GNNs: Overview

Outline

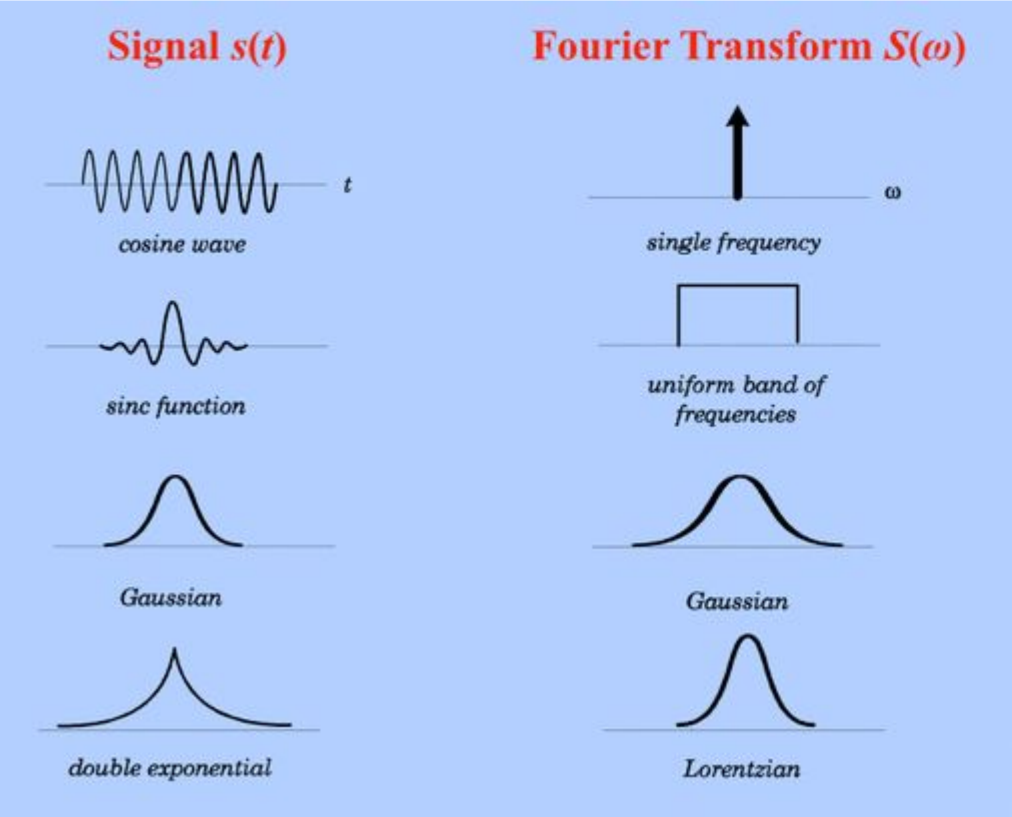
1). The function space of GNNs

2). The basics of GNNs

3). Applications

Signal Processing

- A signal in time (length N), Can be represented by a discrete vector, $x \in \mathbb{R}^N$



https://mriquestions.com/fourier-transform-ft.html

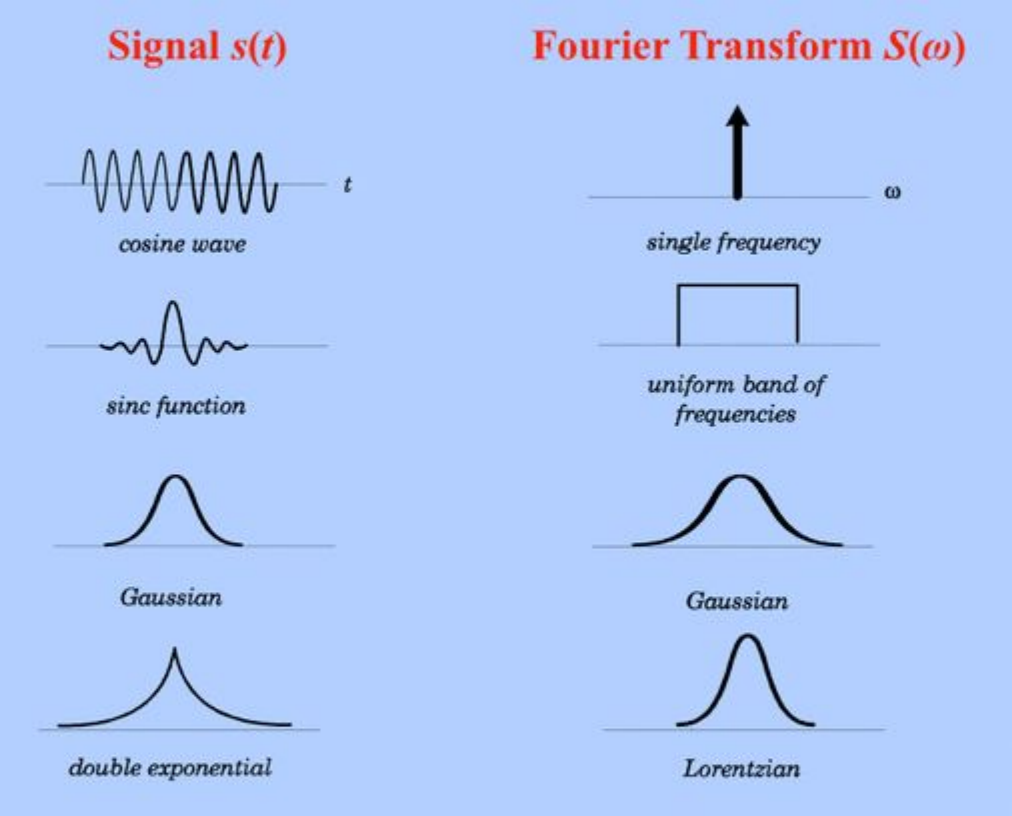
Signal Processing

- A signal in time (length N), Can be represented by a discrete vector, $x \in \mathbb{R}^N$

Fourier transform

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i \xi x}\, dx. \quad \textbf{(Eq.1)}$$

wikipedia



https://mriquestions.com/fourier-transform-ft.html

Signal Processing

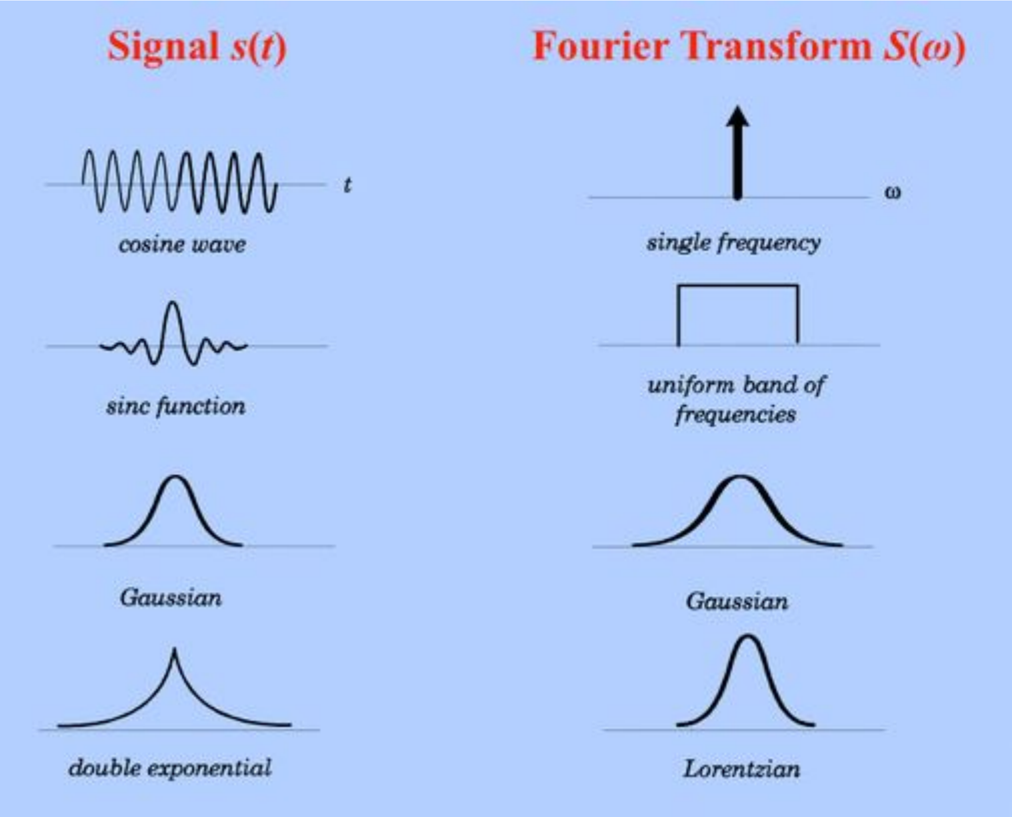
- A signal in time (length N),
  Can be represented by a discrete vector, $x \in \mathbb{R}^N$

Fourier transform

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i \xi x}\, dx. \quad \textbf{(Eq.1)}$$

wikipedia

- Or instead, as a set of N discrete Fourier transform coefficients, $F[x] \in \mathbb{C}^N$



https://mriquestions.com/fourier-transform-ft.html

Advantages of FFT representation

- Each coefficient, $F[x](\omega) \in \mathbb{C}$ has some "global" knowledge of x
- Most real world signals are "bandlimited" – so $n < N$ coefficients control most of variance

Advantages of FFT representation

- Each coefficient, $F[x](\omega) \in \mathbb{C}$ has some "global" knowledge of x
- Most real world signals are "bandlimited" – so $n < N$ coefficients control most of variance
- The basis of FFT is ordered, from low to high frequency

Assumption of FFT

- The “basis” we decompose with is harmonic in time

- Adjacent points in time should behave “similarly”

Assumption of FFT

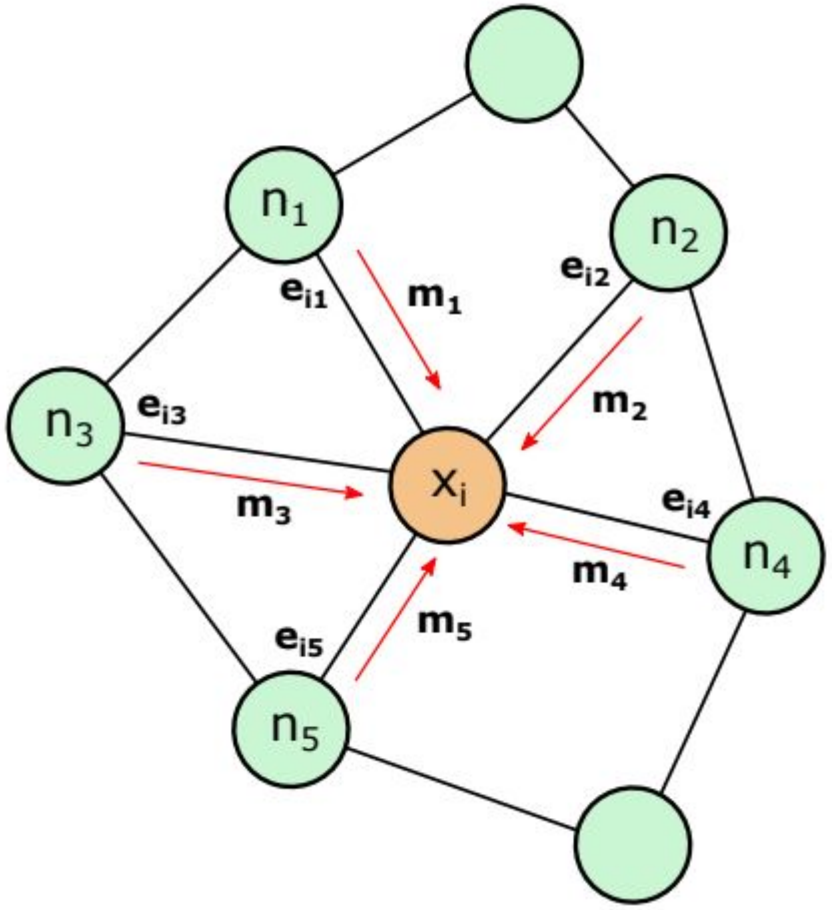
- The “basis” we decompose with is harmonic in time

- Adjacent points in time should behave “similarly”

- Large difference between nearby points implies: high energy signal, non-smooth, high frequency, etc.

Graph Signal Processing

- How do we represent data on a graph?



V : Vertex (or node) set

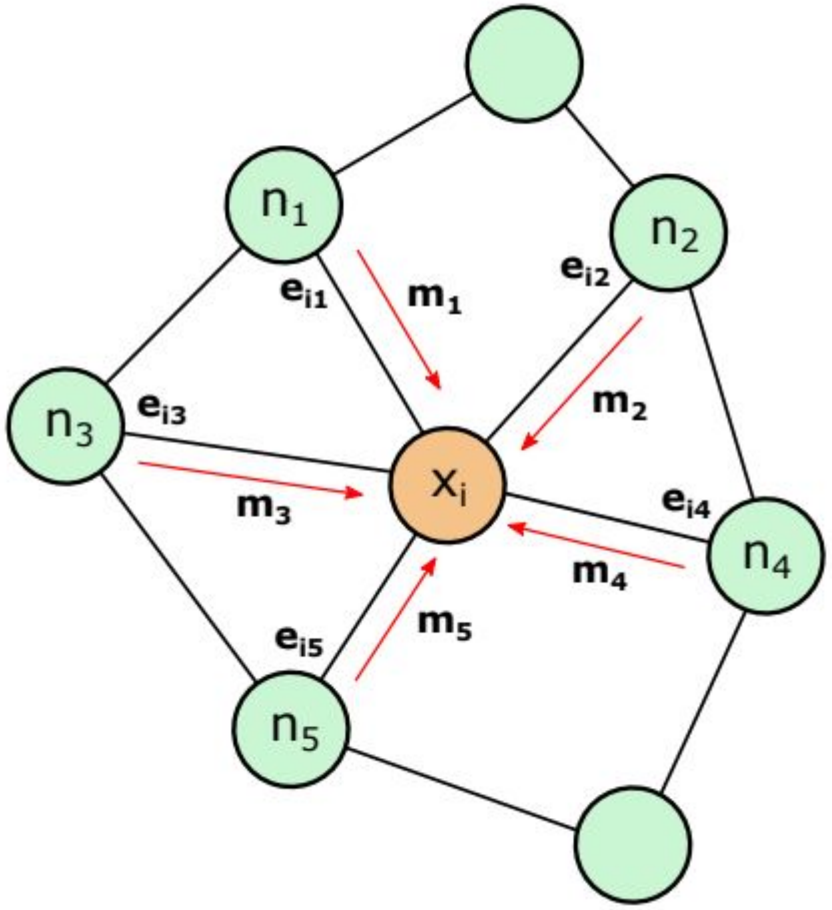
$\mathcal{E}$: Edge set (or Adjacency matrix)

Graph Signal Processing

- How do we represent data on a graph?

- On one hand, it is just $x \in \mathbb{R}^{|V|}$

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



V : Vertex (or node) set

$\mathcal{E}$: Edge set (or Adjacency matrix)

Graph Signal Processing

- How do we represent data on a graph?

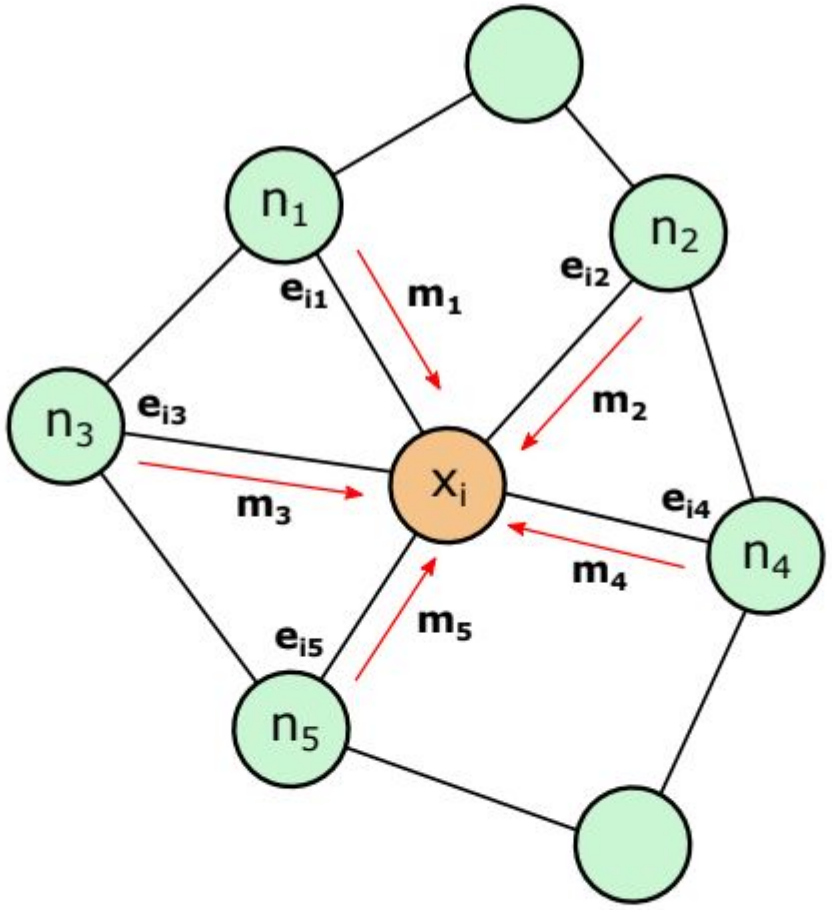
- On one hand, it is just $x \in \mathbb{R}^{|V|}$

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

Problem

- This makes no use of edges (structure)

- Cant use FFT directly, since now “adjacent points” are no longer linearly ordered



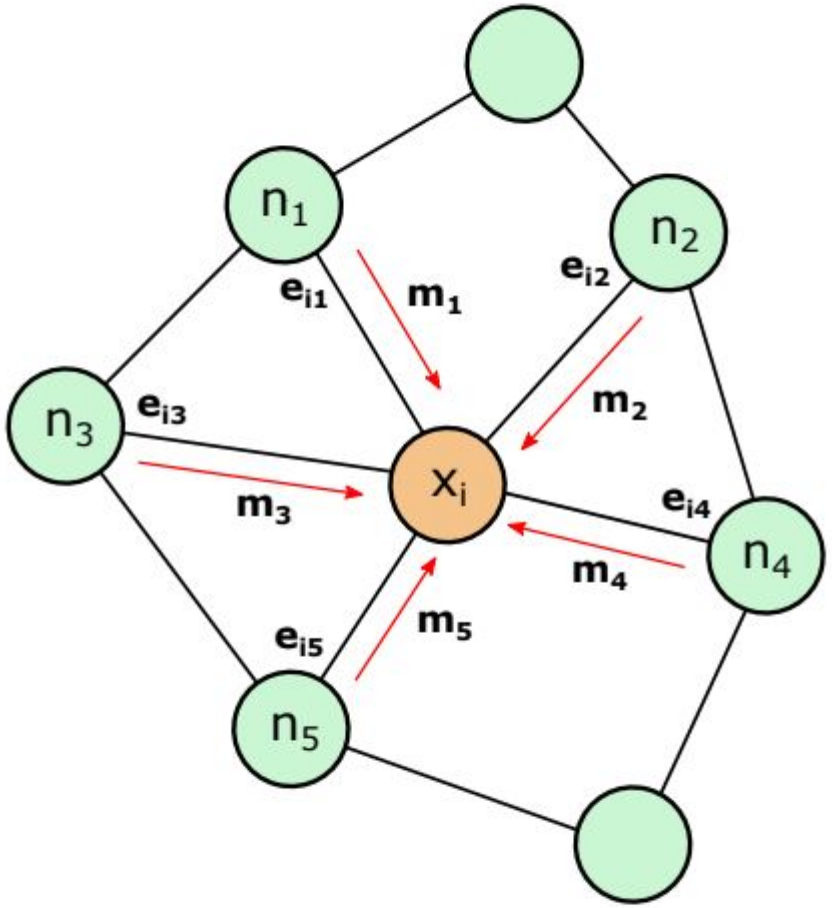
V : Vertex (or node) set

$\mathcal{E}$: Edge set (or Adjacency matrix)

Graph Signal Processing: Graph FFT

- Any graph $G = (V, \mathcal{E})$

Has an intrinsic (orthonormal) basis



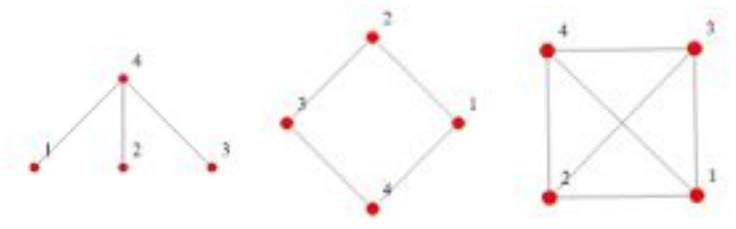
V : Vertex (or node) set

$\mathcal{E}$: Edge set (or Adjacency matrix)

Graph Signal Processing: Graph FFT

- Any graph $G = (V, \mathcal{E})$

Has an intrinsic (orthonormal) basis



$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

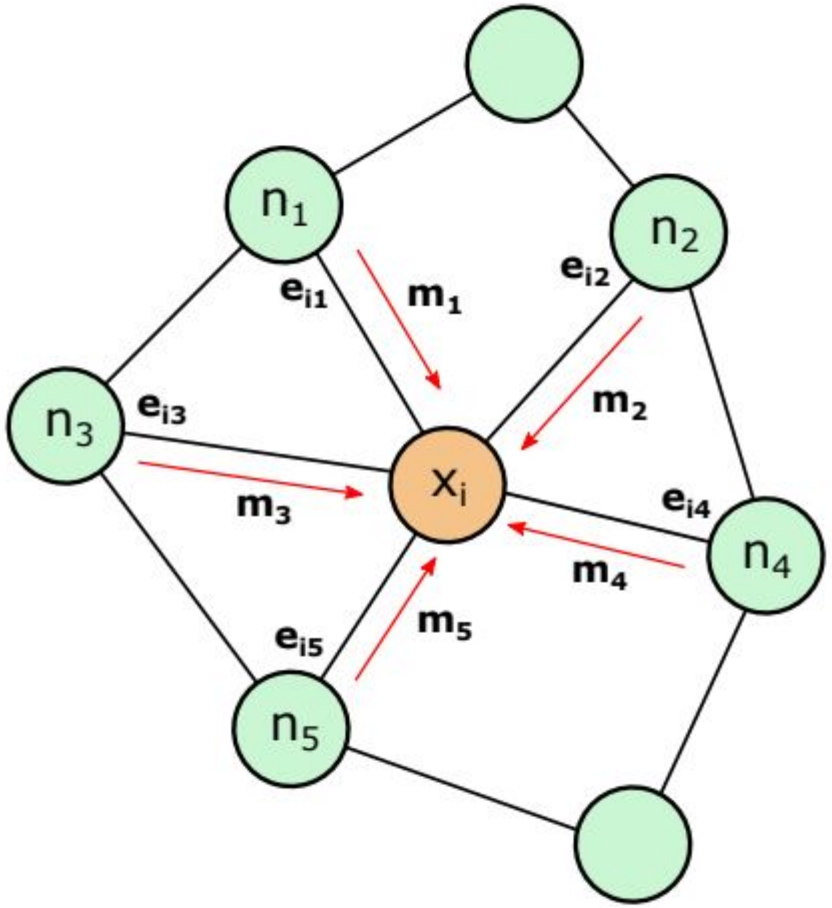
wolfram

adjacencies

$L = D - A$

D : Diagonal matrix

A : Adjacency matrix



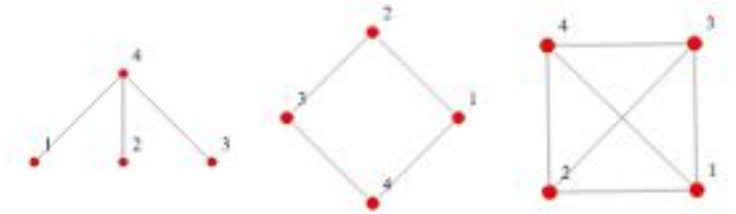
V : Vertex (or node) set

$\mathcal{E}$: Edge set (or Adjacency matrix)

Graph Signal Processing: Graph FFT

- Any graph $G = (V, \mathcal{E})$

Has an intrinsic (orthonormal) basis



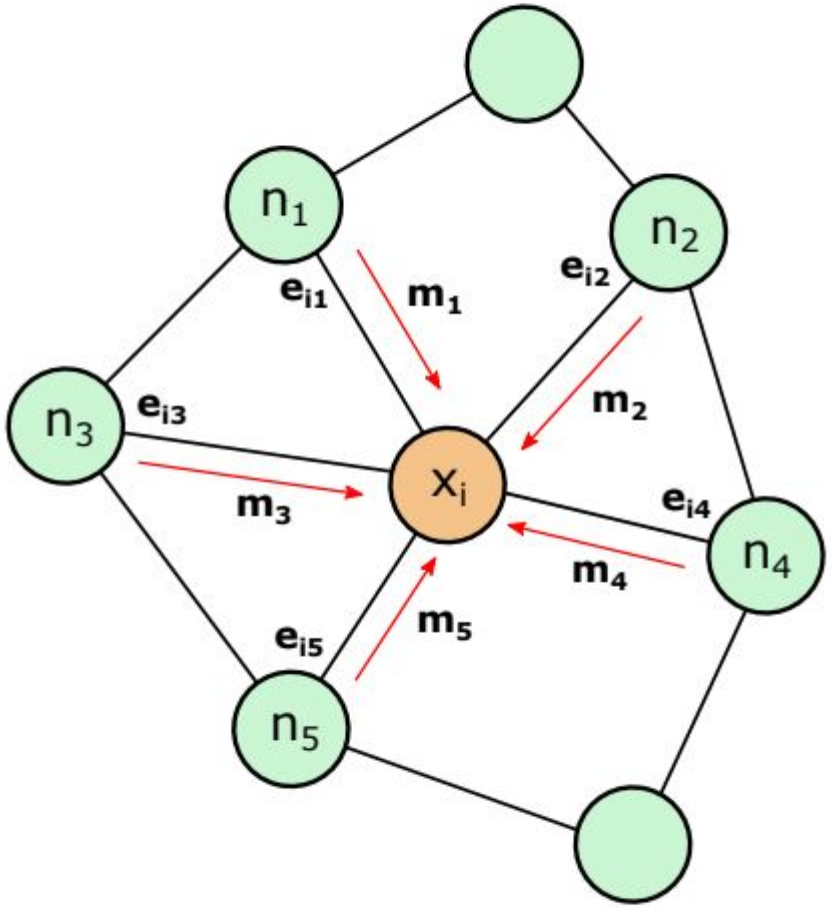
$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

wolfram

adjacencies

$L = U \Lambda U^T \qquad L : Laplacian$

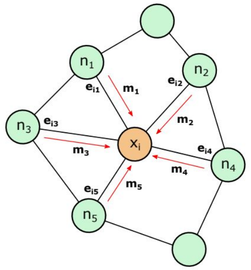
$$\boxed{U_i \in \mathbb{R}^{|V|} : Eigenvectors}$$



V : Vertex (or node) set

$\mathcal{E}$: Edge set (or Adjacency matrix)

Graph FFT Basis



V : Vertex (or node) set

$\mathcal{E}$: Edge set (or Adjacency matrix)

$$L = U \Lambda U^T \qquad U_i \in \mathbb{R}^{|V|} : Eigenvectors$$

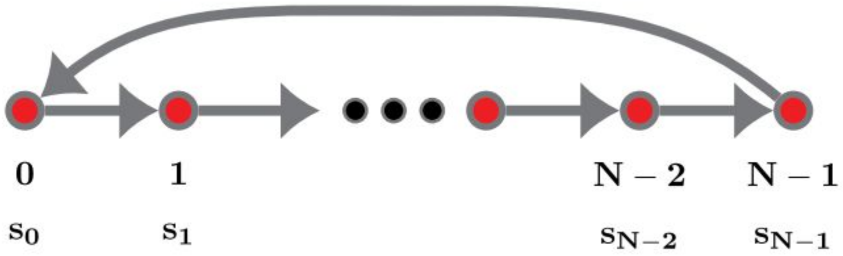
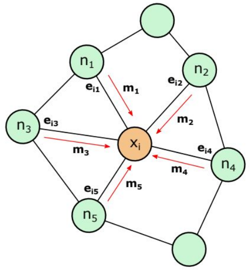


Fig. 2. ***Time graph: Cycle graph*** G_c.

