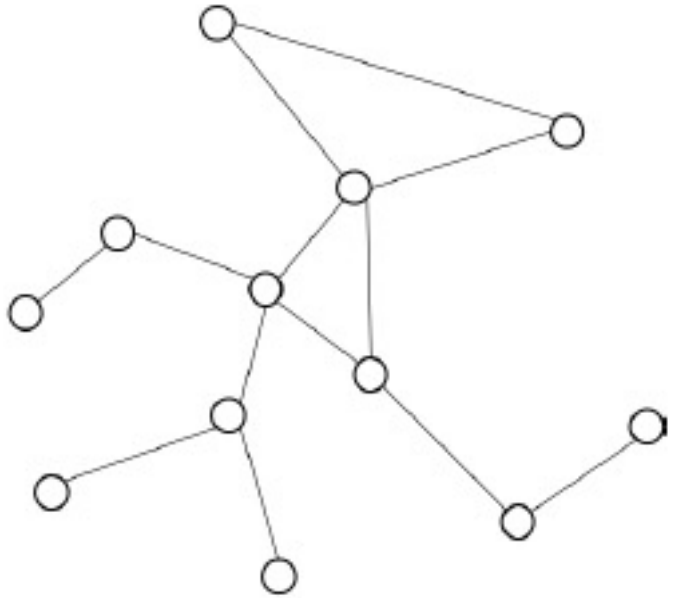
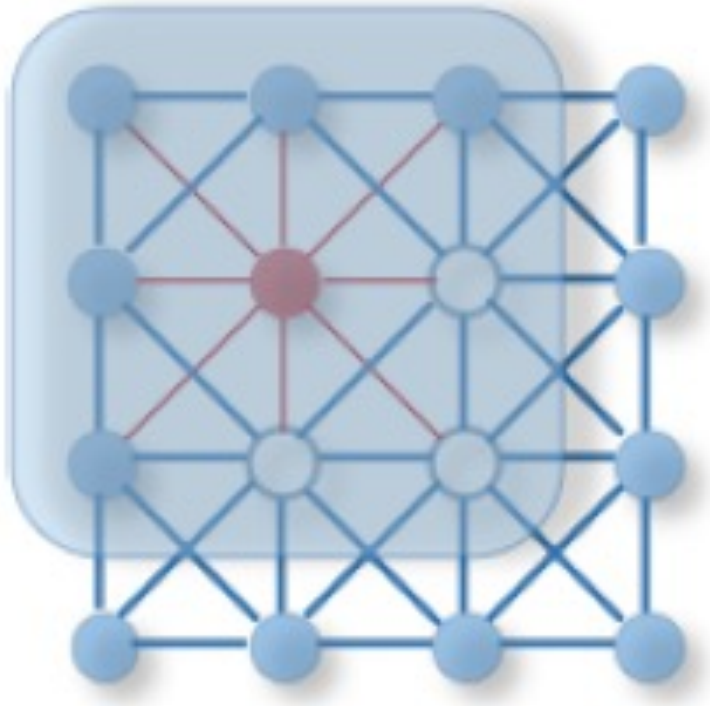
Spectral Graph Convolutional Networks

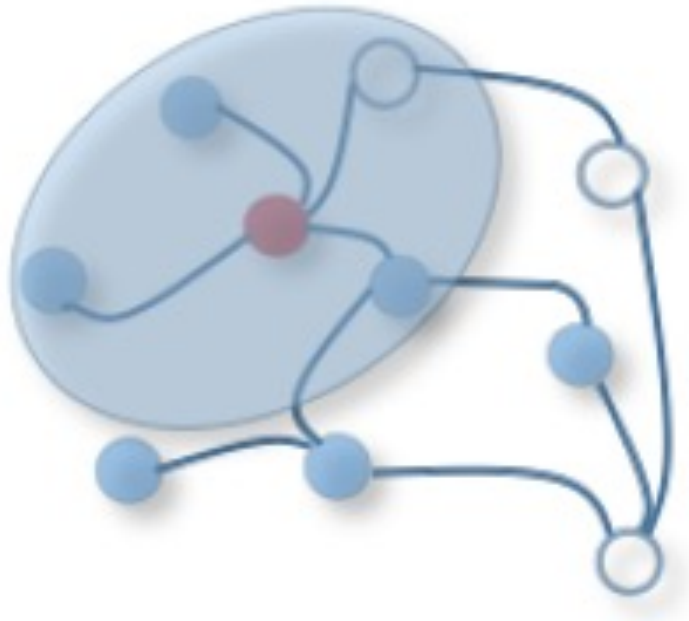
Yale

Images are a specific sort of graph—grid graph



More general graph structures





Recall: CNNs use three basic ideas

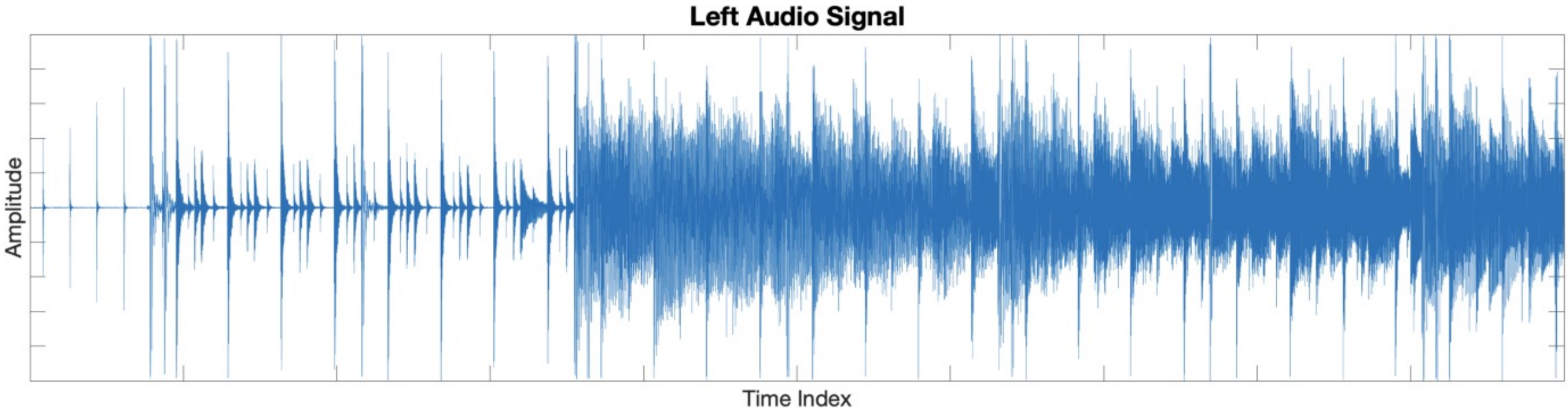
1. Local receptive fields
    1. Problem: There is no clearly defined direction or method of translation over a graph (cannot cleanly slide a convolution)
2. Shared weights
3. Pooling
    1. Hierarchical
    2. Simple

How can we approximate these on more general non-grid graphs

- Spatial construction (Bruna, Hamilton)
  - Apply the same operator and move over the graph space in some fashion
  - Notion of locality based on neighbors of a node

- Spectral construction (Bruna, Defferard)
  - Define a convolutional operation in the graph spectral space
  - Uses signal processing theory to define filters in the graph Fourier domain
  - Notion of locality based on global properties of the filter

Time series signals



https://towardsdatascience.com/the-wavelet-transform-e9cfa85d7b34

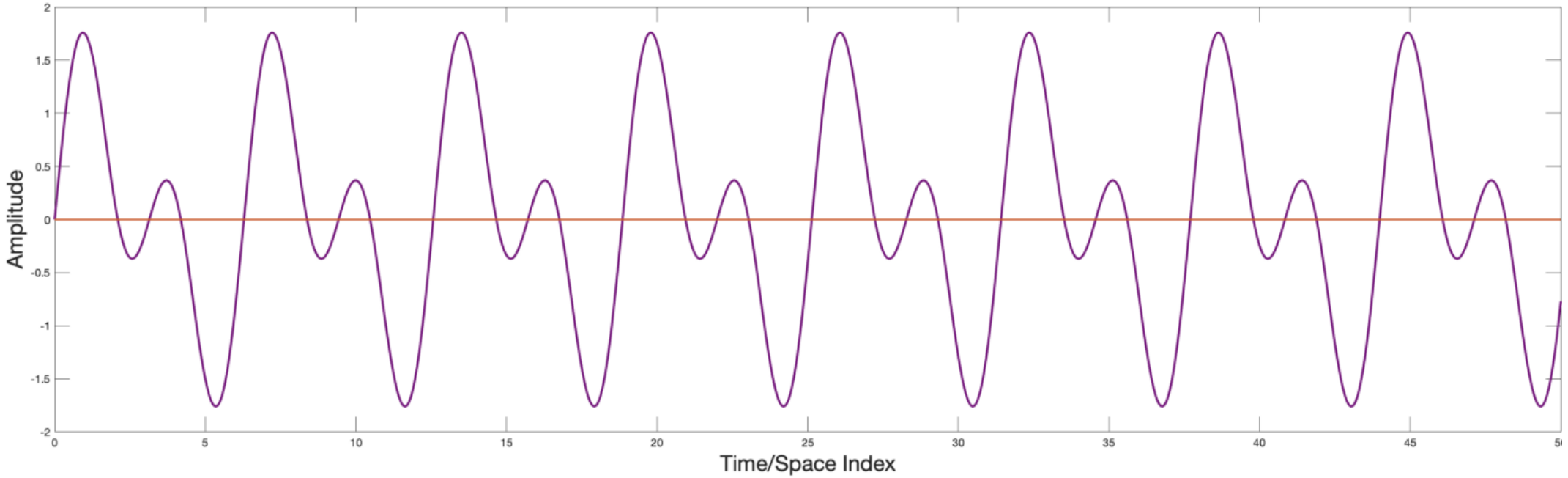
Fourier Transform of a Signal

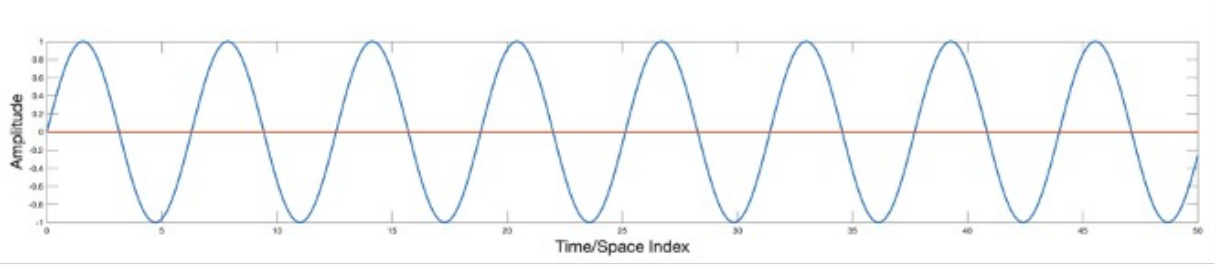
$$f(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$$

Recall Euler's formula:

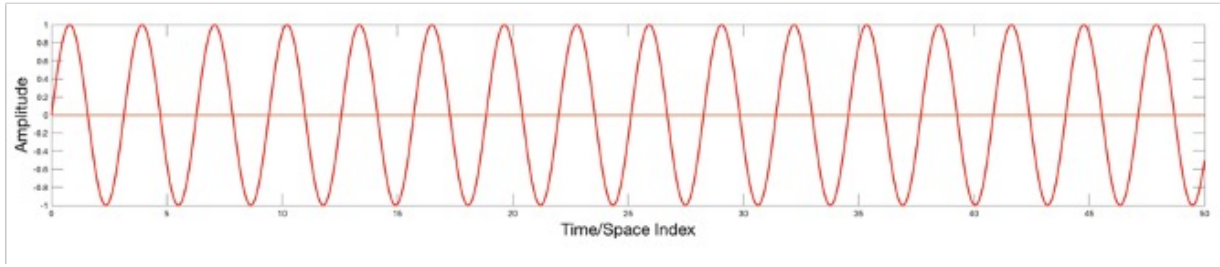
$e^{-i\vartheta x} = \cos(\vartheta x) + i sin(\vartheta x)$

Decomposes signal into frequencies





+



Discrete Fourier Transform

- Usually we don't have the functional form available for computing the integral, so we use discrete samples and discrete wavelengths

$$f(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$$

FT

Discretize

$$f_k = \Sigma_0^{N-1} x_n e^{\frac{-2\pi i k n}{N}}$$

DFT

DFT is a matrix multiplication

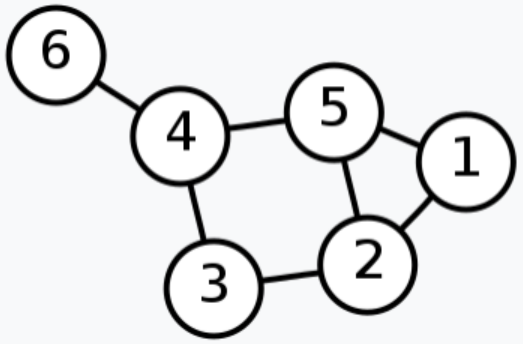
$$\begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{K-1} \end{pmatrix} = \begin{pmatrix} R_{0,0} & R_{0,1} & \cdots & R_{0,N-1} \\ R_{1,0} & R_{1,1} & & \vdots \\ \vdots & & & \vdots \\ R_{K-1,0} & \cdots & \cdots & R_{K-1,N-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix}$$

$R_{k,n}$ is the value of the kth waveform at time index n $R_{kn} = e^{\frac{-2\pi ikn}{N}}$

Fourier transforms have numerous uses

- Power spectral analysis
- Solutions of differential equations (heat equation)
- Audio and image processing
- …much more

