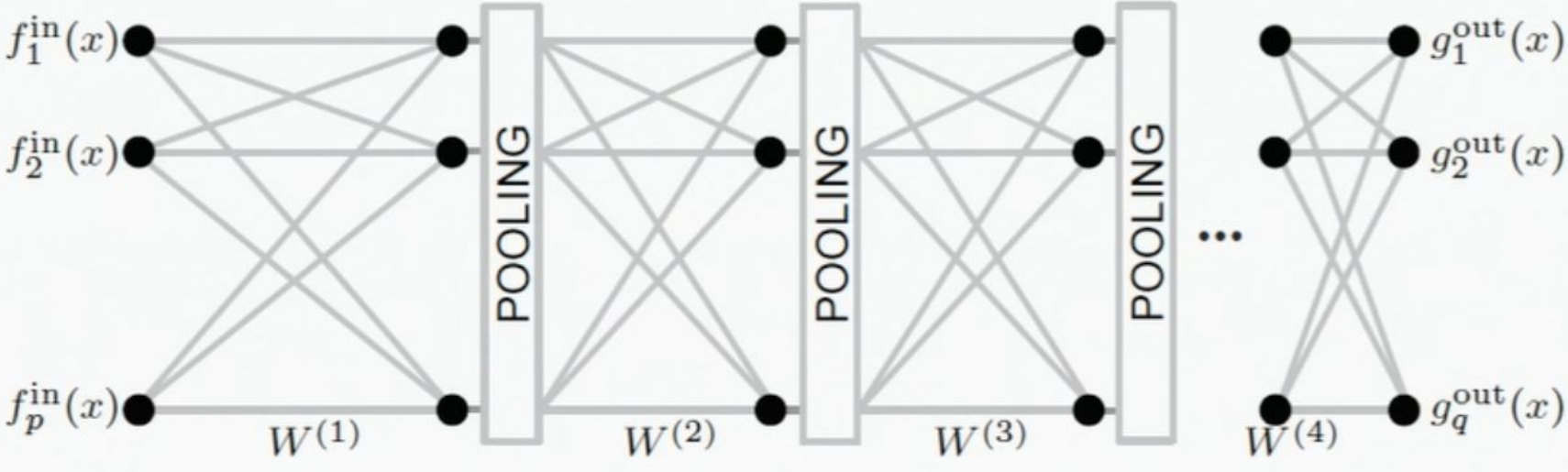
$$g_l(x) = \xi\left(\sum_{l'=1}^{p}(f_{l'} \star w_{l,l'})(x)\right) \qquad \begin{matrix} l = 1, \ldots, q \\ l' = 1, \ldots, p \end{matrix}$$