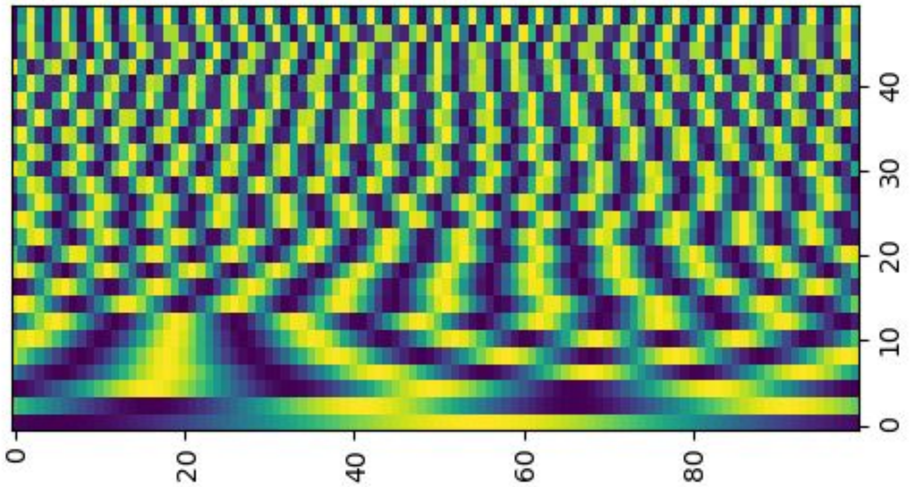
Ortega et a., 2018

Graph FFT Basis



$\mathcal{V}$: Vertex (or node) set

$\mathcal{E}$: Edge set (or Adjacency matrix)



Eigenvector dim

Node dim

$$L = U \Lambda U^T \qquad U_i \in \mathbb{R}^{|V|} : Eigenvectors$$

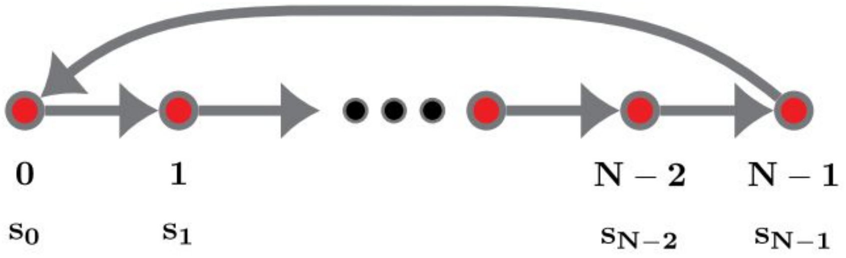
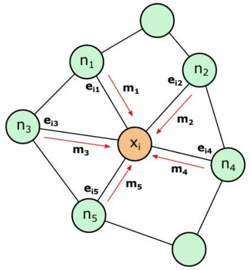


Fig. 2. *Time graph: Cycle graph G_c.*

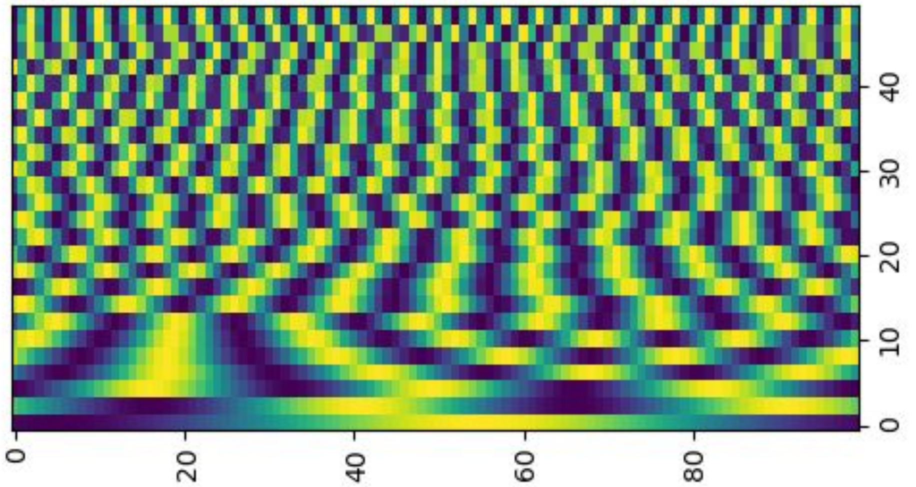
Ortega et a., 2018

Graph FFT Basis



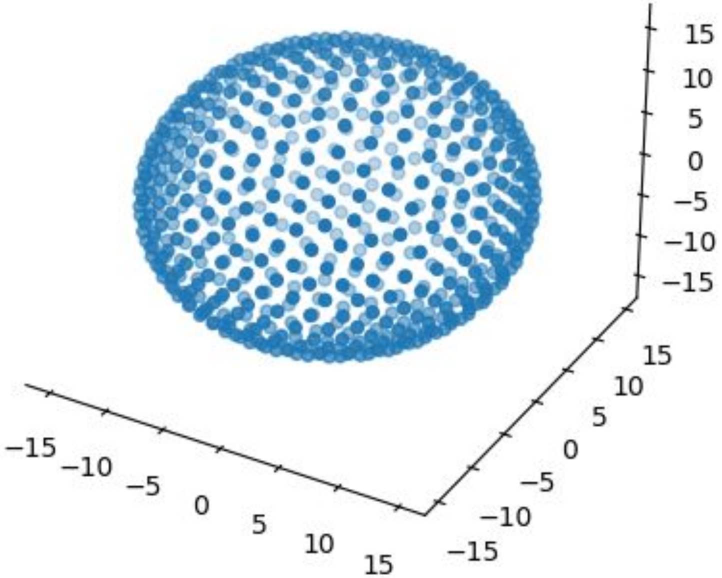
$\mathcal{V}$: Vertex (or node) set

$\mathcal{E}$: Edge set (or Adjacency matrix)



Eigenvector dim

Node dim



$$L = U\Lambda U^T \qquad U_i \in \mathbb{R}^{|V|} : Eigenvectors$$

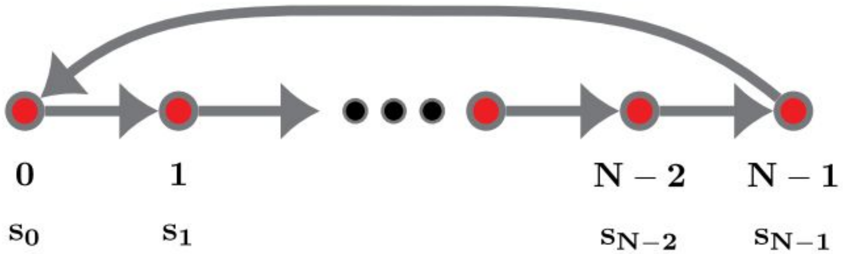
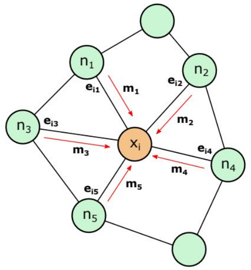


Fig. 2. ***Time graph: Cycle graph*** G_c.

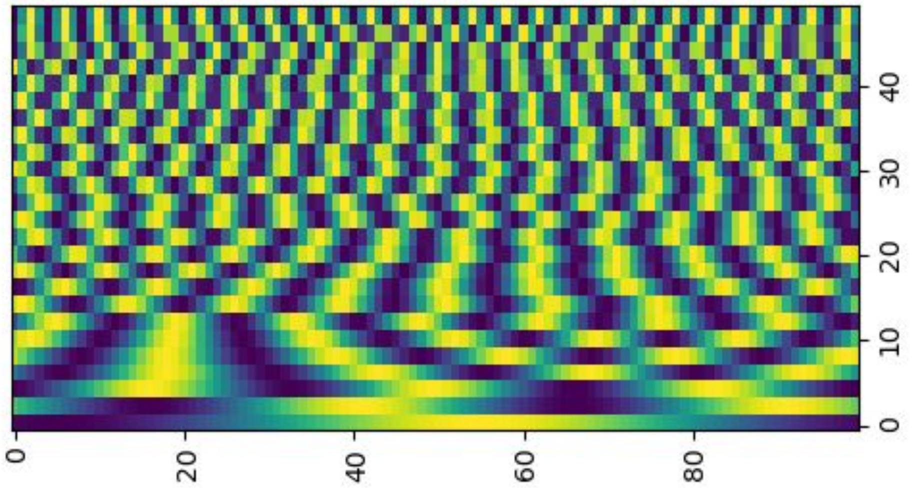
Ortega et a., 2018

Graph FFT Basis



V : Vertex (or node) set

$\mathcal{E}$: Edge set (or Adjacency matrix)



Eigenvector dim

Node dim

$$L = U \Lambda U^T \qquad U_i \in \mathbb{R}^{|V|} : Eigenvectors$$

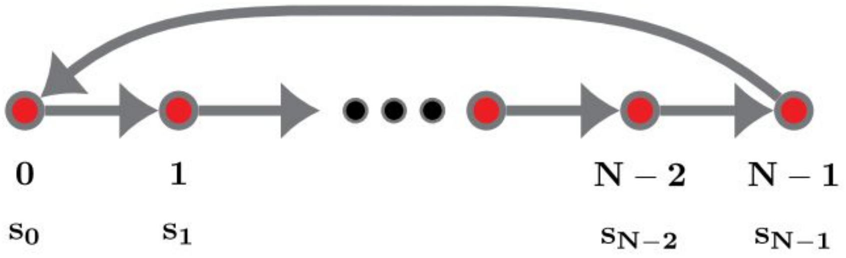
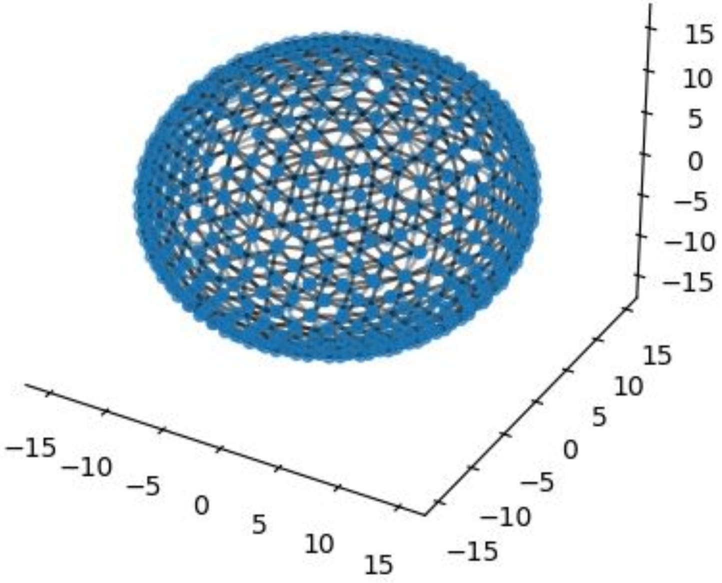
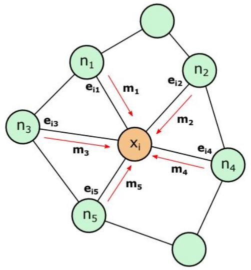


Fig. 2. *Time graph: Cycle graph* G_c.

Ortega et a., 2018

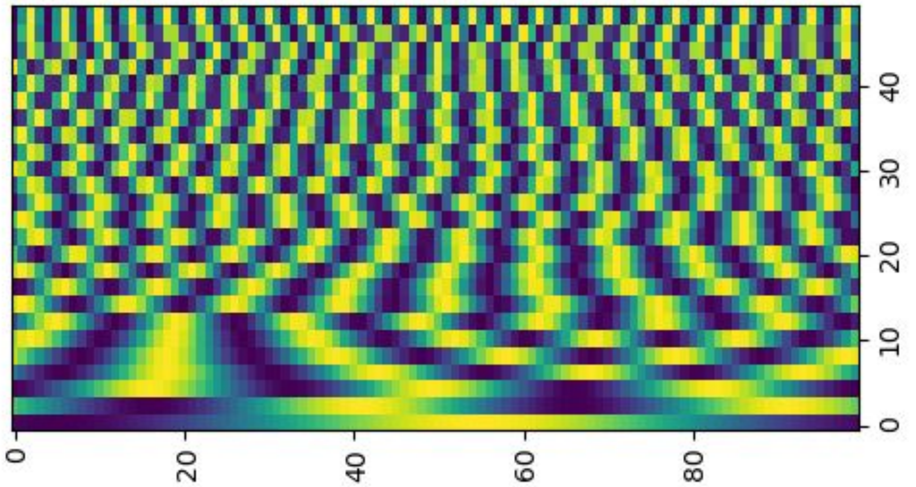


Graph FFT Basis



V : Vertex (or node) set

$\mathcal{E}$: Edge set (or Adjacency matrix)



Eigenvector dim

Node dim

$$L = U \Lambda U^T \qquad U_i \in \mathbb{R}^{|V|} : Eigenvectors$$

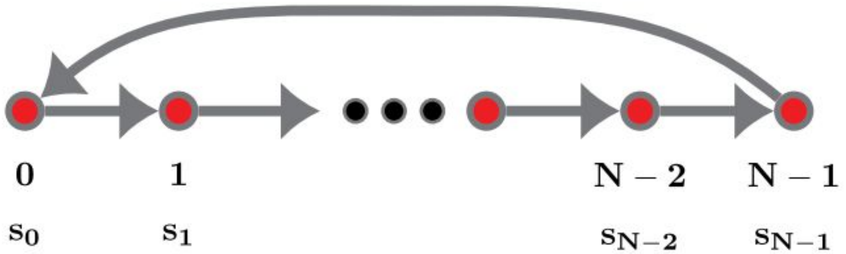
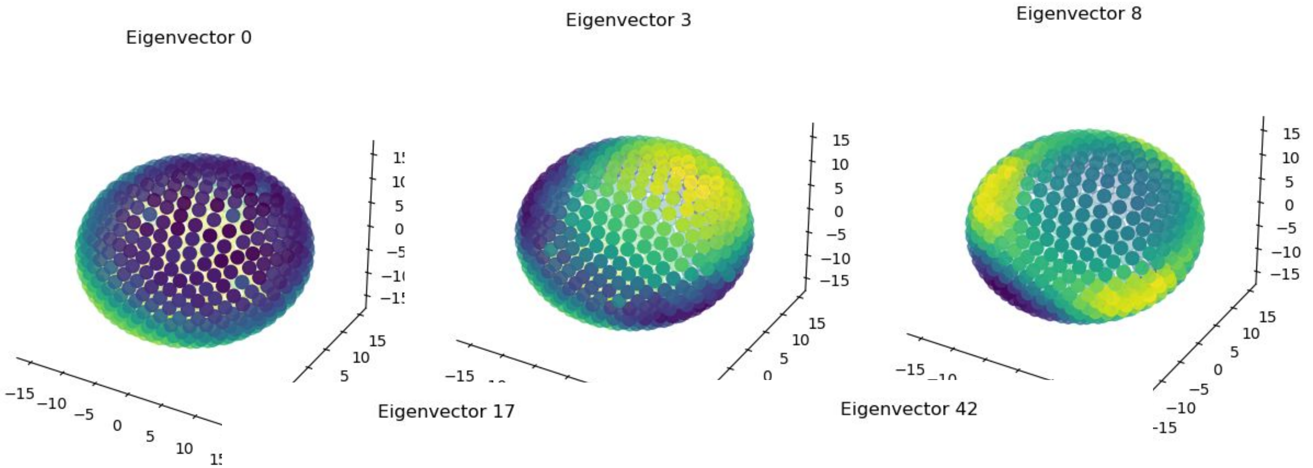
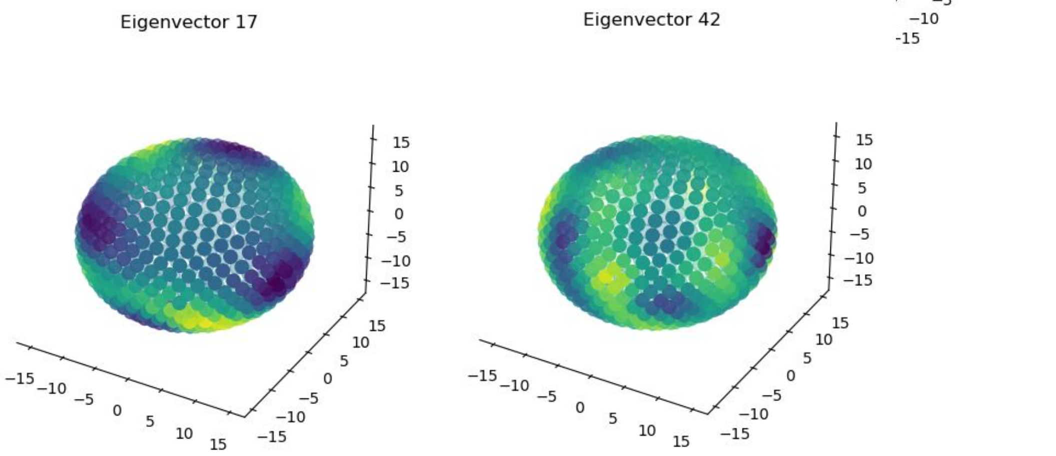


Fig. 2. ***Time graph: Cycle graph*** G_c.

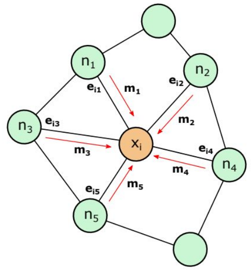
Ortega et a., 2018



Low to high frequency



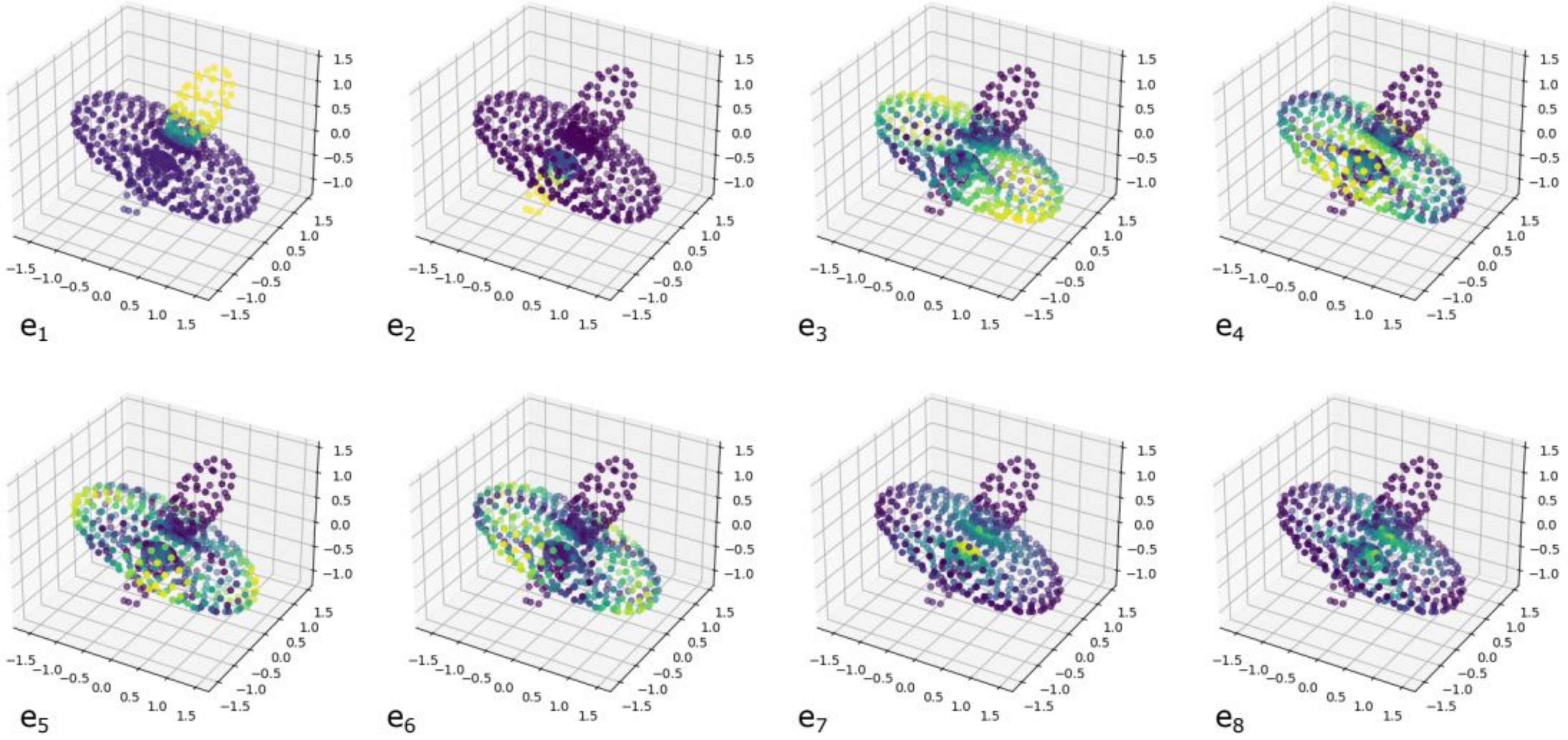
Graph FFT Basis



$\mathcal{V}$: Vertex (or node) set

$\mathcal{E}$: Edge set (or Adjacency matrix)