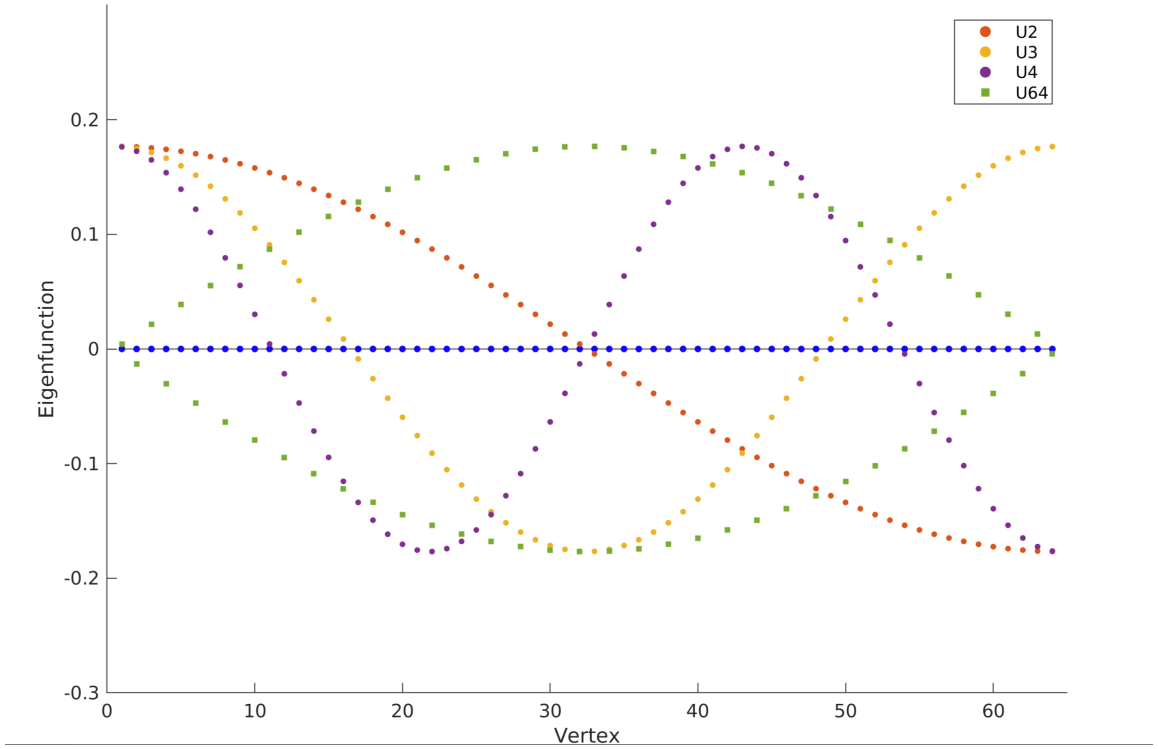
Graphs, Degree Matrix, Adjacency Matrix

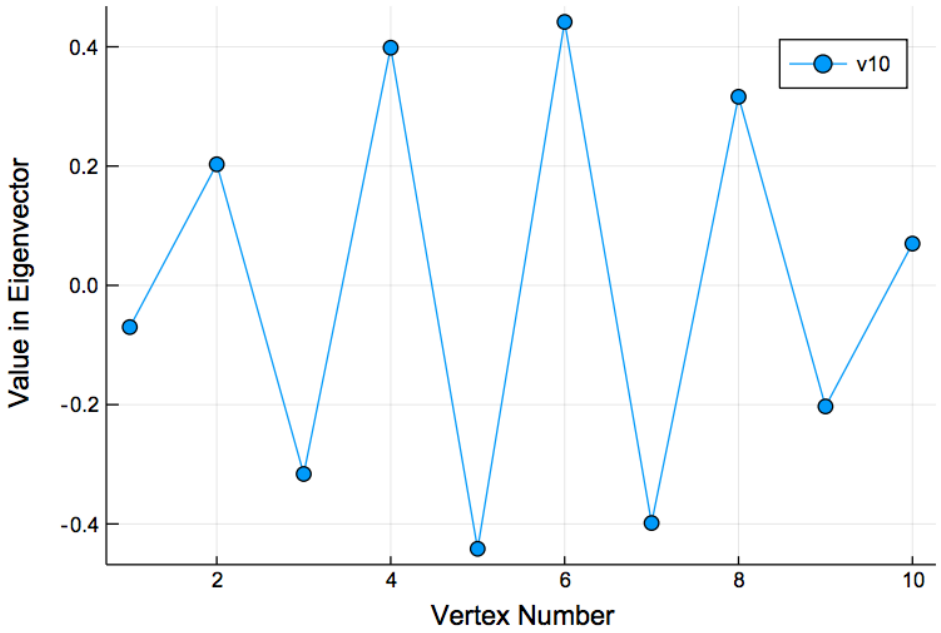
Labelled graph	Degree matrix	Adjacency matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

Graph Laplacian

- A difference operator based on the graph adjacency matrix A.
- (unnormalized Laplacian)$L = D - A$
  - Degree matrix: $D_{ii} = \sum_j A_{ij}$, 0 everywhere else
- (normalized Laplacian) $L = I - D^{-1/2} A D^{-1/2}$
  - I is identity
  - Measures how similar a point is to its neighbors
- (random walk Laplacian) $L = I - D^{-1} A = I - M$
  - Related to Markovian matrix

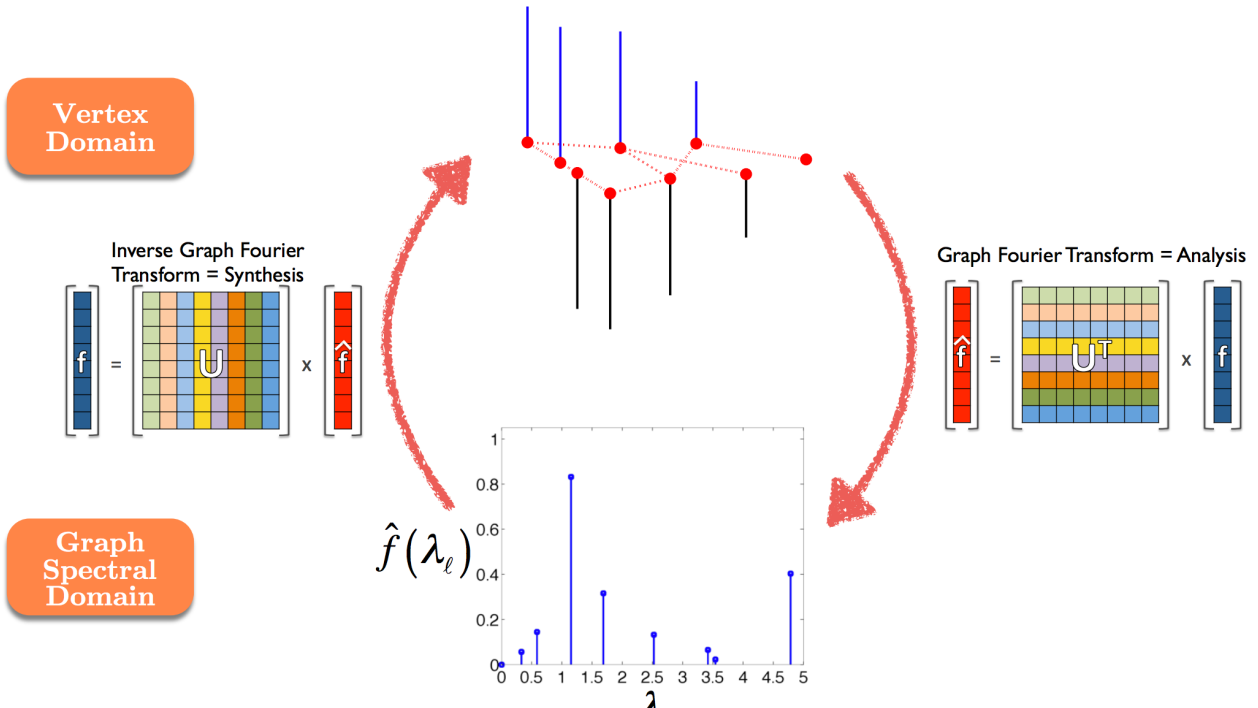
Eigenvectors form a Graph Fourier Spectrum





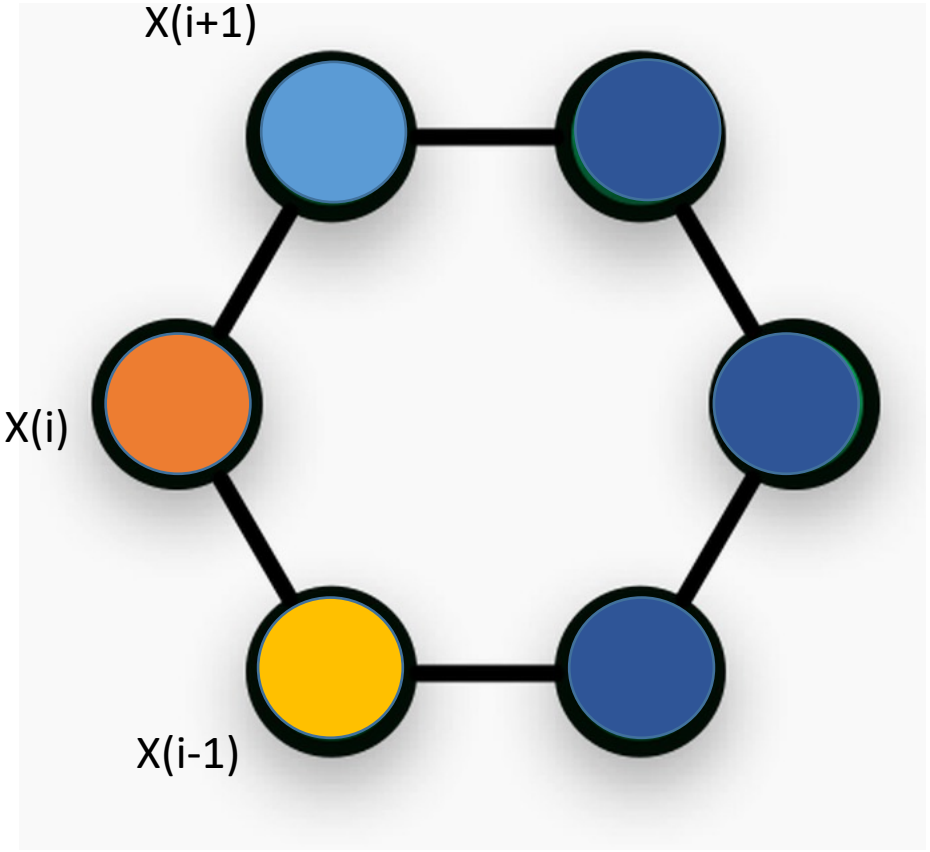
Signals on a graph substrate

- These signals have an analogous Graph Fourier Transform
- Involves using the eigenvectors of the graph Laplacian as the waveforms $L = U \wedge U^T$



Graph Laplacian

- Imagine the Graph Laplacian is created from a dataset with features
- One feature is X
  - X(i) is X evaluate at node i
- First derivative in discrete graph setting is the difference:
  - d(i)= X(i)-X(i-1)
  - d(i+1)=X(i+1)-X(i)
- Second derivative
  - d(i+1)-d(i)
  - =X(i+1)-2X(i)+X(i-1)



Based on Adjacency Matrix

A= $\begin{bmatrix} 0 & 1 & & & & & 1 \\ 1 & 0 & 1 & & & & \\ 0 & 1 & 0 & 1 & & & \\ & & 1 & 0 & 1 & & \\ & & & 1 & 0 & 1 & \\ & & & & 1 & 0 & 1 \\ 1 & & & & & 1 & 0 \end{bmatrix}$

2nd derivative= A-D $\begin{bmatrix} -2 & 1 & & & & & 1 \\ 1 & -2 & 1 & & & & \\ 0 & 1 & -2 & 1 & & & \\ & & 1 & -2 & 1 & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & 1 \\ 1 & & & & & 1 & -2 \end{bmatrix}$

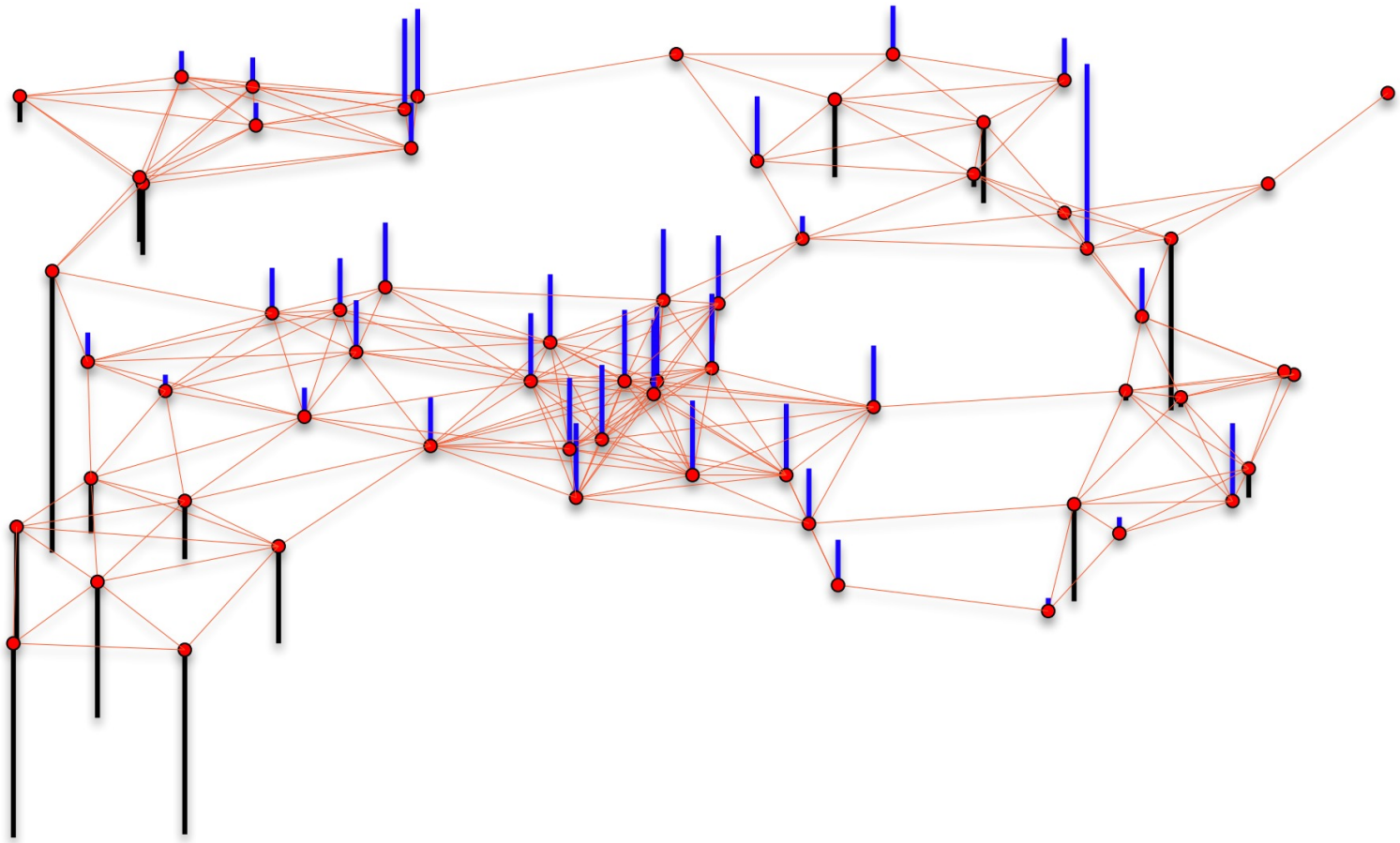
L=D-A $\begin{bmatrix} 2 & -1 & & & & & -1 \\ -1 & 2 & -1 & & & & \\ 0 & -1 & 2 & -1 & & & \\ & & -1 & 2 & -1 & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ -1 & & & & & -1 & 2 \end{bmatrix}$ L $\begin{bmatrix} X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix}$ X $= LX$

