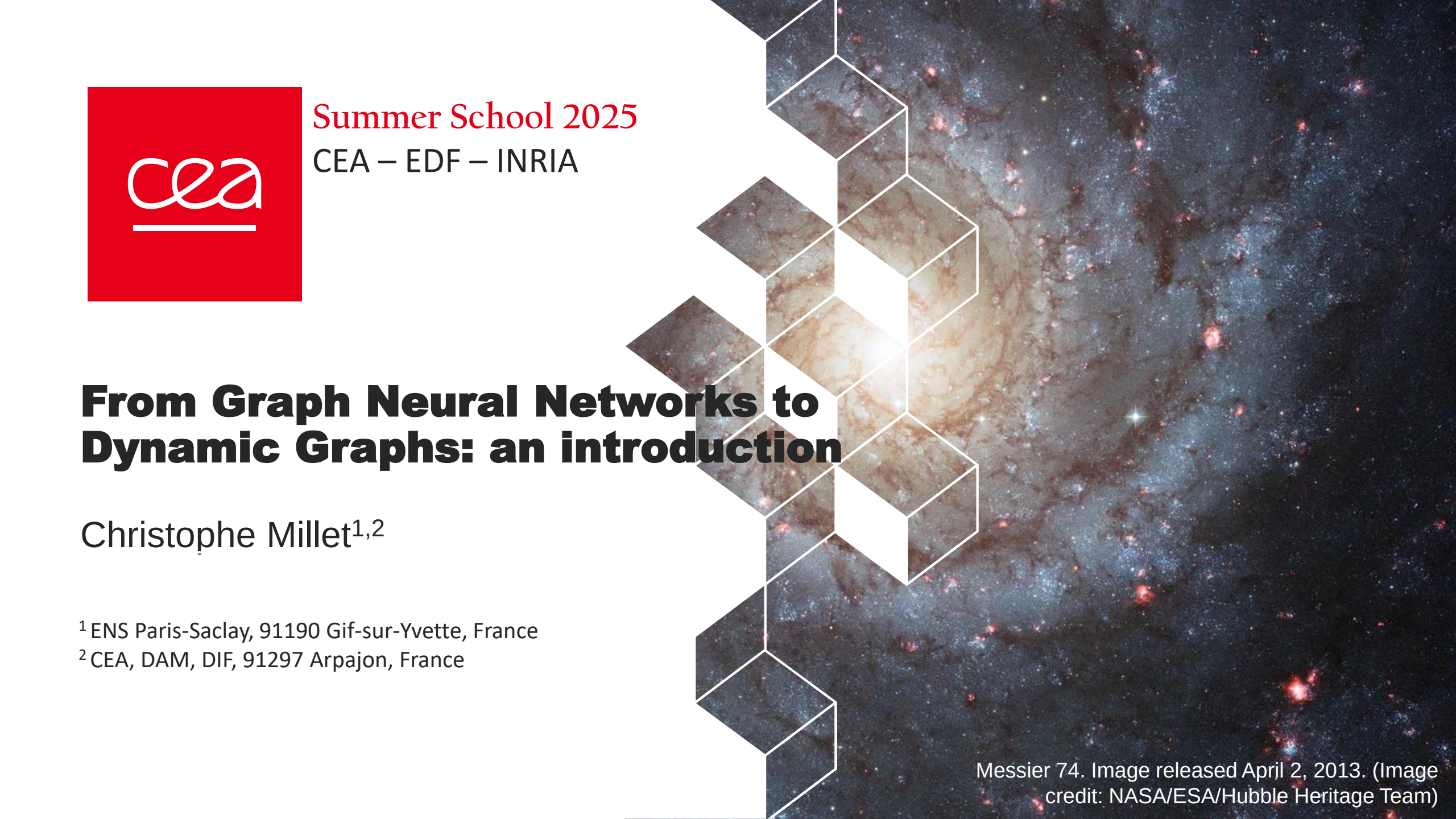


Summer School 2025

CEA – EDF – INRIA

# From Graph Neural Networks to Dynamic Graphs: an introduction

Christophe Millet[1,2]

[1] ENS Paris-Saclay, 91190 Gif-sur-Yvette, France
[2] CEA, DAM, DIF, 91297 Arpajon, France