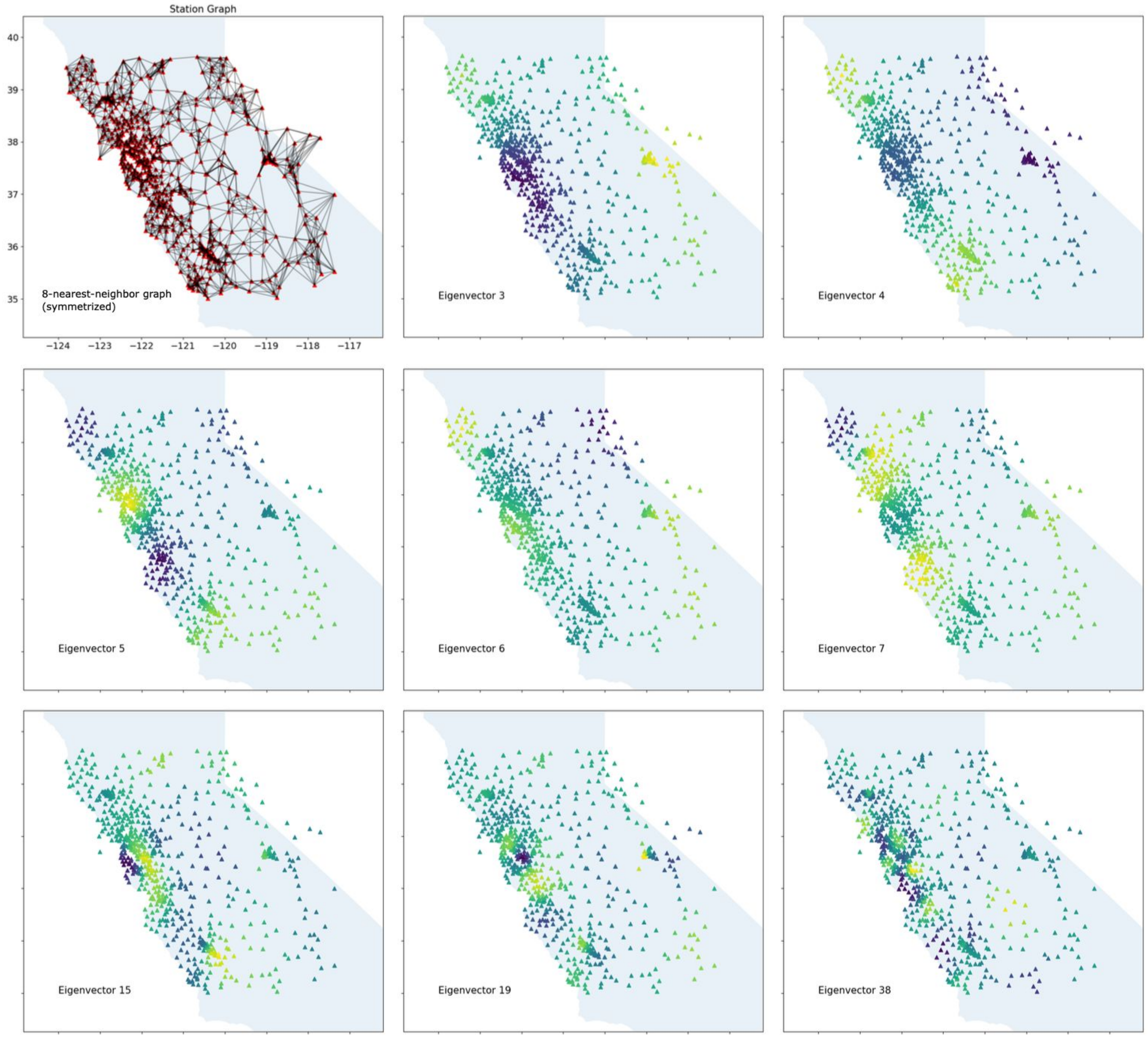
Deformed sphere:



Low to high frequency

And structure aware

Graph FFT Basis



Low to high frequency

And structure aware

Why Graph Neural Networks?

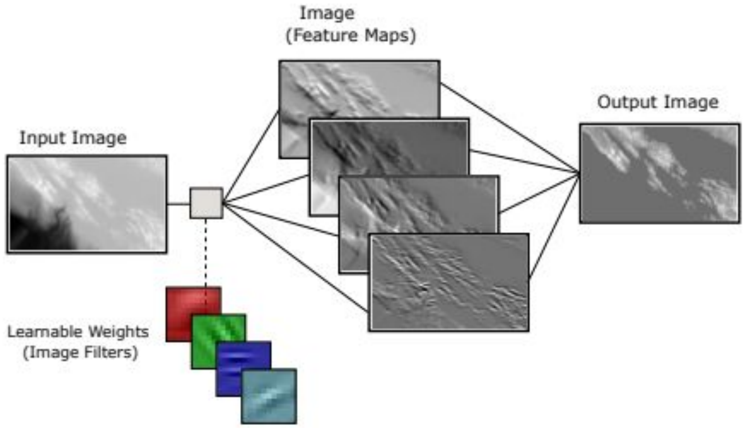
- CNN’s let us “learn” mappings on regular grid domains

Recall convolution theorem:

$$f(t) * g(t) = iFFT[F(\omega)G(\omega)]$$

Convolutional Neural Networks

Effective for Euclidean data (e.g., time series, images)



Relies on the distribution and type of spatial features (e.g., edges, shapes, gradients).

Why Graph Neural Networks?

- CNN’s let us “learn” mappings on regular grid domains

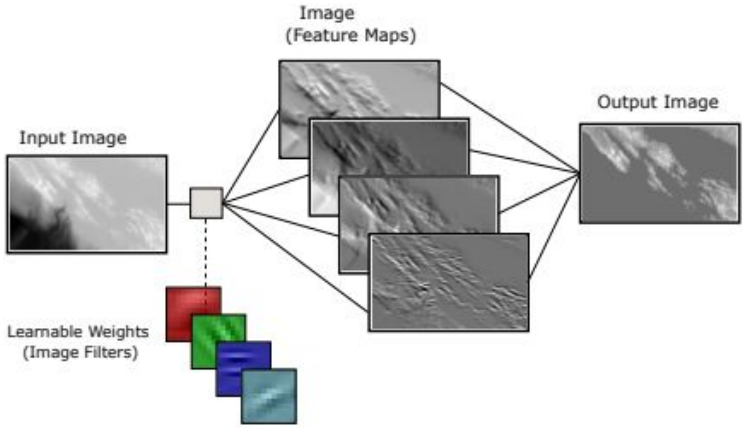
Recall convolution theorem:

$$f(t) * g(t) = iFFT[F(\omega)G(\omega)]$$

- Assumption: multiple layers of “convolution” permits functions that are expressible with ~low order frequency content in Fourier basis

Convolutional Neural Networks

Effective for Euclidean data (e.g., time series, images)



Relies on the distribution and type of spatial features (e.g., edges, shapes, gradients).

Why Graph Neural Networks?

Graph convolution:

• Rather than using eigenvectors directly (global FFT representation),
Use local message passing (local representation)

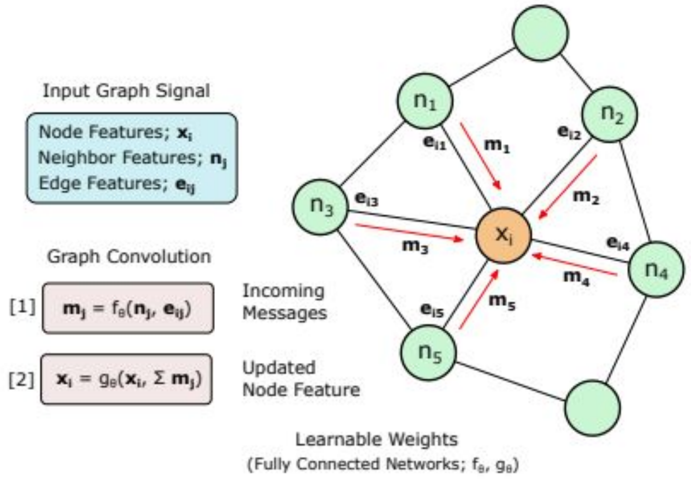
$$L = D - A$$

$$L = U \Lambda U^T$$

$$U_i \in \mathbb{R}^{|V|} : Eigenvectors$$

Graph Neural Networks

Effective for non-Euclidean data (e.g., graphs, sensor arrays)



Relies on local information passing between nodes.

Relaxed conditions on the spatial regularity of data.

Why Graph Neural Networks?

Graph convolution:

• Rather than using eigenvectors directly (global FFT representation),
Use local message passing (local representation)

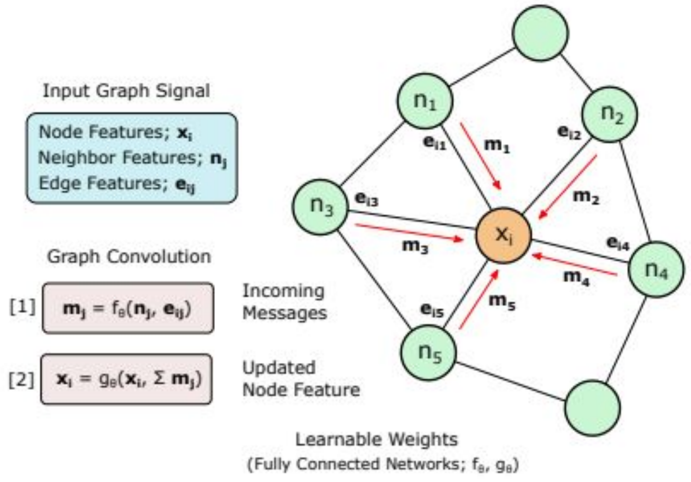
• After several rounds of message passing, will have "access" to functions supported by graph Laplacian eigenvector basis

$$L = D - A$$
$$L = U \Lambda U^T$$

$$U_i \in \mathbb{R}^{|V|} : Eigenvectors$$

Graph Neural Networks

Effective for non-Euclidean data (e.g., graphs, sensor arrays)



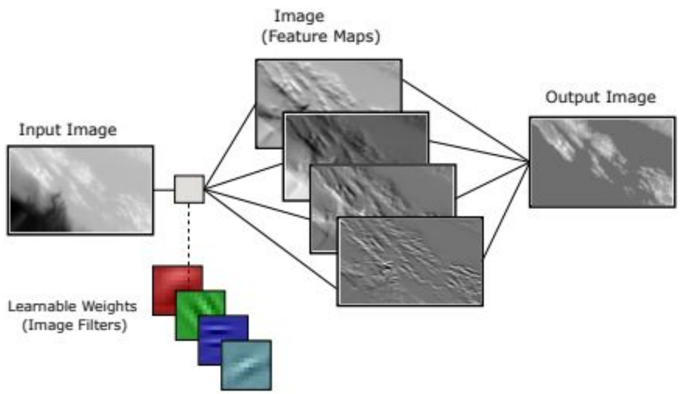
Relies on local information passing between nodes.

Relaxed conditions on the spatial regularity of data.

Why Graph Neural Networks?

Convolutional Neural Networks

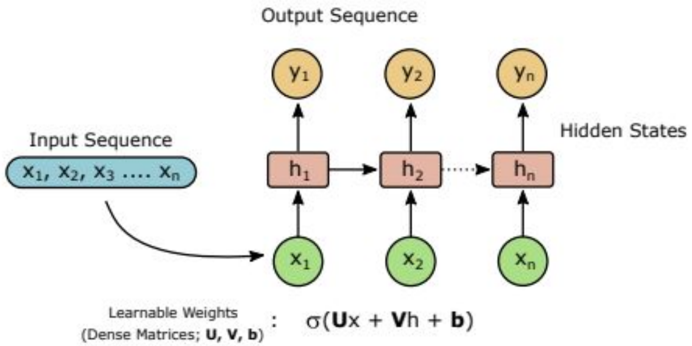
Effective for Euclidean data
(e.g., time series, images)



Relies on the distribution and type of spatial features (e.g., edges, shapes, gradients).

Recurrent Neural Networks

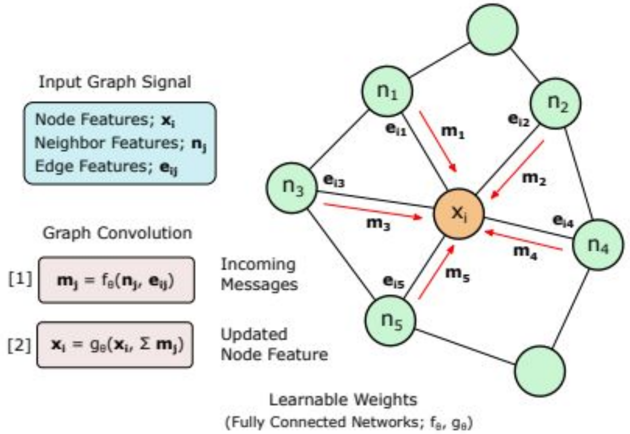
Effective for Euclidean data
(e.g., time series, text)



Relies on the timing/sequencing and strength of temporal signals.

Graph Neural Networks

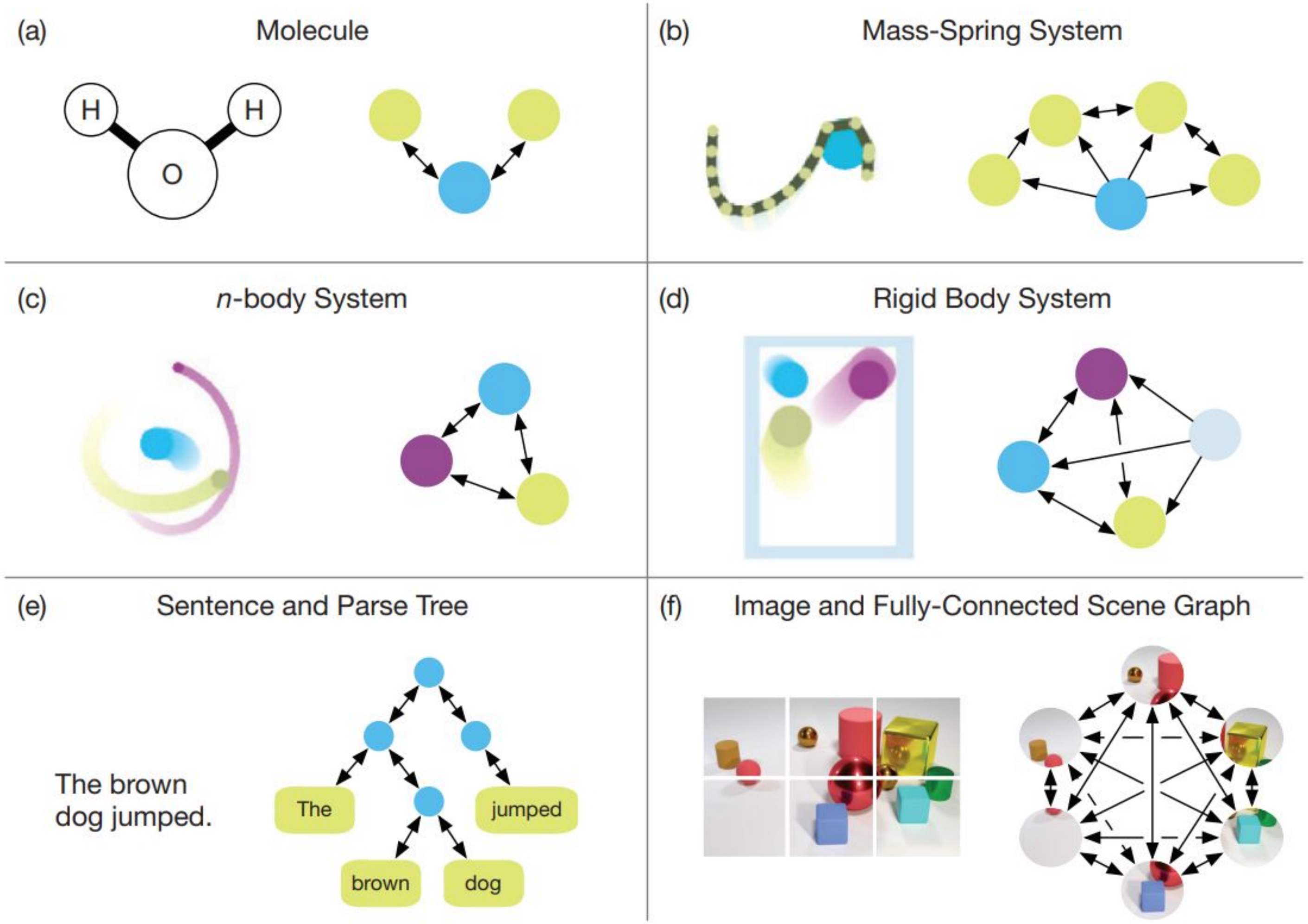
Effective for non-Euclidean data
(e.g., graphs, sensor arrays)



Relies on local information passing between nodes.

Relaxed conditions on the spatial regularity of data.

Graph Examples



Battagilia, et al., 2018

Common examples:

Sensor networks, social networks, smooth irregular surfaces

Less common examples:

Molecules, multi-particle simulation, sparse matrices, etc.

Graph Neural Networks

- Technically, could choose fixed “filters” to apply for graph convolution, but would be hard to hand engineer

- Rather, it’s easier to “learn” filters, and process data in a higher-dimensional (lifted) space

Graph Neural Networks

Message Passing:

The general form of a graph convolution layer is given by

$$\boldsymbol{h}_i^{(k+1)} = \phi^{agg}(\boldsymbol{h}_i^{(k)}, \mathrm{POOL}\{\phi^{msg}(\boldsymbol{h}_j^{(k)}, \boldsymbol{e}_{ij}, \boldsymbol{z}) \mid j \in \mathcal{N}(i)\})$$

Node and edge features:

$h \in \mathbb{R}^K$

h: node feature vector

$\mathbf{e}_{ij}$: edge feature (e.g., offset vector)

Learnable weights:

$\phi_{msg} : \mathbb{R}^{K_1} \longrightarrow \mathbb{R}^{K_2}$

$\phi_{agg} : \mathbb{R}^{K_1+K_2} \longrightarrow \mathbb{R}^{K_3}$

ϕ : Shallow fully connected neaural networks

Graph Neural Networks

Message Passing:

The general form of a graph convolution layer is given by

$$\boldsymbol{h}_i^{(k+1)} = \phi^{agg}(\boldsymbol{h}_i^{(k)}, \mathrm{POOL}\{\phi^{msg}(\boldsymbol{h}_j^{(k)}, \boldsymbol{e}_{ij}, \boldsymbol{z}) \mid j \in \mathcal{N}(i)\})$$

(1). For each node i, “collect” all neighboring nodes of node i

(2). Transform each neighboring latent vectors by phi_msg

(3). POOL (mean, max, or sum pool) over node dimension

(4). Concatenate with latent vector of node i from previous layer

(5). Transform concatenated vector with phi_agg

GNN: Properties

Message Passing:

$$\mathbf{X}' = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Theta},$$

(GCN; Kipf and Welling, 2016)

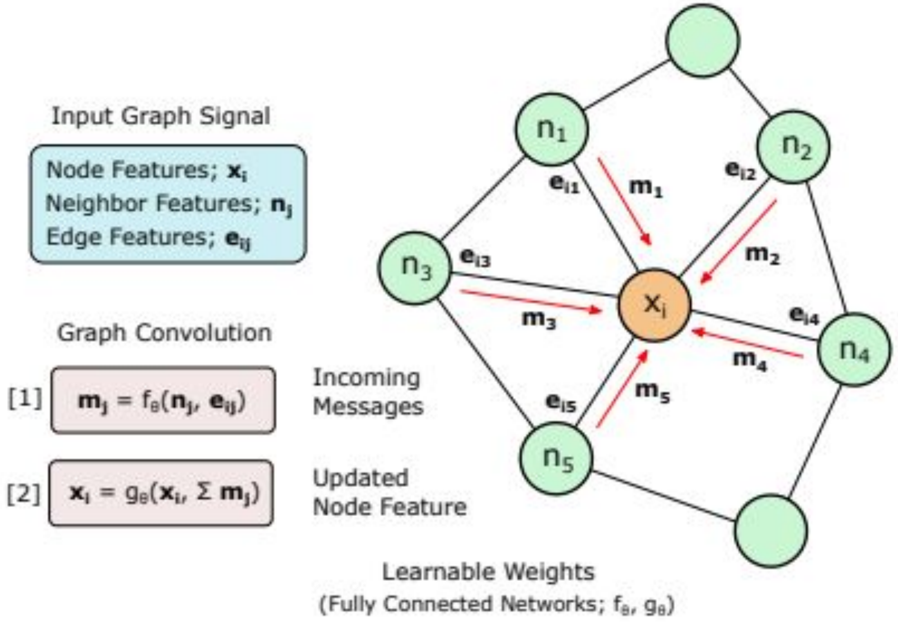
- Can also be expressed in matrix notation using Adjacency, but this limits perspective and extensions

GNN: Properties

Message Passing:

- Information transfers locally, but expands to further “hops” away with every convolution layer

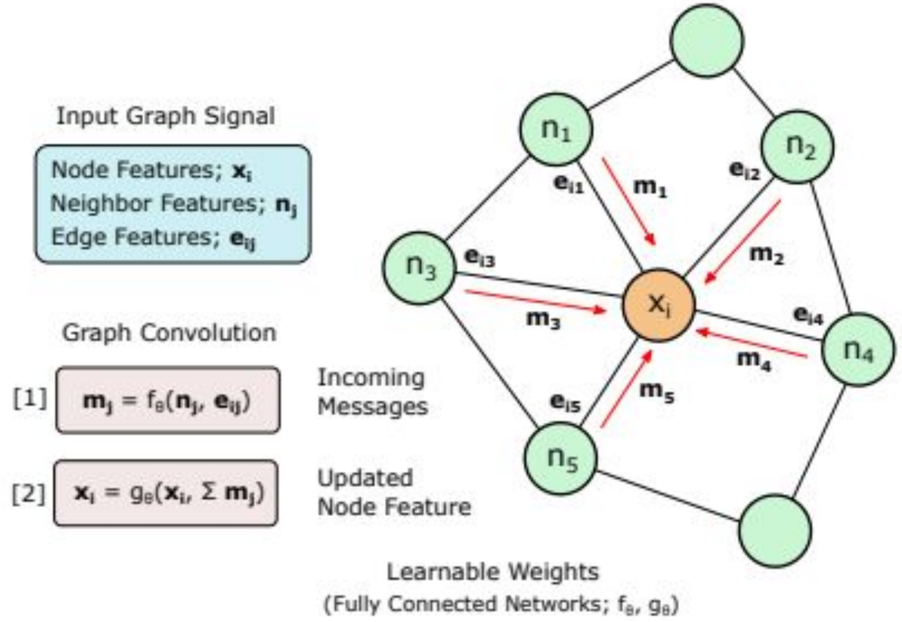
- Can handle very large, sparse graphs well



GNN: Properties

Message Passing:

- Information transfers locally, but expands to further “hops” away with every convolution layer
- Can handle very large, sparse graphs well
- At each layer of GNN, all nodes features vectors, $h_i, i \in V$, are updated based on self, and neighbors



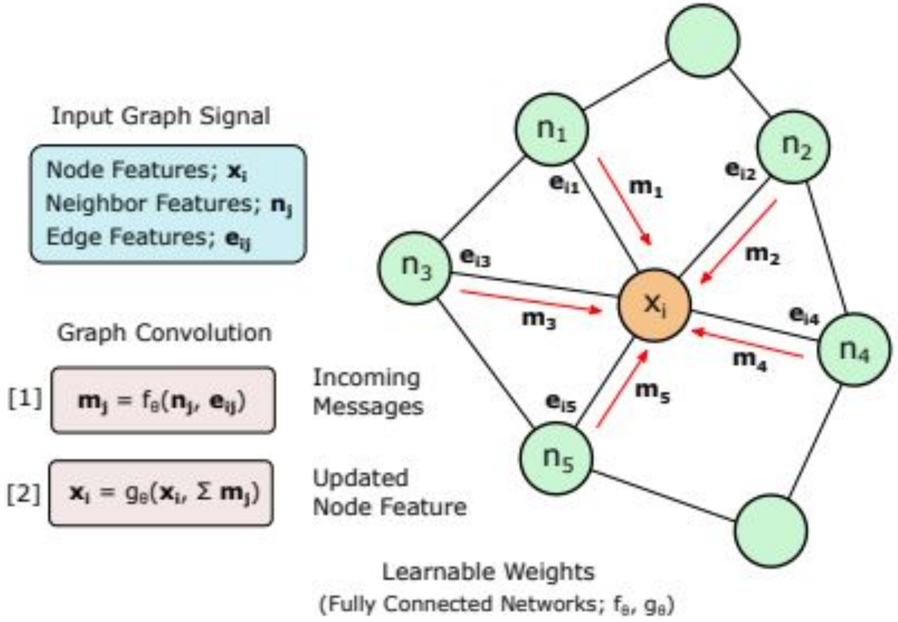
GNN: Properties

Message Passing:

- Information transfers locally, but expands to further "hops" away with every convolution layer

- Can handle very large, sparse graphs well

- Both the **features** and **graph structure** have to be used to guide learned function



GNN: Properties

Message Passing:

Table 1
Node feature vector composition for the Graph Neural Network.