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

filters W

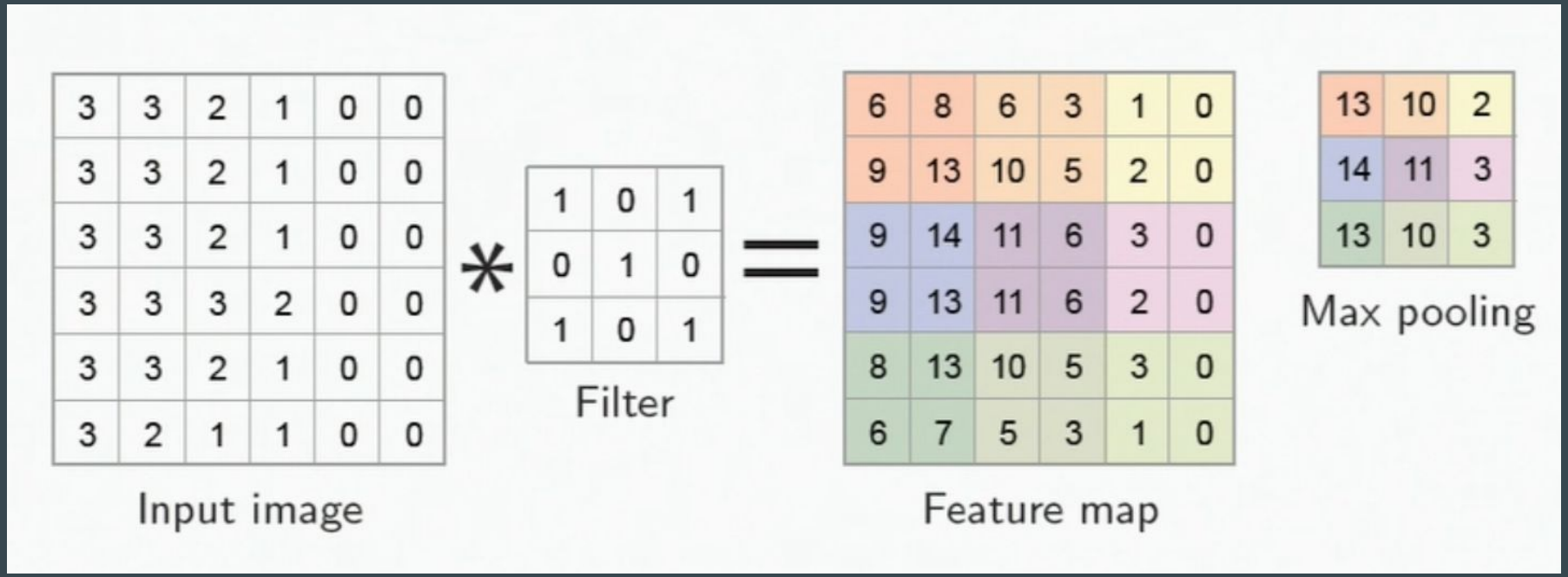


Composing Convolutional Layers



CNN consisting of L convolutional layers interleaved with pooling

Image Convolution Example



Key Properties of CNNs

☺ Convolutional filters **(Translation invariance)**

Key Properties of CNNs

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

Key Properties of CNNs

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

Key Properties of CNNs

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

Key Properties of CNNs

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

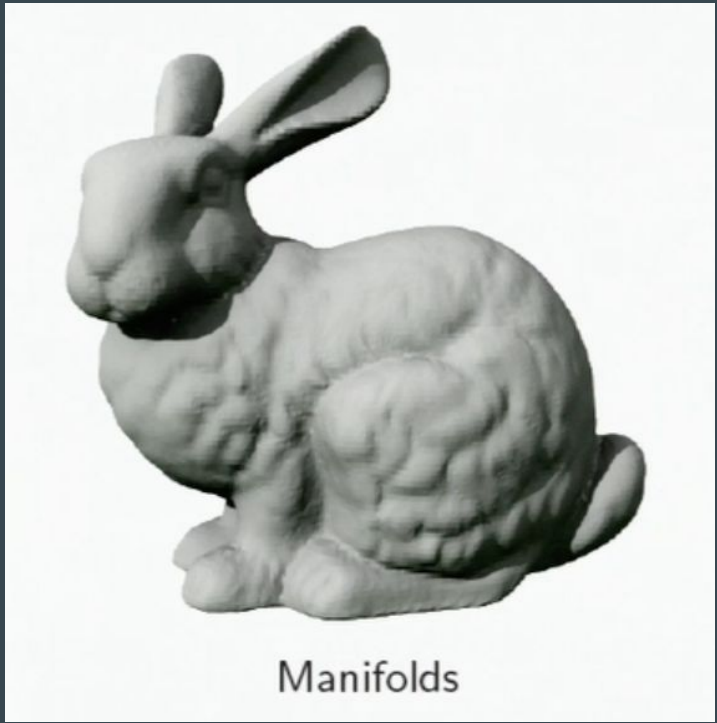
Key Properties of CNNs

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

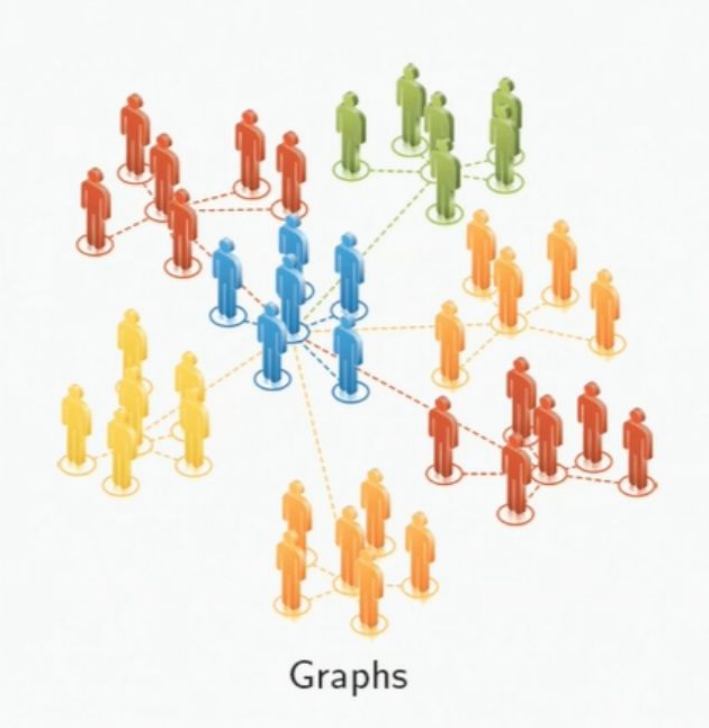
Key Properties of CNNs

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

Non-Euclidean Domains



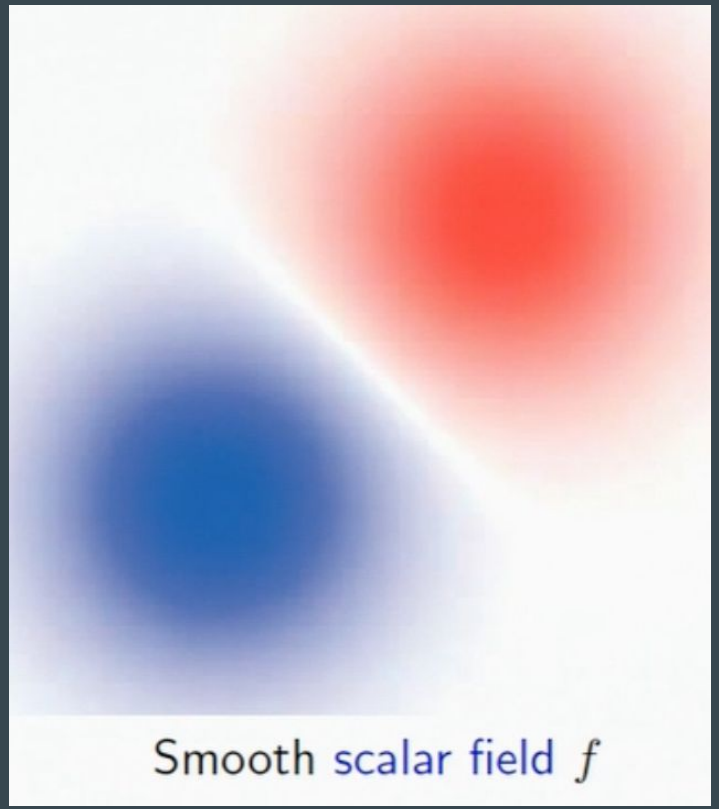
Manifolds



Graphs

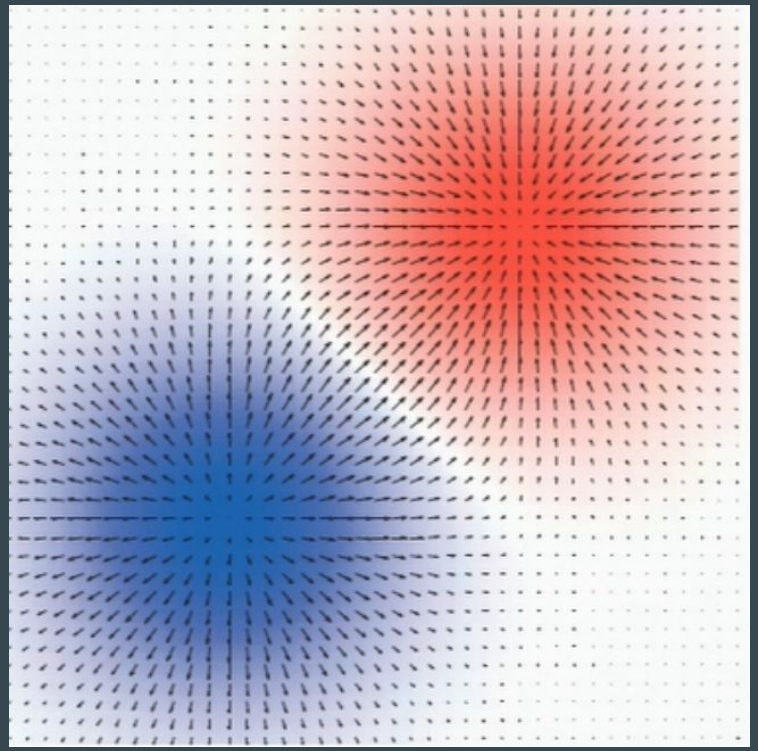
Laplacian Operators

Euclidean Laplacian Operator



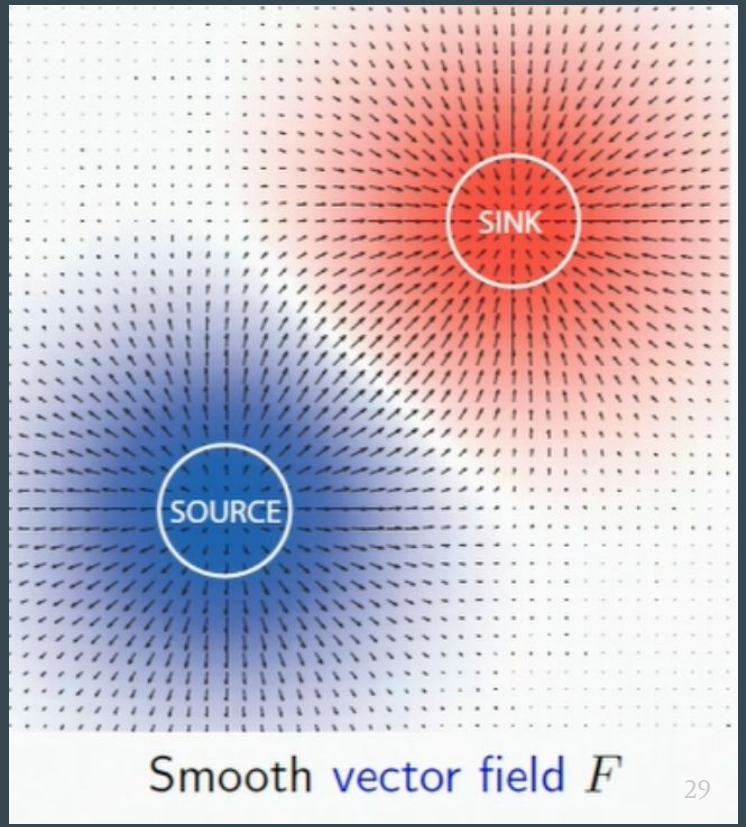
Laplacian Operator

Gradient $\nabla f(x) =$ 'direction of the steepest increase of f at x'



Laplacian Operator

Divergence $\mathrm{div}(F(x)) =$ 'density of an outward flux of F from an infinitesimal volume around x'

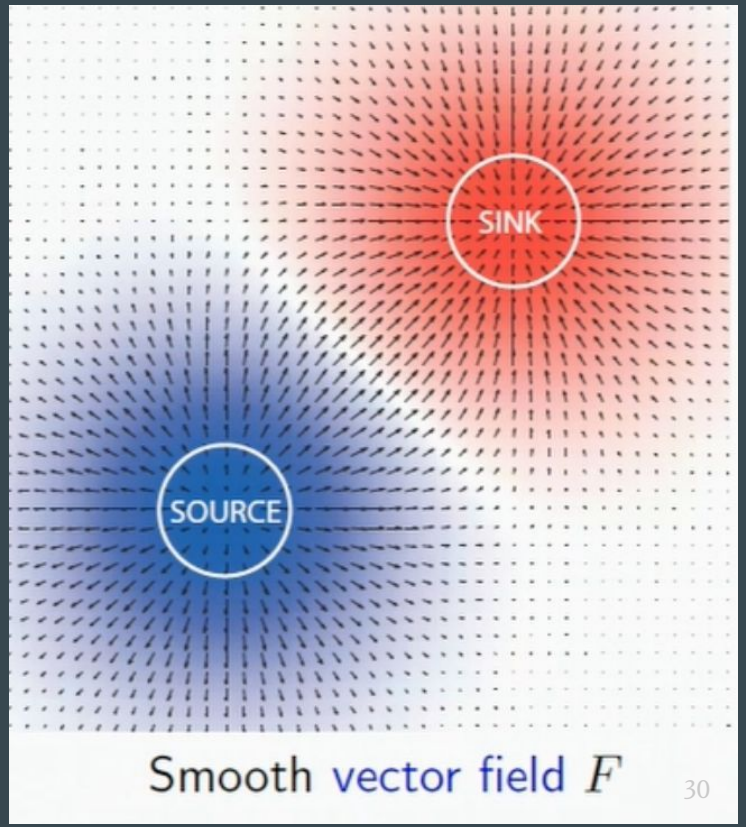


Laplacian Operator

Divergence theorem:

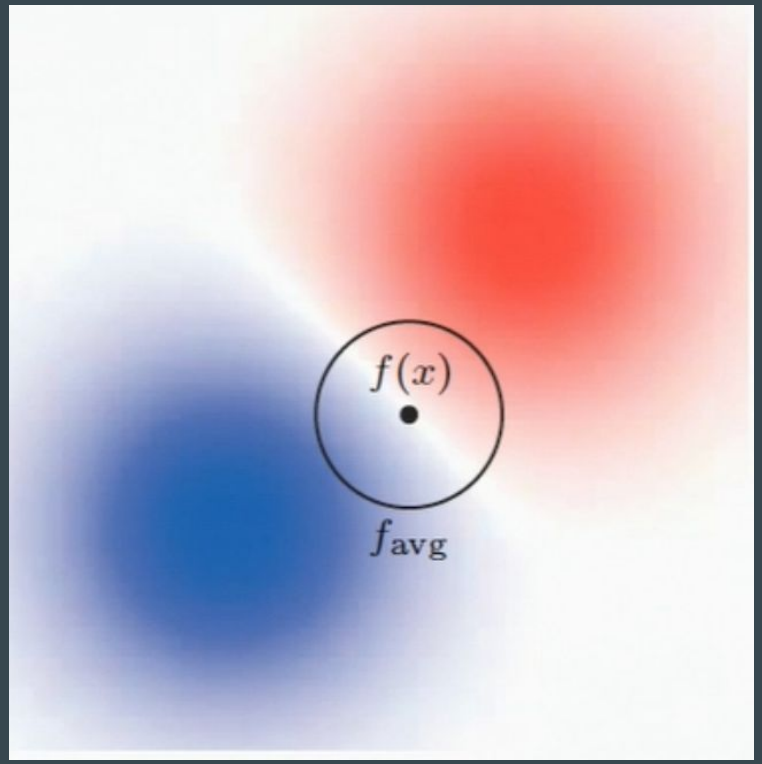
$$\int_V \mathrm{div}(\mathrm{F}) dV = \int_{\partial V} \langle F, \hat{n} \rangle dS$$

'$\sum$ sources + sinks = net flow'



Laplacian Operator

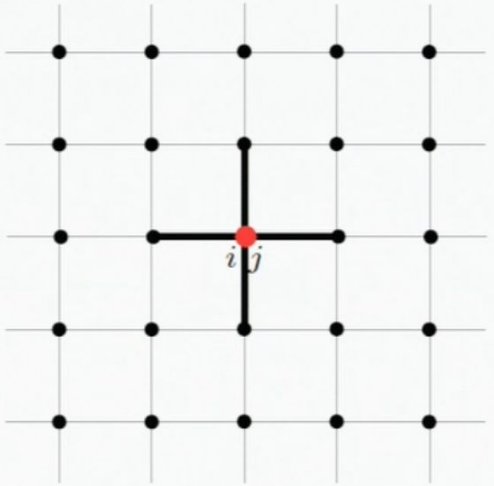
Laplacian $\Delta f(x) = -\mathrm{div}(\nabla f(x))$



Laplacian in 1D and 2D

One-dimensional

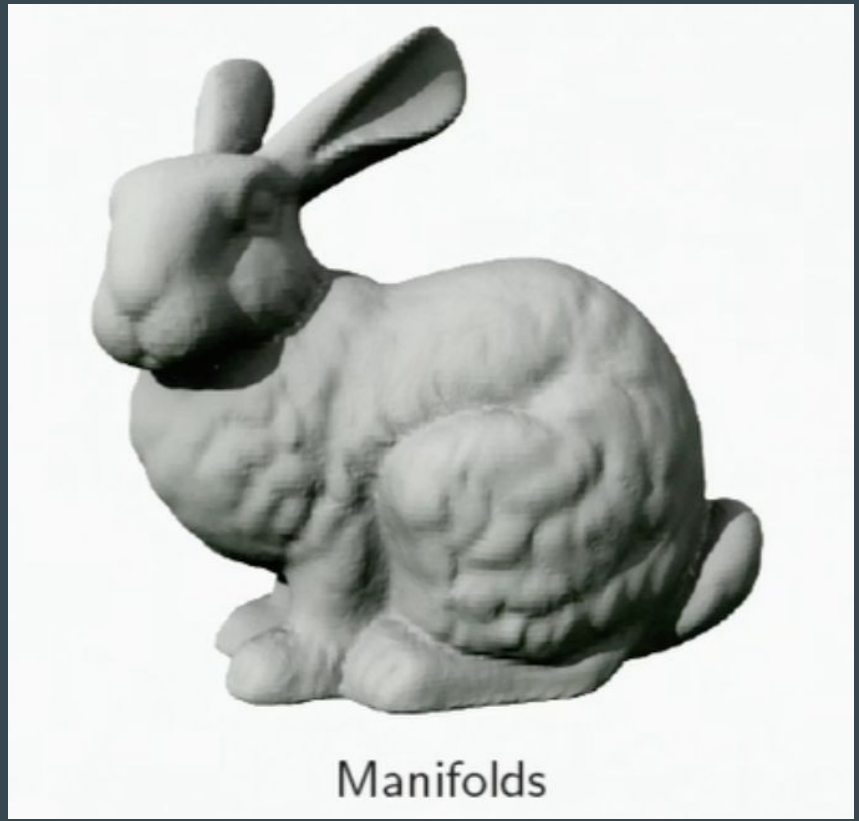
$$(\Delta f)_i \approx 2f_i - f_{i-1} - f_{i+1}$$



Two-dimensional

$$\begin{aligned}(\Delta f)_{ij} &\approx 4f_{ij} - f_{i-1,j} - f_{i+1,j} \\ &- f_{i,j-1} - f_{i,j+1}\end{aligned}$$