Estimates smoothness of X at each point

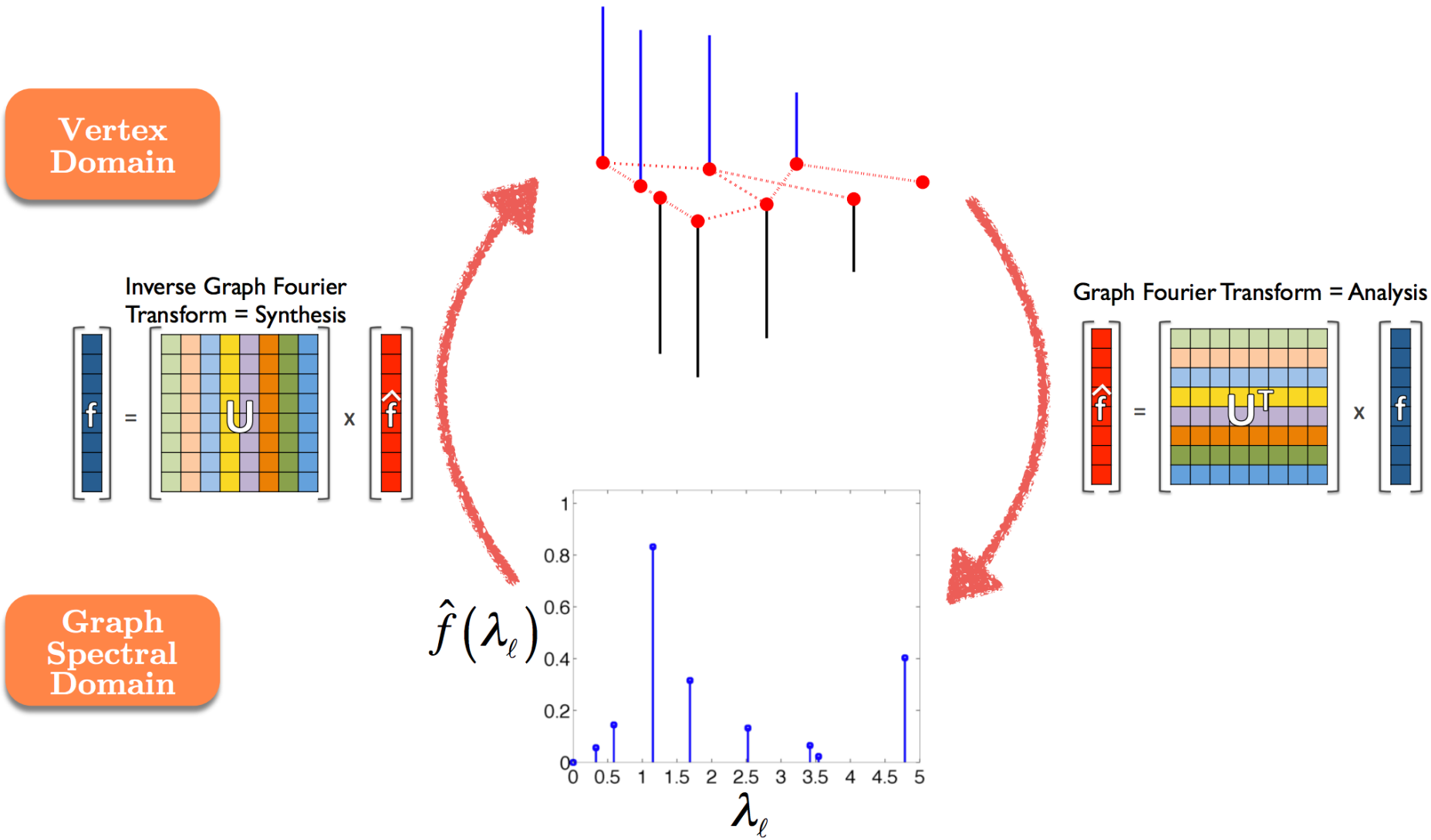
$$s = X^T LX$$

s gives smoothness as a score, lower s is more smooth

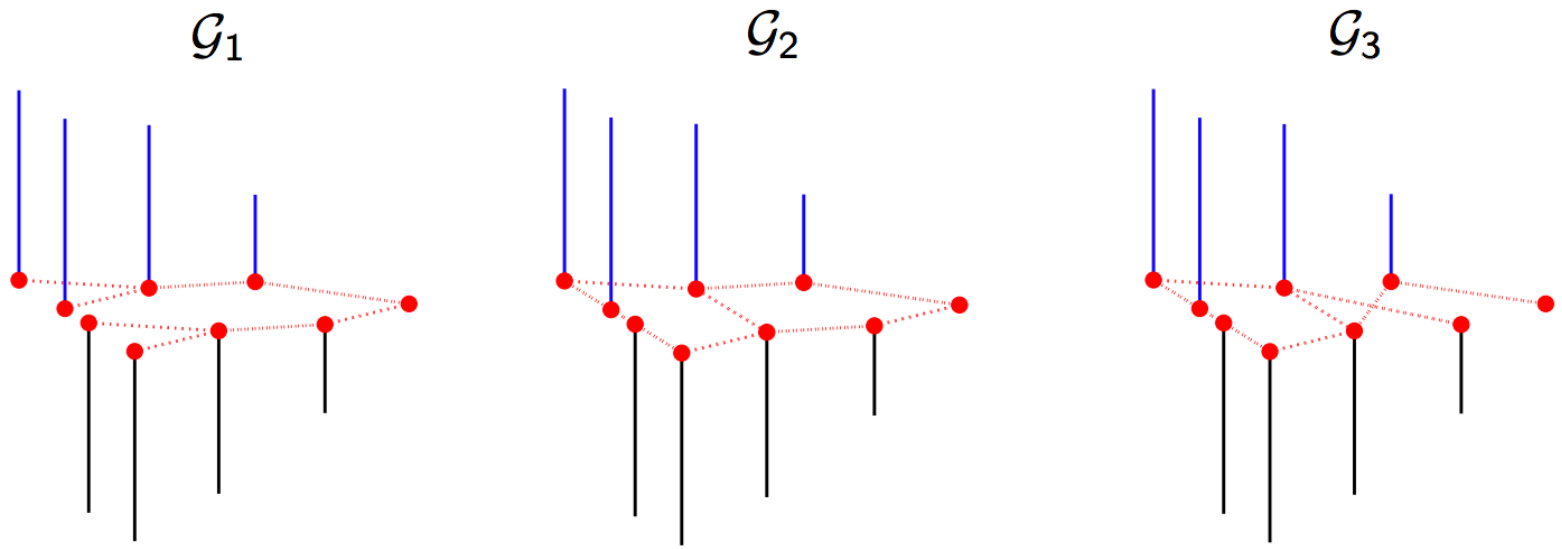
Features are signals on a graph



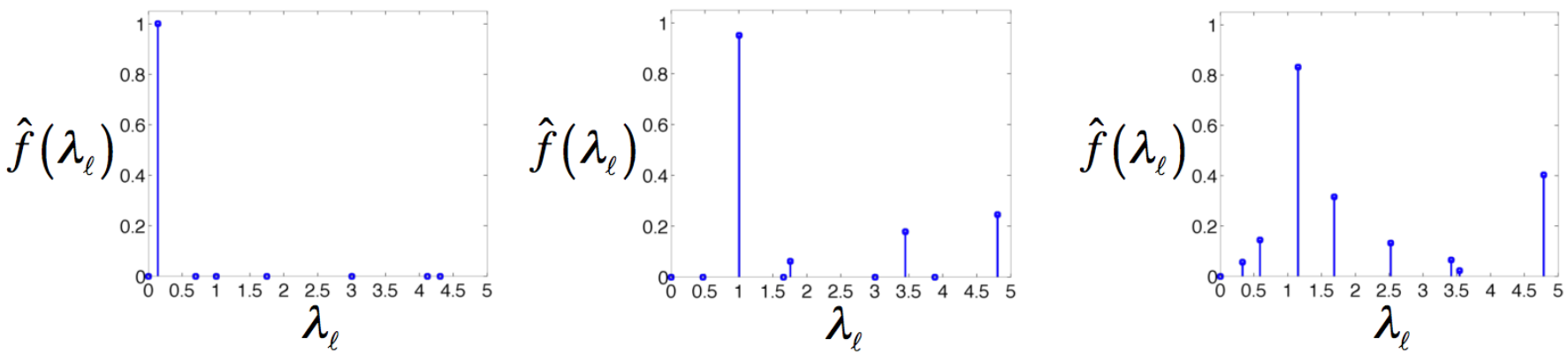
Graph Fourier Transform



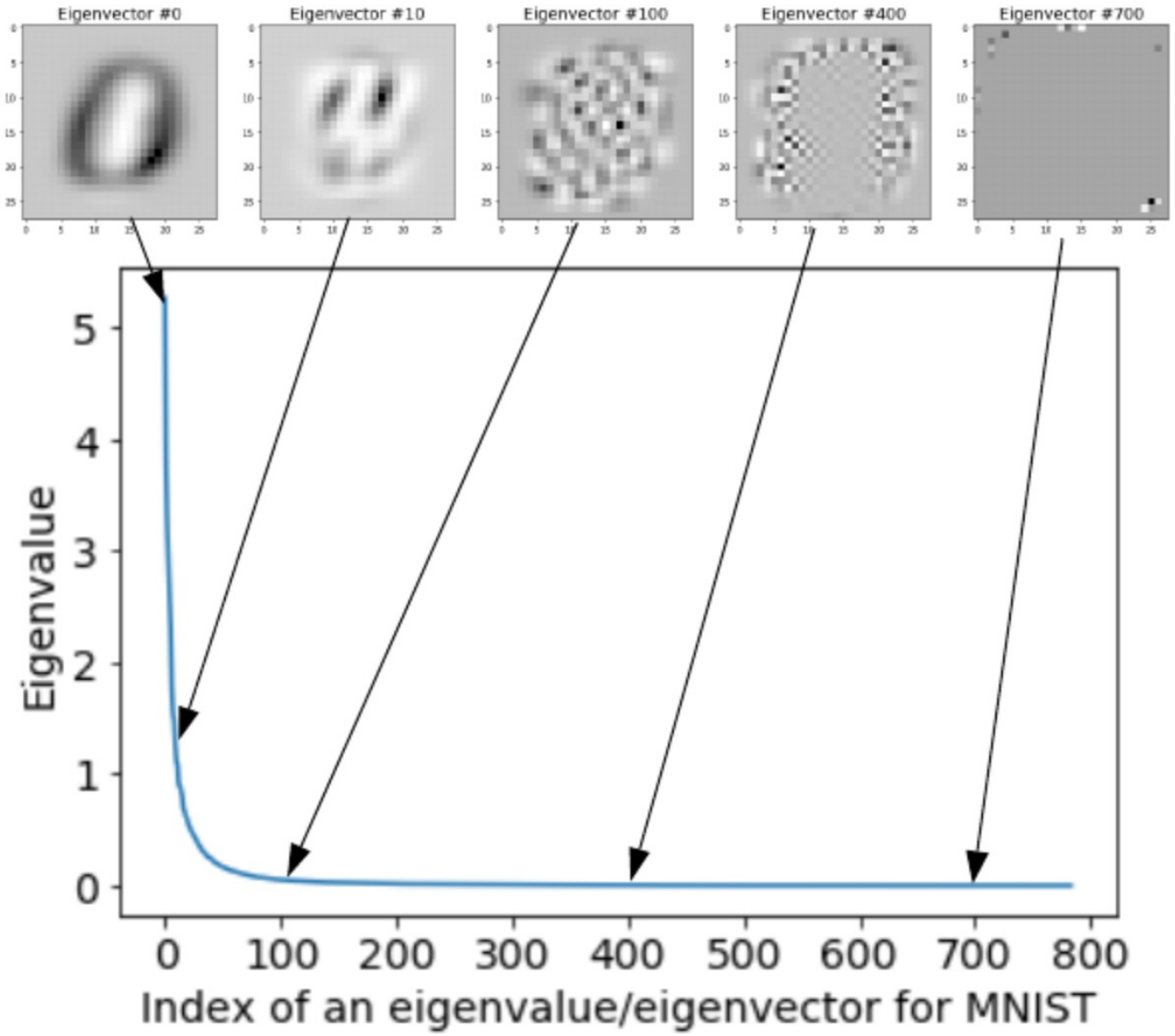
Vertex Domain



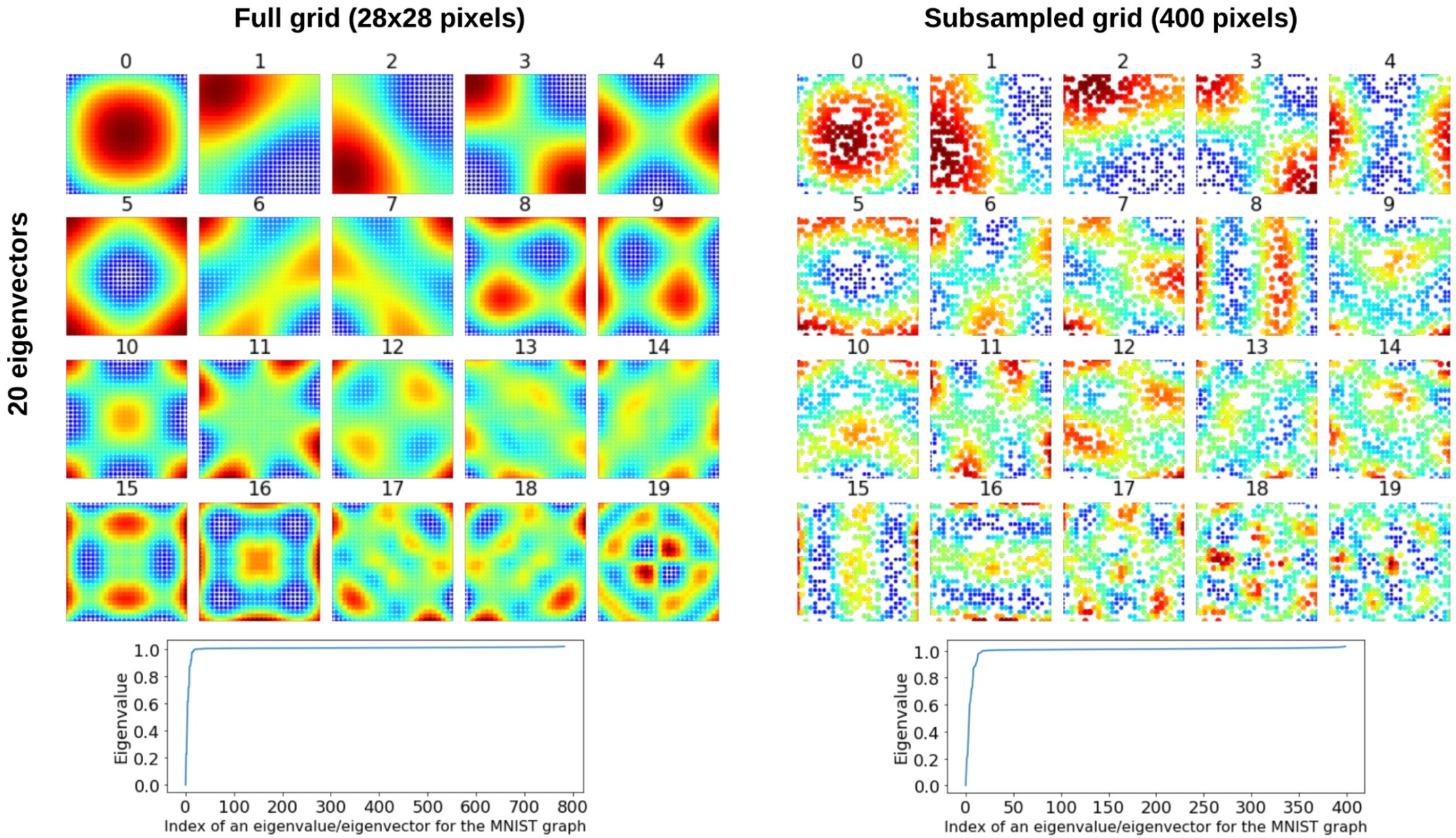
Graph Spectral Domain



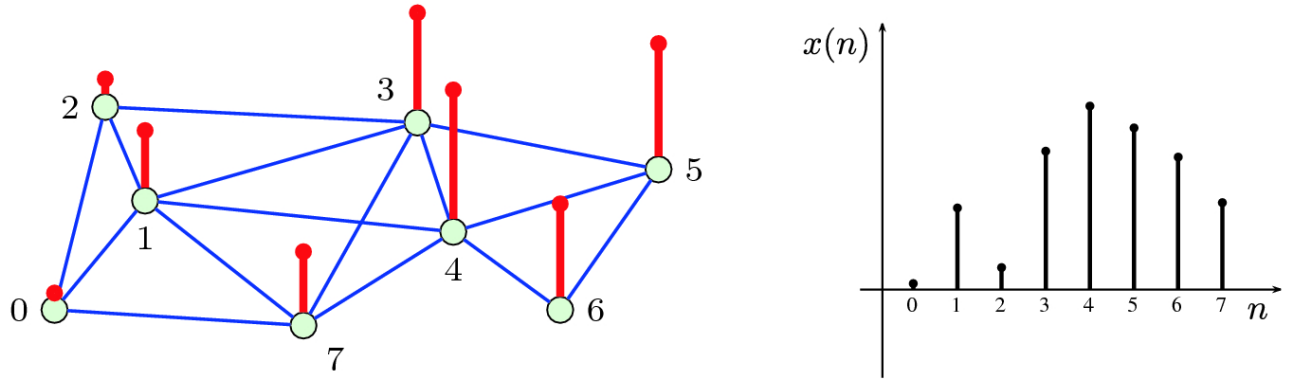
Laplacian Eigenvectors



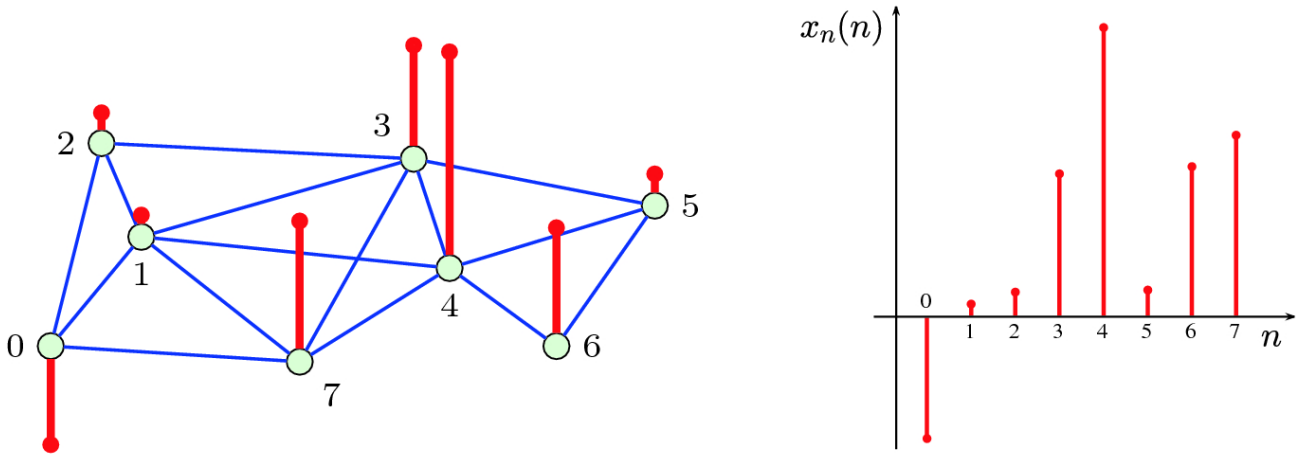
Subsampled Graph



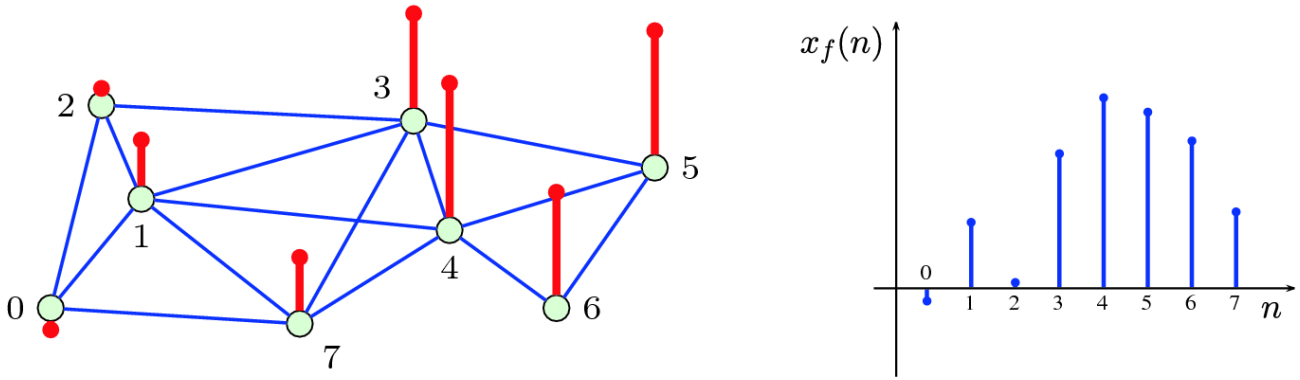
Spectral Graph Filtering



(a) original signal



(b) noisy signal



(c) filtered signal

Stankovic et al. 2018

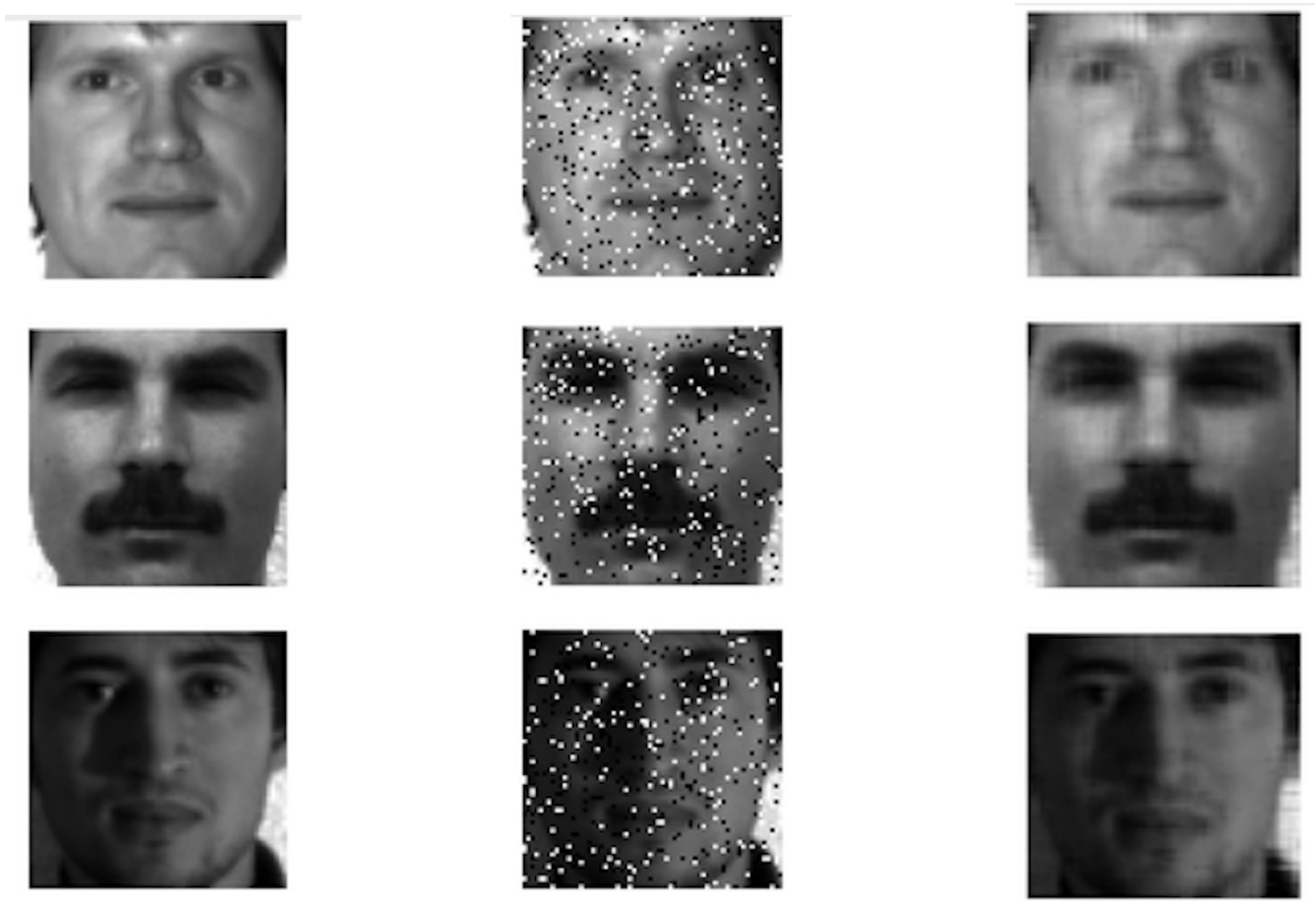
General Filter Construction

- Eigendecompose Graph Laplacian
- $$\begin{bmatrix} & L & \end{bmatrix} = \begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_3 \end{bmatrix} \begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix}^{-1}$$

- Modulate and alter eigenvalues
- $$\begin{bmatrix} & L & \end{bmatrix} = \begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} H(\lambda_1) & & \\ & H(\lambda_2) & \\ & & H(\lambda_3) \end{bmatrix} \begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix}^{-1}$$