Feature	Description	Dimensions
Center coordinates	Spatial position of the node (x, y, z)	3
Cluster dimensions	Size of the cluster (a, b, c)	3
Number of points	Total points in the cluster	1
Node degree	Edges connected to the node	1
Closeness centrality	Inverse sum of shortest distances between the node and all others	1
Eigenvector centrality	Measure of the node's influence based on its connections' quality and quantity	1
Pagerank	Importance of the node in the network based on its links and the significance of its neighboring nodes	1
Phase	Representation for phase (Solid = 1, Pore = 0, and vice versa)	1

- Both the **features** and **graph structure** have to be used to guide learned function

Prediction of effective elastic moduli of rocks using Graph Neural Networks

Jaehong Chung [a,*], Rasool Ahmad [b], WaiChing Sun [c], Wei Cai [b], Tapan Mukerji [a,d]

GNN: Properties

Message Passing:

Table 1
Node feature vector composition for the Graph Neural Network.

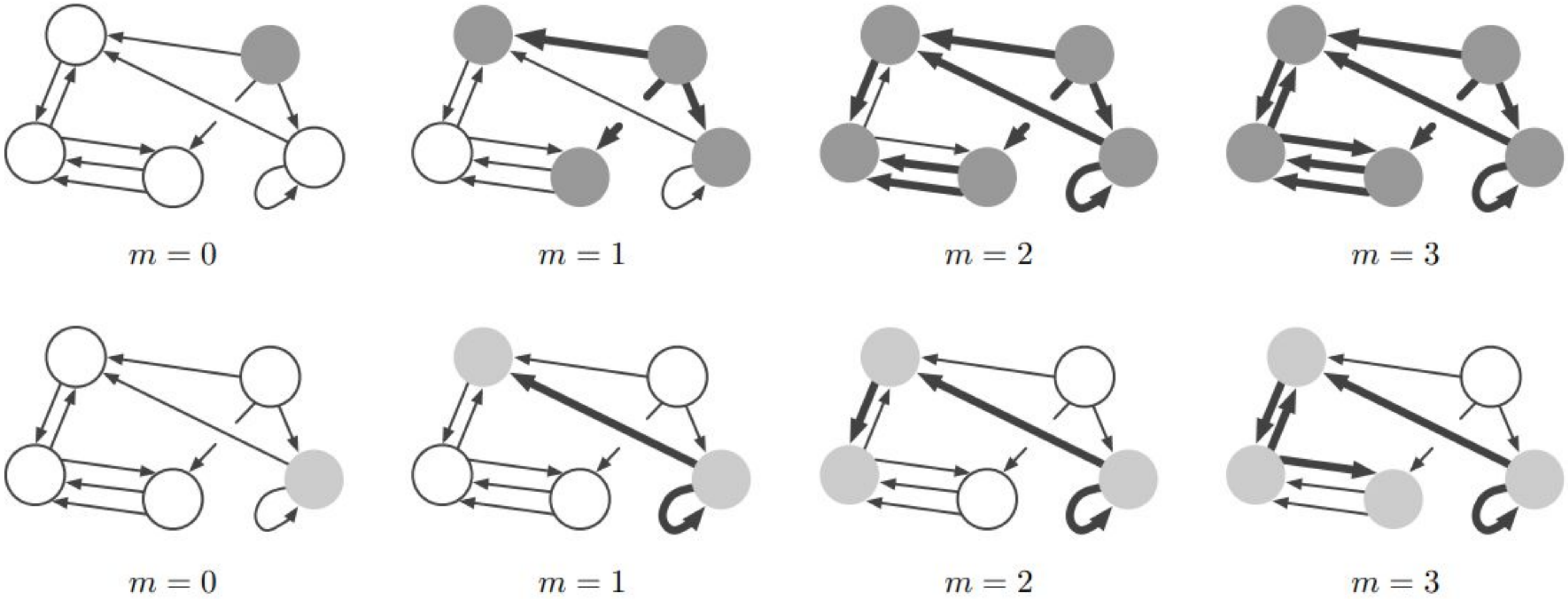
Feature	Description	Dimensions
Center coordinates	Spatial position of the node (x, y, z)	3
Cluster dimensions	Size of the cluster (a, b, c)	3
Number of points	Total points in the cluster	1
Node degree	Edges connected to the node	1
Closeness centrality	Inverse sum of shortest distances between the node and all others	1
Eigenvector centrality	Measure of the node's influence based on its connections' quality and quantity	1
Pagerank	Importance of the node in the network based on its links and the significance of its neighboring nodes	1
Phase	Representation for phase (Solid = 1, Pore = 0, and vice versa)	1

Prediction of effective elastic moduli of rocks using Graph Neural Networks
Jaehong Chung [a,*], Rasool Ahmad [b], WaiChing Sun [c], Wei Cai [b], Tapan Mukerji [a,d]

- The **graph structure** can be the **feature**

GNN: Properties

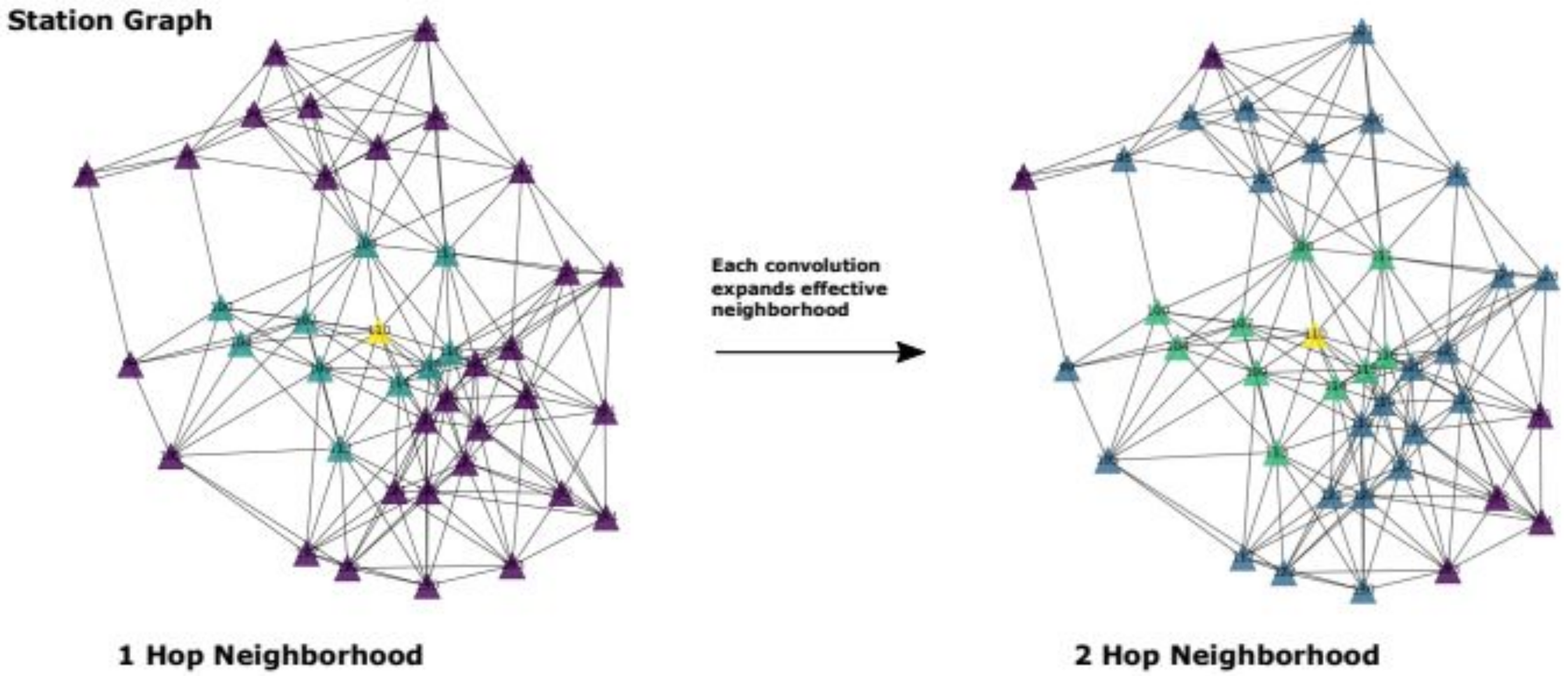
Message



Battagilia, et al.,
2018

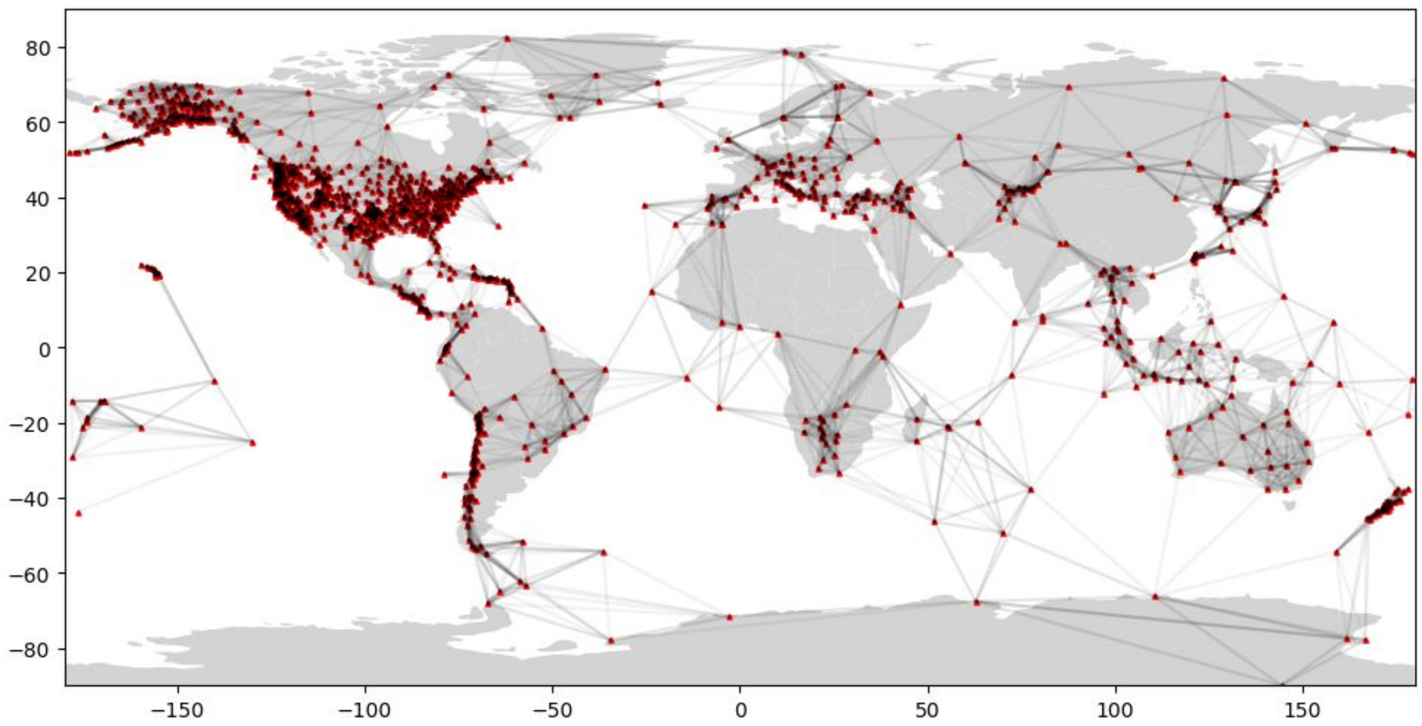
GNN: Properties

Message Passing:



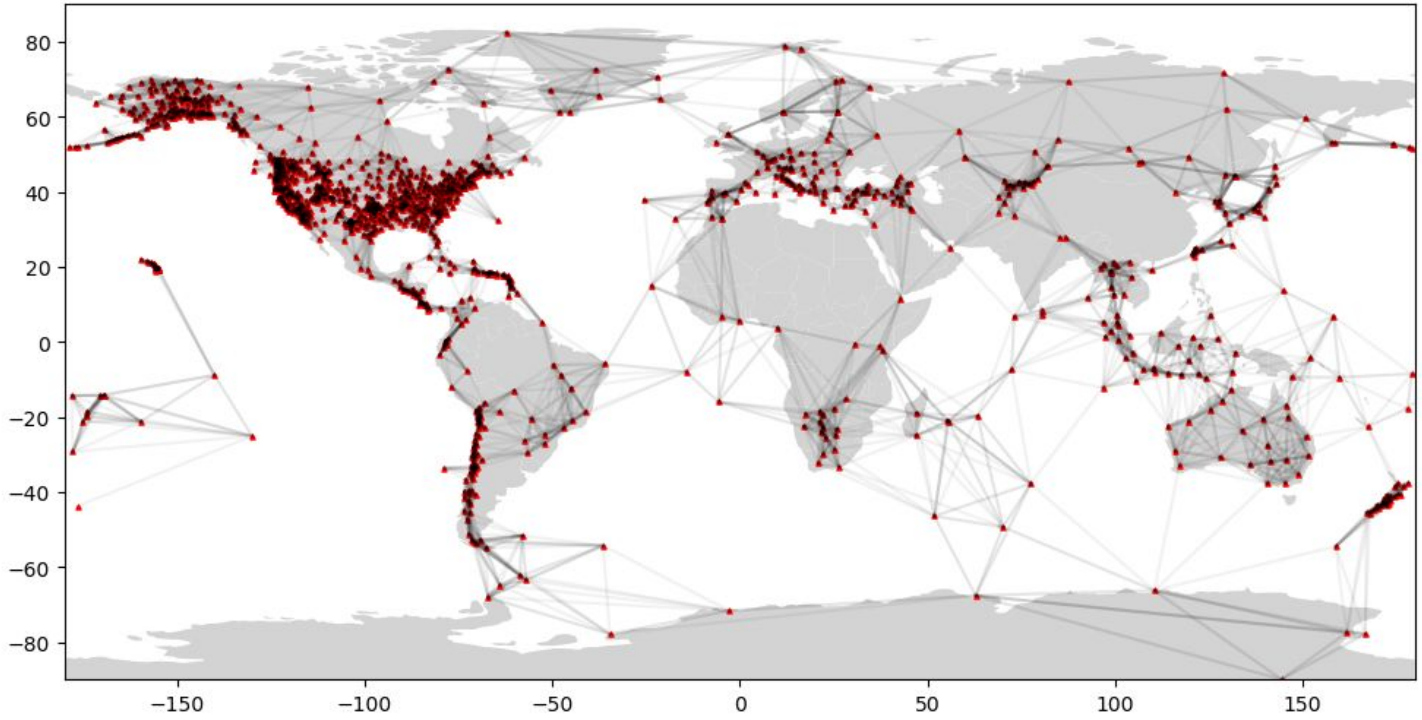
GNN: Considerations

1). Diameter of graph
(longest, shortest path distance; e.g., Distance between most separated nodes)



GNN: Considerations

1). Diameter of graph
(longest, shortest path distance; e.g., Distance between most separated nodes)

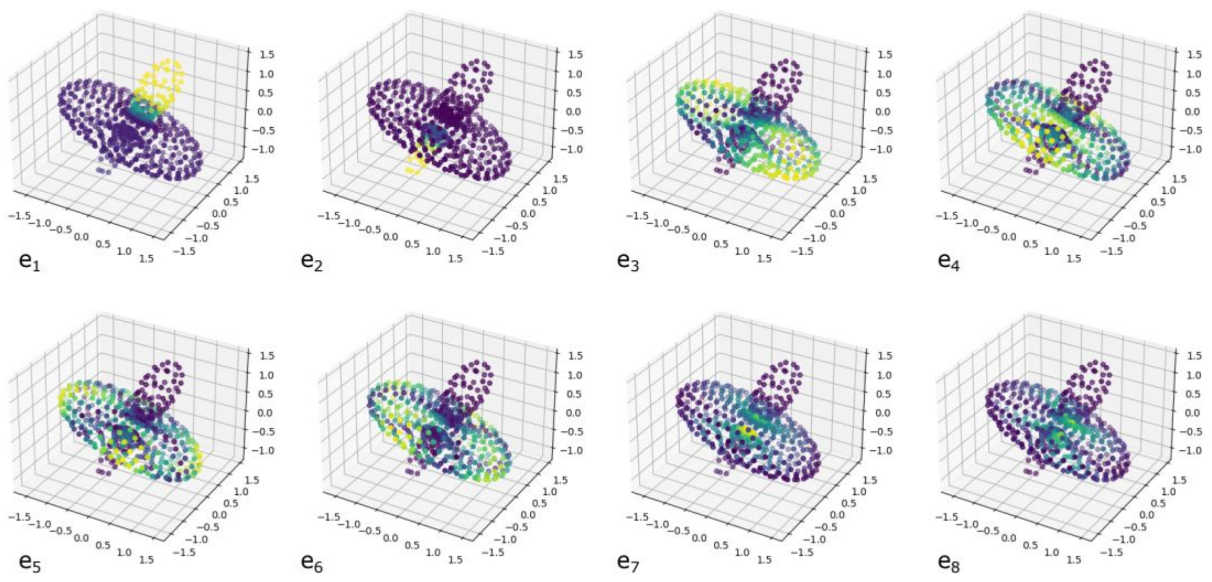


If long-range interactions needed, many ideas proposed:

(i). Add "virtual nodes" connected to all nodes, (ii). Use global summary features, (iii). Add more edges to adjacency (e.g., expander graphs), (iv). Create multi-scale, multi-resolution graphs…

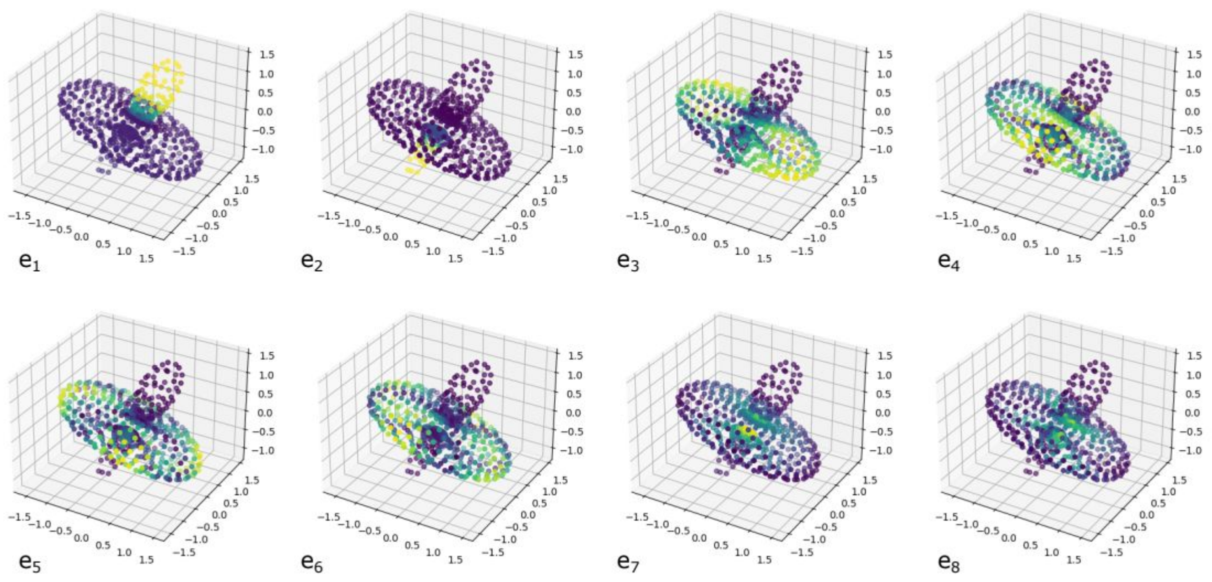
GNN: Considerations

2). Edge features
(Offset vectors between adjacent nodes can be used to infer local curvature)



GNN: Considerations

2). Edge features
(Offset vectors between adjacent nodes can be used to infer local curvature)



$$\boldsymbol{h}_i^{(k+1)} = \phi^{agg}(\boldsymbol{h}_i^{(k)}, \mathrm{POOL}\{\phi^{msg}(\boldsymbol{h}_j^{(k)}, \boldsymbol{e}_{ij}, \boldsymbol{z}) \mid j \in \mathcal{N}(i)\})$$

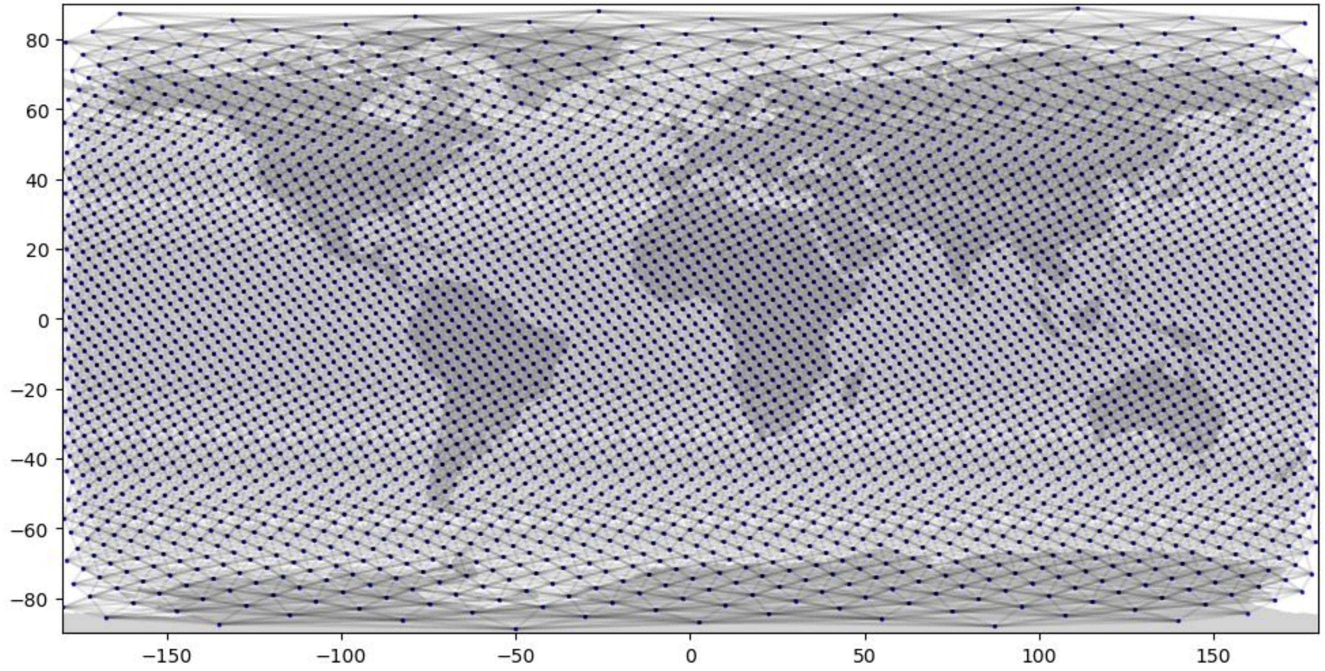
$\mathbf{e}_{ij}$: edge feature (e.g., offset vector)

GNN: Considerations

3). Absolute node positions and extra features

"Position aware GNNs" (You et al., 2019) – nodes "know" absolute position

"Identity aware GNNs" (You et al., 2021) – nodes "know" their unique type; access different learnable GNN message passing functions



GNN: Limitations

1). Over-smoothing : Since message passing iteratively shares information, it roughly emulates a diffusion/smoothing process.

To keep discriminative ability of nodes of deep GNNs, “residual” connections necessary (Hamilton et al., 2017)

GNN: Limitations

1). Over-smoothing : Since message passing iteratively shares information, it roughly emulates a diffusion/smoothing process.

To keep discriminative ability of nodes of deep GNNs, "residual" connections necessary (Hamilton et al., 2017)

$$\boldsymbol{h}_i^{(k+1)} = \phi^{agg}(\boldsymbol{h}_i^{(k)}, \mathrm{POOL}\{\phi^{msg}(\boldsymbol{h}_j^{(k)}, \boldsymbol{e}_{ij}, \boldsymbol{z}) \mid j \in \mathcal{N}(i)\})$$

Node and edge features:

$h \in \mathbb{R}^K$

h: node feature vector

Learnable weights:

$\phi_{msg} : \mathbb{R}^{K_1} \longrightarrow \mathbb{R}^{K_2}$

$\phi_{agg} : \mathbb{R}^{K_1+K_2} \longrightarrow \mathbb{R}^{K_3}$

GNN: Limitations

1). Over-smoothing : Since message passing iteratively shares information, it roughly emulates a diffusion/smoothing process.