Non-Euclidean Domains



Riemannian Manifolds

Manifold $\mathcal{X}$ = topological space

No global Euclidean structure

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

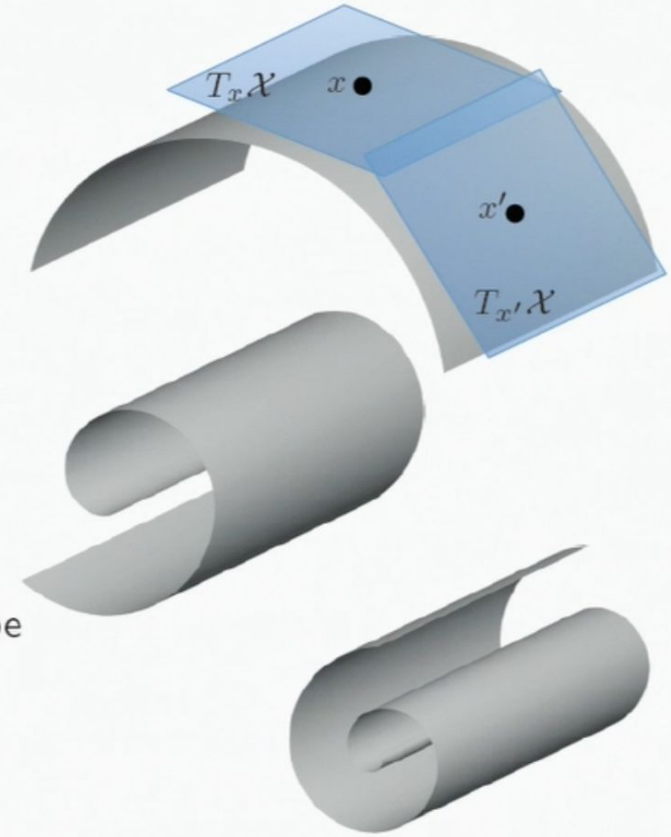
Riemannian metric

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

depending smoothly on x

Isometry = metric-preserving shape deformation

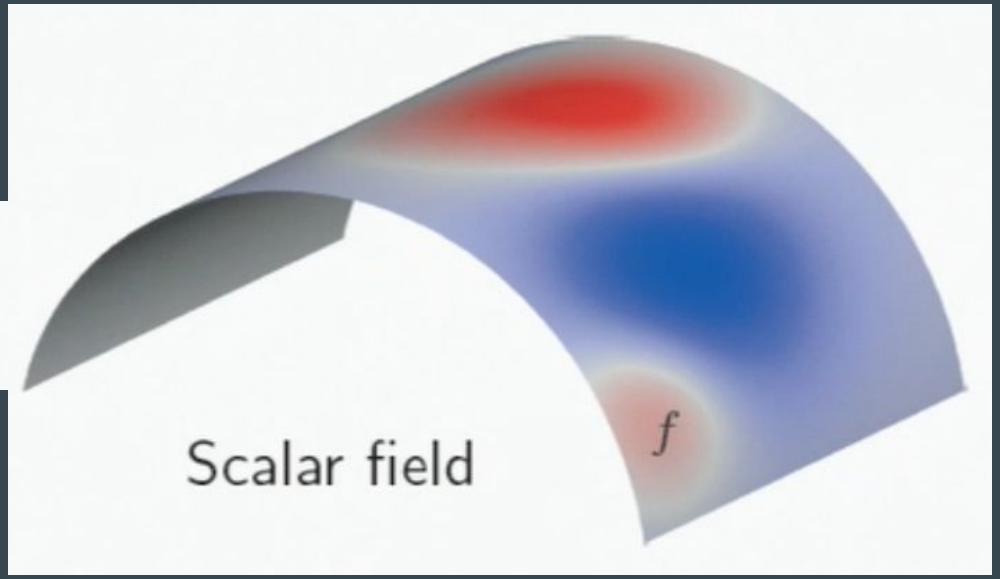
Intrinsic = expressed solely in terms of the Riemannian metric



Manifold Definitions

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

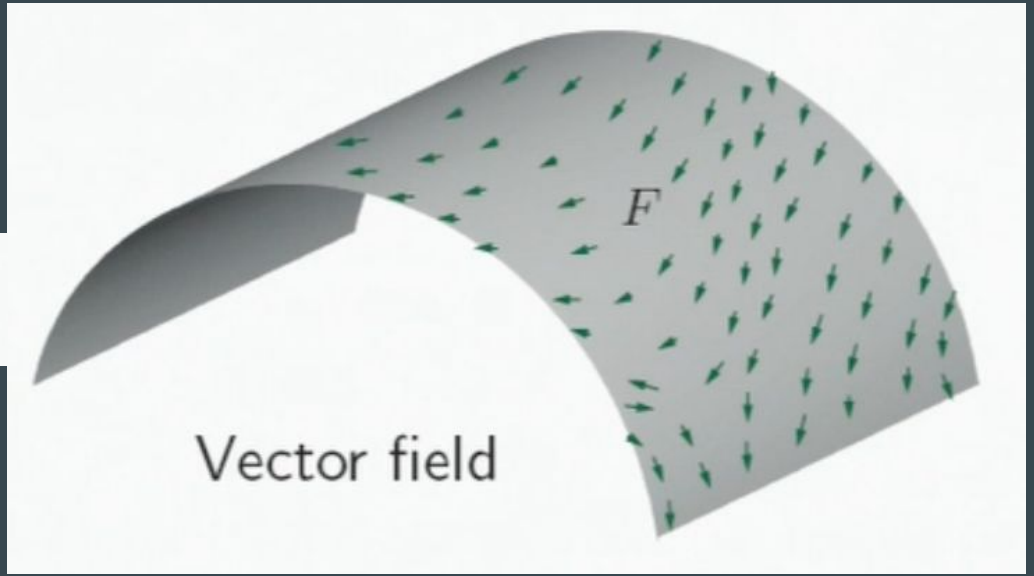
$$\langle f, g \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x) g(x) dx$$



Manifold Definitions

Vector field $F : \mathcal{X} \to T\mathcal{X}$

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

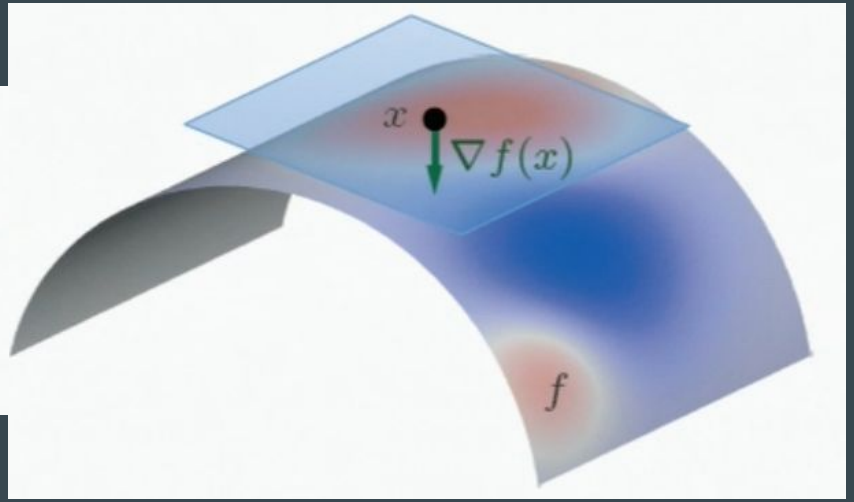


Laplacian on Manifolds

Intrinsic gradient operator

$$\nabla f : L^2(\mathcal{X}) \to L^2(T\mathcal{X})$$

"direction of steepest change of f"

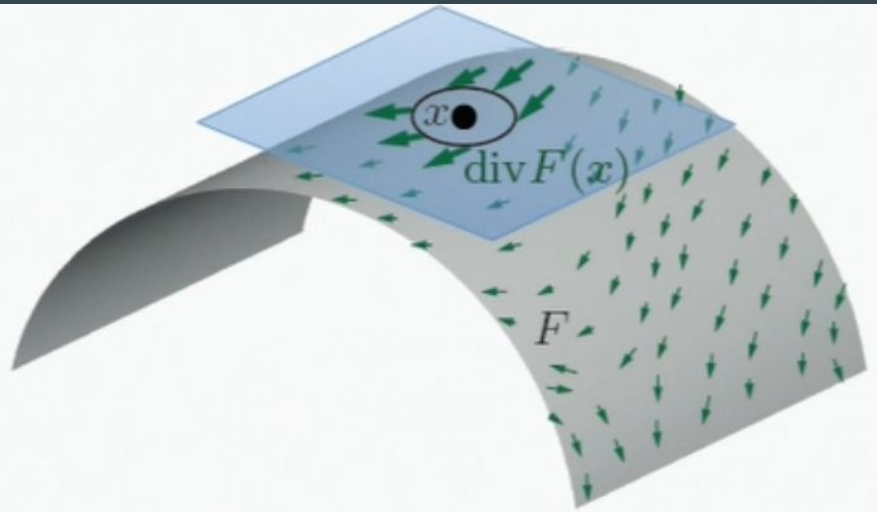


Laplacian on Manifolds

Intrinsic divergence operator

$$\mathrm{div} : L^2(T\mathcal{X}) \to L^2(\mathcal{X})$$

"net flow of field F at x"