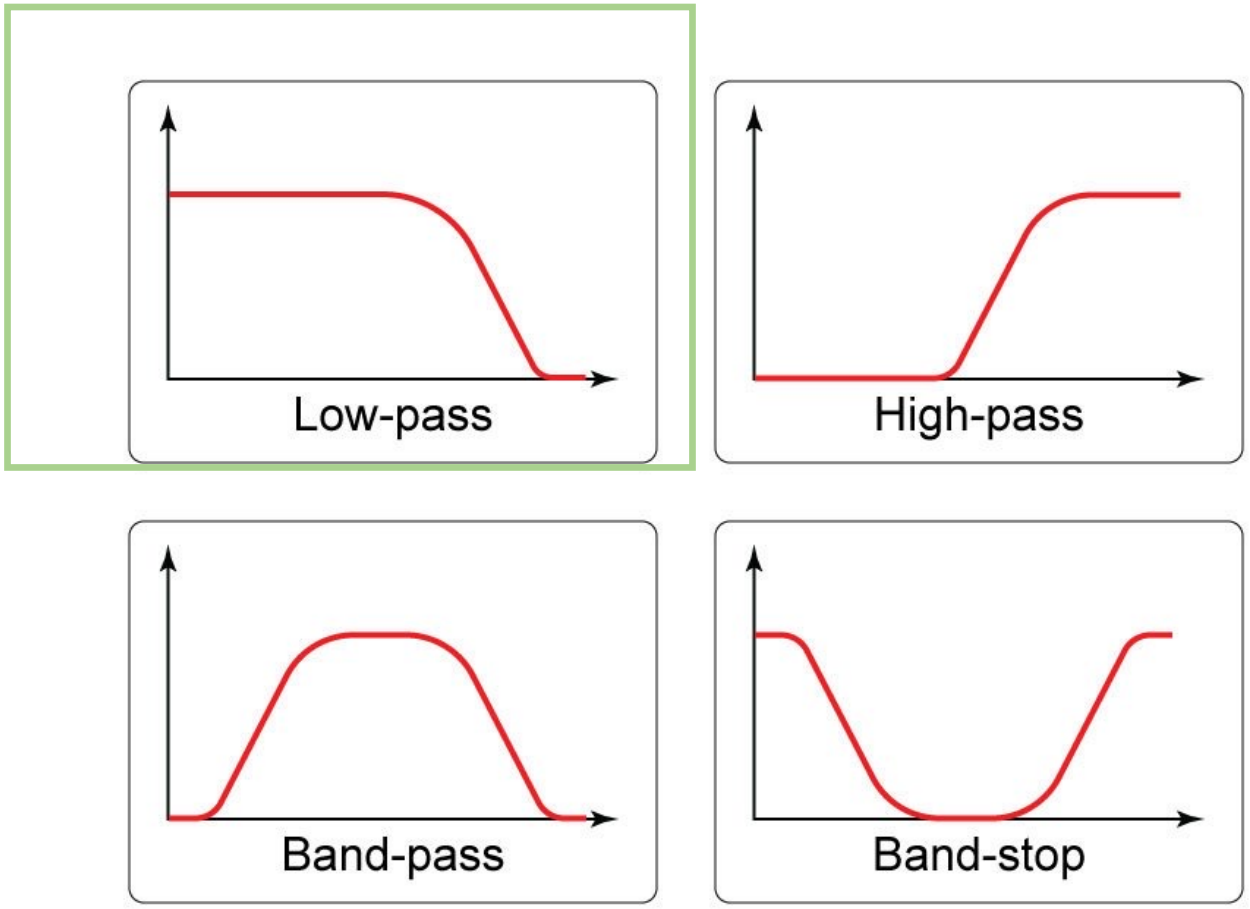
- Apply it to the signal $UH(\Lambda)U^T X$

Data signals are low frequency, Noisy can be high frequency



Can take off noise by taking off high frequency eigenvectors

Low-pass filter



Example of a low pass filter

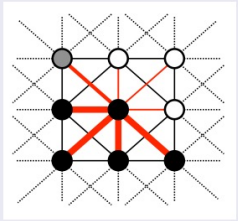
- $$\begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} H(\lambda_1) & & \\ & H(\lambda_2) & \\ & & H(\lambda_3) \end{bmatrix} \begin{bmatrix} | & | & | \\ U_1 & U_2 & U_3 \\ | & | & | \end{bmatrix}^{-1}$$

- $H(\lambda) = e^{-\lambda}$
- Applying $UH(\Lambda)U^T X$ would diminish high frequency components

Image Denoising

$$f(n) \longrightarrow \boxed{\text{GFT}} \longrightarrow \hat{f}(\lambda_\ell) \longrightarrow \boxed{\hat{g}} \longrightarrow \hat{g}(\lambda_\ell)\hat{f}(\lambda_\ell) \longrightarrow \boxed{\text{IGFT}} \longrightarrow \Phi f(n)$$

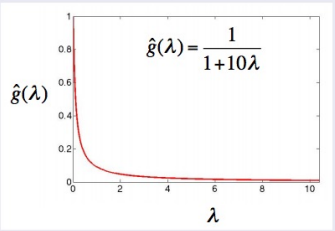
Semi-Local Graph

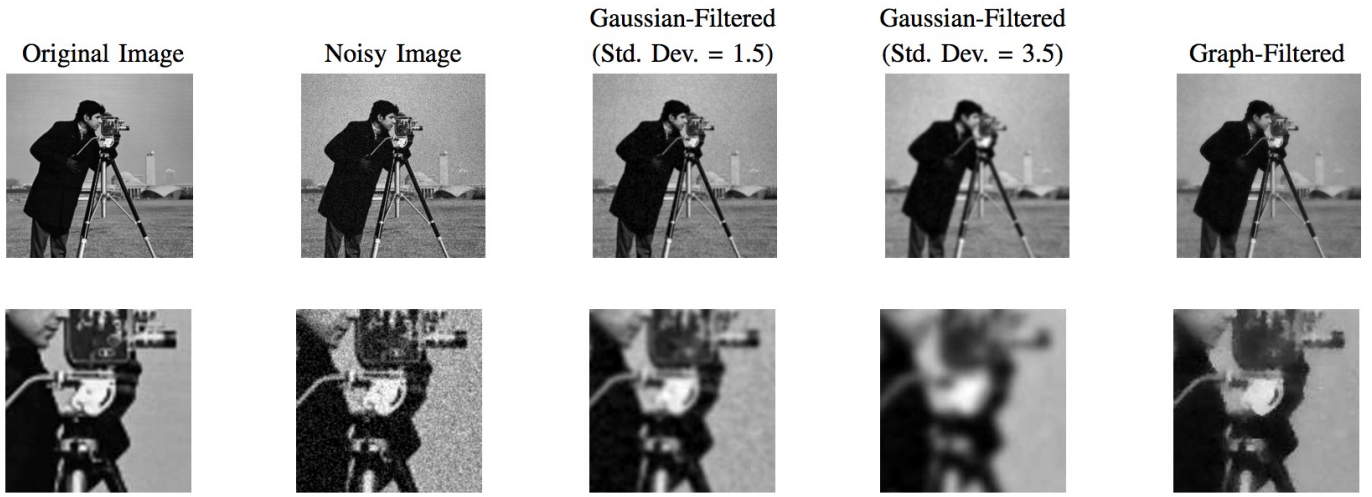


Tikhonov Regularization

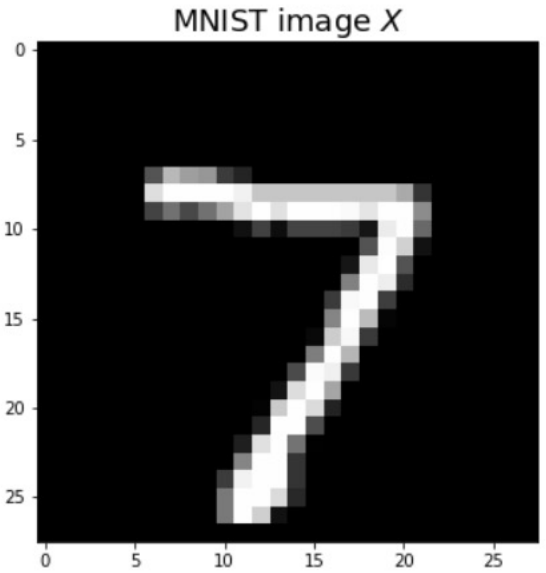
$$\underset{f}{\text{argmin}} \left\{ \|f - y\|_2^2 + \gamma f^{\mathrm{T}} \mathcal{L} f \right\}$$