To keep discriminative ability of nodes of deep GNNs, “residual” connections necessary (Hamilton et al., 2017)

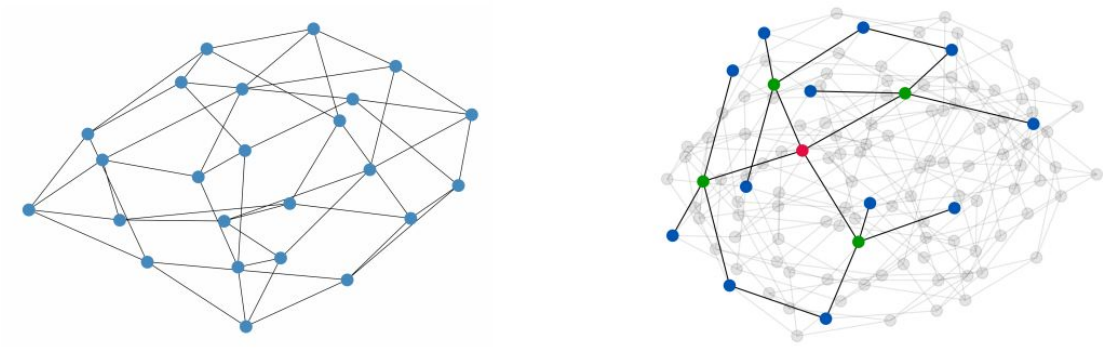
2). Over-squashing: A large “volume” of messages is slowly aggregated into a fixed size representation latent vector, which limits expressiveness of representation. Long-range interactions are weak.

Can improve performance with more well-interconnected (sparse) graphs, e.g., expander graphs.

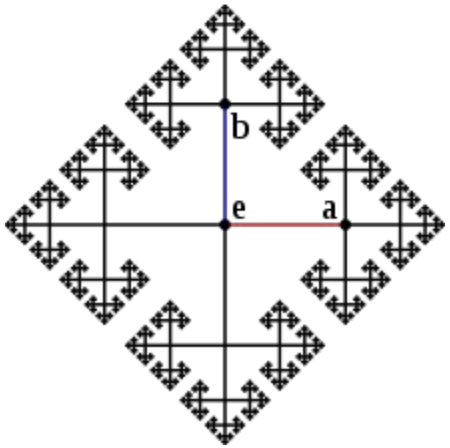
GNN: Limitations

2). Over-squashing: A large “volume” of messages is slowly aggregated into a fixed size representation latent vector, which limits expressiveness of representation. Long-range interactions are weak.

Can improve performance with more well-interconnected (sparse) graphs, e.g., expander graphs.



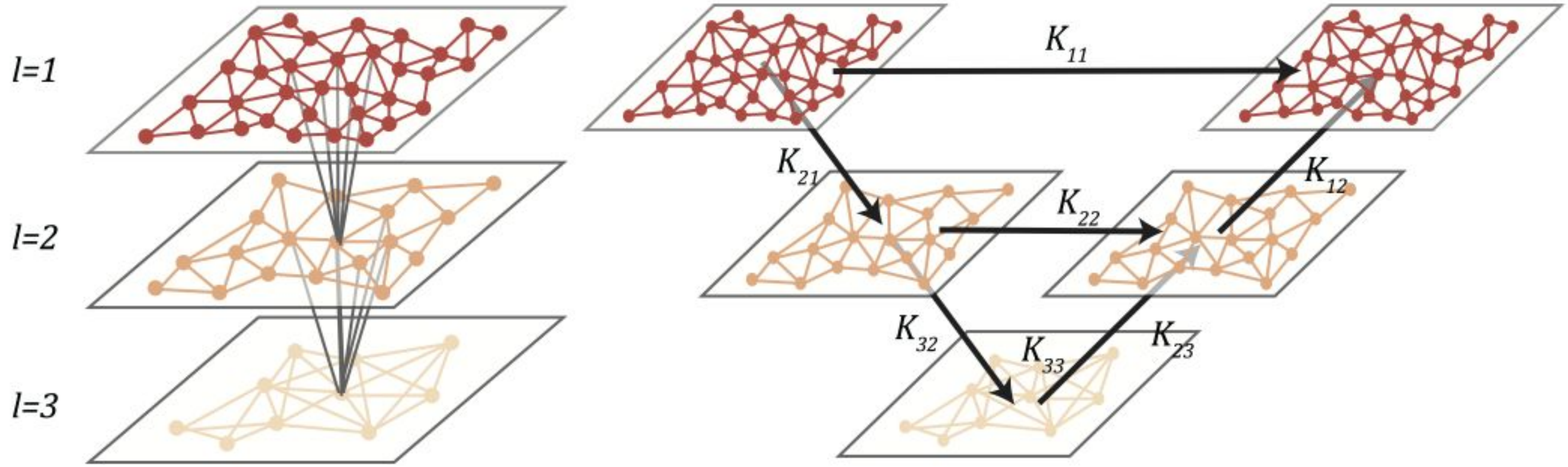
Deac et al., 2022; “Expander Graph Propagation”



Cayley graphs, wikipedia

GNN: Limitations

2). Over-squashing: A large “volume” of messages is slowly aggregated into a fixed size representation latent vector, which limits expressiveness of representation. Long-range interactions are weak.



Li et al., 2020; “Multipole Graph Neural Operator”

Hierarchical GNNs

GNN: Setup

[1]. Choose GNN architecture
(number of layers, latent dimension of features,
Edge features available).

[2]. Choose input field, $h_i^0 : i \in V$

[3]. Choose label targets
(either: node-level, graph-level, edge-level predictions)

$$y_i \in \mathbb{R} : i \in V, \text{ (Node level)}$$
$$y \in \mathbb{R}, \text{ (Graph level)}$$
$$y_{ij} : (i,j) \in \mathcal{E}, \text{ (Edge level)}$$

[4]. Choose training data
(i.e., synthesize data, graph, and label pairs)

The graph and input features typically both vary for each input

GNN: Setup

[1]. Choose GNN architecture
(number of layers, latent dimension of features,
Edge features available).

[2]. Choose input field, $h_i^0 : i \in V$

[3]. Choose label targets
(either: node-level, graph-level, edge-level predictions)

$$y_i \in \mathbb{R} : i \in V, \text{ (Node level)}$$
$$y \in \mathbb{R}, \text{ (Graph level)}$$
$$y_{ij} : (i, j) \in \mathcal{E}, \text{ (Edge level)}$$

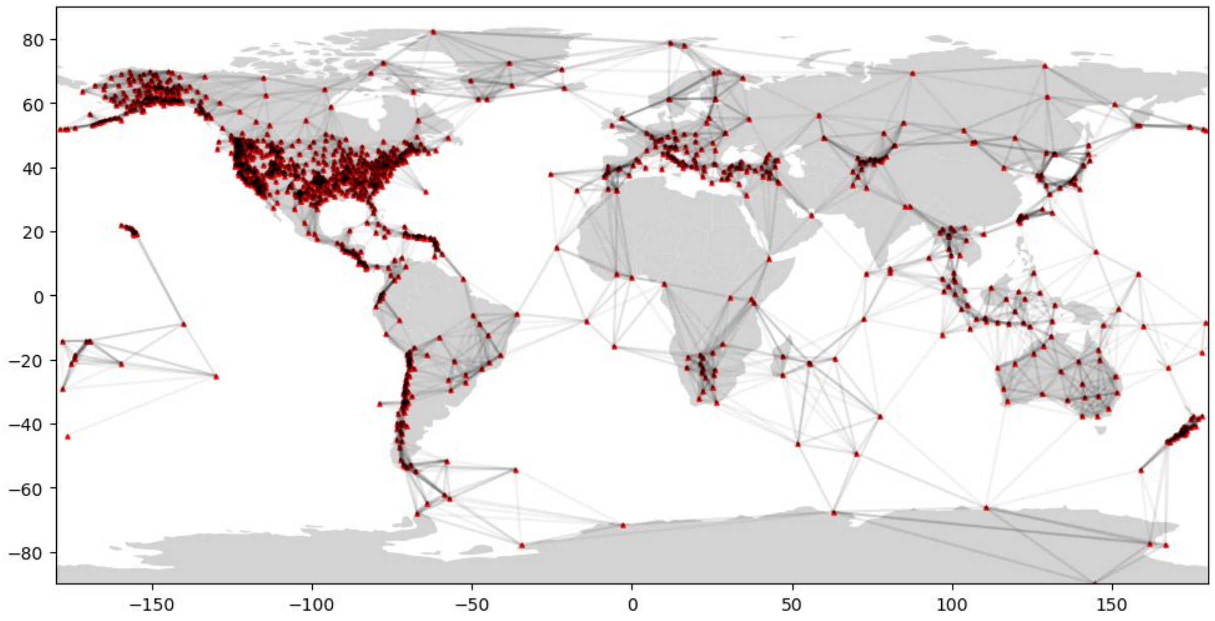
[4]. Choose training data
(i.e., synthesize data, graph, and label pairs)

The graph and input features typically both vary for each input

$$\text{Tuples of } \{(V, \mathcal{E}, h^0, y)_j \text{ for } j \text{ datapoints}\}$$

GNN: Limitations

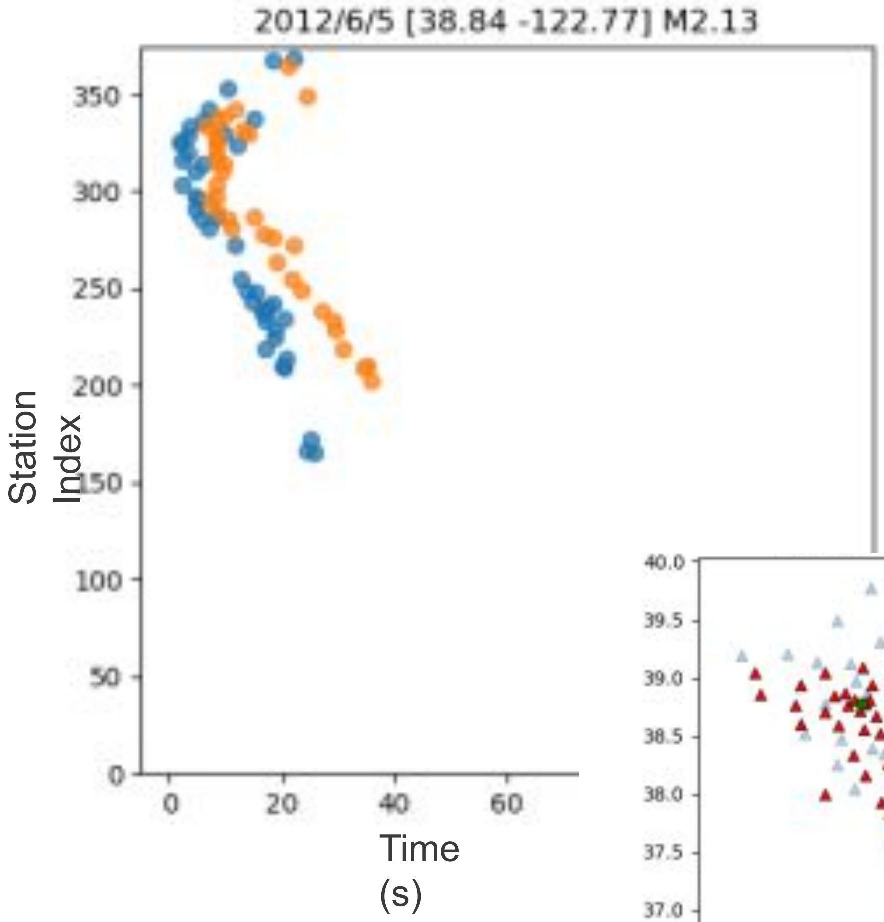
1). Diameter of graph
(longest, shortest path distance; e.g., Distance between most separated nodes)

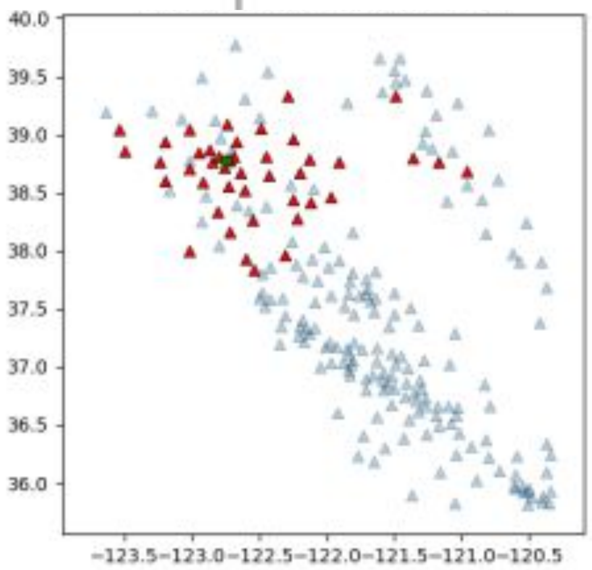


2). Over-smoothing : Since message passing iteratively shares information, it roughly emulates a diffusion/smoothing process.

3). Over-squashing: A large "volume" of messages is slowly aggregated into a fixed size representation latent vector, which limits expressiveness of representation. Long-range interactions are weak.

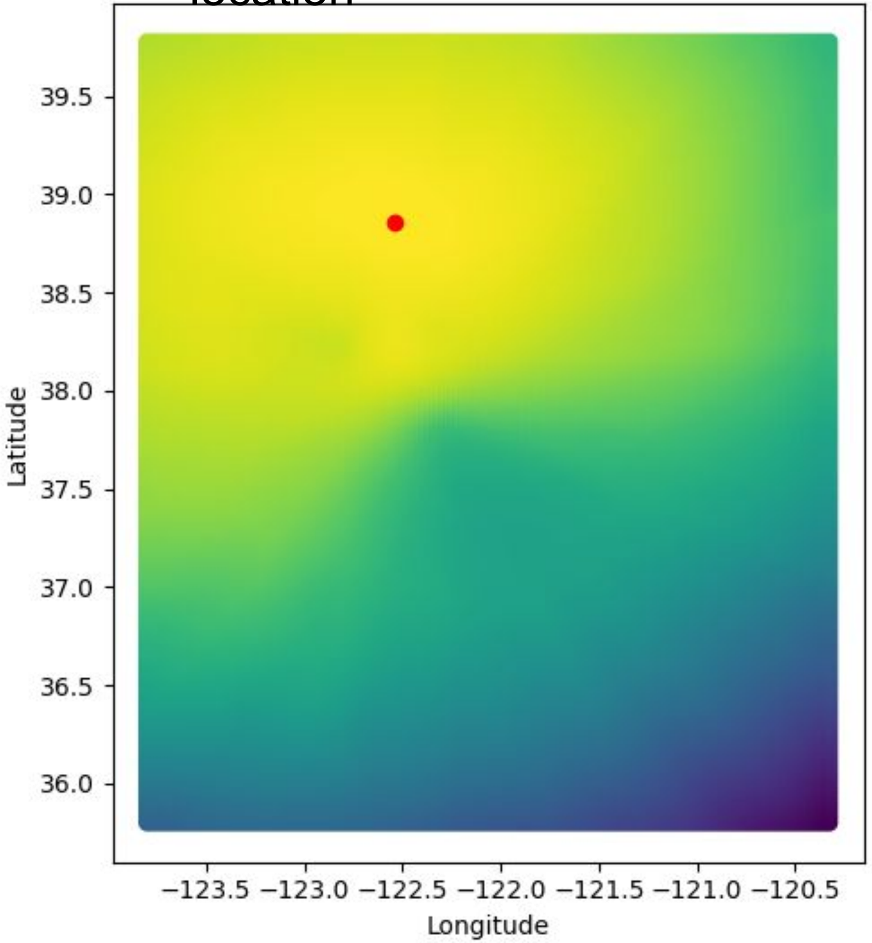
Application: Earthquake Location



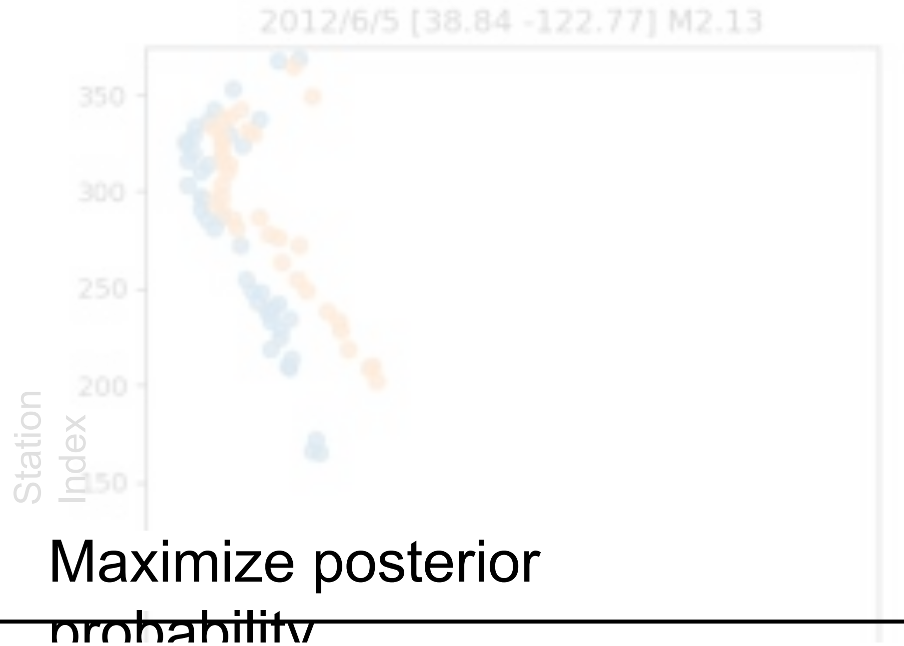


Inversion

Maximum likelihood location



Application: Earthquake Location



Inversion

Maximum likelihood location

Maximize posterior probability

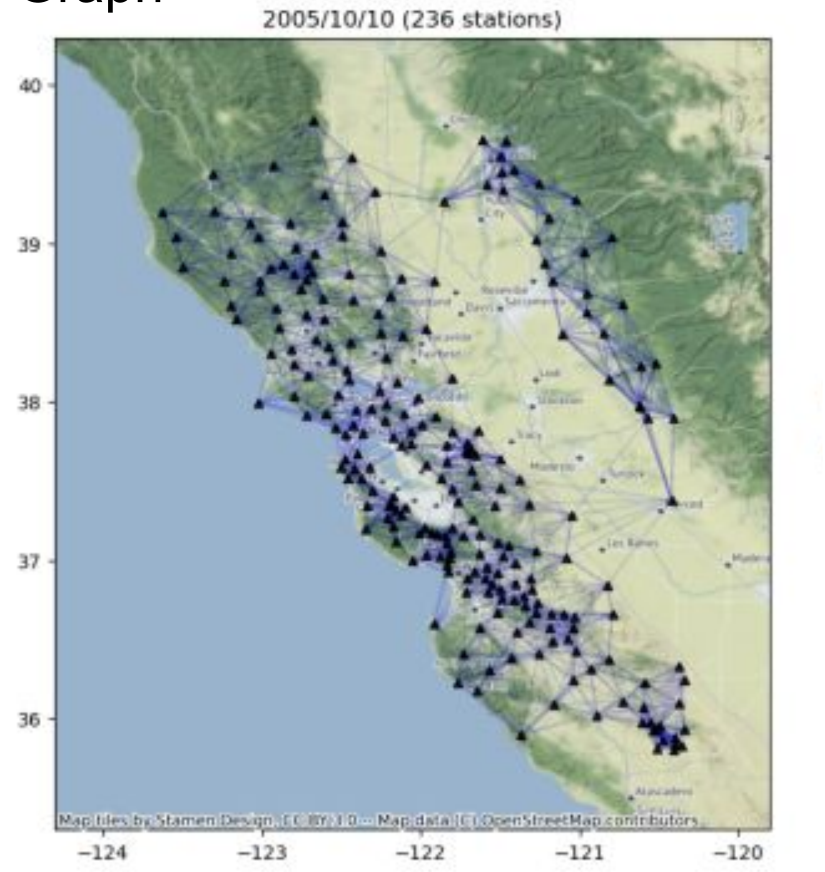
$$p(\boldsymbol{x}, t) = \exp\Big(-\frac{(t + T(\boldsymbol{x}, \boldsymbol{s}_i) - \tau_i)^T C_{cov}^{-1}(t + T(\boldsymbol{x}, \boldsymbol{s}_i) - \tau_i)}{2}\Big)$$