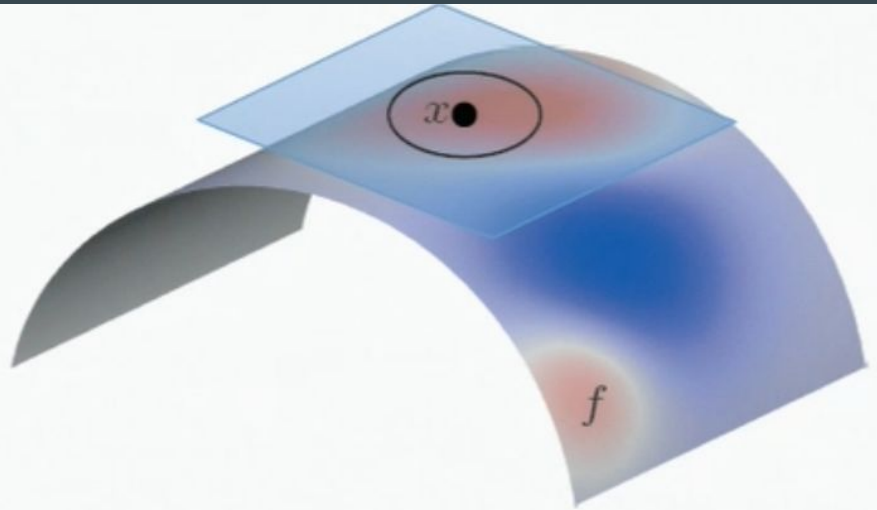


Laplacian on Manifolds

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

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

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



Laplacian on Manifolds

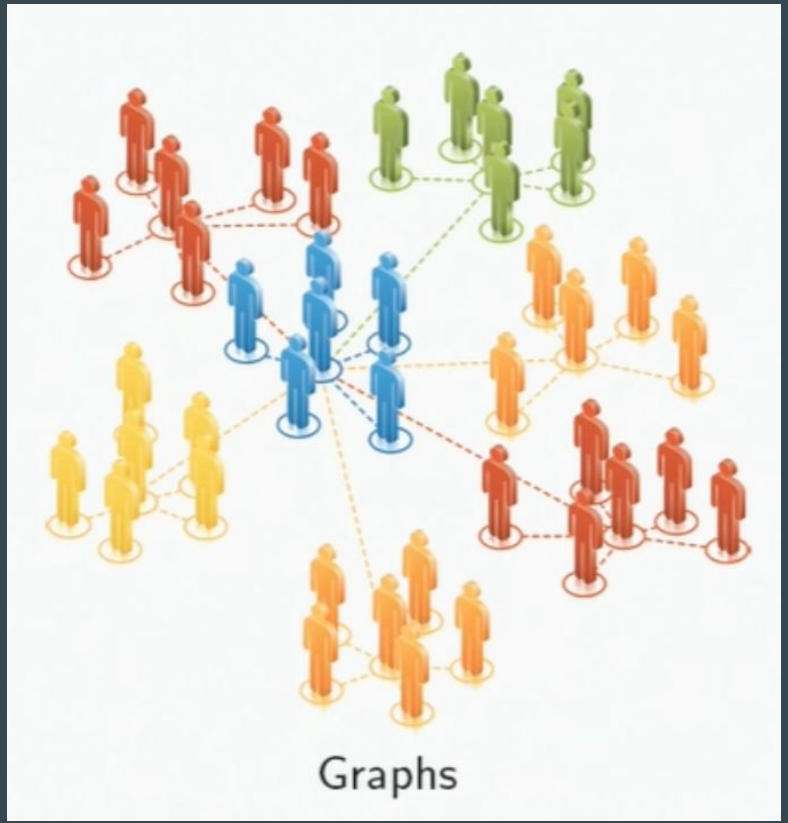
Intrinsic

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

Non-Euclidean Domains

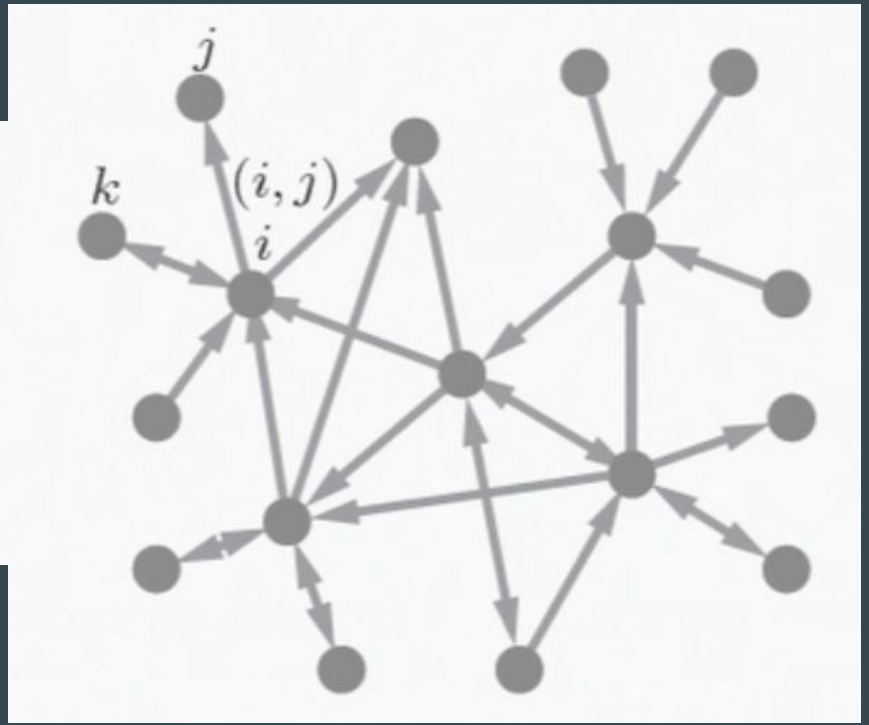


Graph Definitions

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

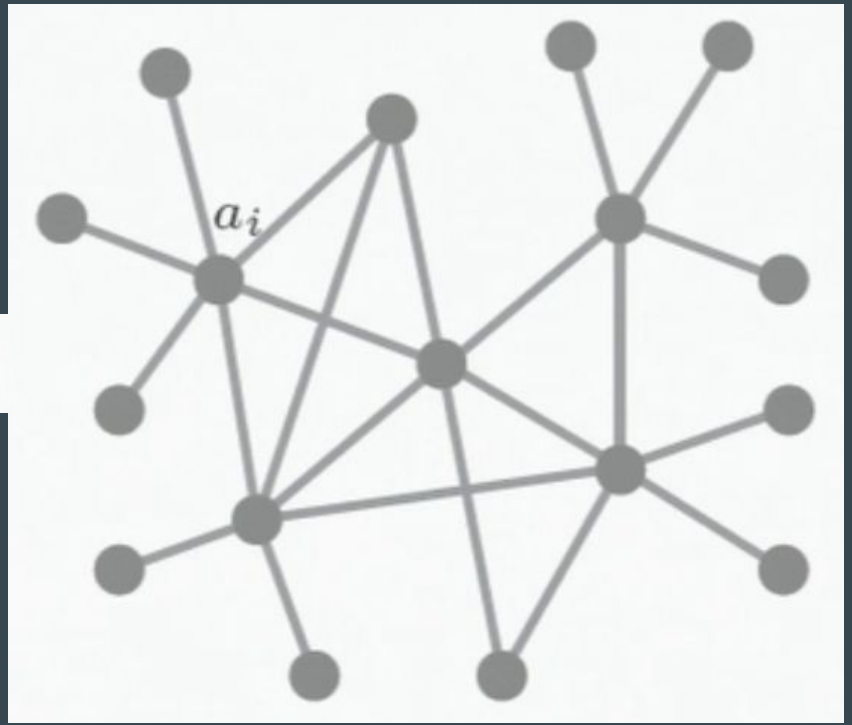
Vertices $\mathcal{V} = \{1, \ldots, n\}$

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



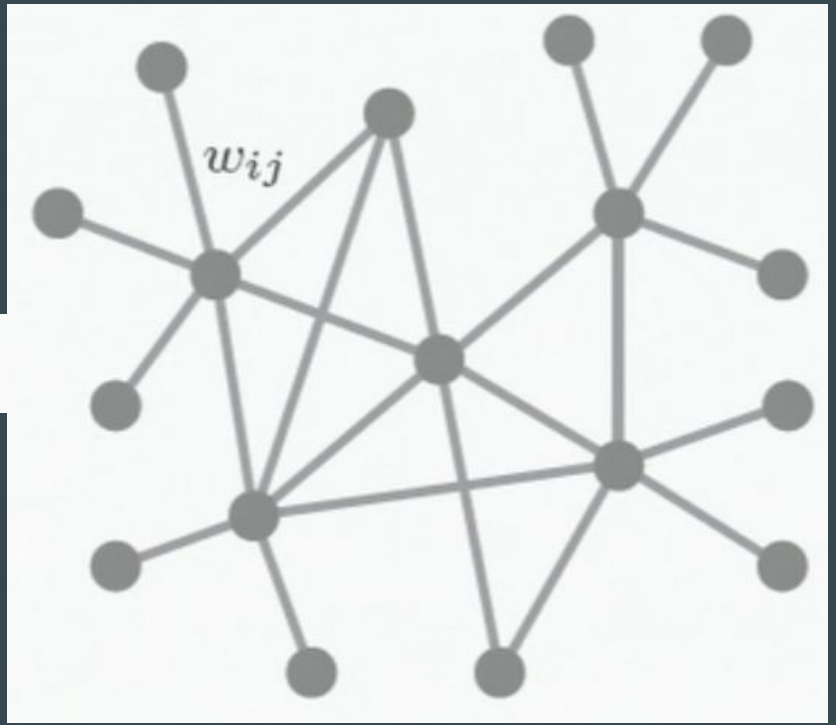
Graph Definitions

Vertex weights $a_i > 0$ for $i \in \mathcal{V}$



Graph Definitions

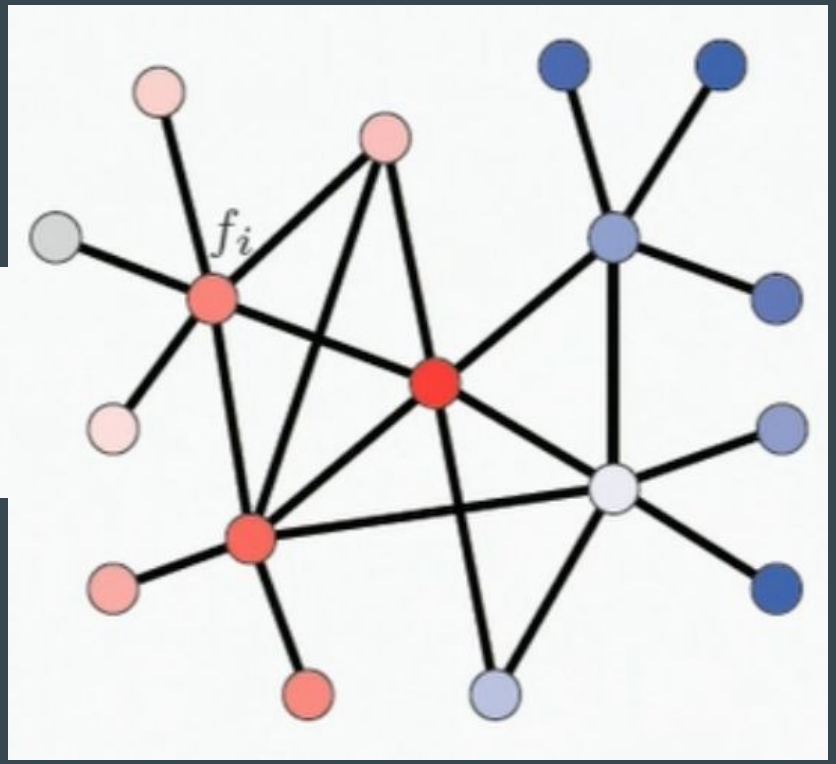
Edge weights $w_{ij} \geq 0$ for $(i, j) \in \mathcal{E}$



Graph Definitions

Vertex field $f : \mathcal{V} \to \mathbb{R}$

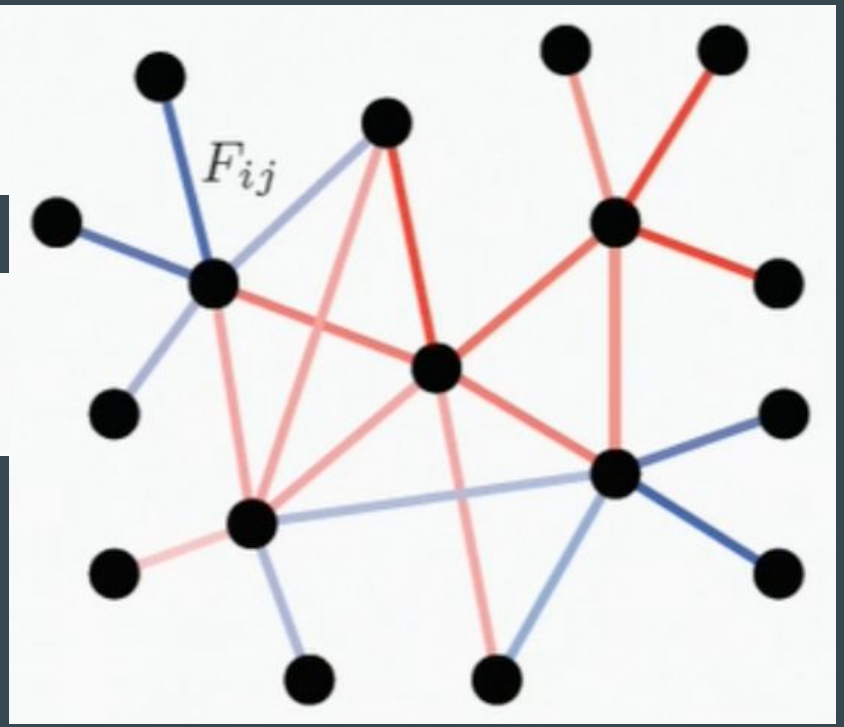
$$\langle f, g \rangle_{L^2(\mathcal{V})} = \sum_{i \in \mathcal{V}} a_i f_i g_i$$



Graph Definitions

Edge field $F : \mathcal{E} \to \mathbb{R}$
assumed alternating $F_{ij} = -F_{ji}$

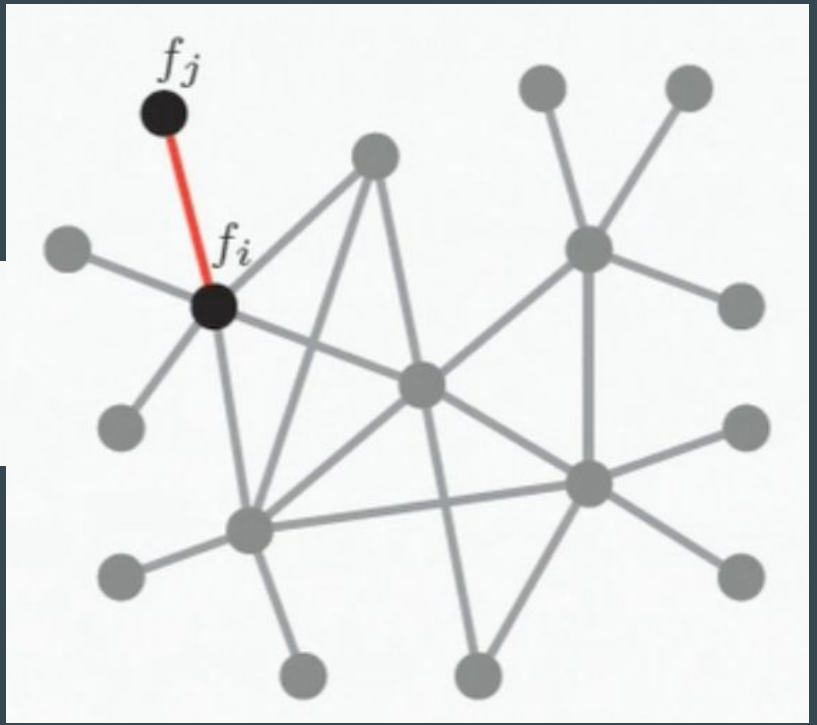
$$\langle F, G \rangle_{L^2(\mathcal{E})} = \sum_{i \in \mathcal{E}} w_{ij} F_{ij} G_{ij}$$