Messier 74. Image released April 2, 2013. (Image credit: NASA/ESA/Hubble Heritage Team)

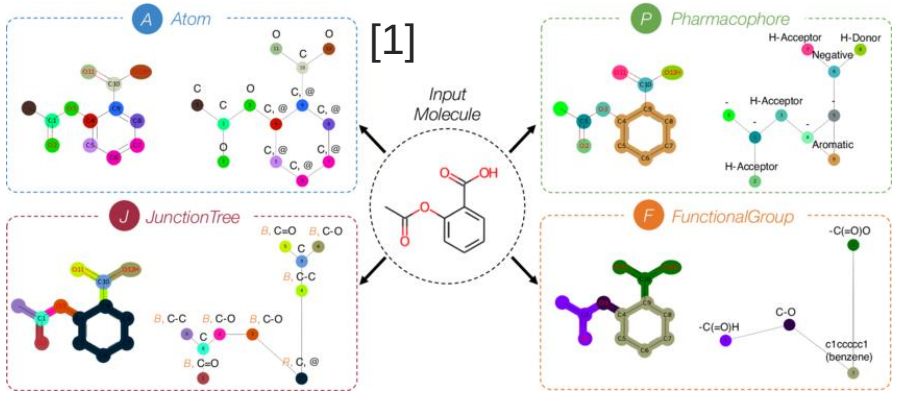

"But as a proof of concept it's a total success."

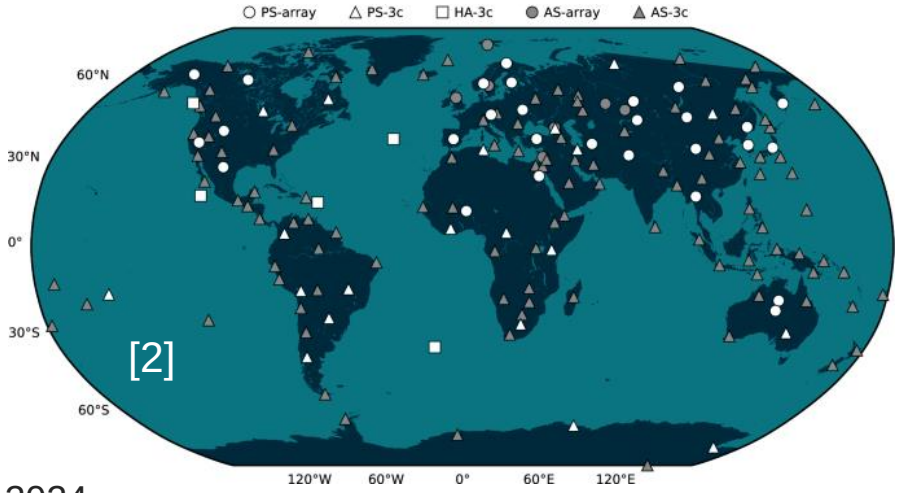
# Motivation

## ●○○○ Graphs are everywhere

- **Graphs = vertices + edges**
  - Universal representation for structured data.
  - Social networks (people, messages), biology (atoms/proteins, chemical bonds), sensor networks (stations, geodesic), recommendation (users/items, ratings), transportation (cities, roads/flights), etc.
- **Graphs help us ask — and sometimes answer — fundamental questions about structure and interaction**
  - What is the structure of a network ? Are there parts? (clustering)
  - Do the parts look the same? (similarity, isomorphism)
  - How can we model a set of graphs? (models)

**Fig. :** molecular graph abstractions [1], IMS [2].





[1] Kengkanna & Ohue, doi: 10.1038/s42004-024-0115-w, 2024.
[2] Gaebler & Ceranna, doi:10.1007/s00024-020-02604-y, 2021.

# Motivation

## ○●○○ When graphs are ignored

### About the event

- Accidental explosion (Beirut, Aug. 4, 2020).
- 2.7 kt of Ammonium Nitrate ($NH_4NO_3$).