$\boldsymbol{x}$: source location $\boldsymbol{s}_i$: i^{th} station

t: origin time τ_i: pick time on i^{th} station

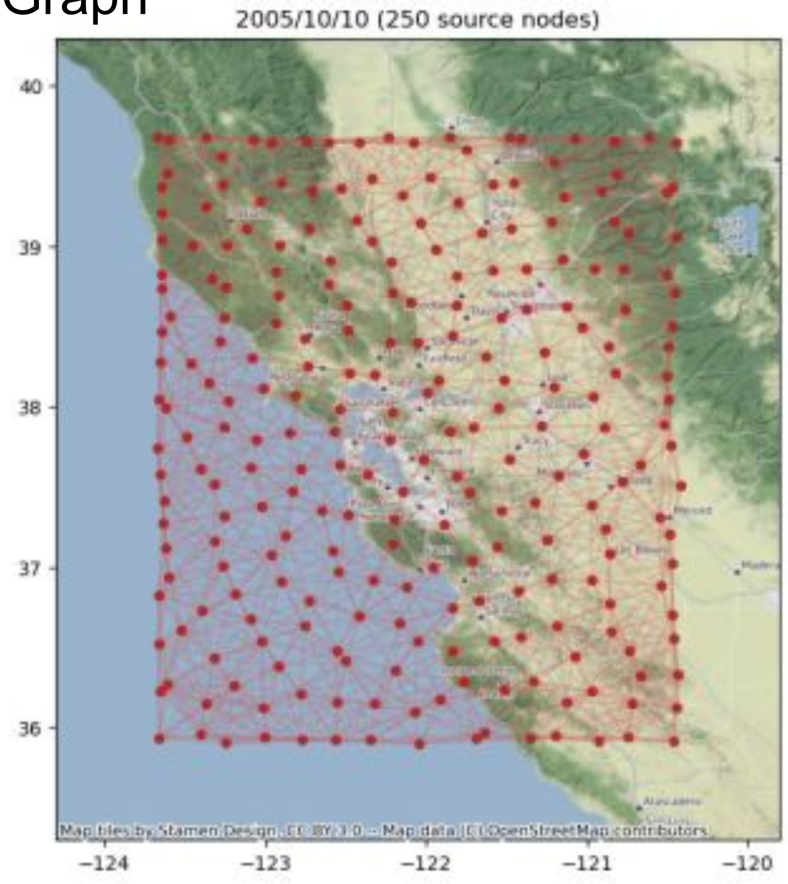
Input Graphs

Station
Graph



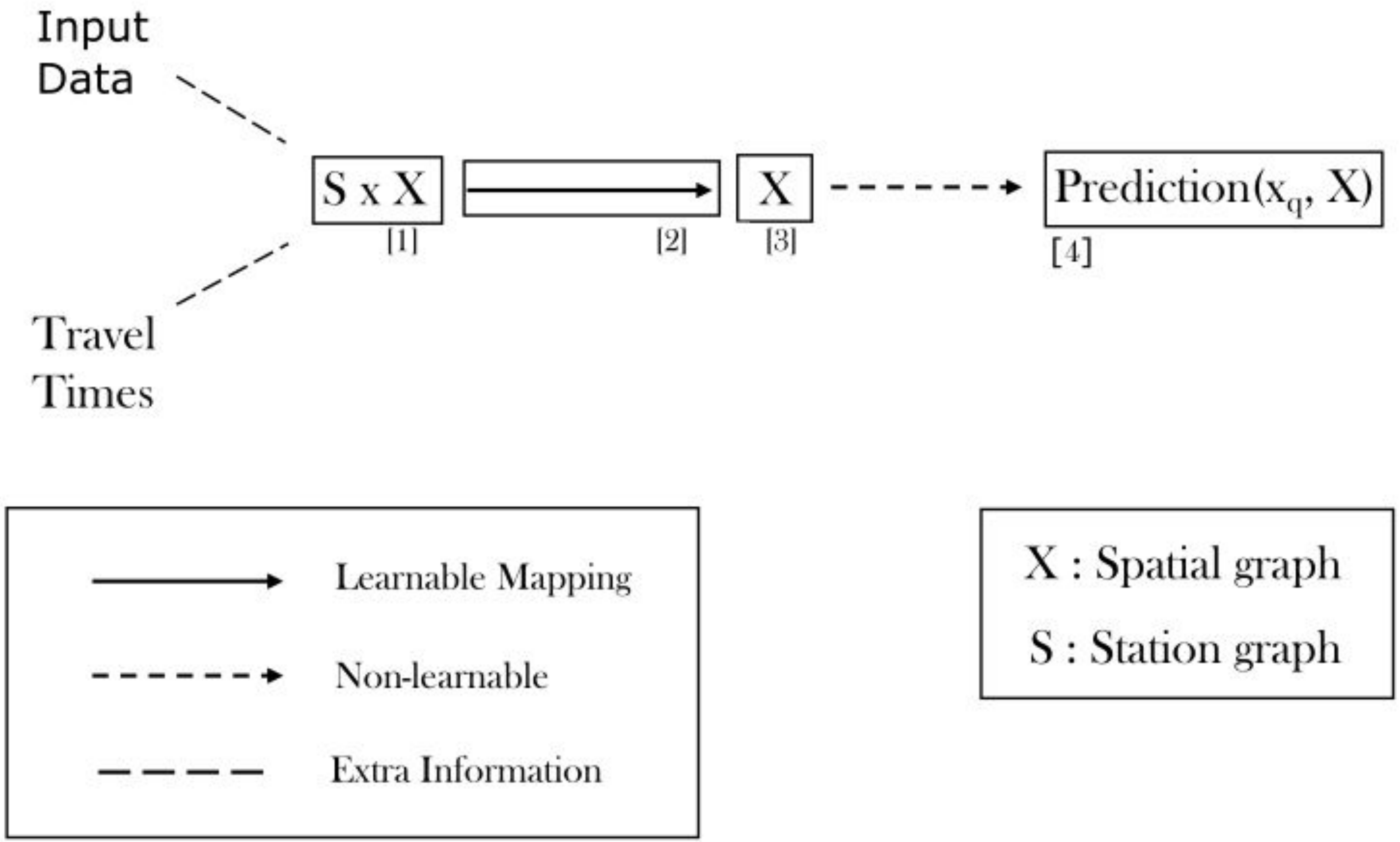
8-nearest-neighbo
rs

Spatial
Graph

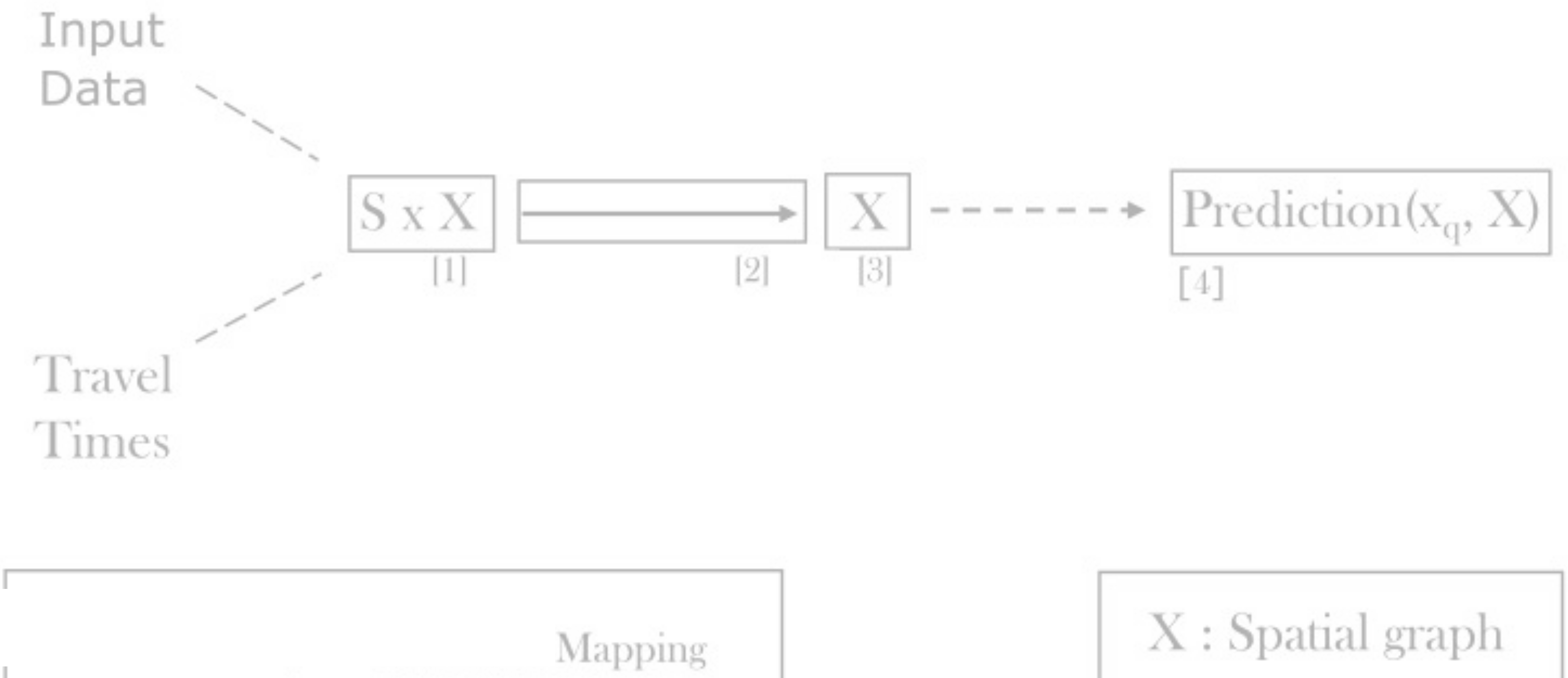


15-nearest-neighbo
rs

GNN: Architecture



GNN: Architecture



Input feature:

$$\boldsymbol{h}_k^{\mathcal{X}}(\boldsymbol{s}_i, \boldsymbol{x}) = \exp\left(-\frac{\left(t_0 + T_k(\boldsymbol{s}_i, \boldsymbol{x}) - \tau_i^k\right)^2}{2\sigma_t^2}\right)$$

pre-stack BP metric

GNN: Architecture

Cartesian Product graph:

S x X

Nodes: all pairs of (s,x)

Edges:

$$\mathcal{E}_{\mathcal{X}\leftarrow\mathcal{X},\mathcal{S}} = \left\{(i,j) \mid \boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i) \wedge (\boldsymbol{s}_j = \boldsymbol{s}_i)\right\}$$

$$\mathcal{E}_{\mathcal{S}\leftarrow\mathcal{S},\mathcal{X}} = \left\{(i,j) \mid \boldsymbol{s}_j \in \mathcal{N}(\boldsymbol{s}_i) \wedge (\boldsymbol{x}_j = \boldsymbol{x}_i)\right\}$$

Prediction(x_q, X)

[4]

Learnable Mapping

Non-learnable

Extra Information

X : Spatial graph

S : Station graph

GNN: Architecture

Cartesian Product graph:

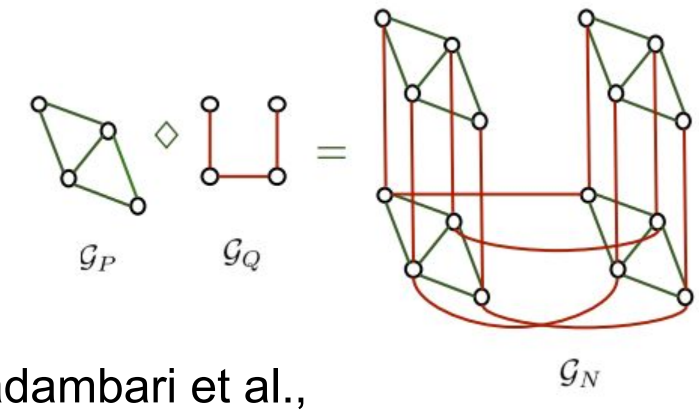
$\boxed{\mathrm{S \times X}}$

Nodes: all pairs of (s,x)

Edges:

$$\mathcal{E}_{\mathcal{X}\leftarrow\mathcal{X},\mathcal{S}} = \{(i,j) \mid \boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i) \wedge (\boldsymbol{s}_j = \boldsymbol{s}_i)\}$$

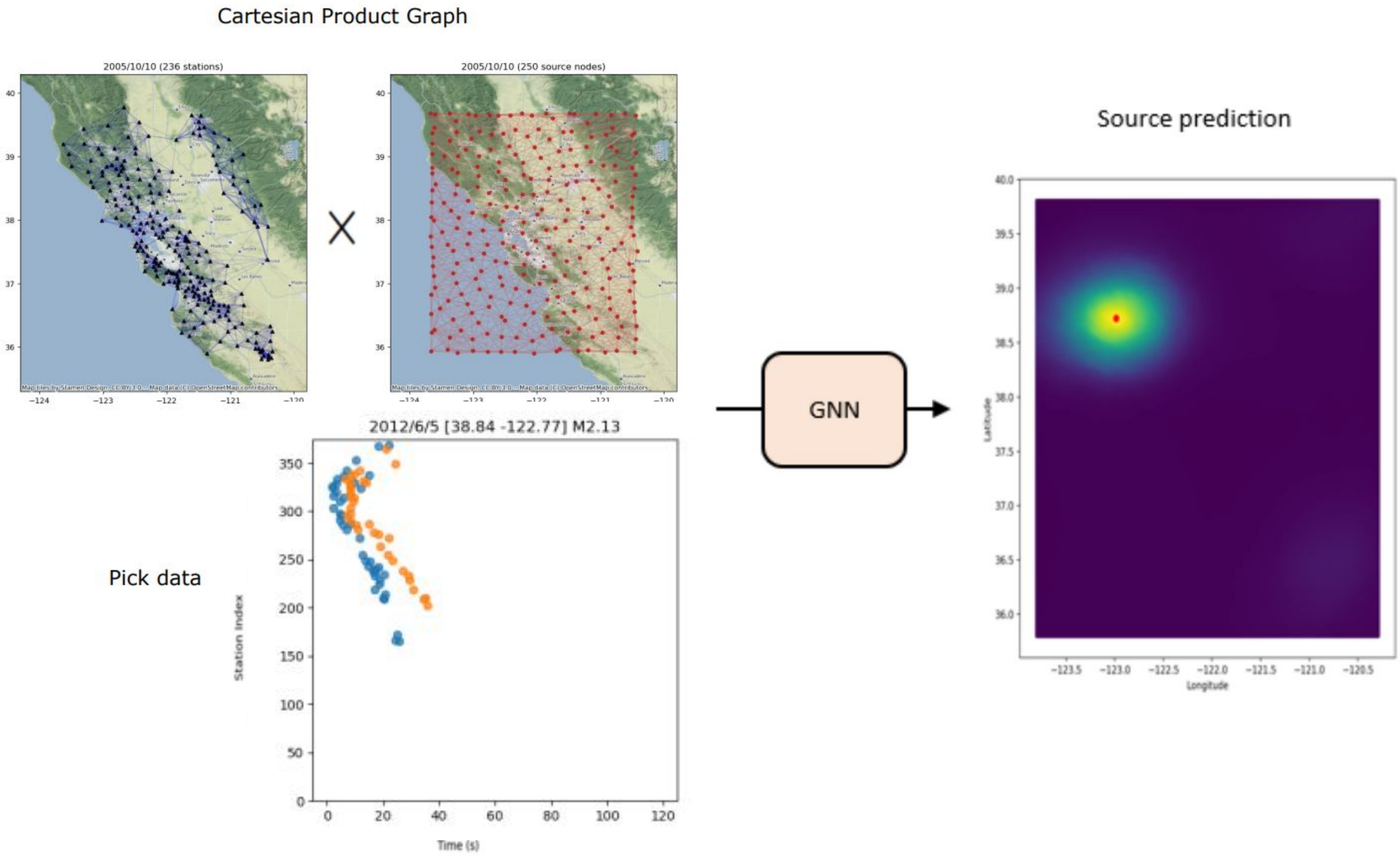
$$\mathcal{E}_{\mathcal{S}\leftarrow\mathcal{S},\mathcal{X}} = \{(i,j) \mid \boldsymbol{s}_j \in \mathcal{N}(\boldsymbol{s}_i) \wedge (\boldsymbol{x}_j = \boldsymbol{x}_i)\}$$



Kadambari et al., 2021

- Cartesian product, Kronecker product,
Strong product, Lexicographic product…

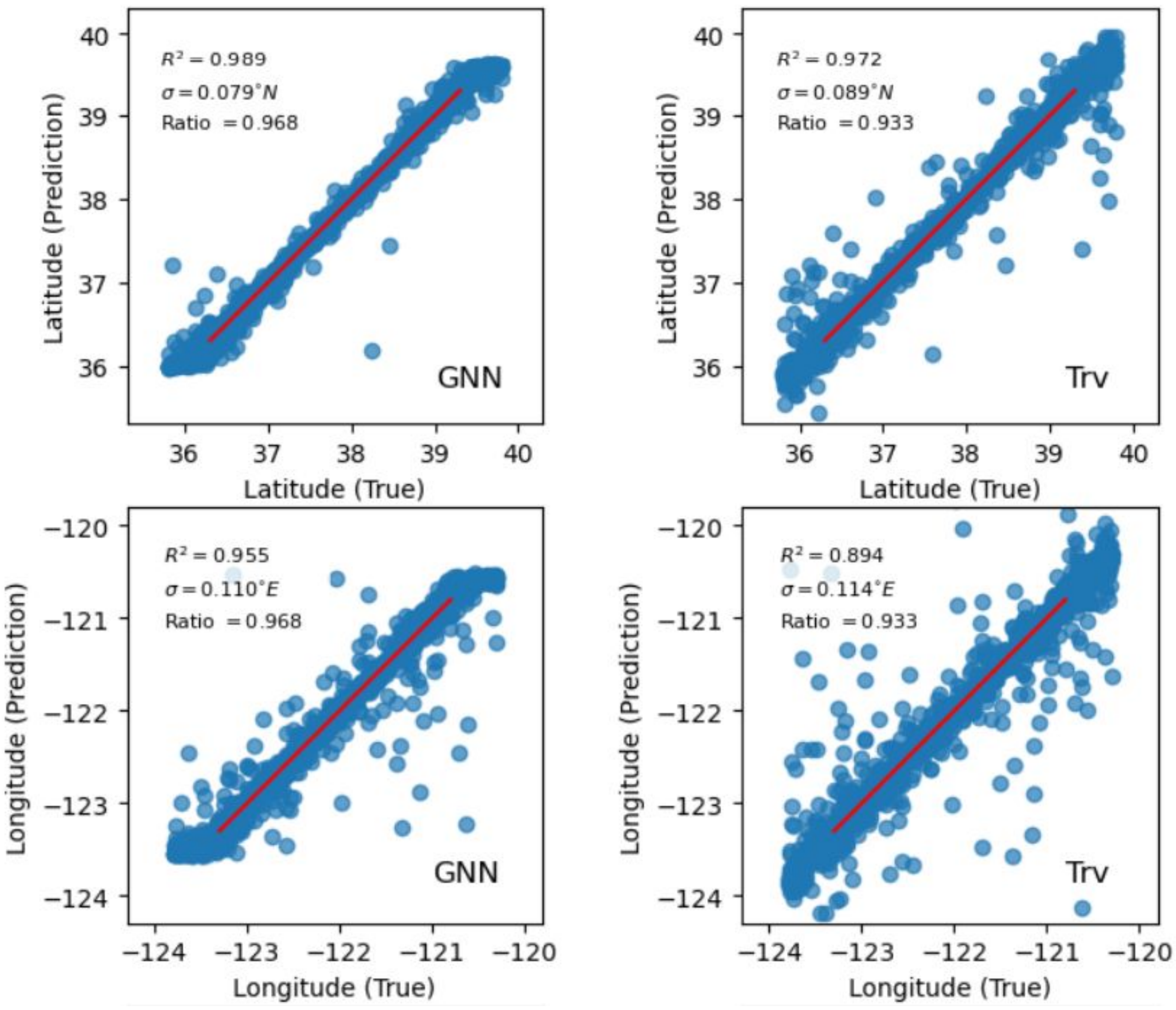
Forward Map



Cartesian Product Graph
2005/10/10 (236 stations)
2005/10/10 (250 source nodes)
X
Pick data
2012/6/5 [38.84 -122.77] M2.13
Station Index
Time (s)
GNN
Source prediction
Latitude
Longitude

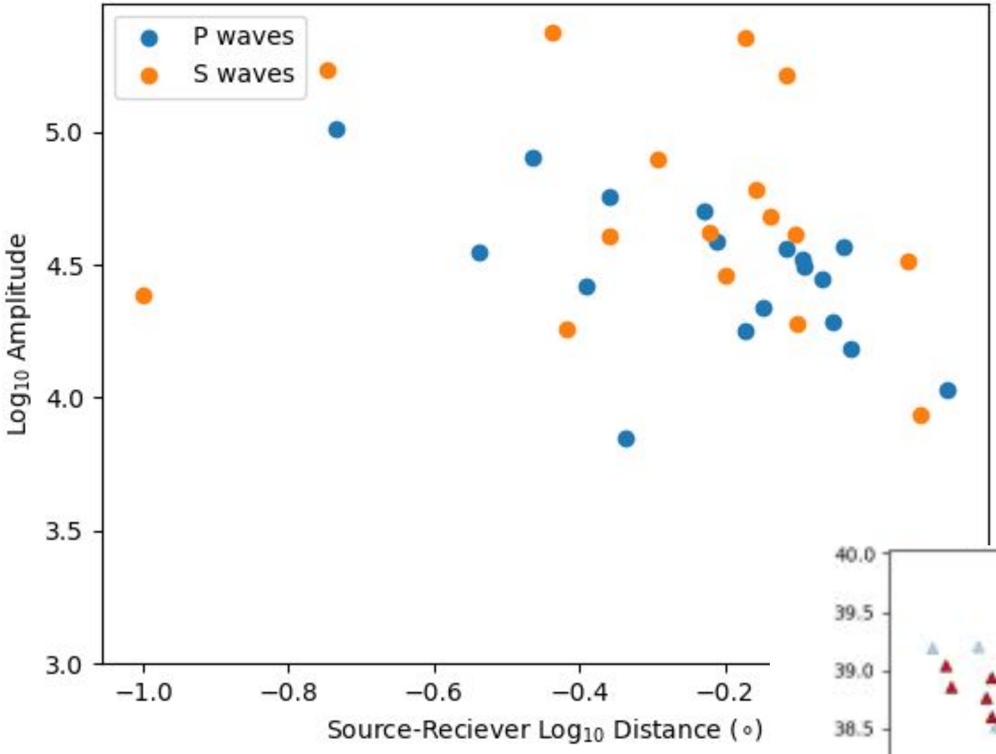
Results on Synthetic Data

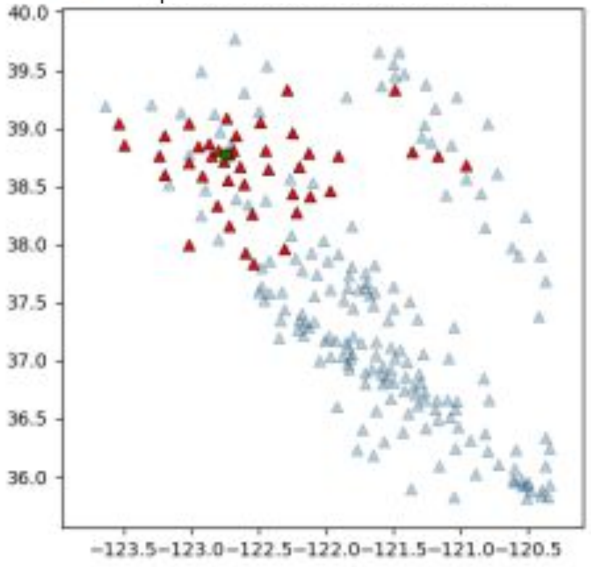
Simulated many realizations of pick data for sources (and different sets of stations) over large spatial aperture, with high levels of noise



GNN predicted source locations
Improve upon locations obtained
with traditional inversion

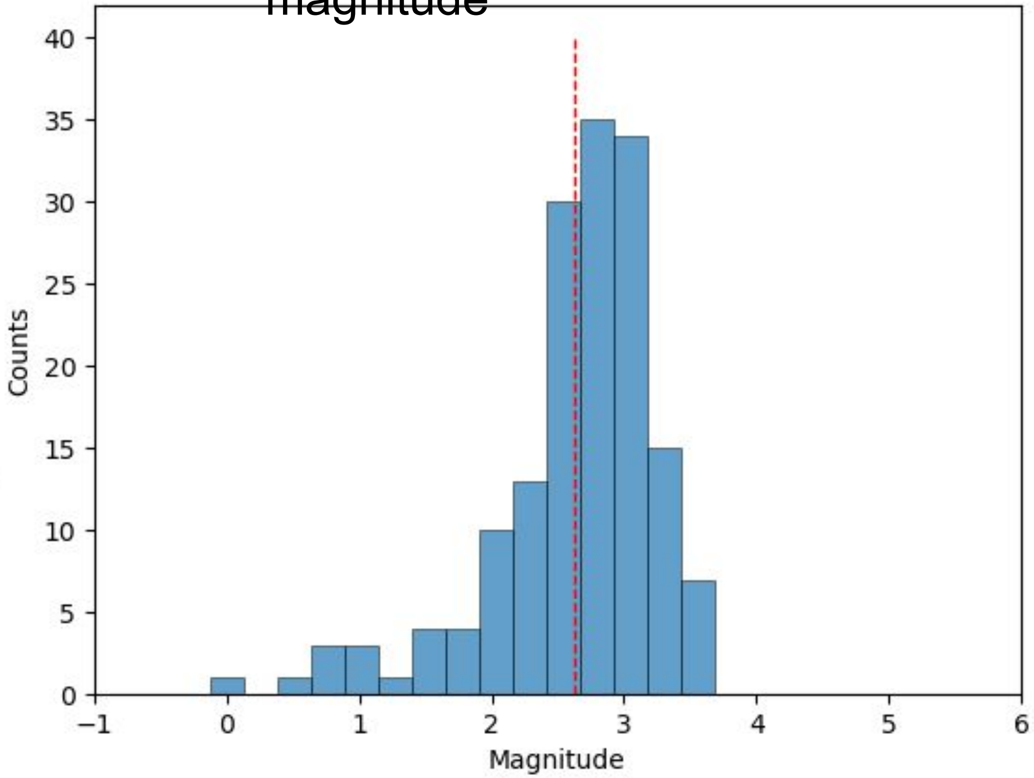
Magnitude Problem



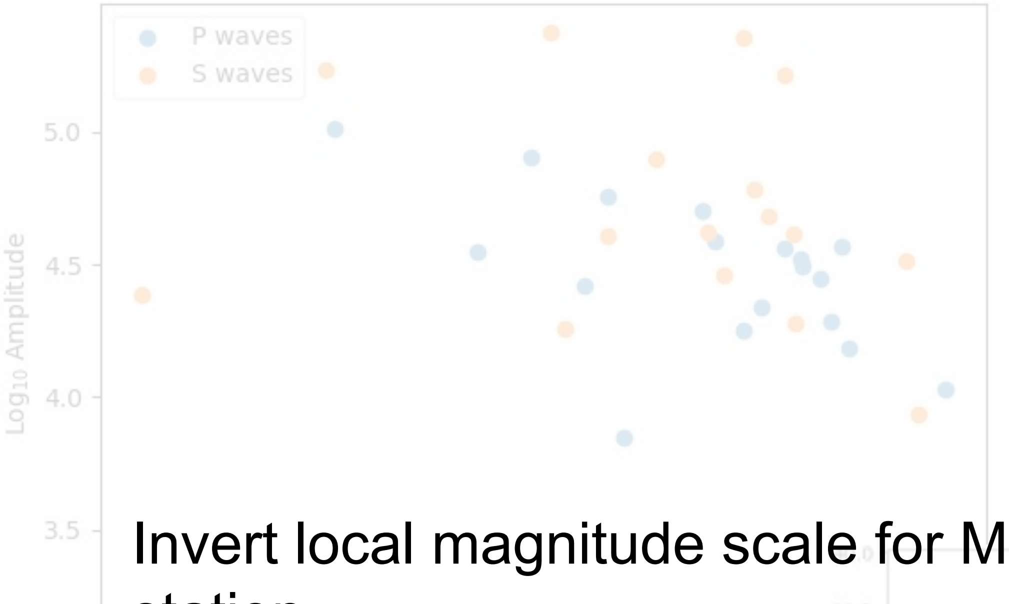


**Inversio
n**

Maximum likelihood
magnitude

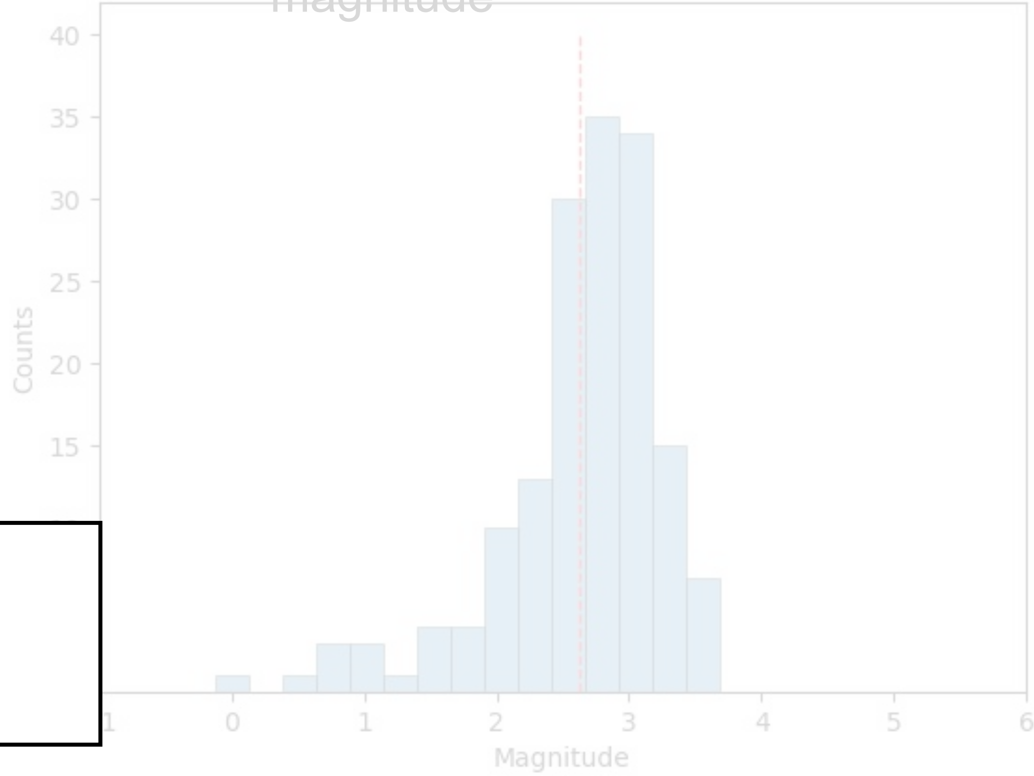


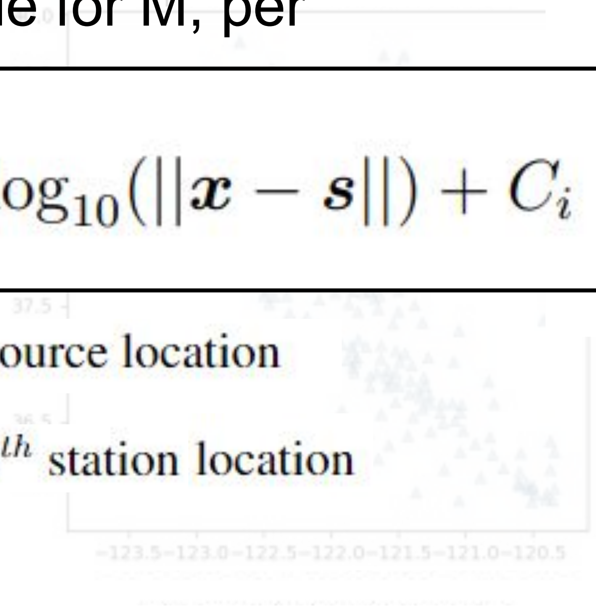
Magnitude Problem



Inversion

Maximum likelihood magnitude





Invert local magnitude scale for M, per station

$$\log_{10}(a_i) = C_1 M + C_2 \log_{10}(||\boldsymbol{x} - \boldsymbol{s}||) + C_i$$