Laplacian on Graphs

Gradient operator $\nabla : L^2(\mathcal{V}) \to L^2(\mathcal{E})$

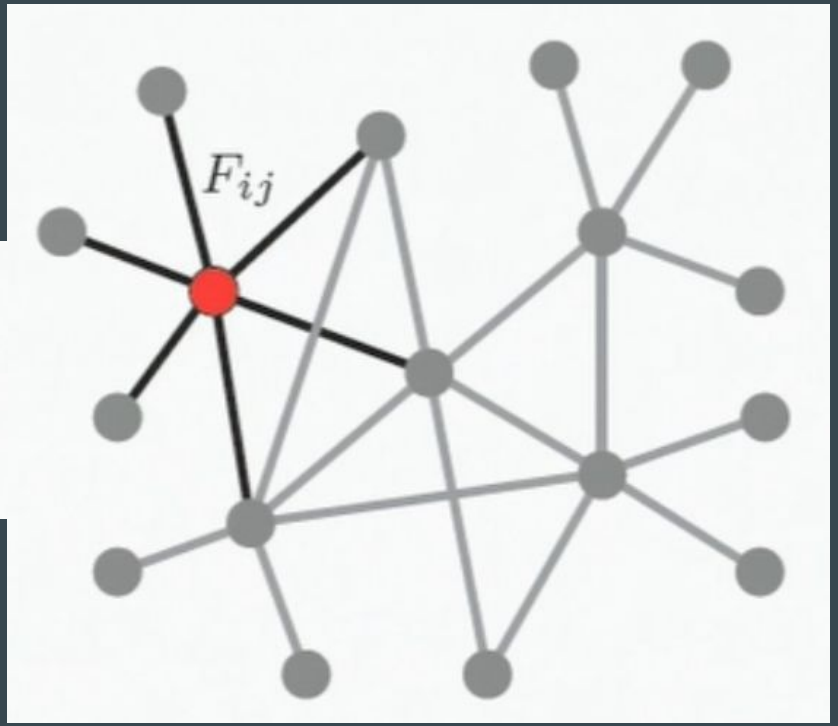
$$(\nabla f)_{ij} = f_i - f_j$$



Laplacian on Graphs

Divergence operator $\mathrm{div} : L^2(\mathcal{E}) \to L^2(\mathcal{V})$

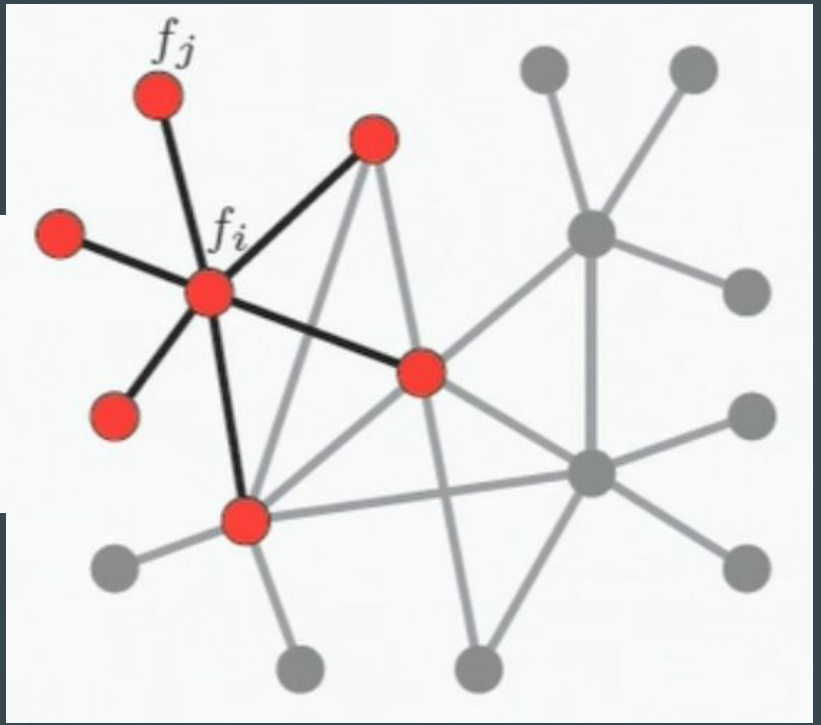
$$(\mathrm{div} F)_i = \frac{1}{a_i} \sum_{j:(i,j)\in\mathcal{E}} w_{ij} F_{ij}$$



Laplacian on Graphs

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

$$(\Delta f)_i = \frac{1}{a_i} \sum_{j:(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j)$$



Laplacian on Graphs

Represented as a positive semi-definite $n \times n$ matrix

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

where $\mathbf{W} = (w_{ij})$, $\mathbf{A} = \mathrm{diag}(a_1, \ldots, a_n)$, and $\mathbf{D} = \mathrm{diag}(\sum_{j \neq i} w_{ij})$

Laplacian on Graphs

Represented as a positive semi-definite $n \times n$ matrix

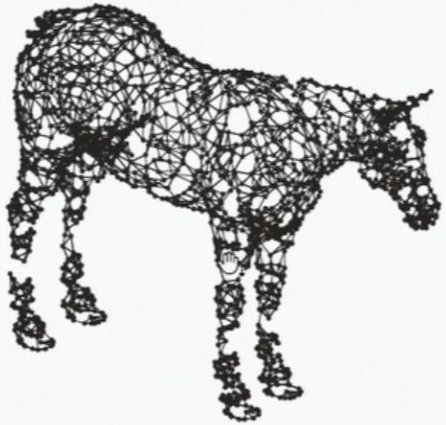
$$\mathbf{\Delta} = \mathbf{A}^{-1}(\mathbf{D} - \mathbf{W})$$

where $\mathbf{W} = (w_{ij})$, $\mathbf{A} = \mathrm{diag}(a_1, \ldots, a_n)$, and $\mathbf{D} = \mathrm{diag}(\sum_{j \neq i} w_{ij})$

Unnormalized Laplacian: $\mathbf{\Delta} = \mathbf{D} - \mathbf{W}$ $\quad (\mathbf{A} = \mathbf{I})$

Random walk Laplacian: $\mathbf{\Delta} = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{W})$ $\quad (\mathbf{A} = \mathbf{D})$

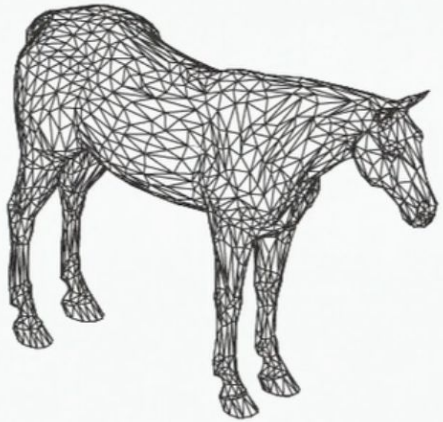
Graphs as Discrete Manifolds



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

Spectral Analysis

Eigendecomposition of the Laplacian

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

$$\Delta\phi_k(x) = \lambda_k\phi_k(x), \qquad k = 1, 2, \ldots$$

Eigenfunctions are real and orthonormal

Eigenvalues are non-negative $0 = \lambda_1 \leq \lambda_2 \leq \ldots$

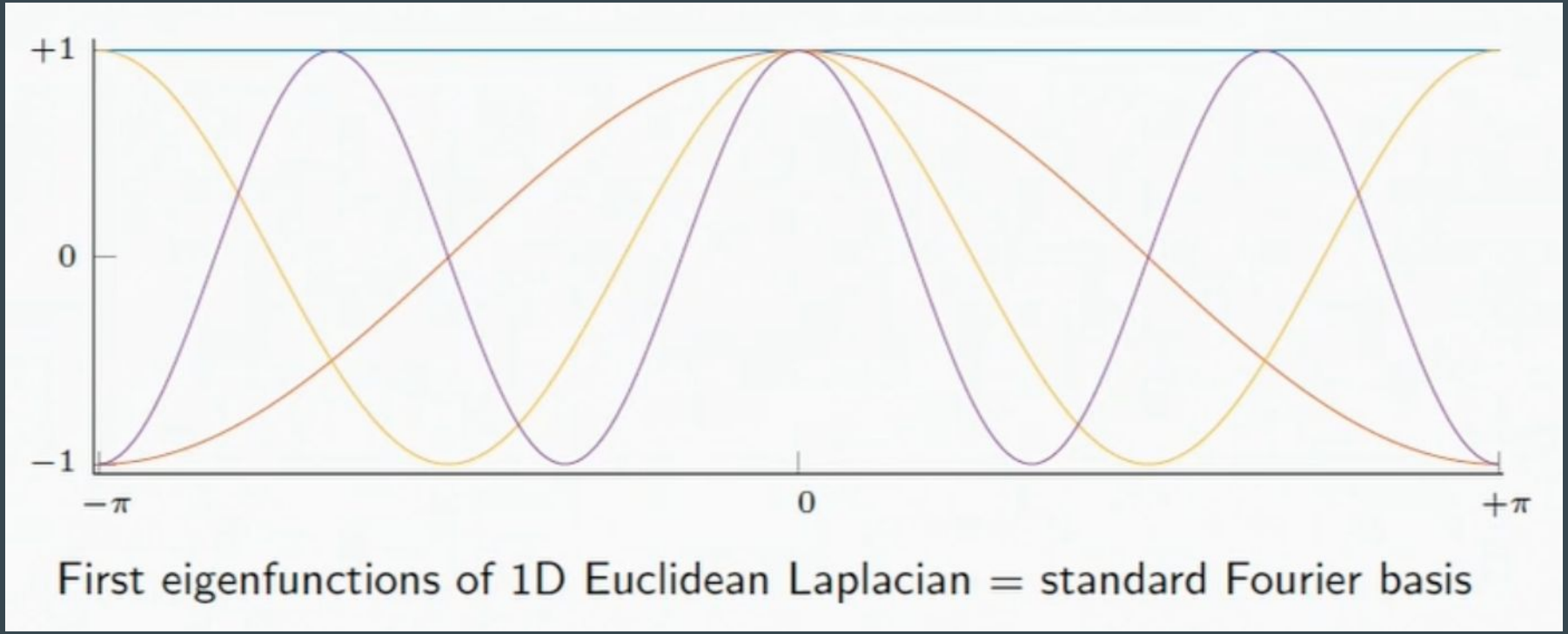
Eigendecomposition of the Laplacian

Eigendecomposition of a graph Laplacian

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

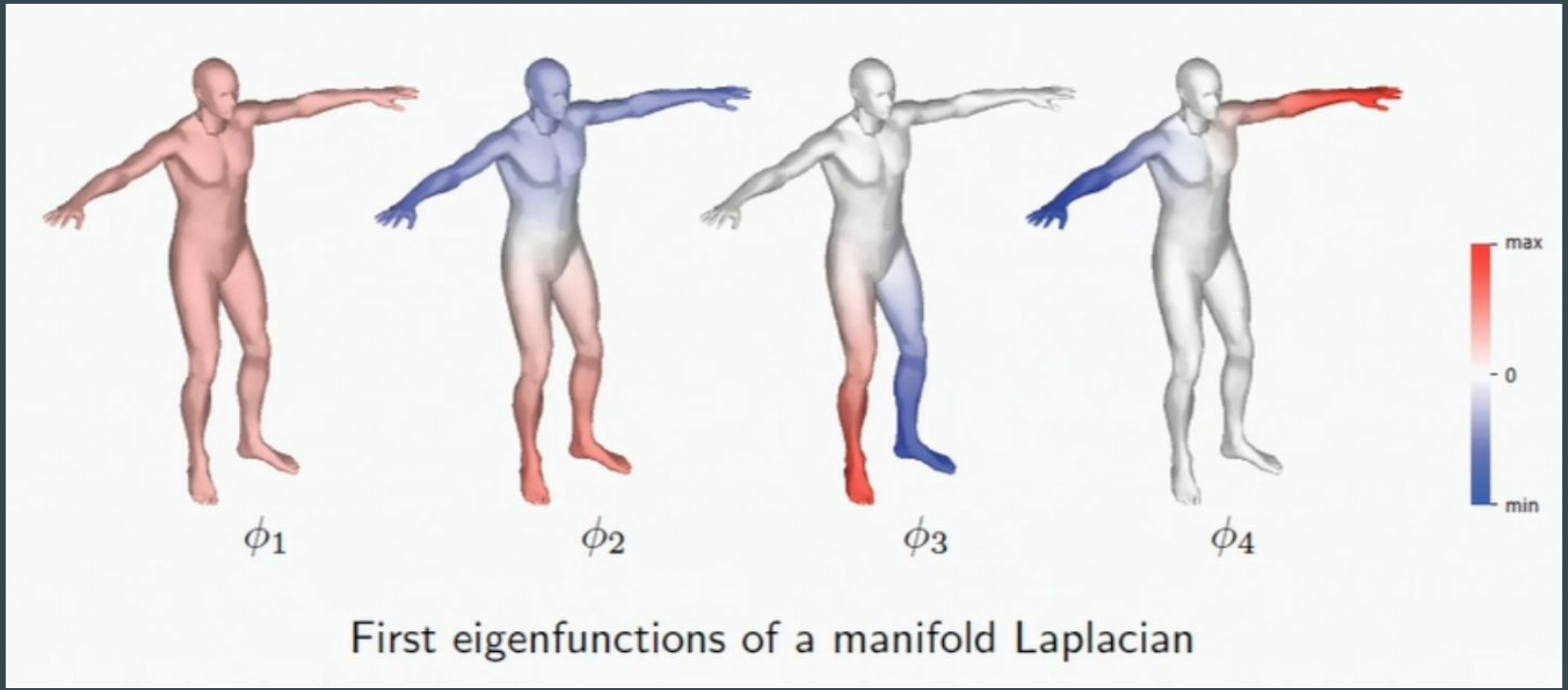
where $\mathbf{\Phi} = (\phi_1, \ldots, \phi_n)$ and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$