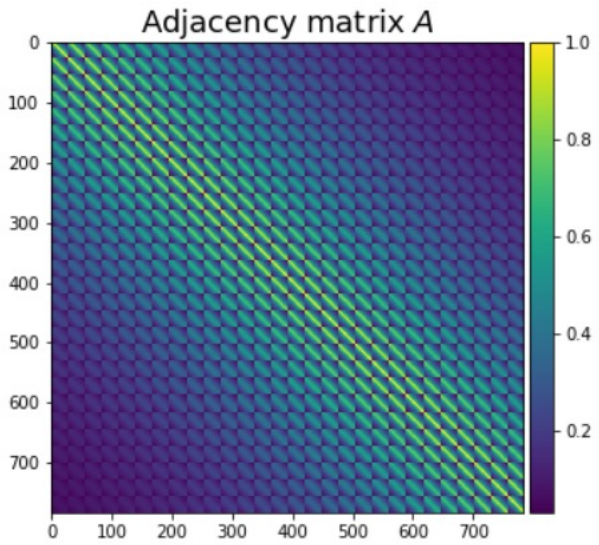
$$\Longrightarrow \hat{g}(\lambda_\ell) = \frac{1}{1 + \gamma \lambda_\ell}$$

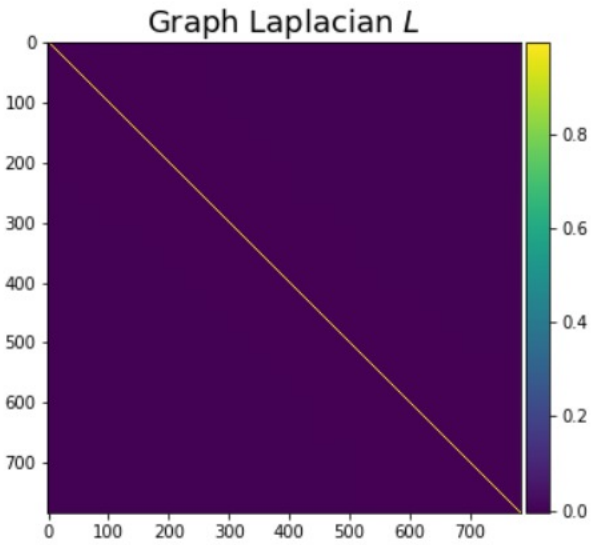




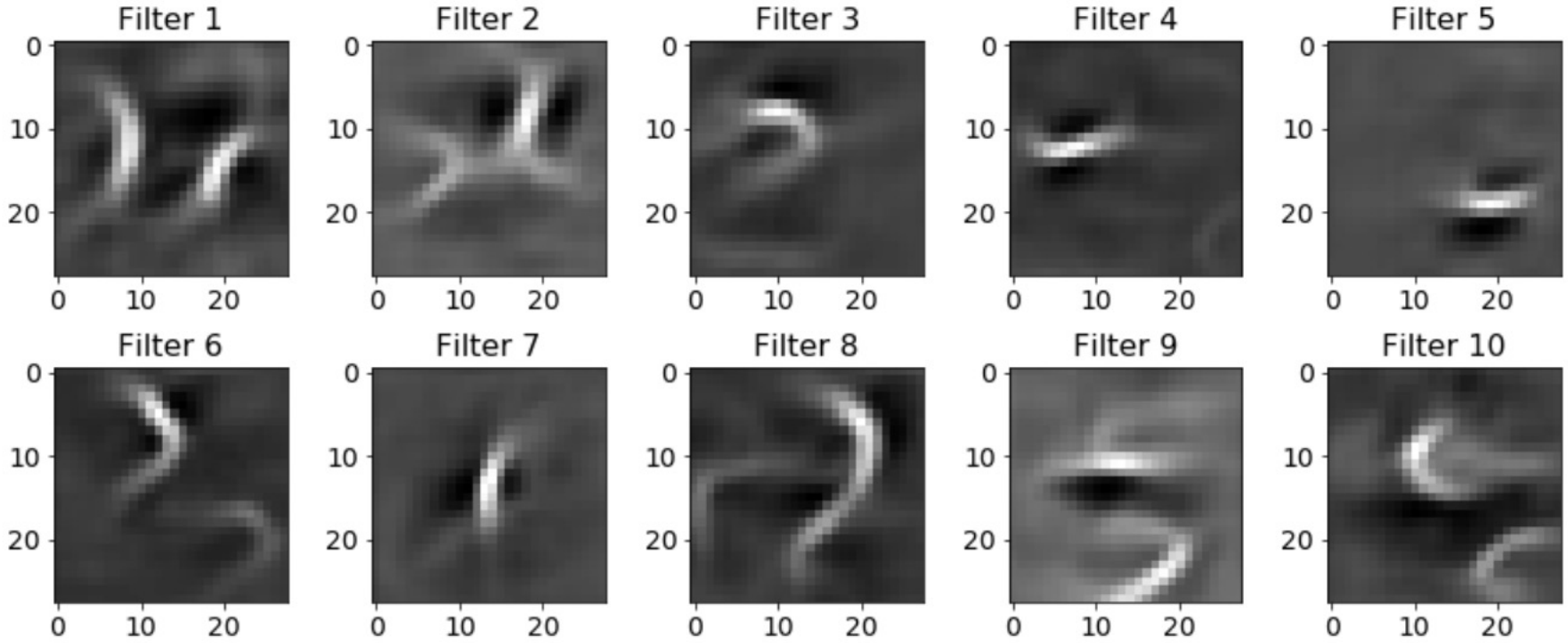
Graph Spectral Convolution MNIST







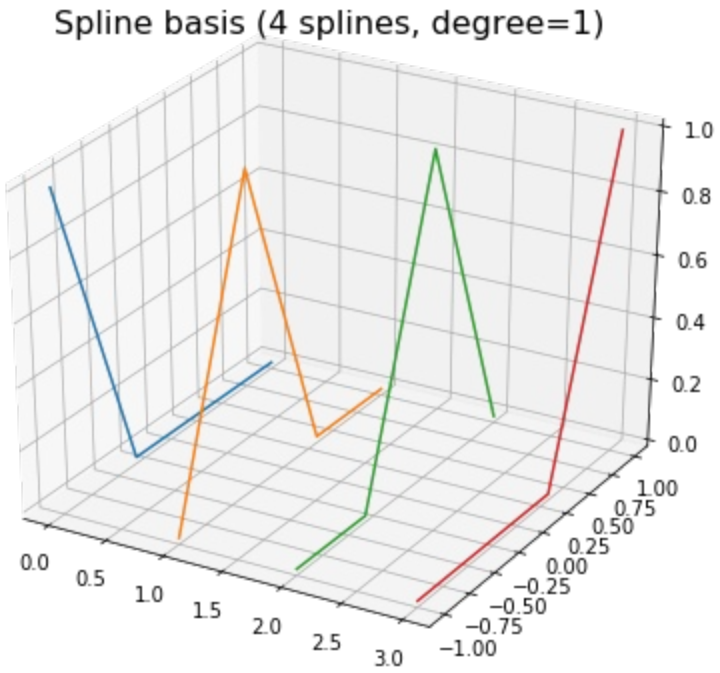
Learned Graph Spectral Filters

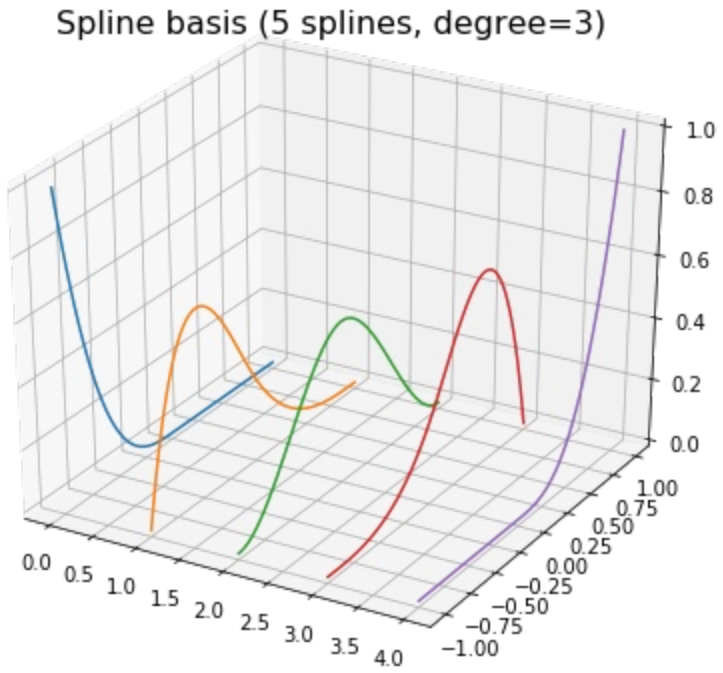


Problems

- **Problem 1**: The filters are no longer localized operations
  - They are operating globally based on frequency characteristics
- Fix: To localize in the spatial domain you need smoothness in the Fourier domain
  - Smoothness achieved by restricting filters to polynomial filters with K coefficients
  - K coefficients gives K-node localization!
- Fix 2: Use wavelet filters instead of fourier
  - Wavelets are localized in time and space
- **Problem 2:** Fourier transform is expensive, eigendecomposition of an NxN matrix each time
- Fix: Chebyshev approximation
- Fix2: Diffusion based operations

K filter coefficients





$$W_{\text{spectral}}^{(l)} \approx \sum_{k=1}^{K} \alpha_k f_k \qquad (4)$$

Chebyshev Polynomials

- Avoiding eigendecomposition which is expensive
- Chebyshev polynomials of the graph Laplacian defined recursively

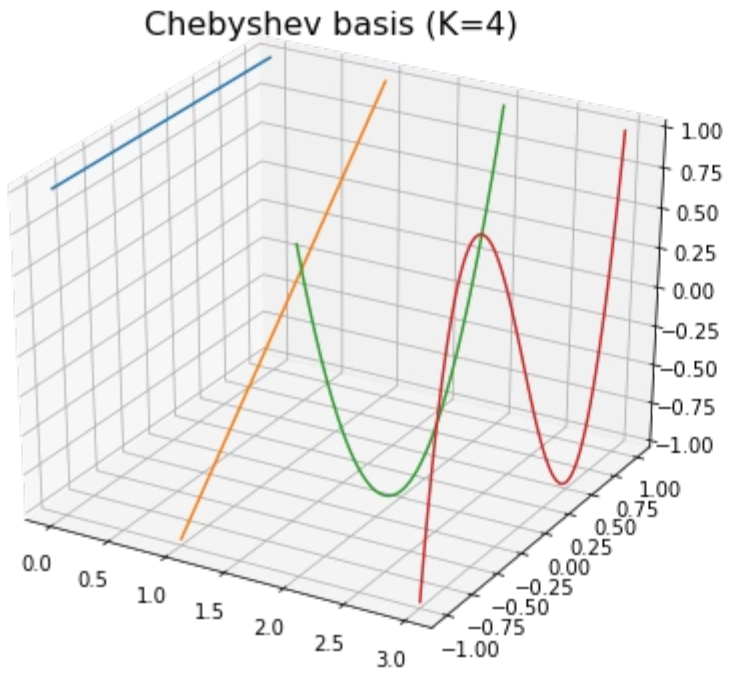
$$T_k(y) := \begin{cases} 1, & \text{if } k = 0 \\ y, & \text{if } k = 1 \\ 2yT_{k-1}(y) - T_{k-2}(y), & \text{if } k \geq 2 \end{cases}.$$

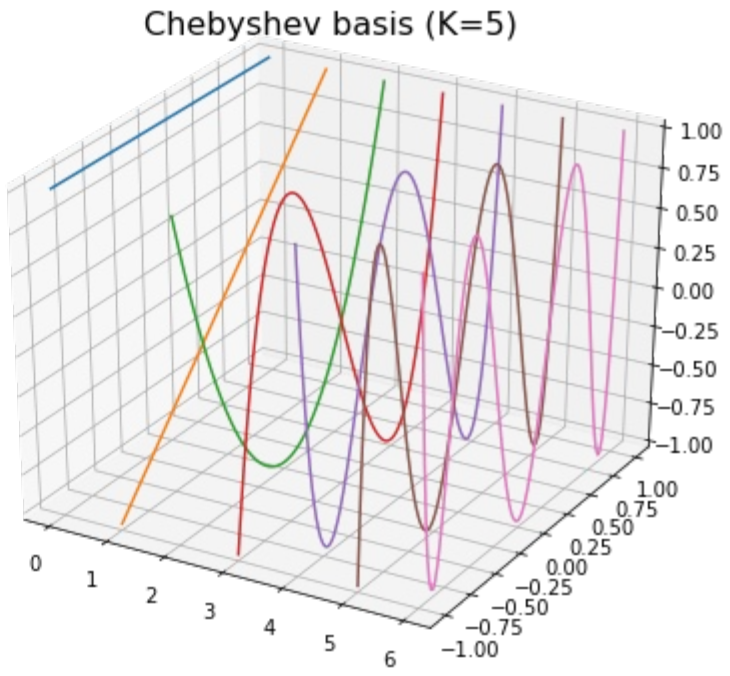
Original Spectral Filters $$y = g_\theta(L)x = g_\theta(U\Lambda U^T)x = Ug_\theta(\Lambda)U^T x.$$

Polynomial Filters $$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k,$$

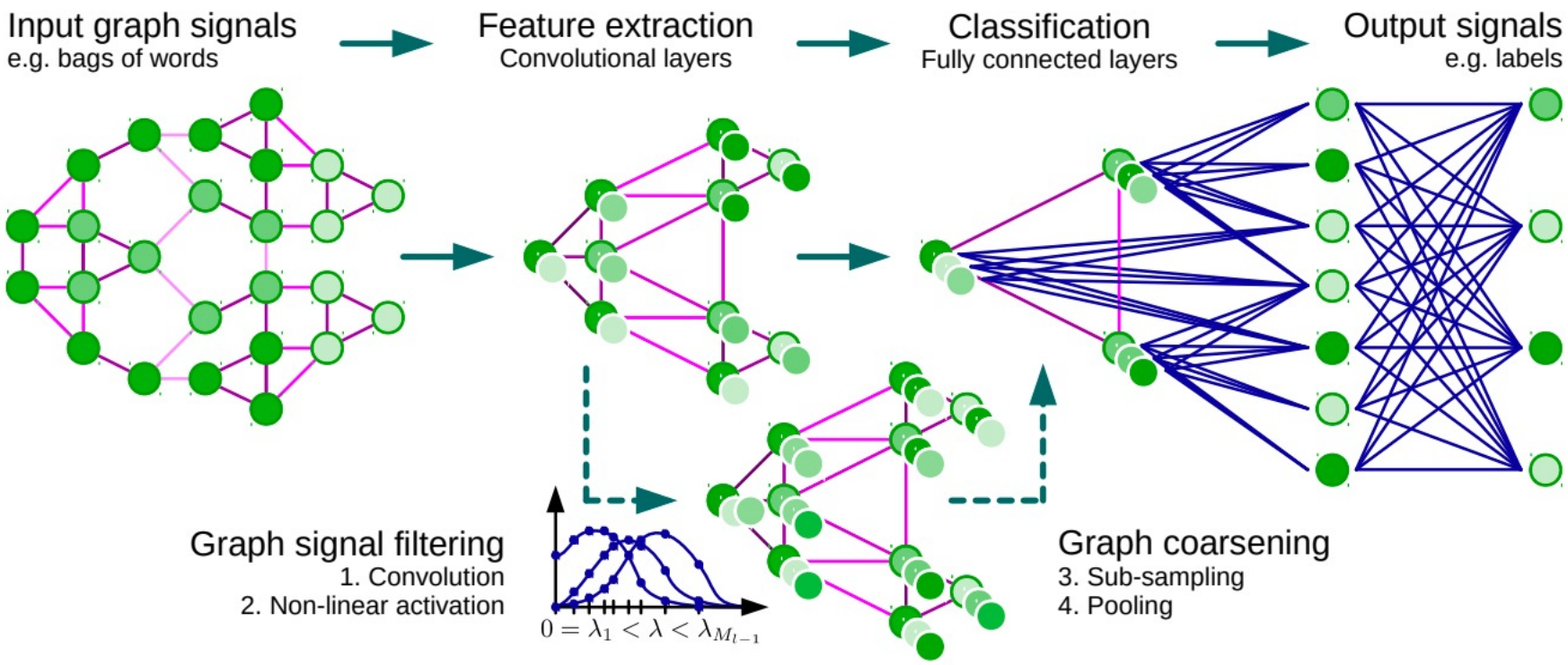
Chebyshev Filters $$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}),$$