a: arrival amplitude

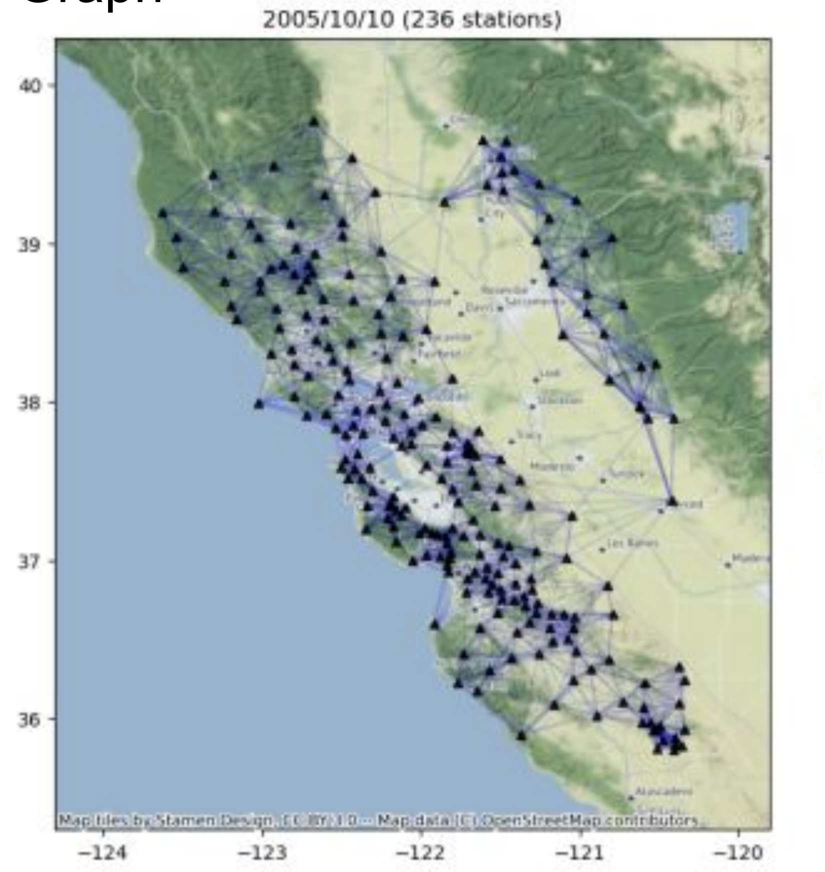
$\boldsymbol{x}$: source location

C_1, C_2, C_i: coefficients

$\boldsymbol{s}_i$: i^{th} station location

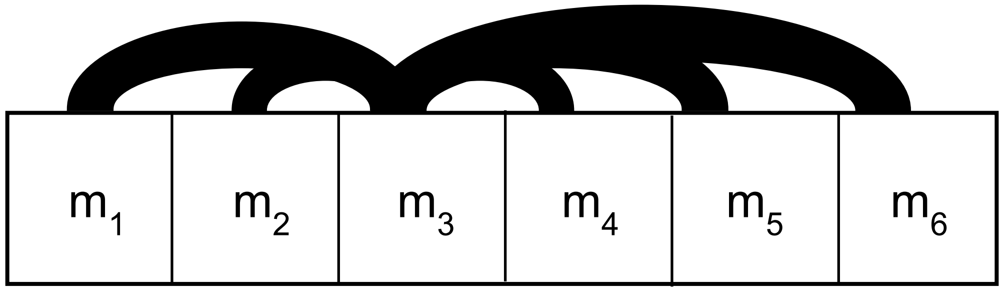
Input Graphs

Station Graph



8-nearest-neighbors

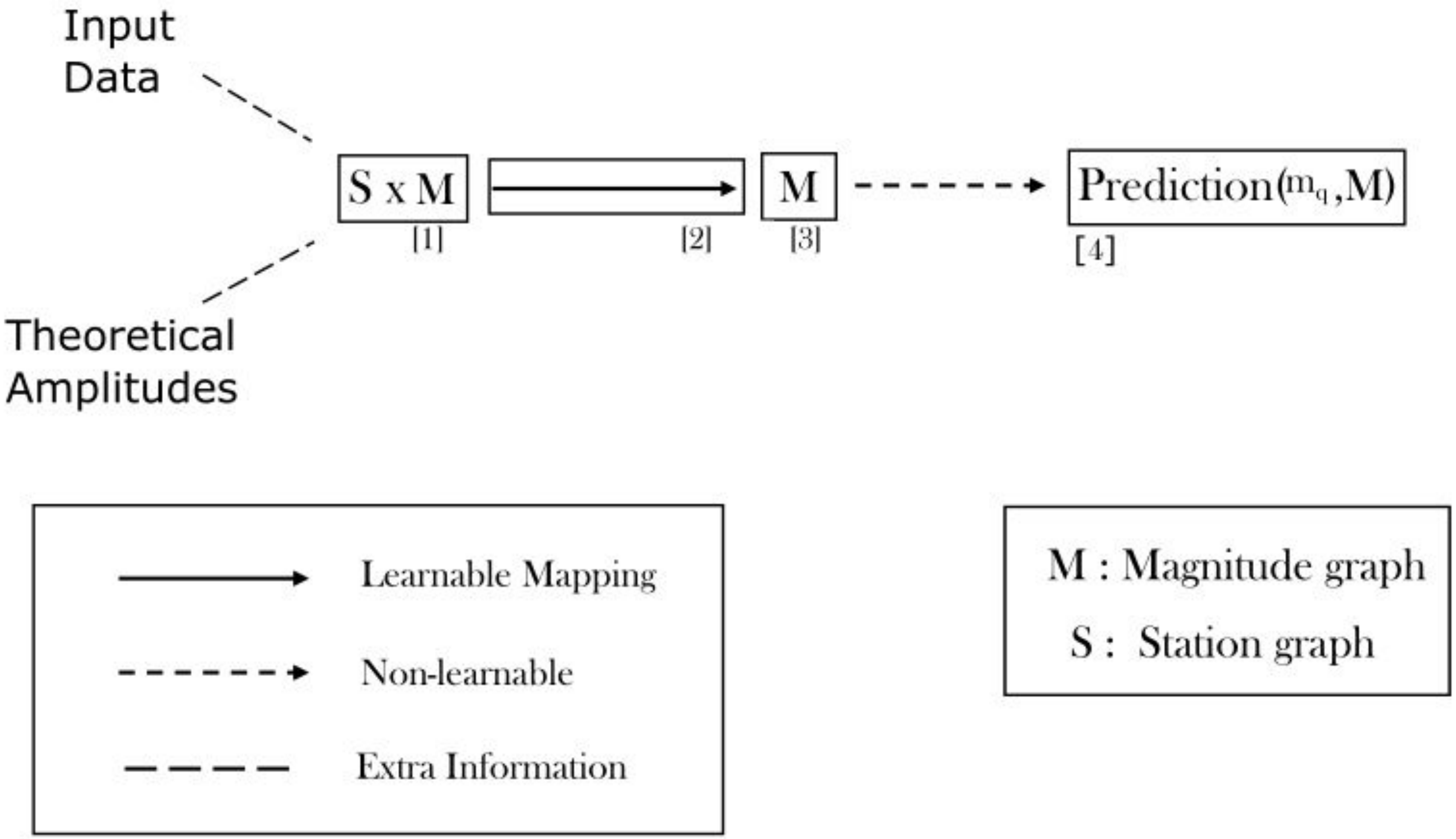
Magnitude Graph



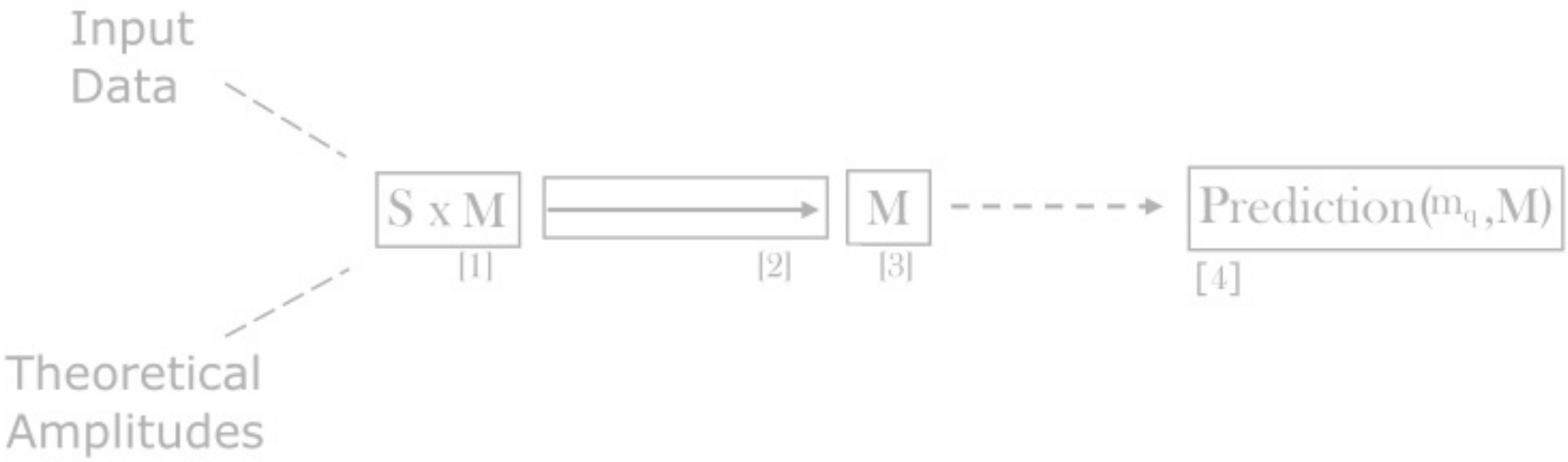
M: -3 to 7, with 0.1 M increment

10-nearest-neighbors

GNN: Architecture



GNN: Architecture



M : Magnitude graph

S : Station graph

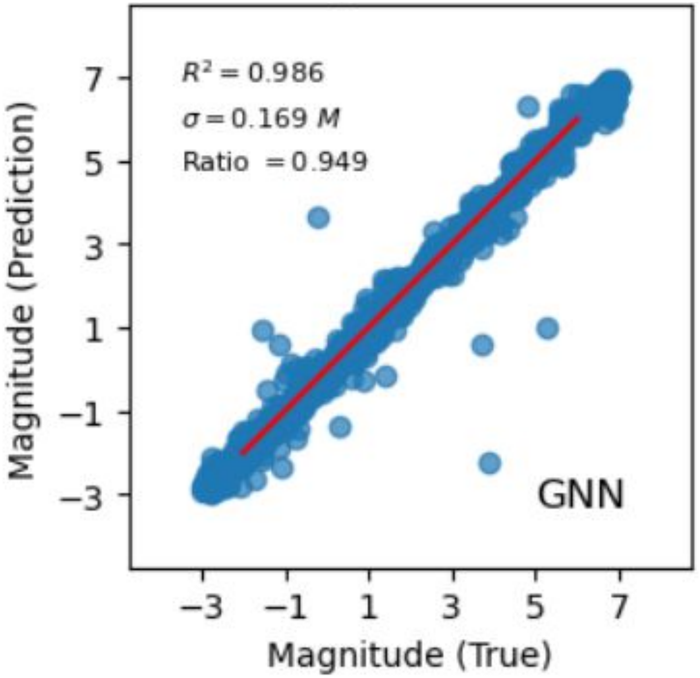
Input feature:

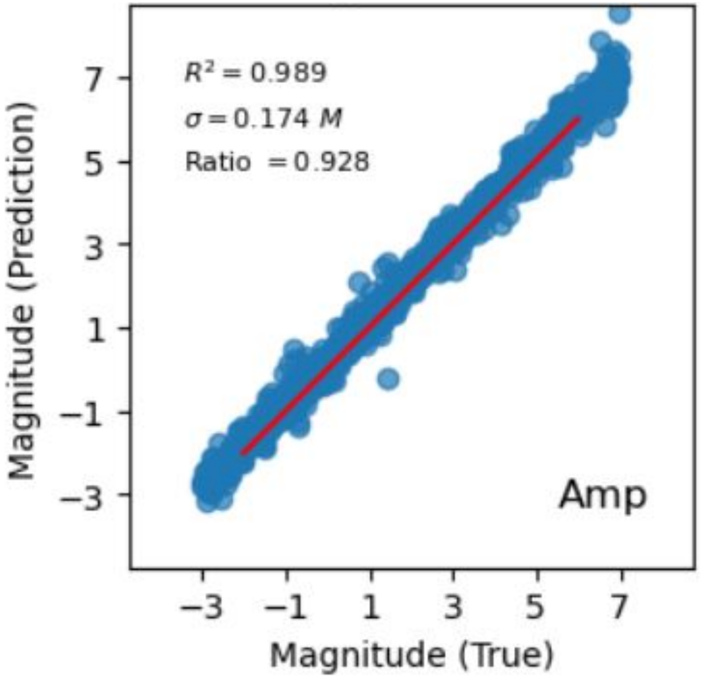
$$\boldsymbol{h}_k^{\mathcal{M}}(\boldsymbol{s}_i, \boldsymbol{m}) = \exp\left(- \frac{\left(A_k(\boldsymbol{s}_i, \boldsymbol{m}, \boldsymbol{x}) - \log_{10}(a_i^k)\right)^2}{2\sigma_a^2} \right)$$

Results on Synthetic Data

Simulated many realizations of pick data for sources (and different sets of stations) over large spatial aperture, with high levels of noise

GNN predicted source magnitudes
Improve upon magnitudes obtained
with traditional inversion





GNN: A general inverse approach

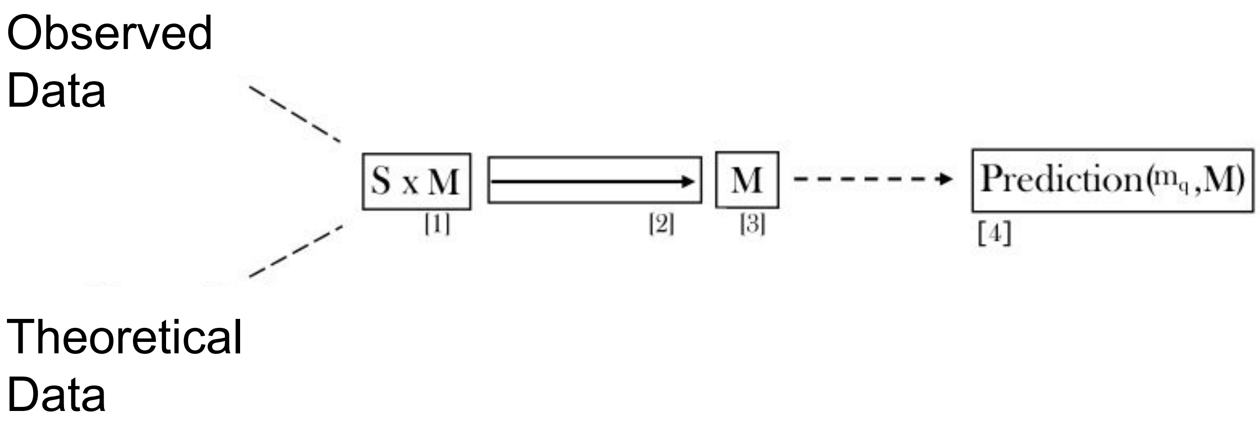
Input feature (**location**):

$$\boldsymbol{h}_k^{\mathcal{X}}(\boldsymbol{s}_i, \boldsymbol{x}) = \exp\left(-\frac{(t_0 + T_k(\boldsymbol{s}_i, \boldsymbol{x}) - \tau_i^k)^2}{2\sigma_t^2}\right)$$

Input feature (**magnitude**):

$$\boldsymbol{h}_k^{\mathcal{M}}(\boldsymbol{s}_i, \boldsymbol{m}) = \exp\left(-\frac{(A_k(\boldsymbol{s}_i, \boldsymbol{m}, \boldsymbol{x}) - \log_{10}(a_i^k))^2}{2\sigma_a^2}\right)$$

GNN: A general inverse approach



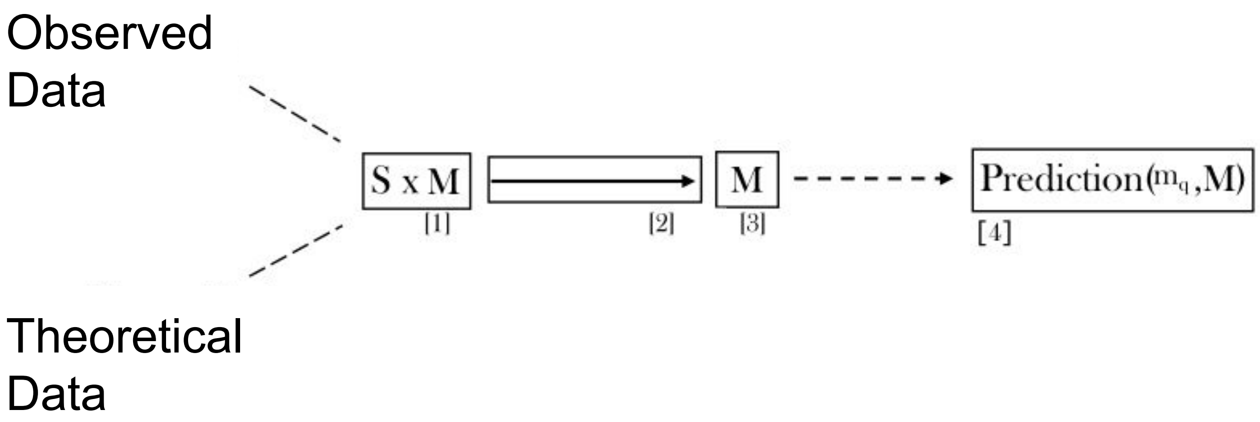
Input feature (**location**):

$$\boldsymbol{h}_k^{\mathcal{X}}(\boldsymbol{s}_i, \boldsymbol{x}) = \exp\left(-\frac{(t_0 + T_k(\boldsymbol{s}_i, \boldsymbol{x}) - \tau_i^k)^2}{2\sigma_t^2}\right)$$

Input feature (**magnitude**):

$$\boldsymbol{h}_k^{\mathcal{M}}(\boldsymbol{s}_i, \boldsymbol{m}) = \exp\left(-\frac{(A_k(\boldsymbol{s}_i, \boldsymbol{m}, \boldsymbol{x}) - \log_{10}(a_i^k))^2}{2\sigma_a^2}\right)$$

GNN: A general inverse approach



Input feature (**location**):

$$\boldsymbol{h}_k^{\mathcal{X}}(\boldsymbol{s}_i, \boldsymbol{x}) = \exp\left(-\frac{(t_0 + T_k(\boldsymbol{s}_i, \boldsymbol{x}) - \tau_i^k)^2}{2\sigma_t^2}\right)$$

Input feature (**magnitude**):

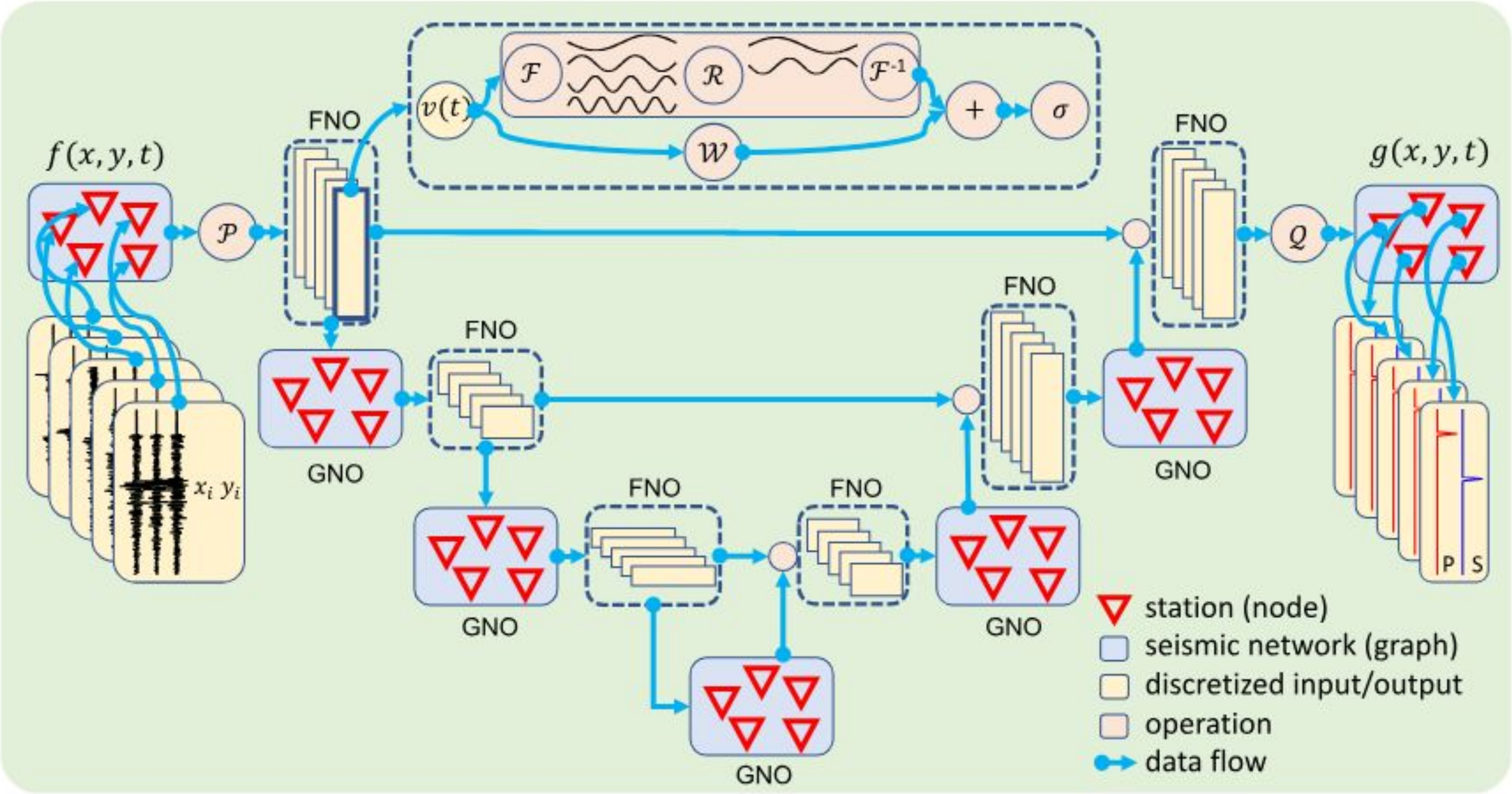
$$\boldsymbol{h}_k^{\mathcal{M}}(\boldsymbol{s}_i, \boldsymbol{m}) = \exp\left(-\frac{(A_k(\boldsymbol{s}_i, \boldsymbol{m}, \boldsymbol{x}) - \log_{10}(a_i^k))^2}{2\sigma_a^2}\right)$$