Eigendecomposition of the Laplacian



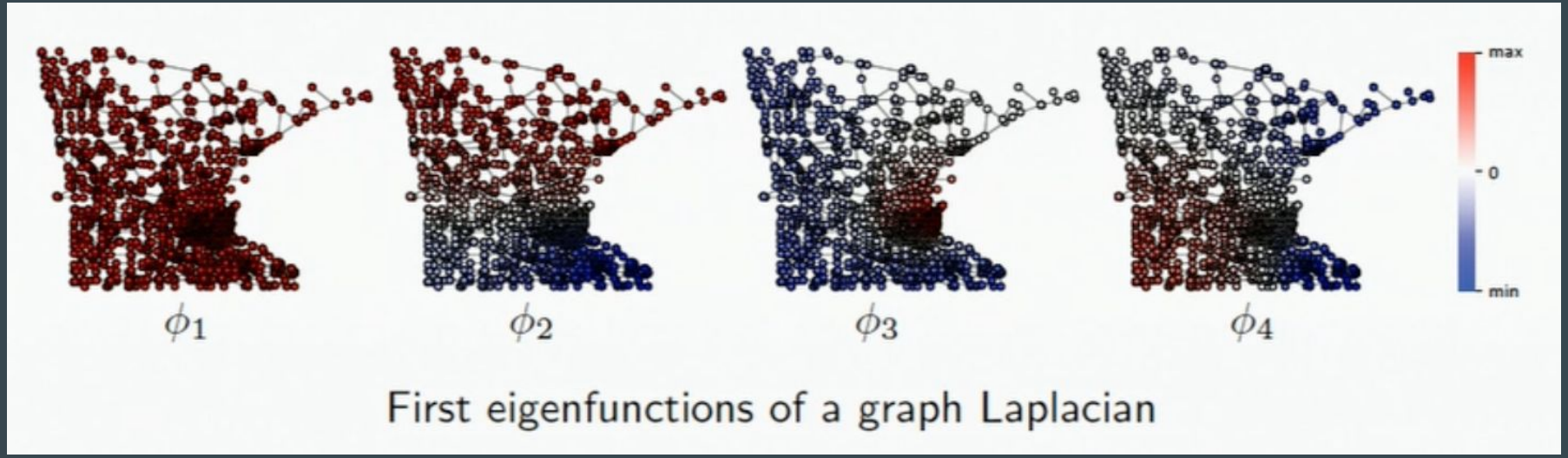
First eigenfunctions of 1D Euclidean Laplacian = standard Fourier basis

Eigendecomposition of the Laplacian



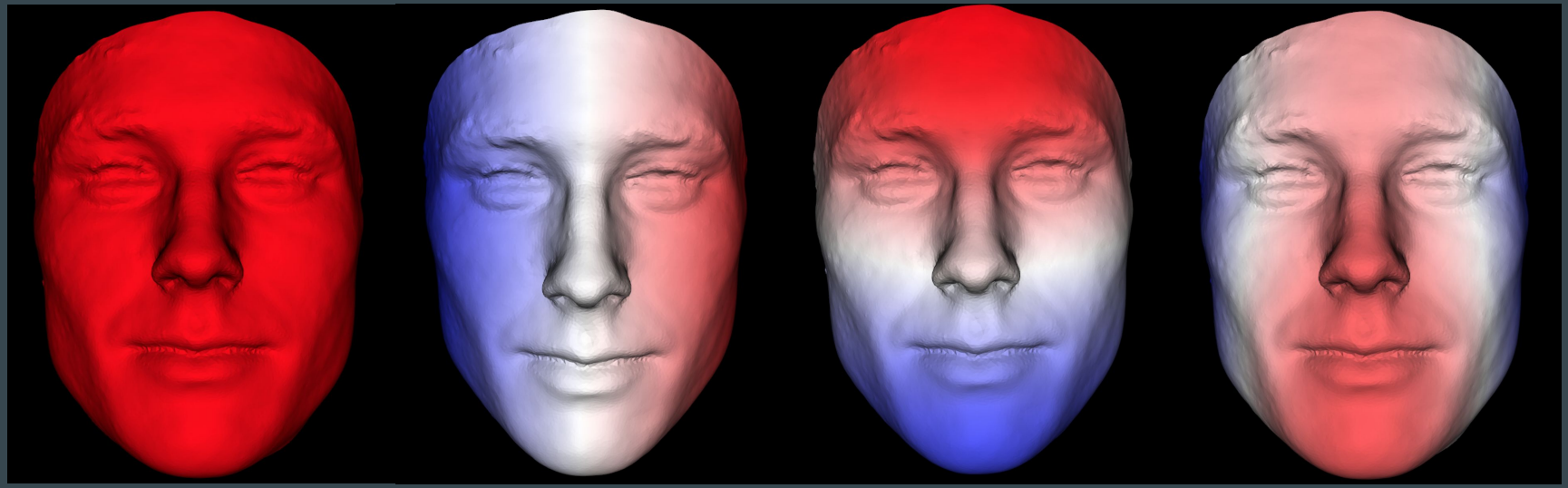
First eigenfunctions of a manifold Laplacian

Eigendecomposition of the Laplacian

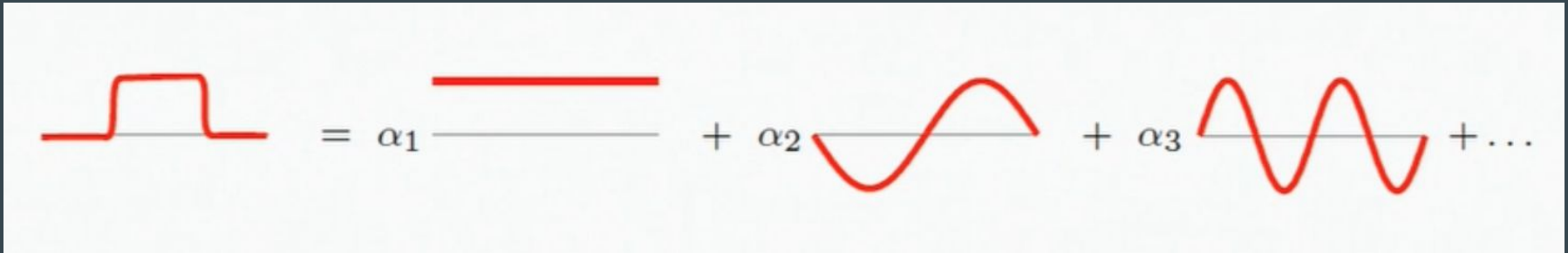


First eigenfunctions of a graph Laplacian

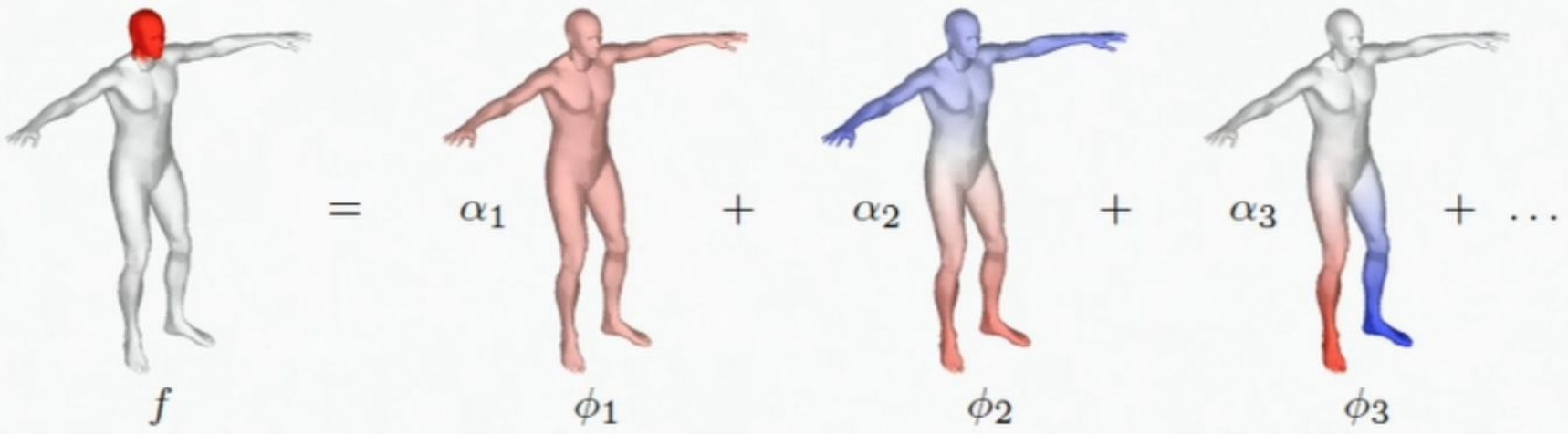
Eigendecomposition of the Laplacian



Fourier Analysis



Fourier Analysis



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

Convolution in the Euclidean Domain

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

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

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

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

- Efficient computation using FFT with $\mathcal{O}(n \log n)$ complexity

Convolution in the Euclidean Domain

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

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

Convolution in the Spectral Domain

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

Convolution in the Spectral Domain

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

Convolution in the Spectral Domain

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

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

Generalized Spectral Convolution

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

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

Generalized Spectral Convolution

In matrix-vector notation

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

Generalized Spectral Convolution

In matrix-vector notation

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

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

Generalized Spectral Convolution

In matrix-vector notation

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

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

Generalized Spectral Convolution

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

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

Commutes with Laplacian: $\mathbf{G}\mathbf{\Delta}\mathbf{f} = \mathbf{\Delta}\mathbf{G}\mathbf{f}$

Filter coefficients depend on basis $\phi_1, \ldots, \phi_n$

Generalized Spectral CNNs

Spectral Convolutional Layer

Convolutional layer expressed in the spectral domain

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

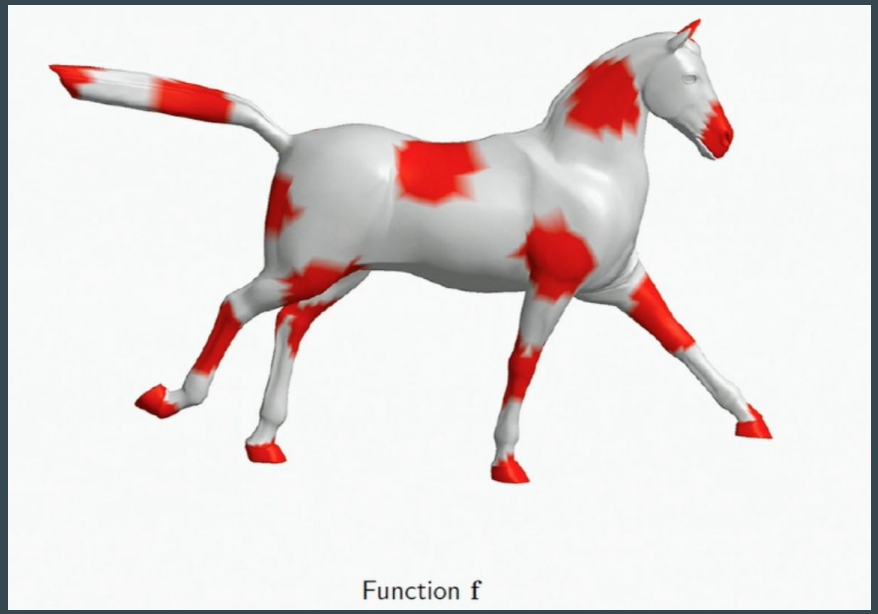
where $\mathbf{W}_{l,l} = n \times n$ diagonal matrix of filter coefficients