Chebyshev Basis





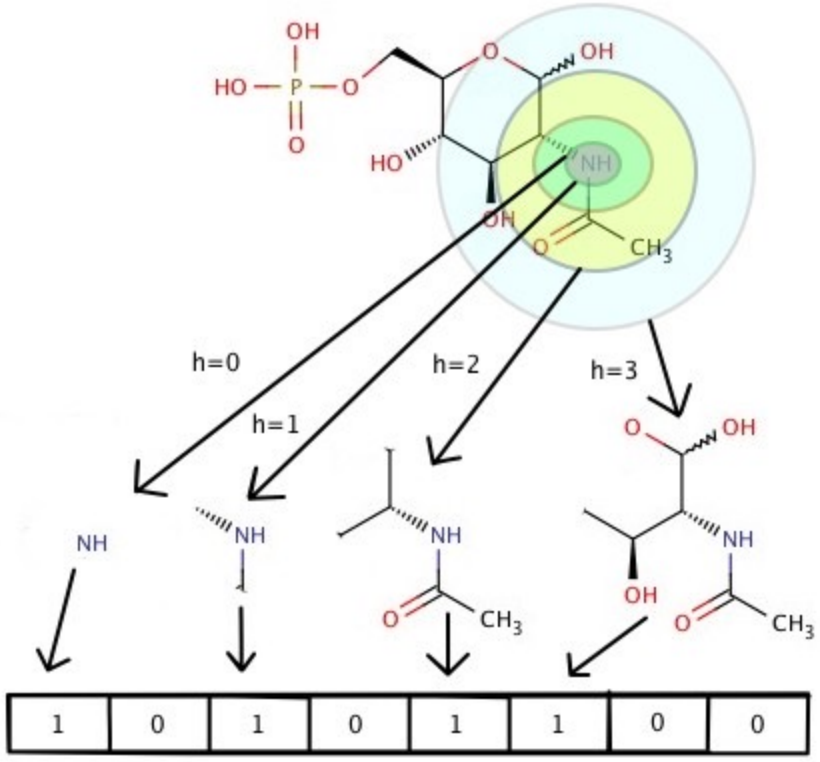
Summarized Architecture of GCN

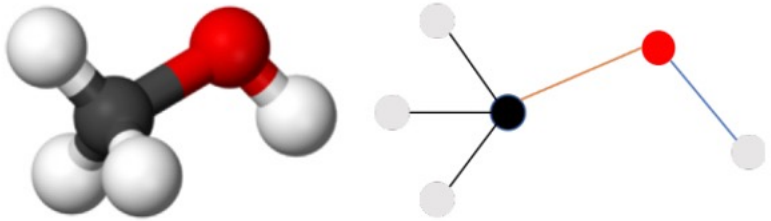


Kipf et al.

Applications

- Predicting Molecular properties by encoding molecules as graphs





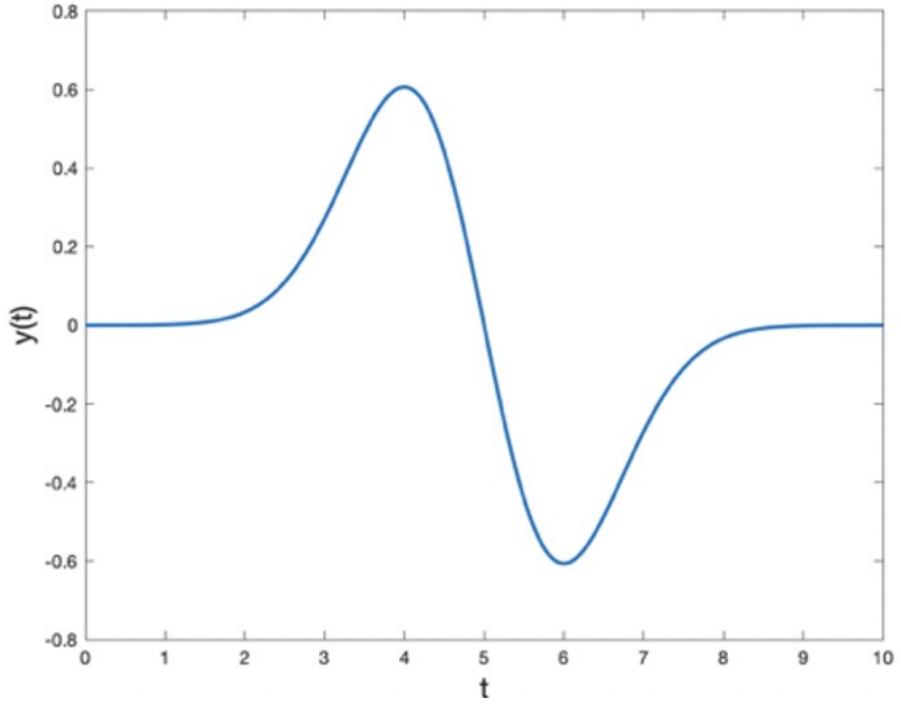
(b) molecule

Global vs Local Frequency Domain Analysis

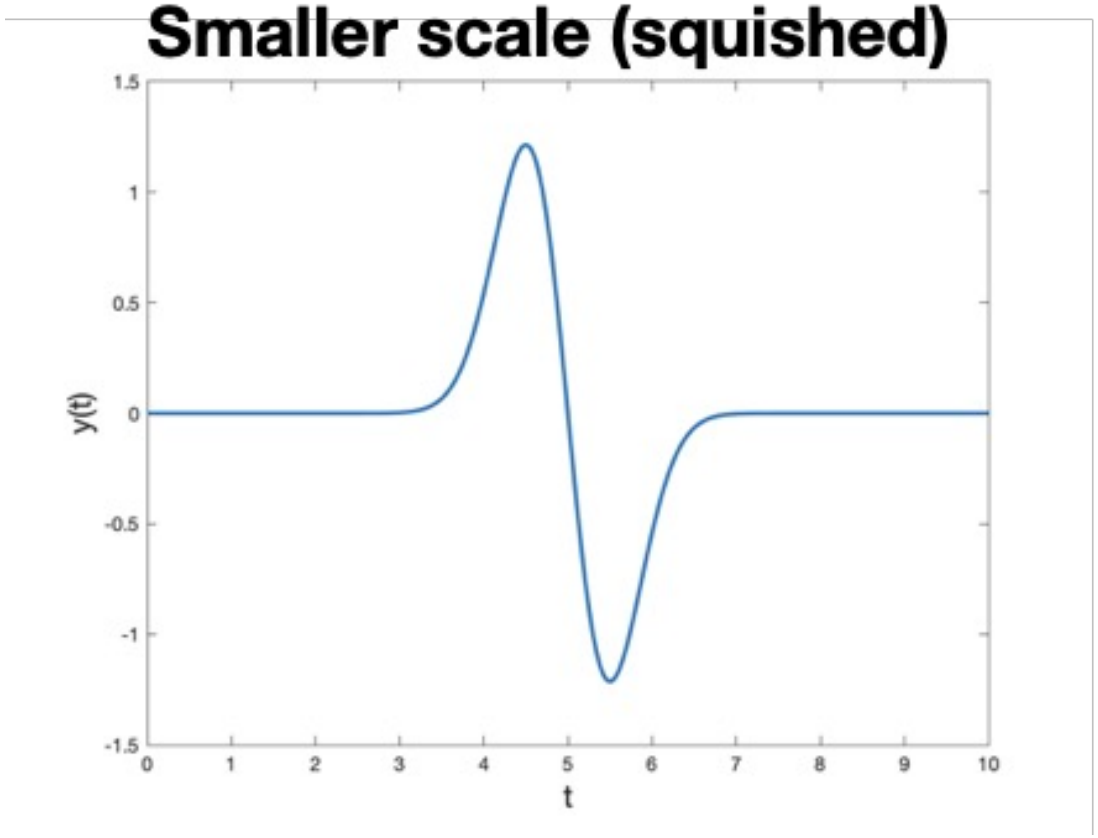
- Fourier transforms capture global frequency information, frequencies that persist over the entire length of the signal
- What about small bursts of frequency changes?
- Uneven frequency distributions over time?
- These might benefit from a more localized frequency domain analysis