Input feature (**misfit**):

$$\boldsymbol{h}_k^{\mathcal{M}}(\boldsymbol{s}_i, \boldsymbol{m}) = \mathrm{Misfit}(s_i, f_i(m_j))$$

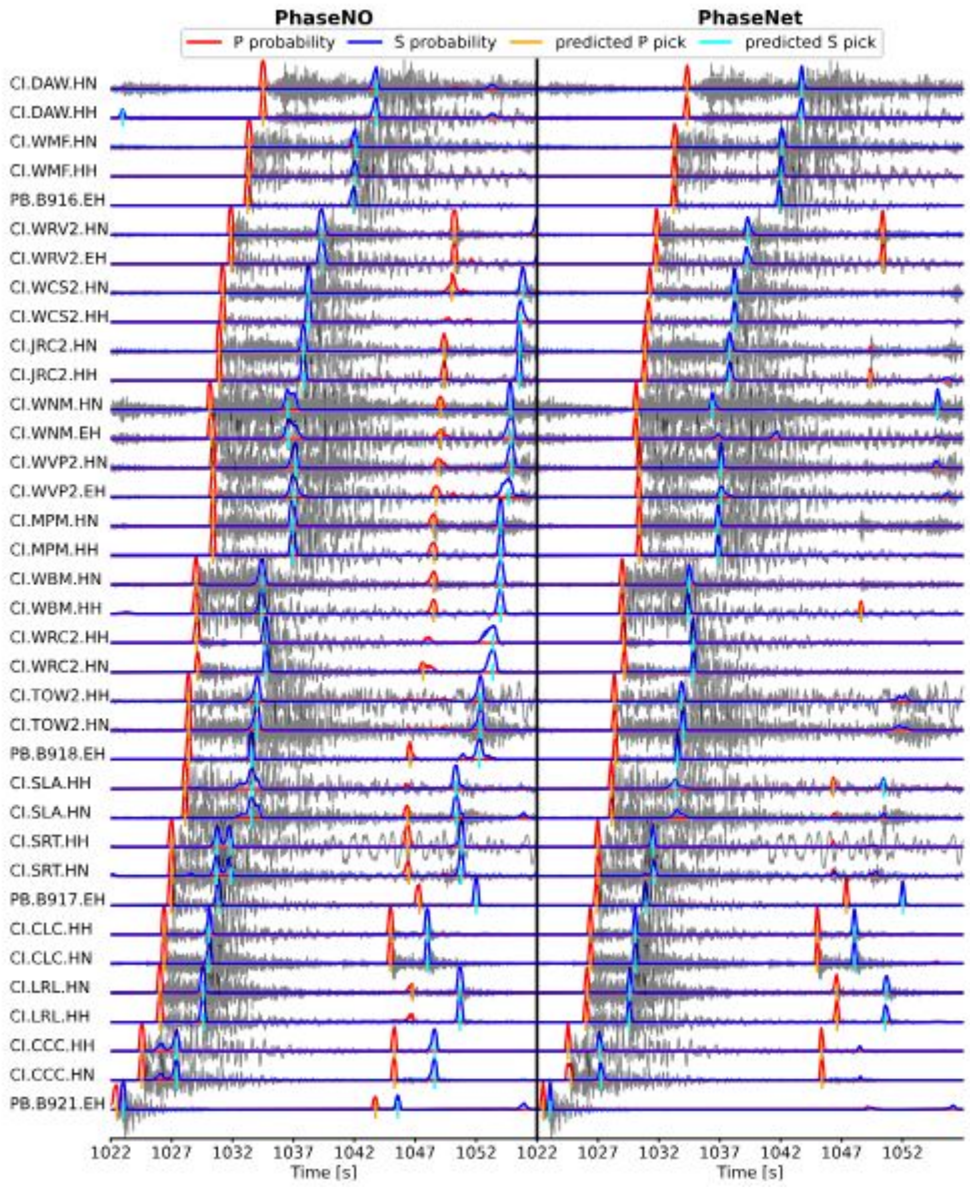
- One graph to represent **Data domain**
- One graph to represent **Model domain**

Applications



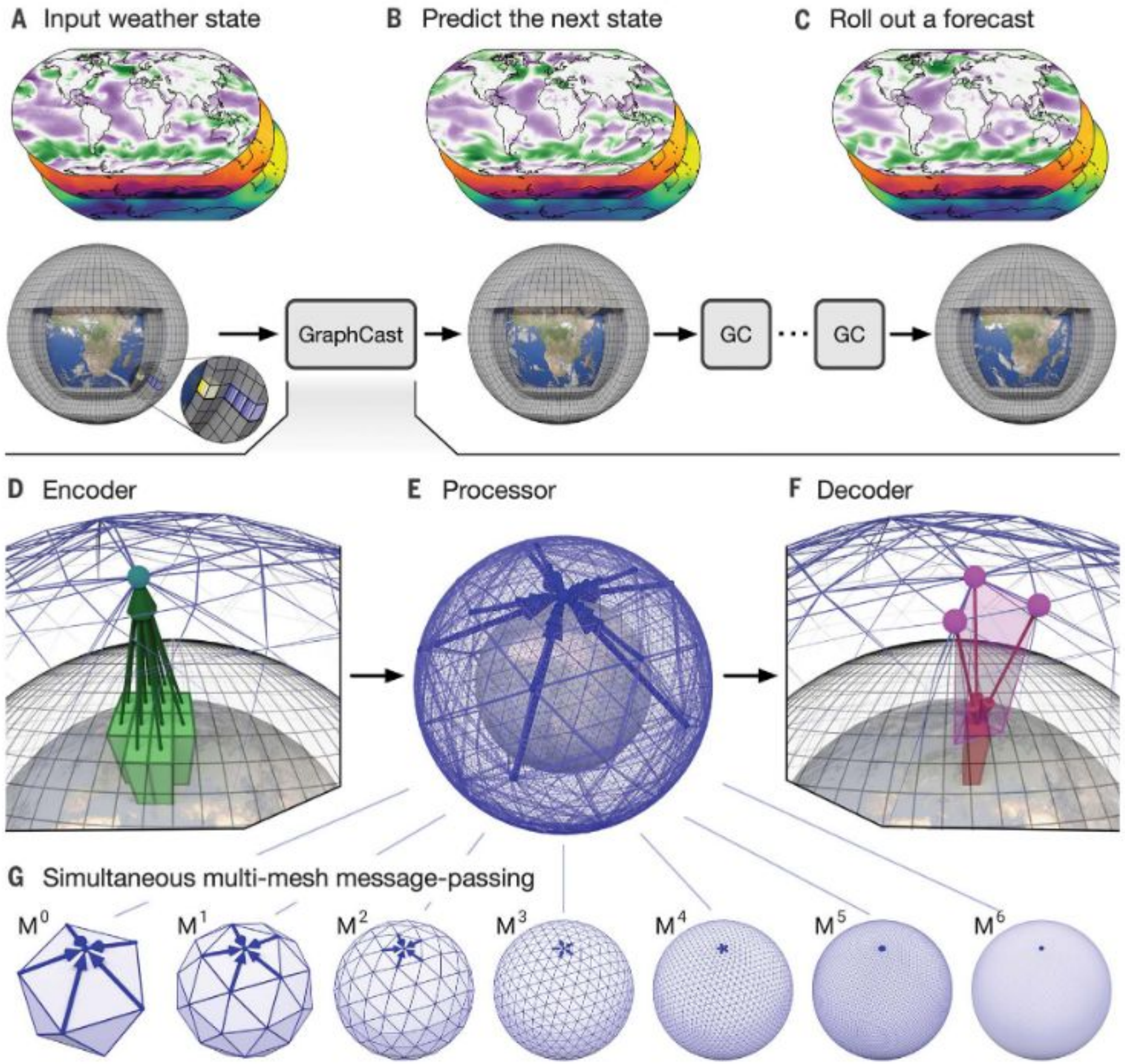
Sun et al., 2023

Applications



Sun et al., 2023

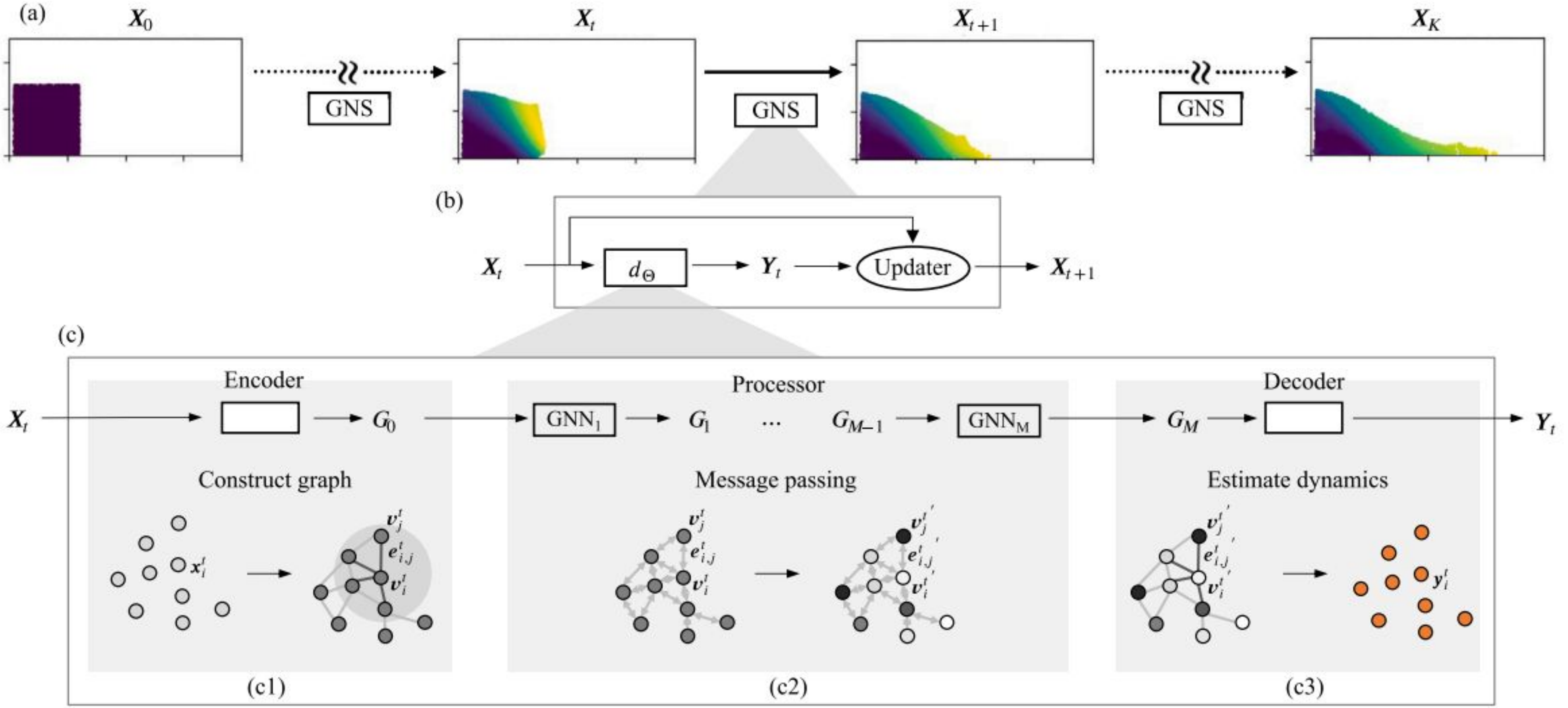
Applications



GraphCast

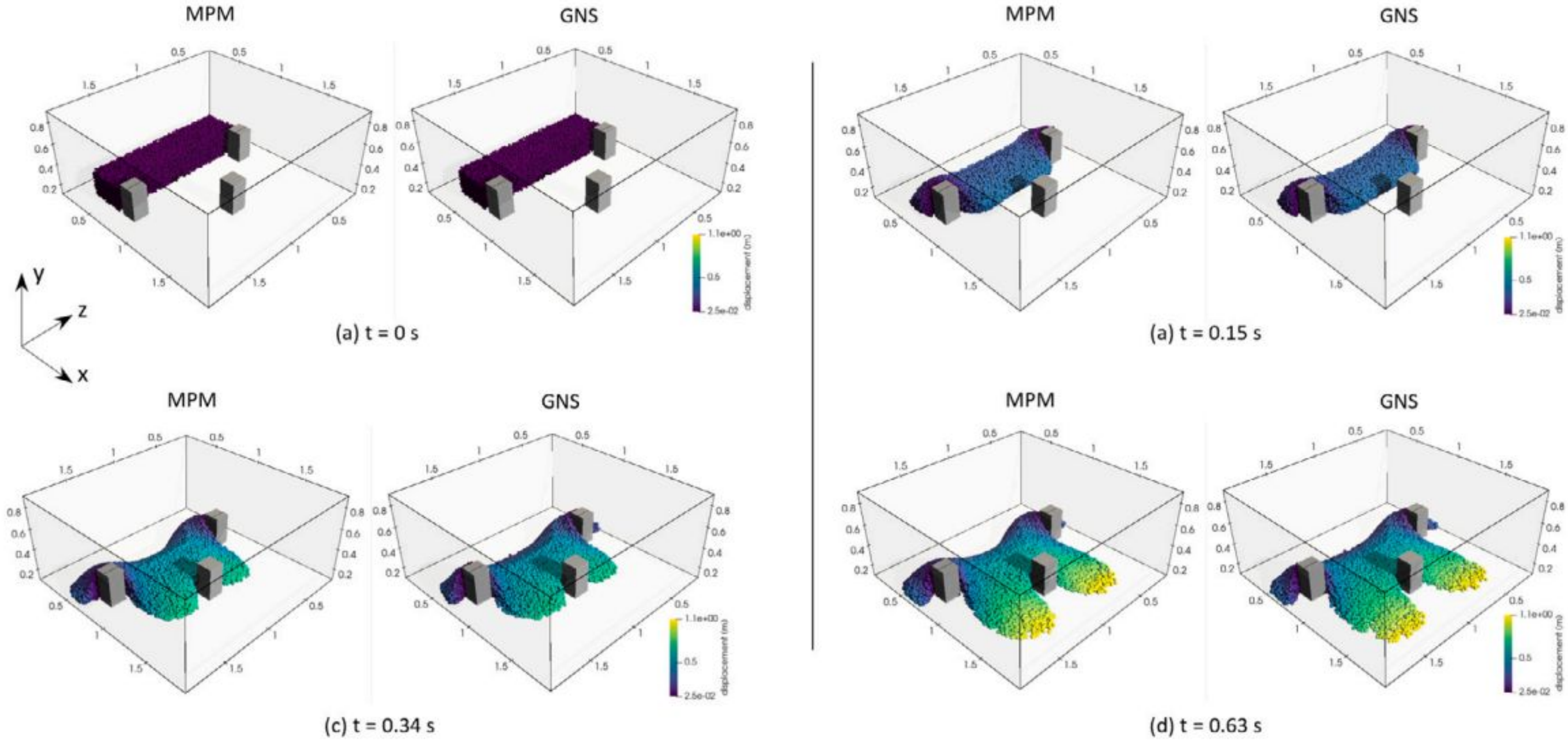
Lam et al., 2023

Applications



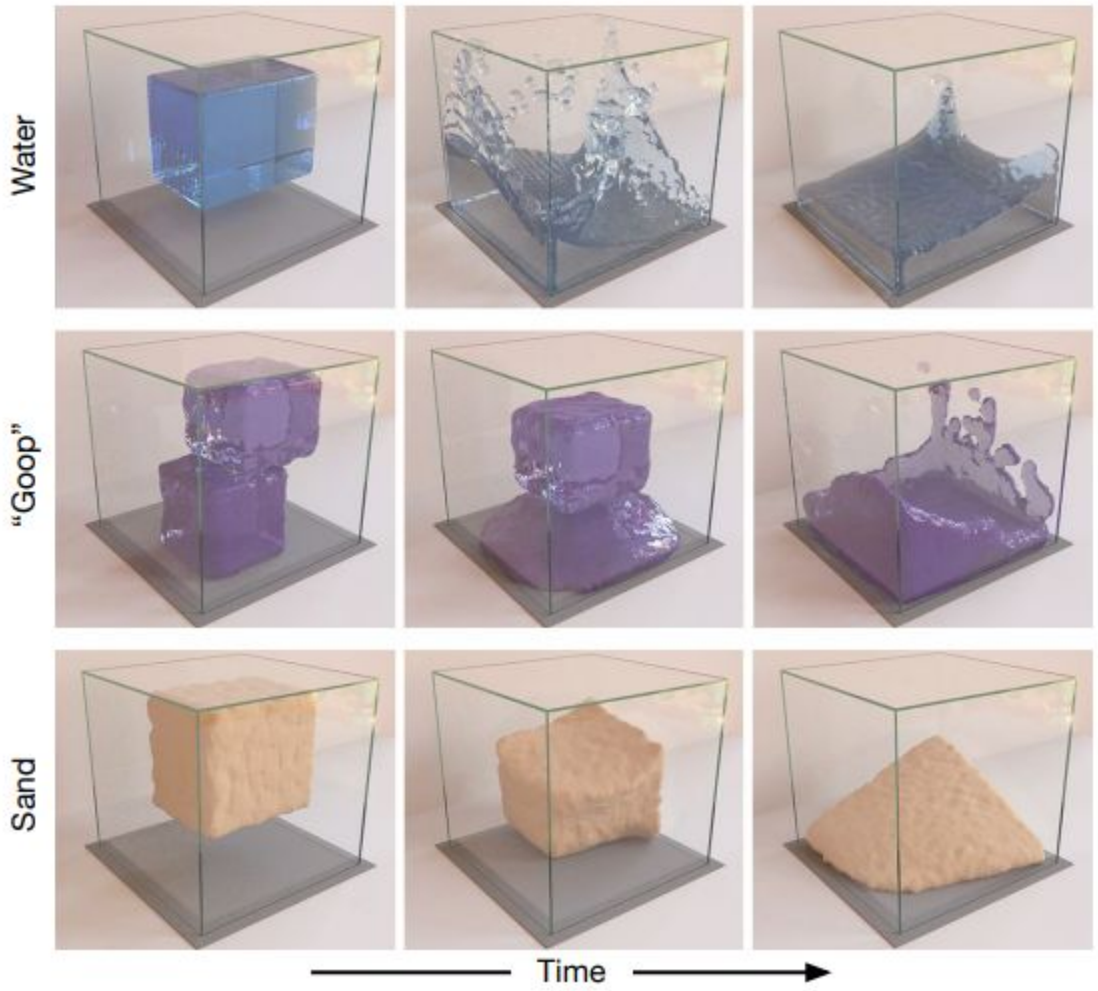
Choi and Kumar, 2024

Applications



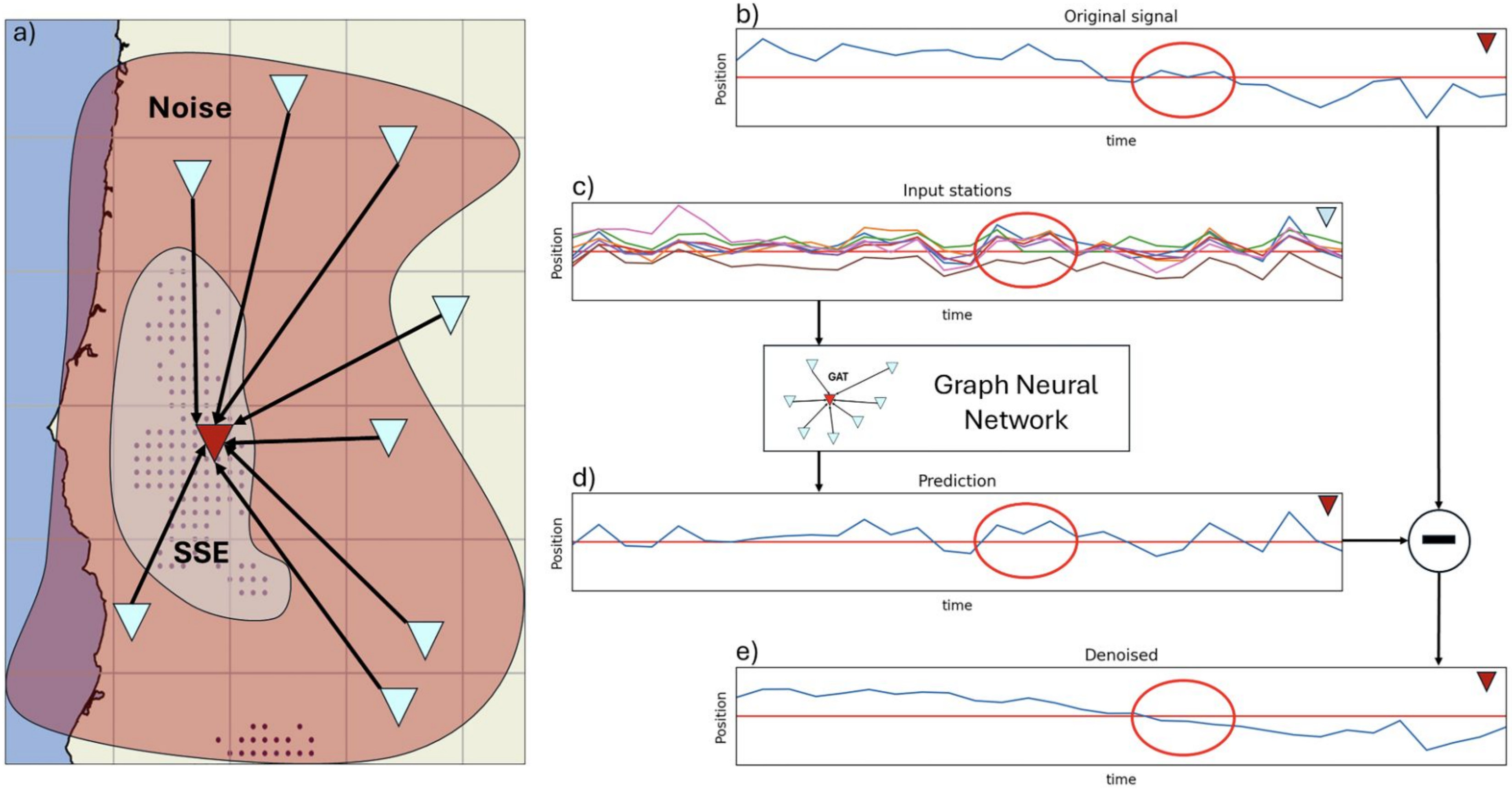
Choi and Kumar, 2024

Applications



Sanchez-Gonzalez et al., 2020

Applications



Cascadia Daily GNSS Time Series Denoising: Graph Neural Network and Stack Filtering

L. Bachelot [1], A. M. Thomas [1,2], D. Melgar [1], J. Searcy [3], Y-S. Sun [1]

[1]Department of Earth Sciences, University of Oregon, Eugene, OR, USA, [2]Department of Earth and Planetary Sciences, University of California, Davis, CA, USA, [3]School of Computer and Data Sciences, University of Oregon, Eugene, OR, USA

Applications

Automated Seismic Source Characterization Using Deep Graph Neural Networks

M. P. A. van den Ende, J.-P. Ampuero

Denoising of Geodetic Time Series Using Spatiotemporal Graph Neural Networks: Application to Slow Slip Event Extraction

Publisher: IEEE Cite This PDF

Giuseppe Costantino ; Sophie Giffard-Roisin ; Mauro Dalla Mura ; Anne Socquet **All Authors**