Spectral Convolutional Layer

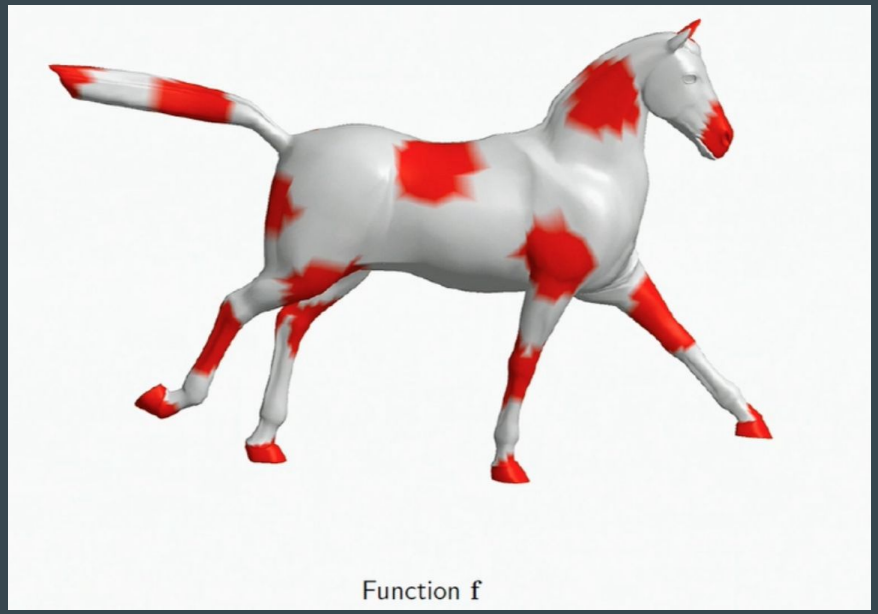
$$\mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{\Phi}\mathbf{W}_{l,l'}\mathbf{\Phi}^\top \mathbf{f}_{l'}\right) \qquad \begin{matrix} l = 1, \ldots, q \\ l' = 1, \ldots, p \end{matrix}$$