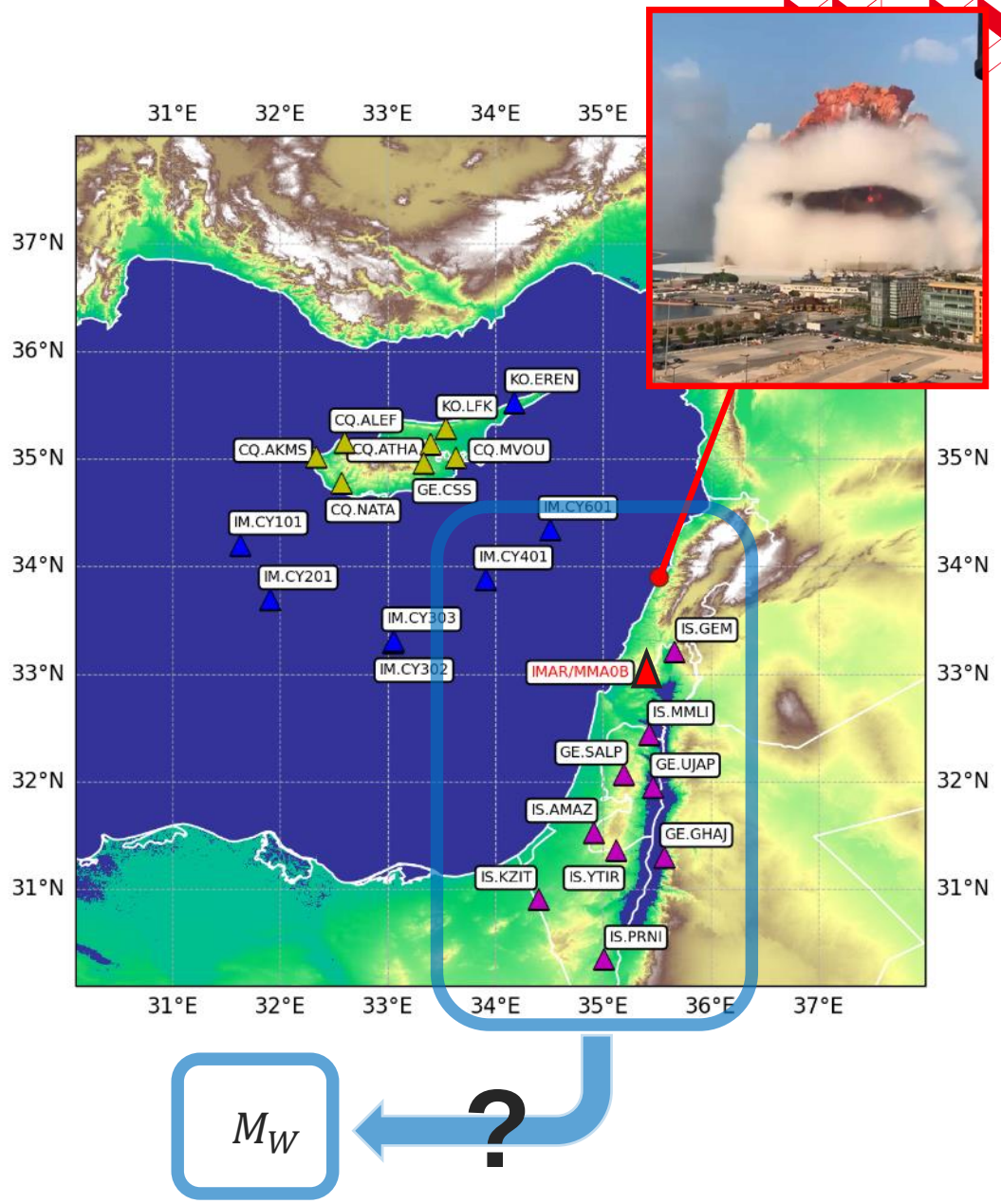
### Global to local graphs

- Recorded by the IMS and regional networks (INSN, Israel National Seismic Network).
  - Infrasound stations: $\sim 10^3$ km.
  - Seismic stations (▲▲▲): $\sim 10^2$ km.
- IS + seismic at $10^2$ km (▲).
- Publicly available videos ($\sim 10^2$ m).