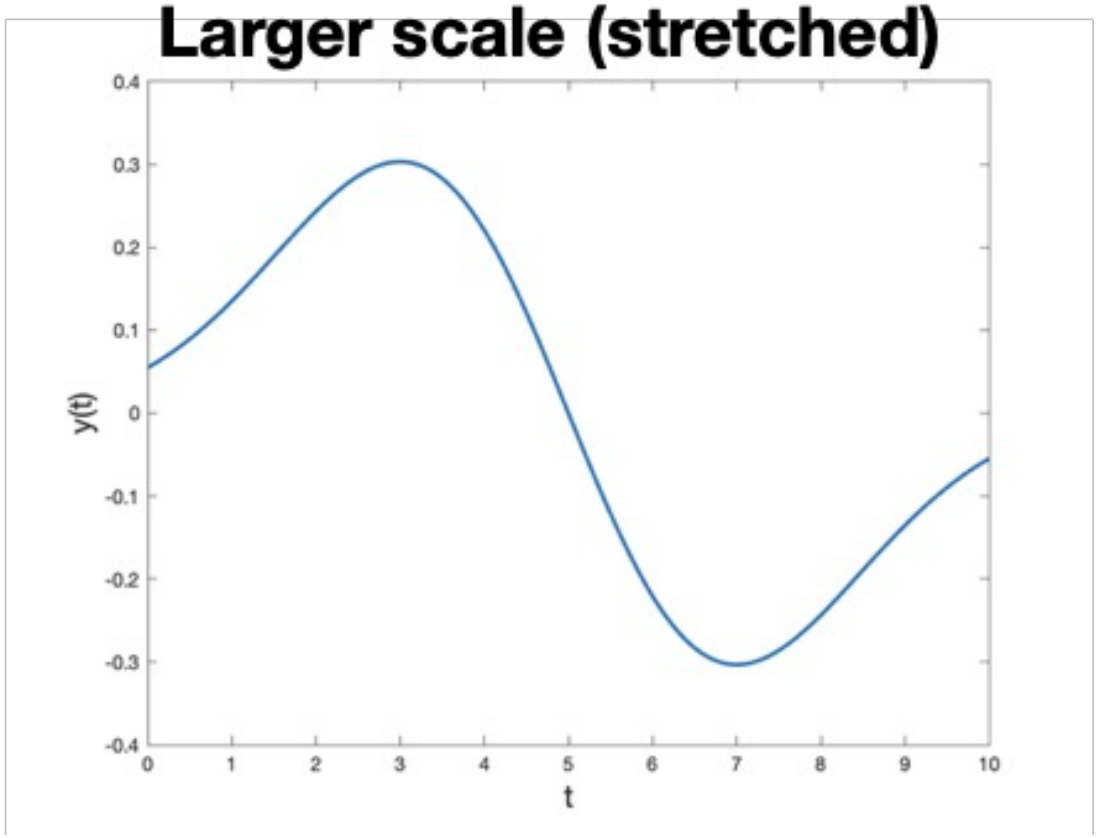
Wavelets

- Unlike a sine or cosine function a wavelet is a localized oscillation

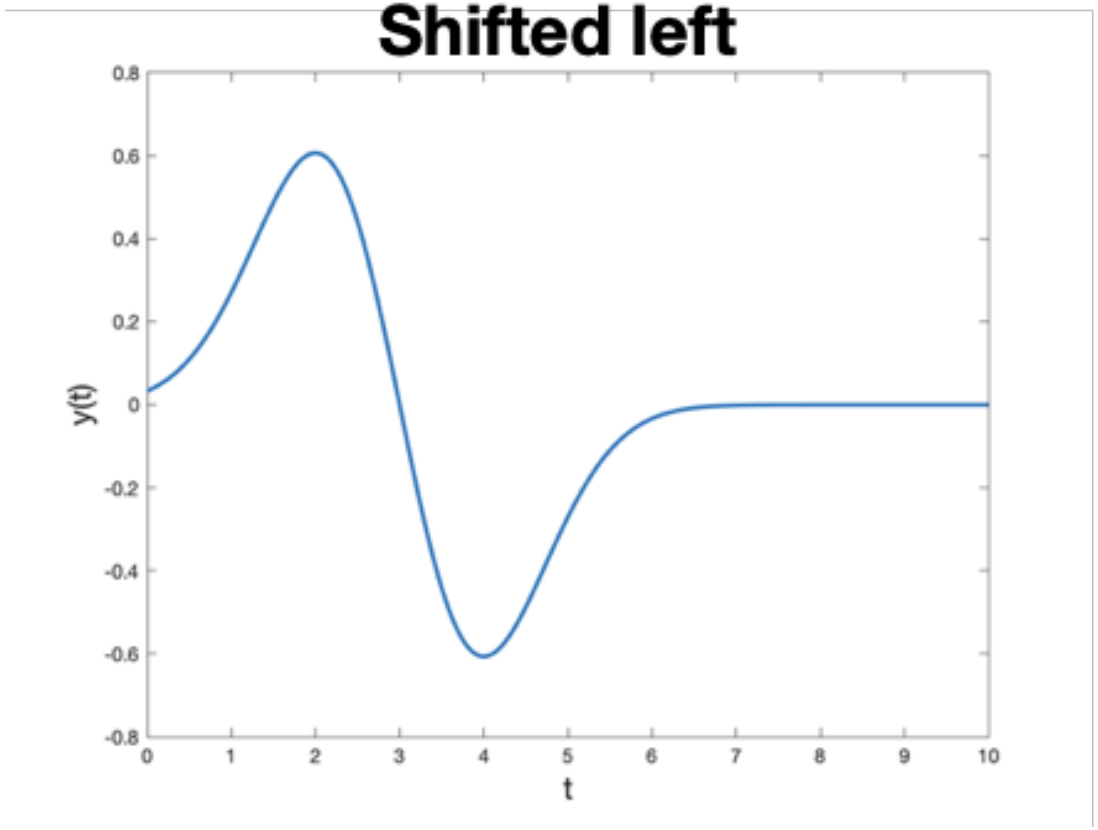


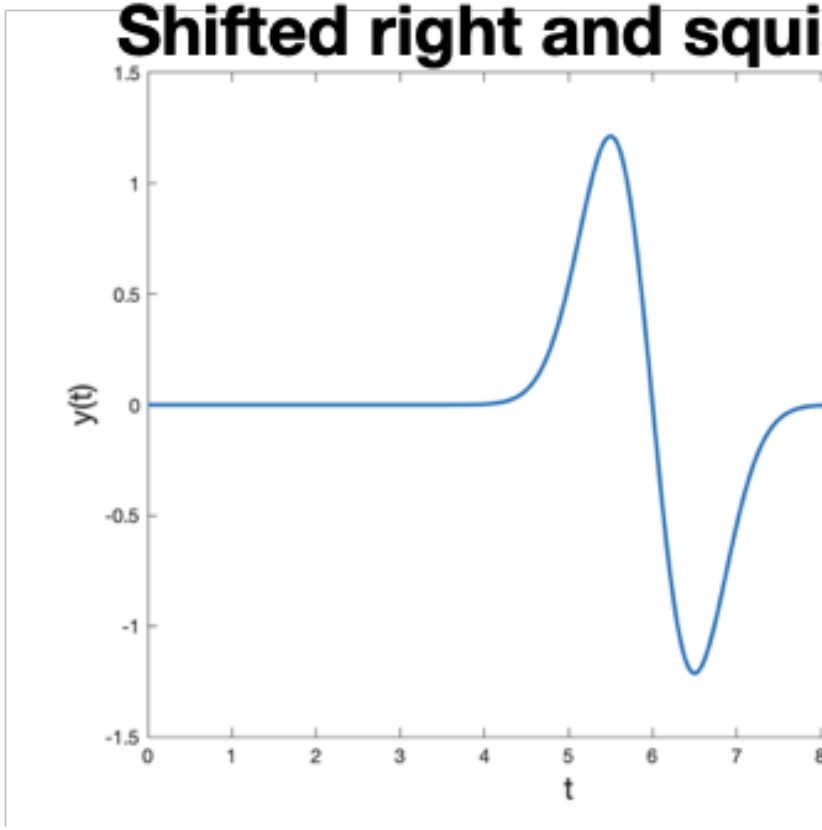
Scaling wavelets



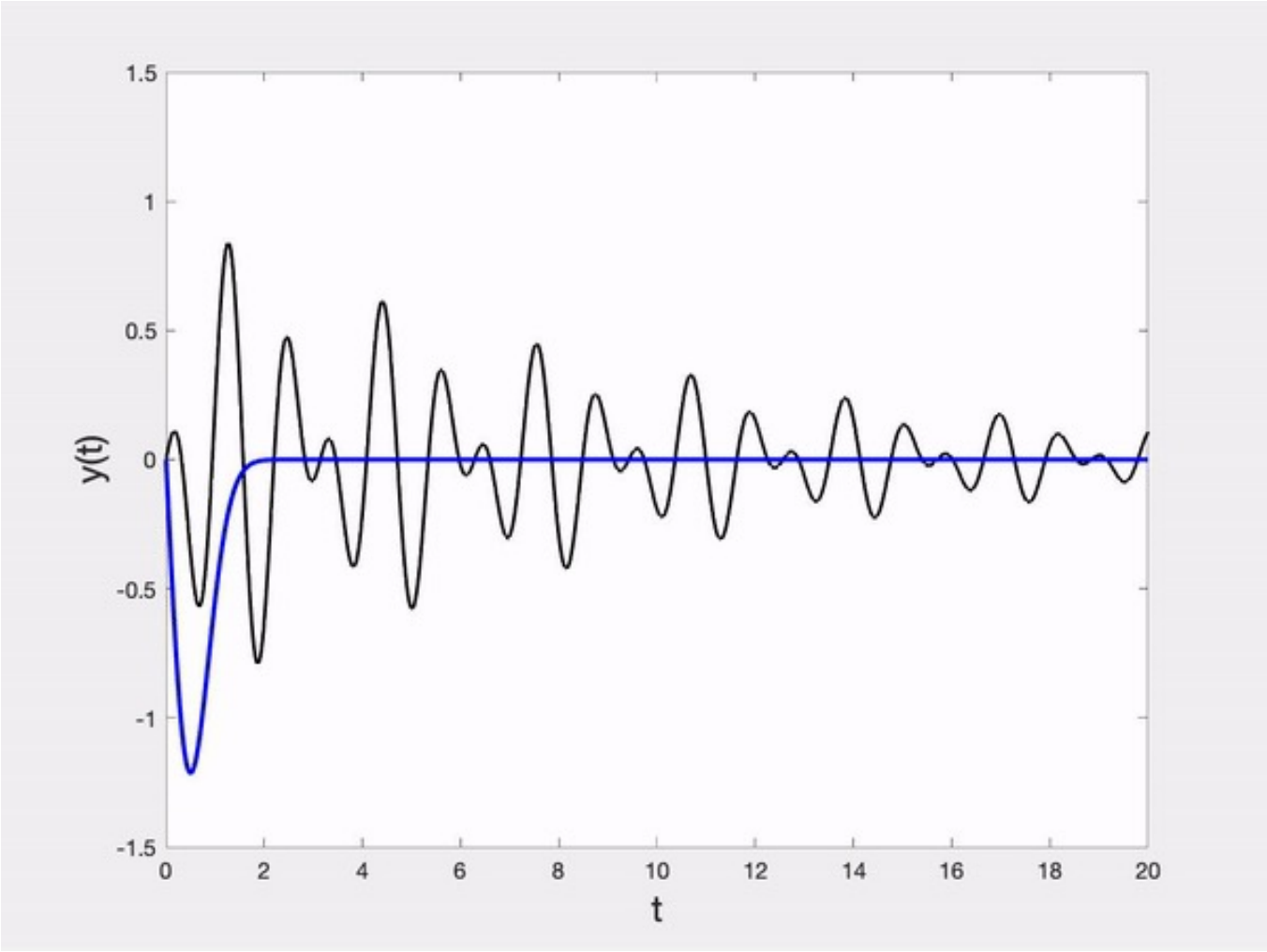


Shifting wavelets in time





Discrete Wavelet Transform



Wavelet Transform

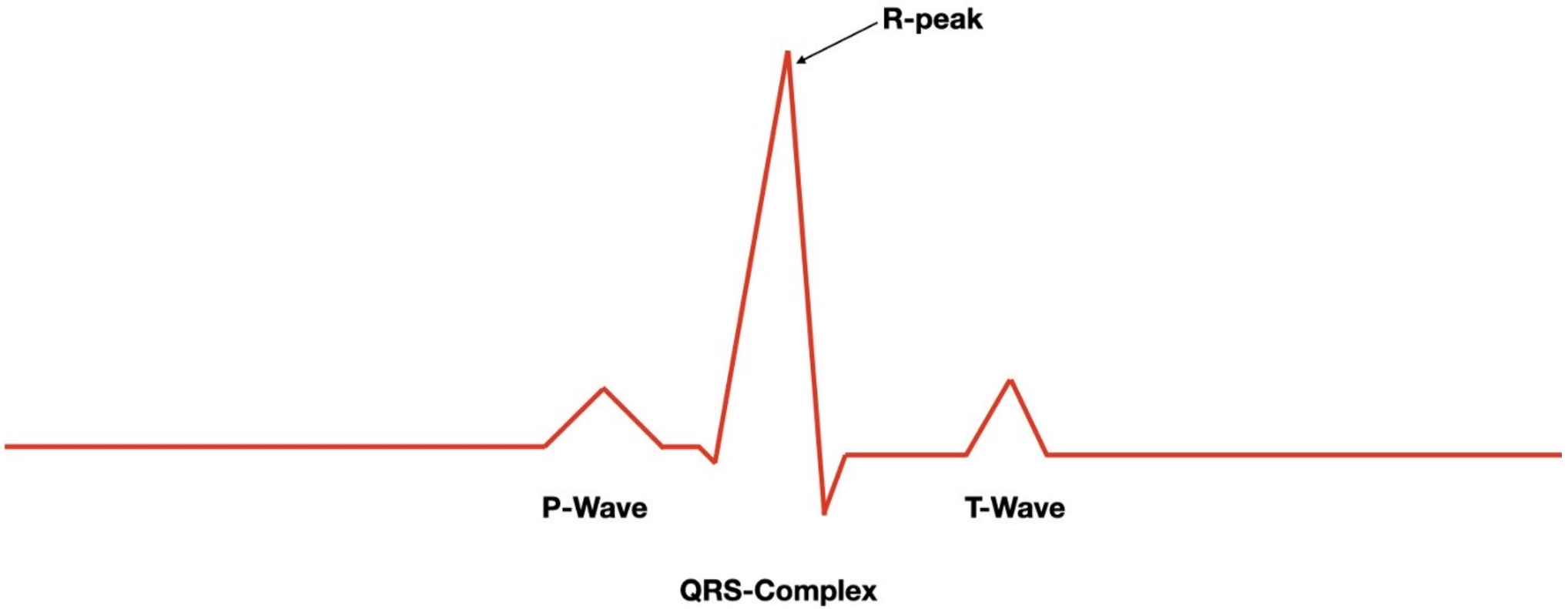
Continuous Wavelet Transform (CWT)

$$T(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\, \psi^* \frac{(t-b)}{a} dt$$

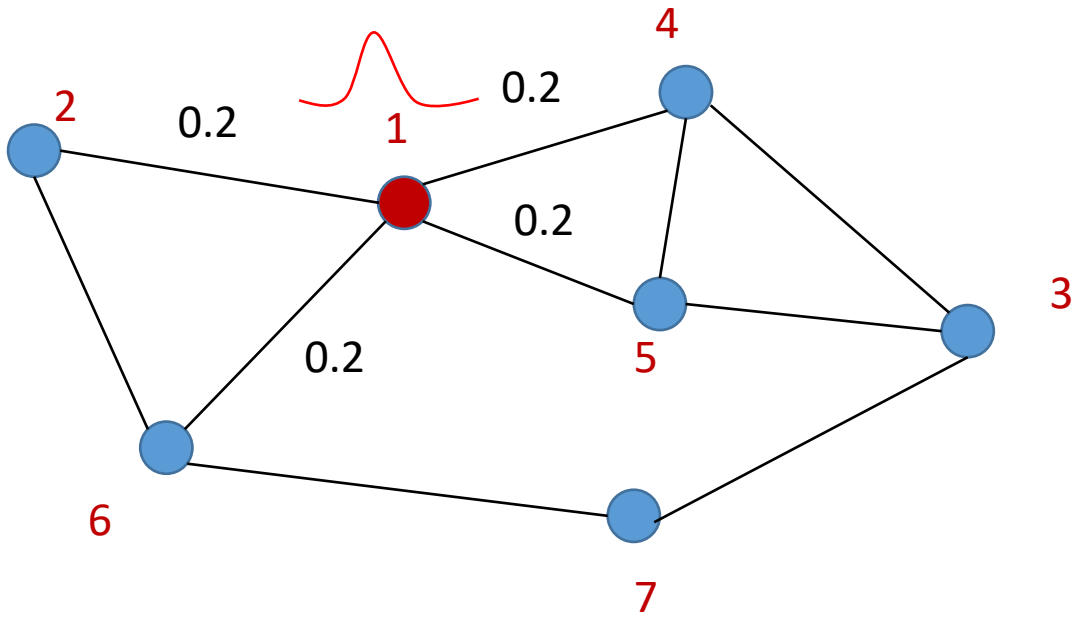
Discrete Wavelet Transform (DWT)

$$T_{m,n} = \int_{-\infty}^{\infty} x(t)\, \psi_{m,n}(t)\, dt$$

Sample Application peak detection in EEG



Diffusion Operator



.2	.2	0	.2	.2	.2	0
.333	.333	0	0	0	.333	0
0	0	.25	.25	.25	0	.25
.25	0	.25	.25	.25	0	0
.25	0	.25	.25	.25	0	0
.25	.25	0	0	0	.25	.25
0	0	.333	0	0	.333	.33

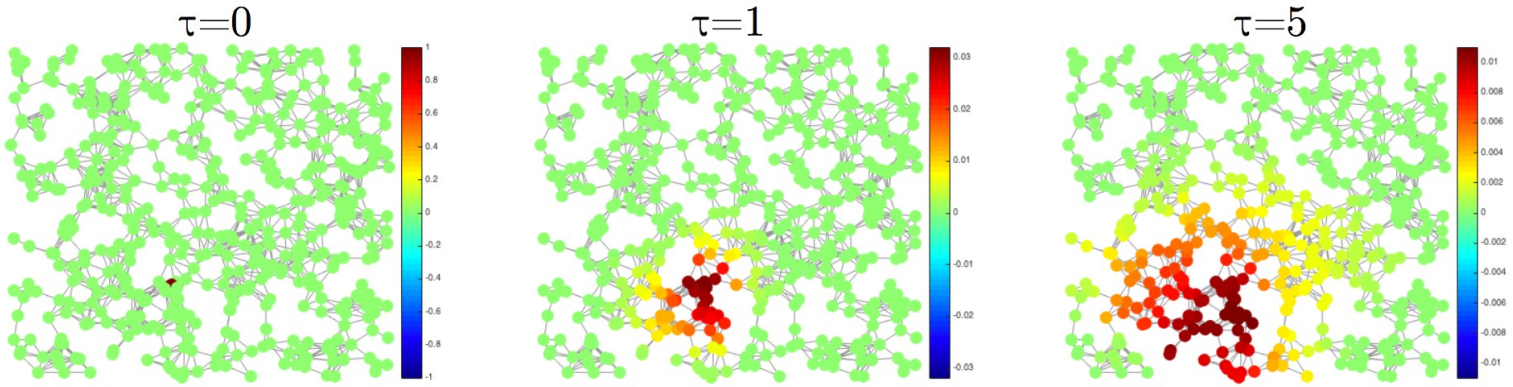
$$P = D^{-1}A$$

Same eigenvectors as the graph laplacian

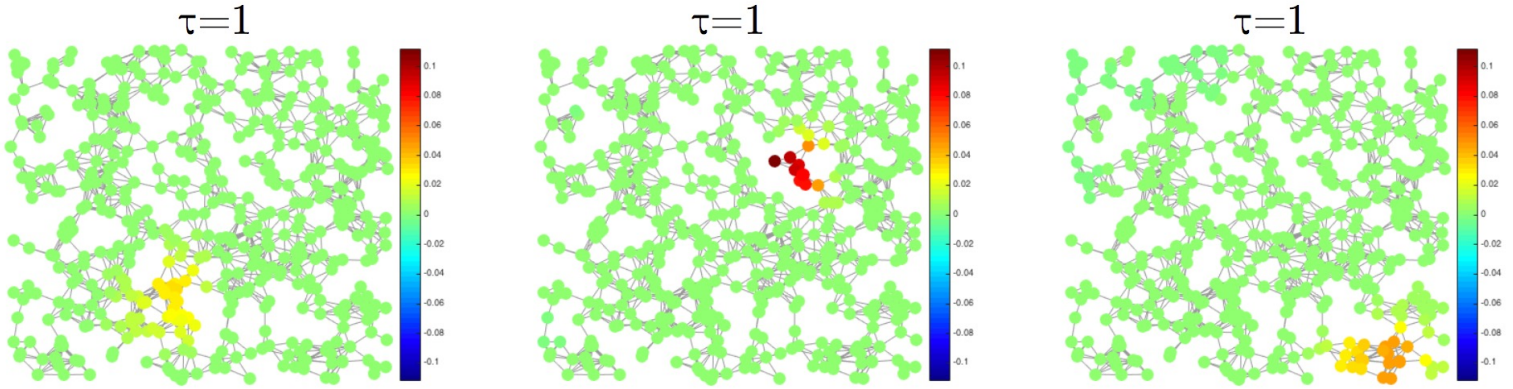
Diffusions can be used to create wavelets

- Diffusion Atoms
- P^t s gives diffused version of S
- Diffusion based filters don't require eigendecomposition

- Start with a unit of energy at a single vertex and let it diffuse:



- How much it diffuses over a fixed time depends on the graph structure:



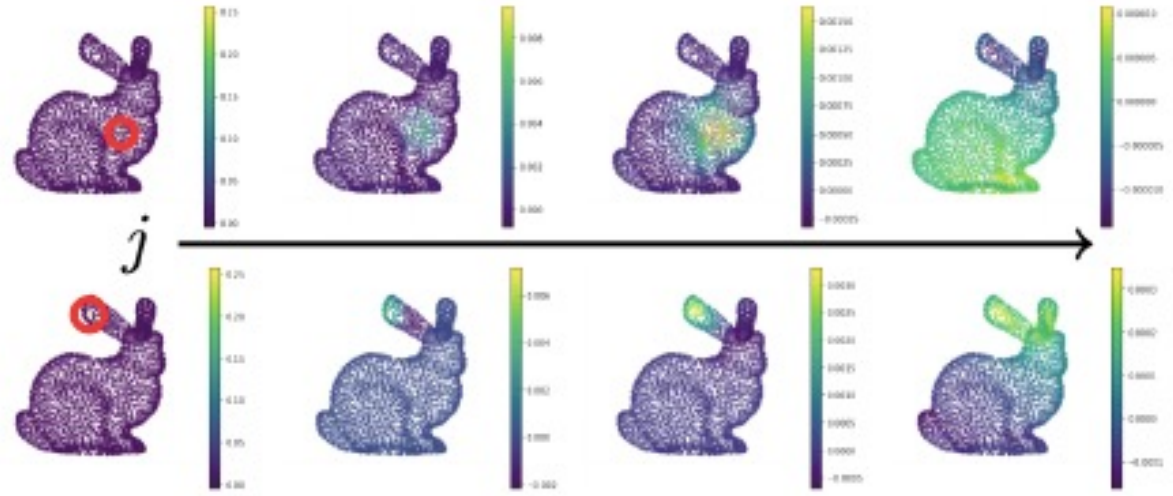
Coifman and Lafon, Diffusion maps, ACHA, 2006

Diffusion wavelets: Differences between lazy Random walks

D

$$\mathbf{P} = \frac{1}{2}\left(\mathbf{I} + \mathbf{A}\mathbf{D}^{-1}\right)$$

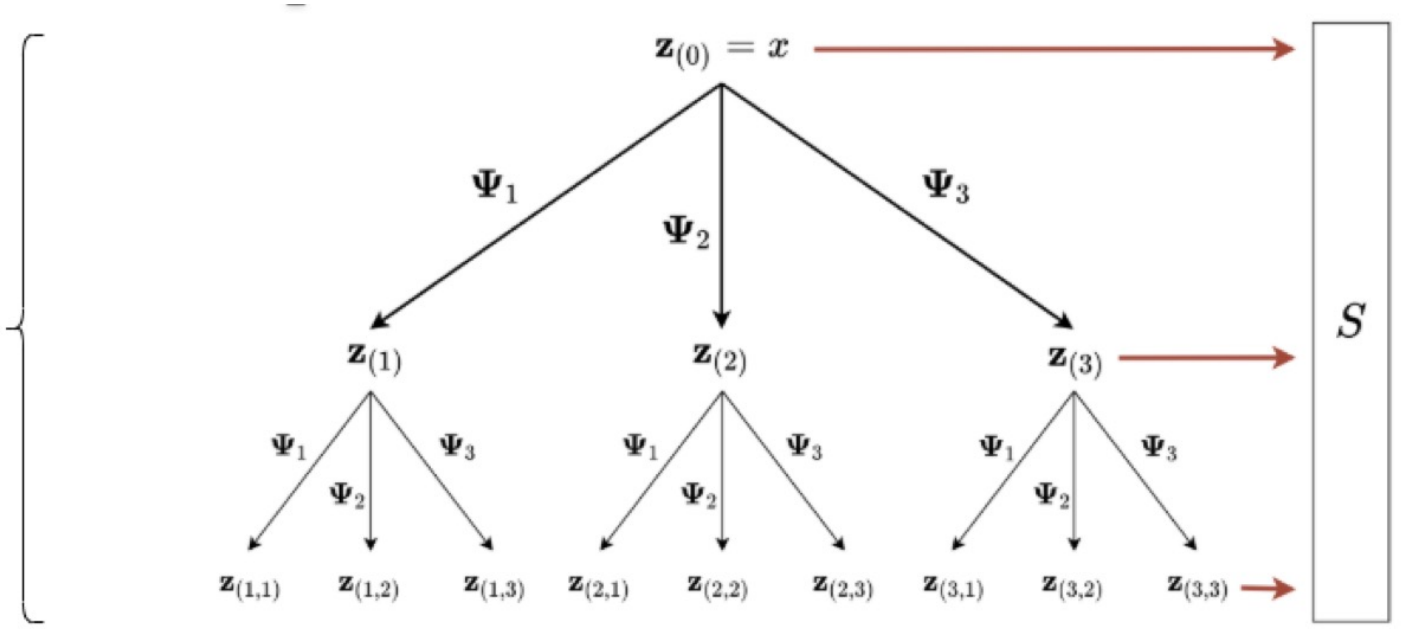
$$\boldsymbol{\Psi}_j = \mathbf{P}^{2^{j-1}} - \mathbf{P}^{2^j} = \mathbf{P}^{2^{j-1}}\left(\boldsymbol{I} - \mathbf{P}^{2^{j-1}}\right)$$



Coifman and Maggioni

Graph Scattering transform

- Multiscale Wavelet transform (ψ)
- Coefficients collected via a non-linearity (S)
- Hierarchically constructed



Gao et al. ICML 2019
Gama et al. NeurIPS 2019

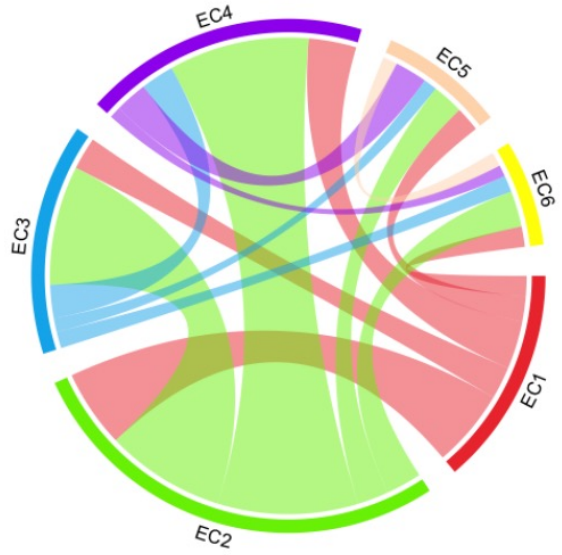
Graph Classification

	COLLAB	IMDB-B	IMDB-M	REDDIT-B	REDDIT-5K	REDDIT-12K	
WL	77.82 ± 1.45	71.60 ± 5.16	N/A	78.52 ± 2.01	50.77 ± 2.02	34.57 ± 1.32	Graph kernel
Graphlet	73.42 ± 2.43	65.40 ± 5.95	N/A	77.26 ± 2.34	39.75 ± 1.36	25.98 ± 1.29	
WL-OA	80.70 ± 0.10	N/A	N/A	89.30 ± 0.30	N/A	N/A	
DGK	73.00 ± 0.20	66.90 ± 0.50	44.50 ± 0.50	78.00 ± 0.30	41.20 ± 0.10	32.20 ± 0.10	
DGCNN	73.76 ± 0.49	70.03 ± 0.86	47.83 ± 0.85	N/A	48.70 ± 4.54	N/A	Deep learning
2D CNN	71.33 ± 1.96	70.40 ± 3.85	N/A	89.12 ± 1.70	52.21 ± 2.44	48.13 ± 1.47	
PSCN ($k = 10$)	72.60 ± 2.15	71.00 ± 2.29	45.23 ± 2.84	86.30 ± 1.58	49.10 ± 0.70	41.32 ± 0.42	
GCAPS-CNN	77.71 ± 2.51	71.69 ± 3.40	48.50 ± 4.10	87.61 ± 2.51	50.10 ± 1.72	N/A	
S2S-P2P-NN	81.75 ± 0.80	73.80 ± 0.70	51.19 ± 0.50	86.50 ± 0.80	52.28 ± 0.50	42.47 ± 0.10	
GIN-0 (MLP-SUM)	80.20 ± 1.90	75.10 ± 5.10	52.30 ± 2.80	92.40 ± 2.50	57.50 ± 1.50	N/A	
GS-SVM	79.94 ± 1.61	71.20 ± 3.25	48.73 ± 2.32	89.65 ± 1.94	53.33 ± 1.37	45.23 ± 1.25	

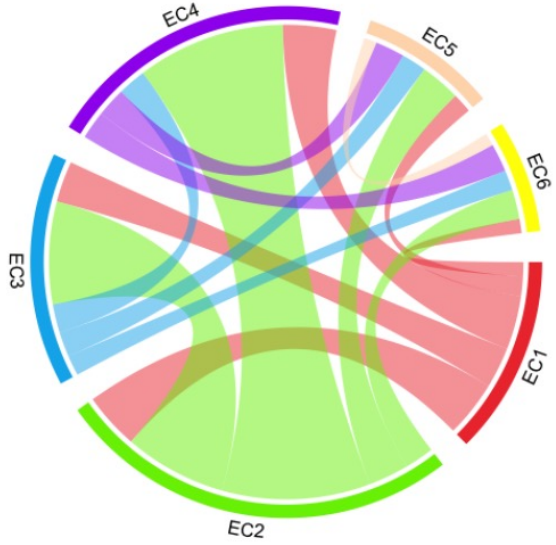
Graph Classification

	COLLAB	IMDB-B	IMDB-M	REDDIT-B	REDDIT-5K	REDDIT-12K	
WL	77.82 ± 1.45	71.60 ± 5.16	N/A	78.52 ± 2.01	50.77 ± 2.02	34.57 ± 1.32	Graph kernel
Graphlet	73.42 ± 2.43	65.40 ± 5.95	N/A	77.26 ± 2.34	39.75 ± 1.36	25.98 ± 1.29	
WL-OA	80.70 ± 0.10	N/A	N/A	89.30 ± 0.30	N/A	N/A	
DGK	73.00 ± 0.20	66.90 ± 0.50	44.50 ± 0.50	78.00 ± 0.30	41.20 ± 0.10	32.20 ± 0.10	
DGCNN	73.76 ± 0.49	70.03 ± 0.86	47.83 ± 0.85	N/A	48.70 ± 4.54	N/A	Deep learning
2D CNN	71.33 ± 1.96	70.40 ± 3.85	N/A	89.12 ± 1.70	52.21 ± 2.44	48.13 ± 1.47	
PSCN ($k = 10$)	72.60 ± 2.15	71.00 ± 2.29	45.23 ± 2.84	86.30 ± 1.58	49.10 ± 0.70	41.32 ± 0.42	
GCAPS-CNN	77.71 ± 2.51	71.69 ± 3.40	48.50 ± 4.10	87.61 ± 2.51	50.10 ± 1.72	N/A	
S2S-P2P-NN	81.75 ± 0.80	73.80 ± 0.70	51.19 ± 0.50	86.50 ± 0.80	52.28 ± 0.50	42.47 ± 0.10	
GIN-0 (MLP-SUM)	80.20 ± 1.90	75.10 ± 5.10	52.30 ± 2.80	92.40 ± 2.50	57.50 ± 1.50	N/A	
GS-SVM	79.94 ± 1.61	71.20 ± 3.25	48.73 ± 2.32	89.65 ± 1.94	53.33 ± 1.37	45.23 ± 1.25	

Embedding Analysis

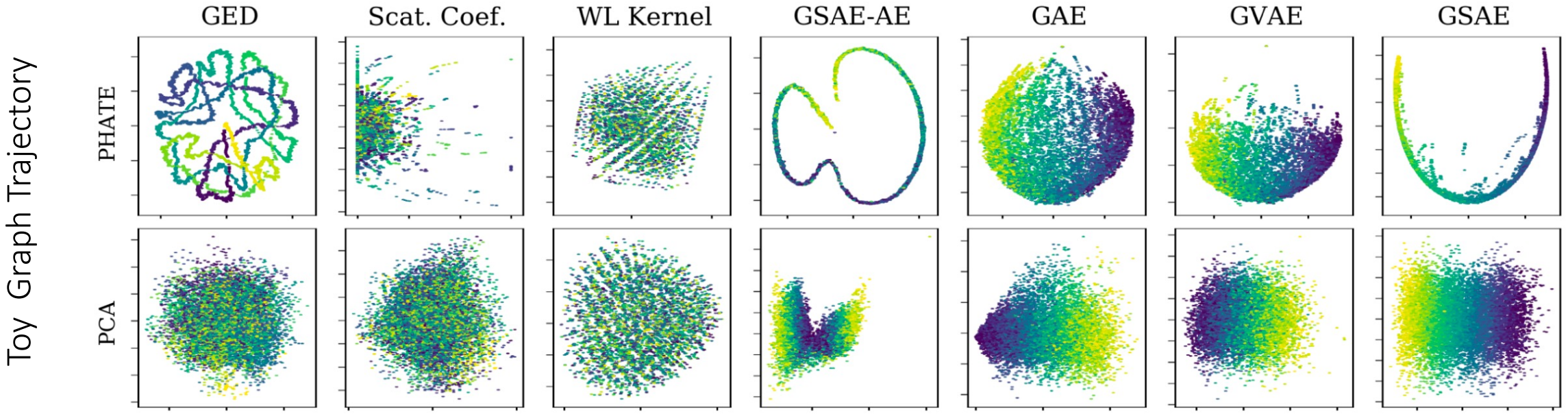


(a) Observed



(b) Inferred

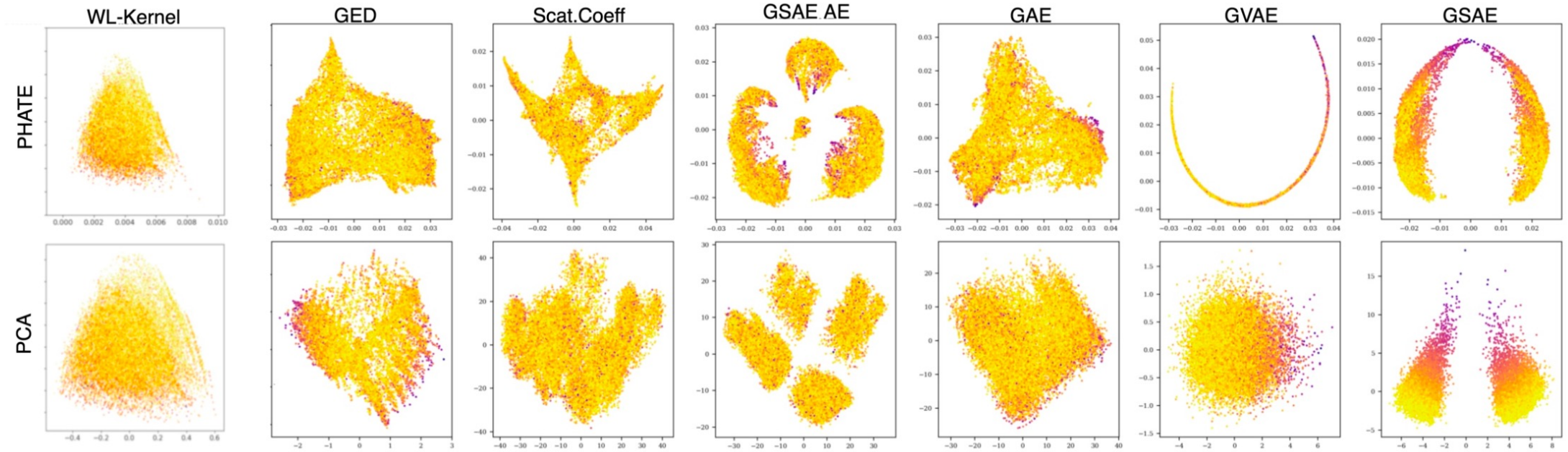
Graph Scattering Autoencoder Embeddings



Castro et al. 2019

Bistable RNA Structure

SEQ3



Reading List

- Defferrard et al. Convolutional Neural Networks on Graphs

- Kipf & Welling Semisupervised Graph Classification

- https://towardsdatascience.com/spectral-graph-convolution-explained-and-implemented-step-by-step-2e495b57f801
- Graph wavelet neural network https://arxiv.org/pdf/1904.07785.pdf

- Graph scattering transform: https://proceedings.mlr.press/v97/gao19e.html
- https://proceedings.neurips.cc/paper/2019/file/3ce3bd7d63a2c9c81983cc8e9bd02ae5-Paper.pdf

- Graph scattering Autoencoder: https://arxiv.org/abs/2006.06885
- Learnable scattering network: https://arxiv.org/abs/2010.02415

- Diffusion wavelets: https://www.sciencedirect.com/science/article/pii/S106352030600056X