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

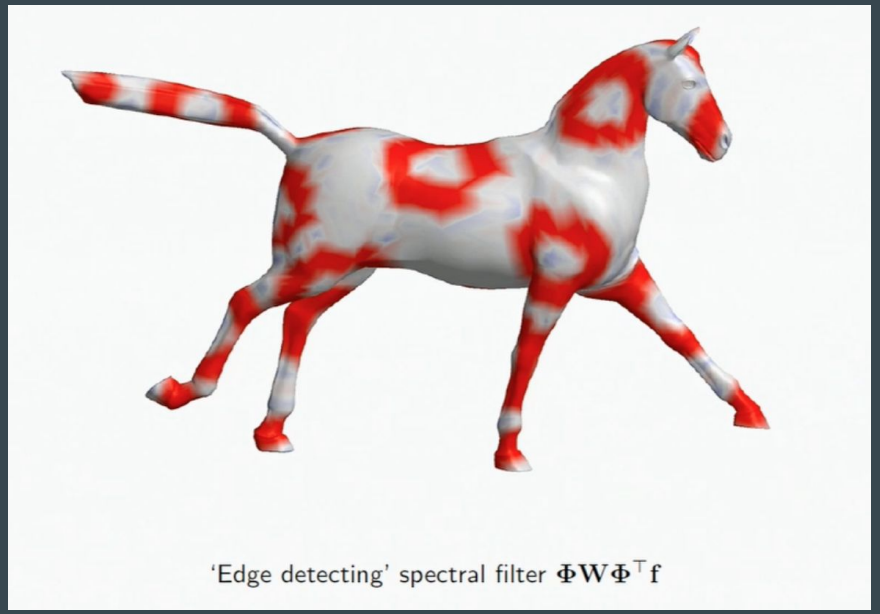
Lack of Generalizability



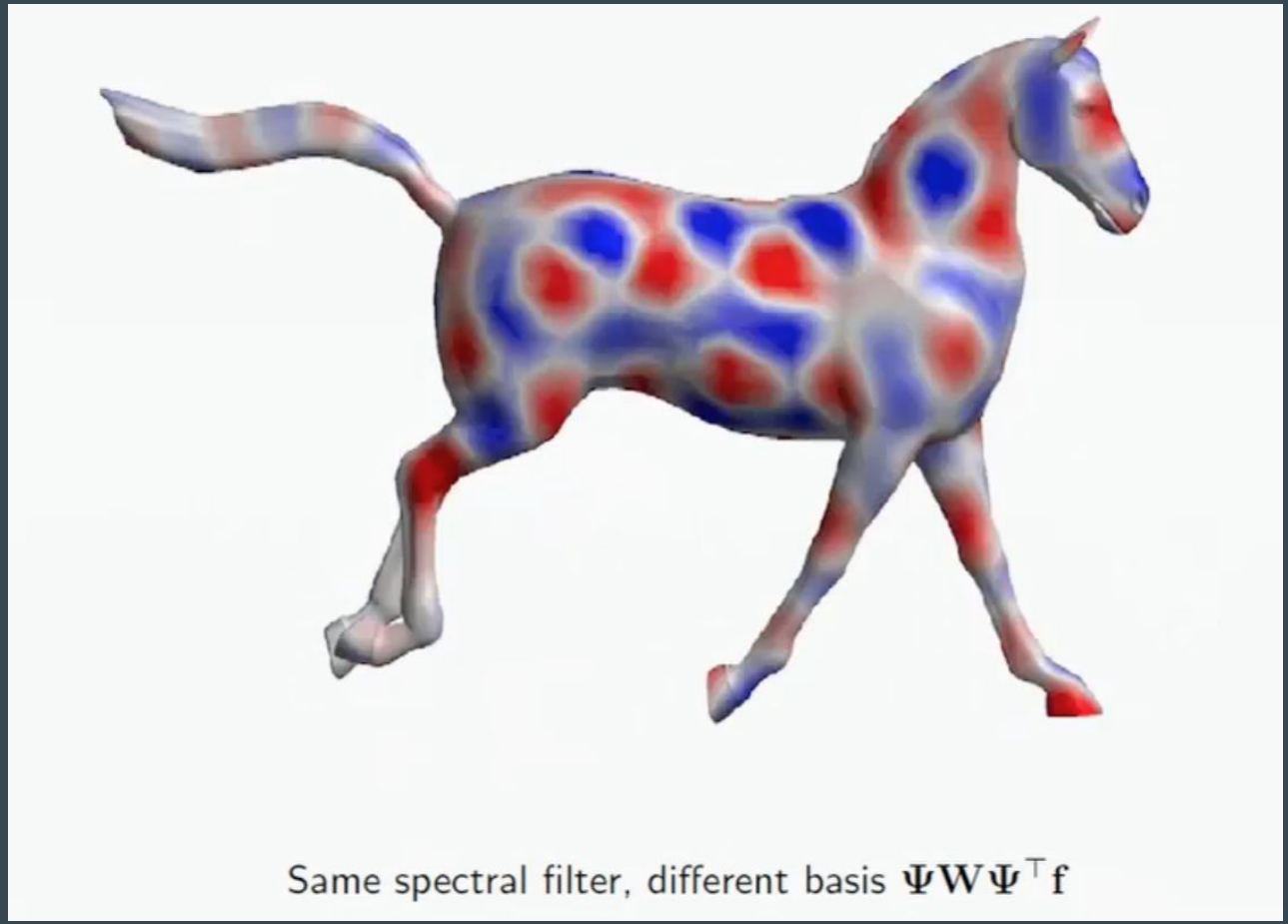
Lack of Generalizability



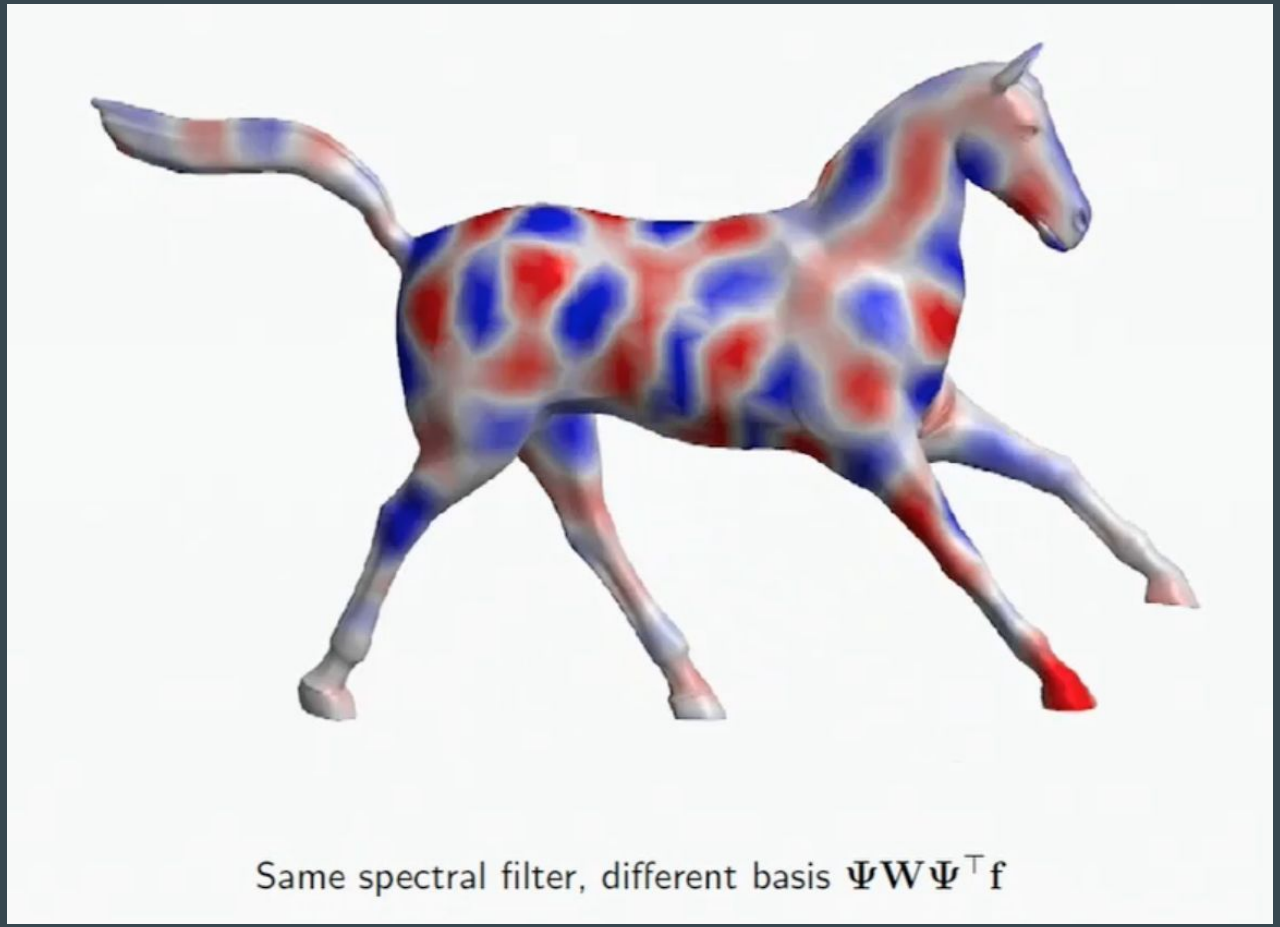
Function $\mathbf{f}$



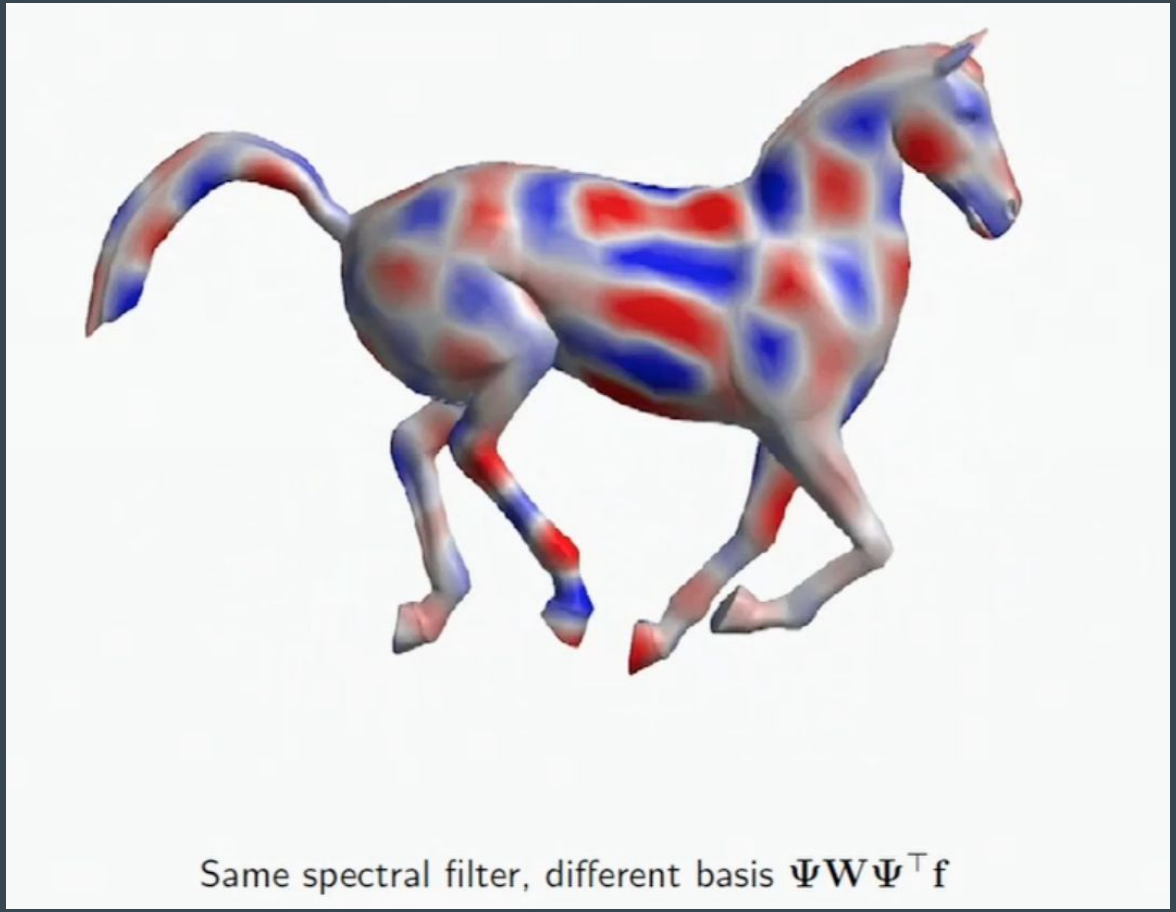
'Edge detecting' spectral filter $\mathbf{\Phi W \Phi}^{\top}\mathbf{f}$



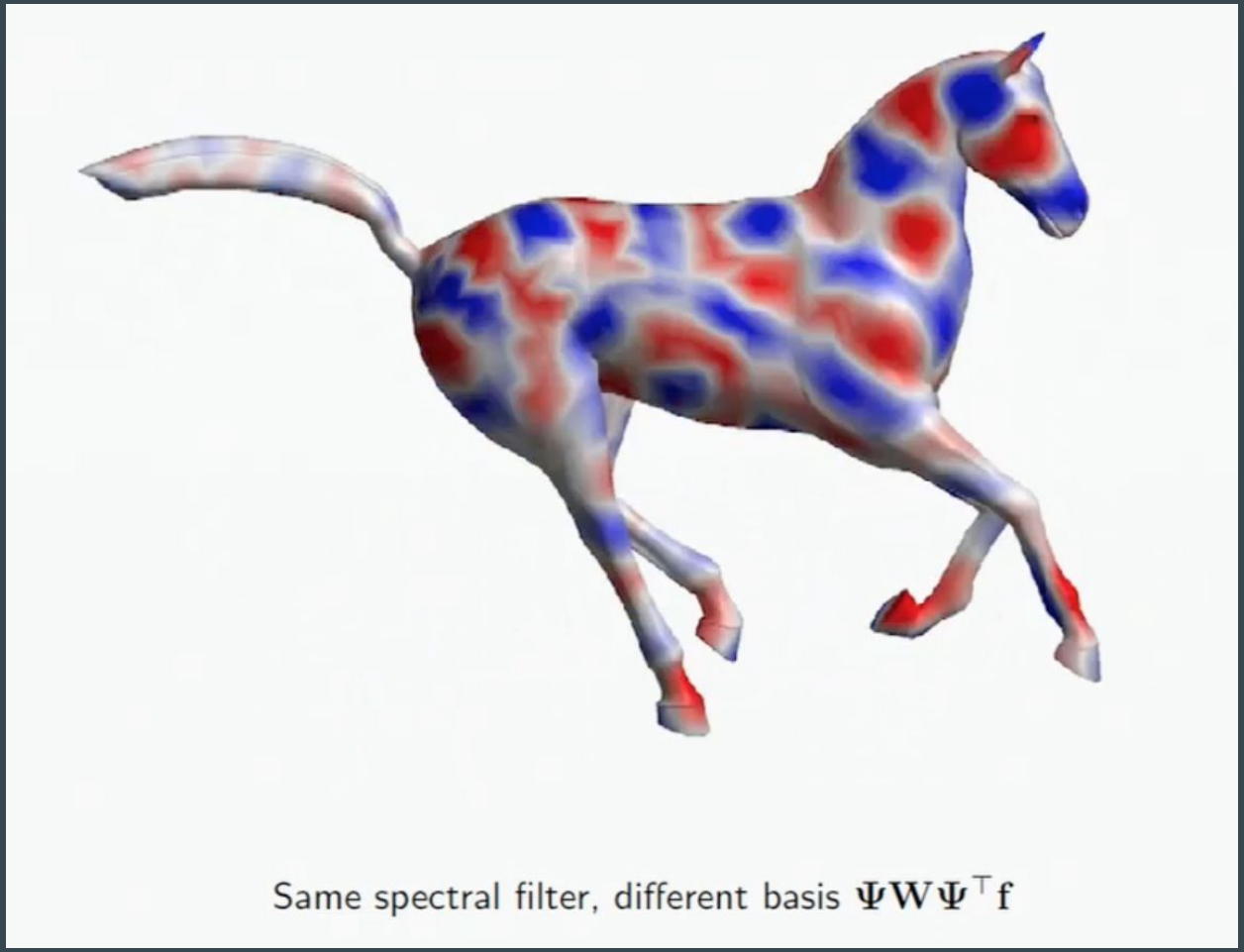
Same spectral filter, different basis $\Psi \mathbf{W} \Psi^\top \mathbf{f}$



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



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



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

Spatial Localization

Localization in space = smoothness in frequency domain

Spatial Localization

Localization in space = smoothness in frequency domain

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

Spatial Localization

Localization in space = smoothness in frequency domain

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

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

Spatial Localization

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

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

Spatial Localization

Represent spectral transfer function as a polynomial or order r

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

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

Application of the filter $\tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \sum_{j=0}^{r} \alpha_j \boldsymbol{\Delta}^j \mathbf{f}$

Spatial Localization

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

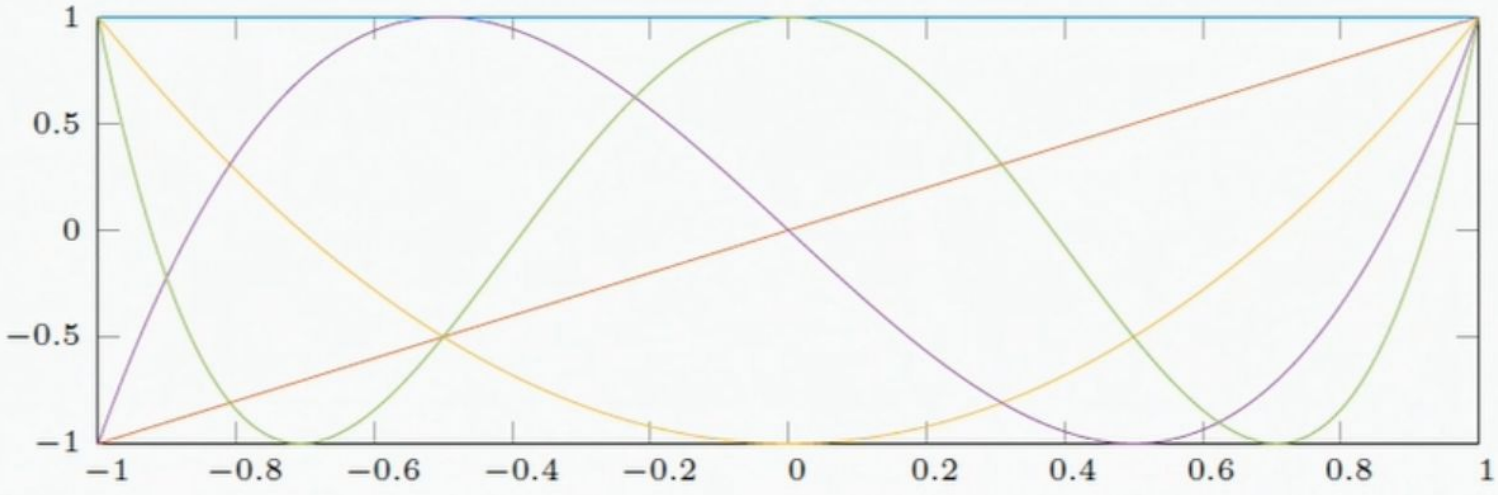
- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed r-hops support
- ☺ No explicit computation of $\boldsymbol{\Phi}^\top, \boldsymbol{\Phi} \Rightarrow \mathcal{O}(nr)$ computational complexity (assuming sparsely-connected graph)

Almost There...

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

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed r-hops support
- ☺ No explicit computation of $\boldsymbol{\Phi}^\top, \boldsymbol{\Phi} \Rightarrow \mathcal{O}(nr)$ computational complexity (assuming sparsely-connected graph)
- ☹ Unstable under coefficients perturbation (hard to optimize)

Chebyshev polynomials



Chebyshev polynomials of order r defined recursively

$$T_j(\tilde{\lambda}) = 2\tilde{\lambda}T_{j-1}(\tilde{\lambda}) - T_{j-2}(\tilde{\lambda}) \qquad T_0(\tilde{\lambda}) = 1, \quad T_1(\tilde{\lambda}) = \tilde{\lambda}$$

Chebyshev approximation

Represent spectral transfer function as a Chebyshev polynomial of order r

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

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters and $-1 \leq \tilde{\lambda} \leq 1$

Application of the filter to scaled Laplacian $\tilde{\boldsymbol{\Delta}} = 2\lambda_n^{-1}\boldsymbol{\Delta} - \mathbf{I}$ recursively

$$\mathbf{g}^{(j)} = 2\tilde{\boldsymbol{\Delta}}\mathbf{g}^{(j-1)} - \mathbf{g}^{(j-2)} \qquad \mathbf{g}^{(0)} = \mathbf{f}, \quad \mathbf{g}^{(1)} = \tilde{\boldsymbol{\Delta}}\mathbf{f}$$

Final Construction!

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

$$\mathbf{g}^{(j)} = 2\tilde{\boldsymbol{\Delta}}\mathbf{g}^{(j-1)} - \mathbf{g}^{(j-2)} \qquad \mathbf{g}^{(0)} = \mathbf{f}, \quad \mathbf{g}^{(1)} = \tilde{\boldsymbol{\Delta}}\mathbf{f}$$

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

Where to go from here?

- Generalizing across graphs
- Pooling on graphs
- Spatial Convolution on graphs
- Wavelet Convolution
- Applications and current results

Thank you!