$\lambda \sim 10^1$ - $10^6$ m
$N \sim 2$ - $10^2$

### Magnitude/yield?

- Many magnitude estimates from empirical laws and various tech. $\Rightarrow W \simeq 0.13 - 2$ kt TNT [*].
- Solving the inverse wave propagation problem.



# Motivation

## ○○●○ Does simulation help?

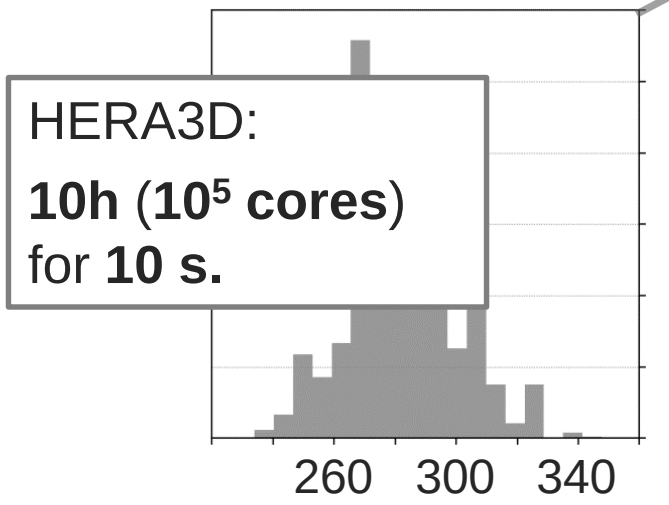
### Yield estimates (Beirut, 2020)

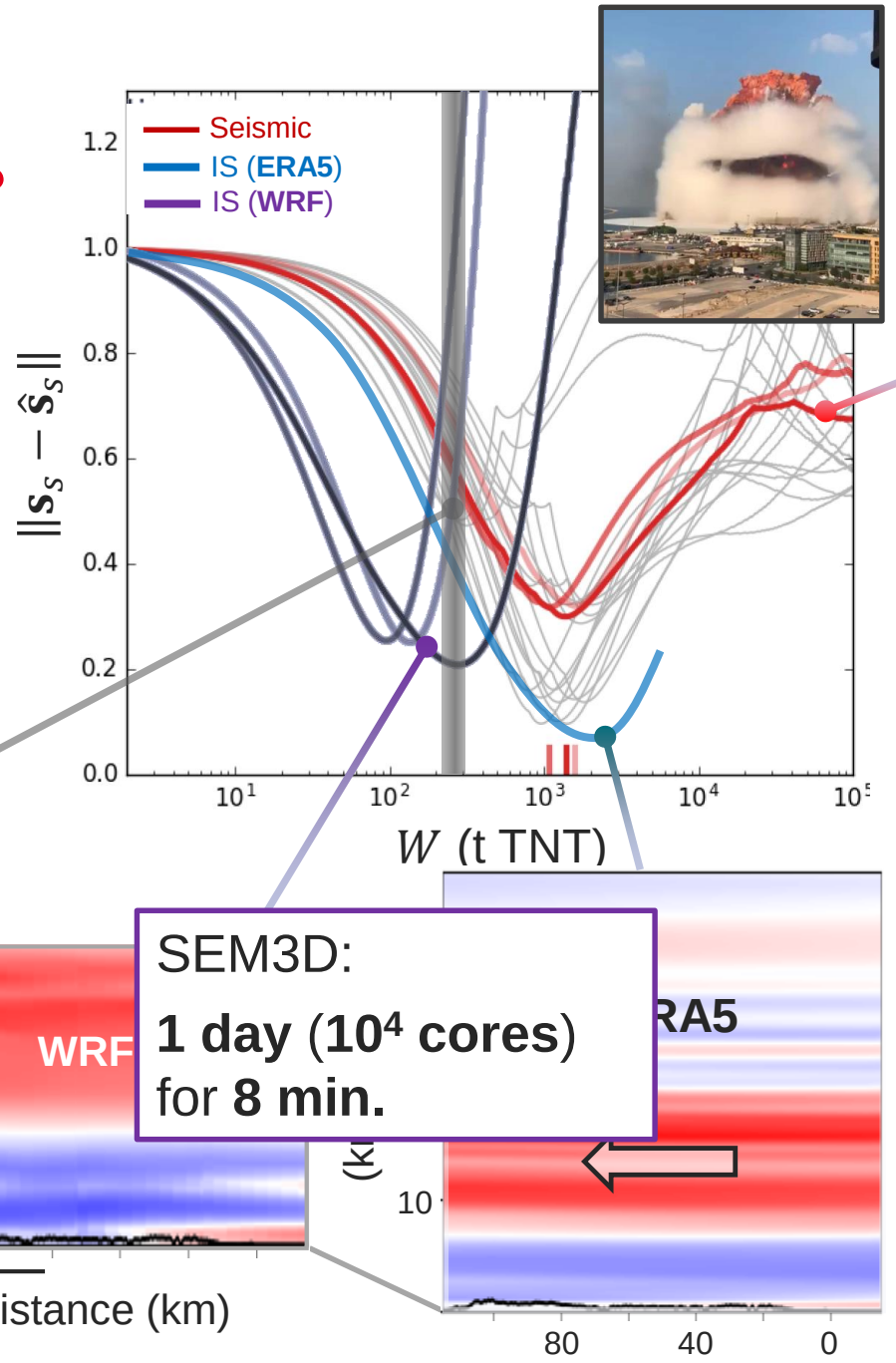
- Using **Green**'s functions of the wave equation (SEM3D) + signals.

$$\min_{W} \sum_{s=1}^{N} \|\mathbf{s}_s - \hat{\mathbf{s}}_s\|,$$

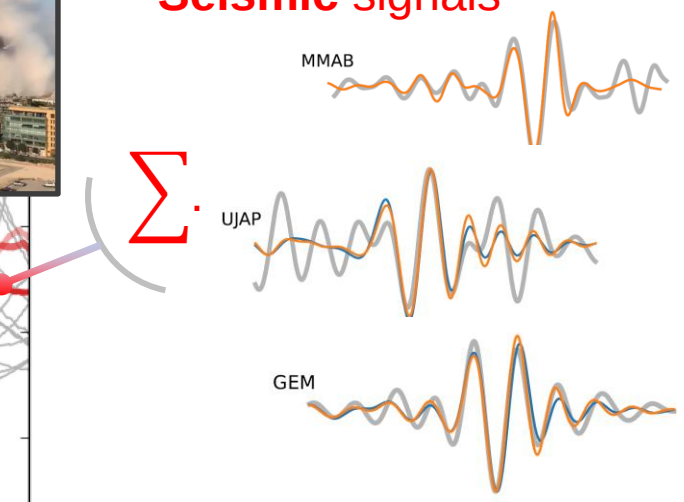
  - $\mathbf{s}_s = \mathbf{s}_0(W) *_t \mathbf{G}$; $s_0$: source model.
  - $\hat{s}$, $s$: recorded and simulated signals.

- Using a compressible flow solver and videos of **shock dynamics**.

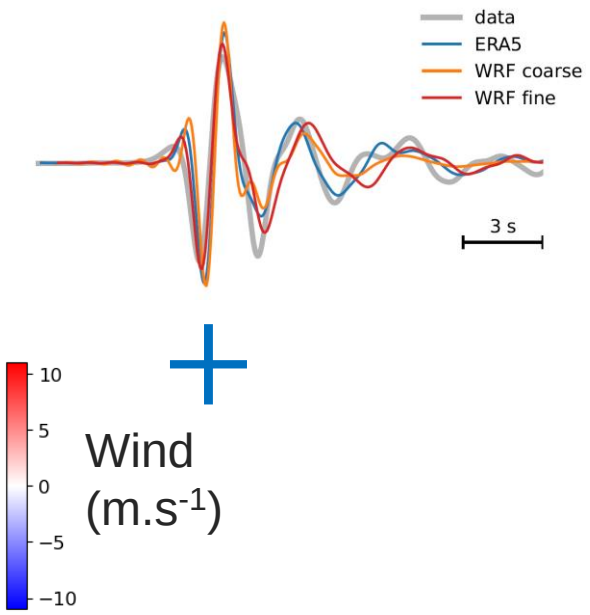



**Seismic** signals




**InfraSound (IS)**



# Motivation

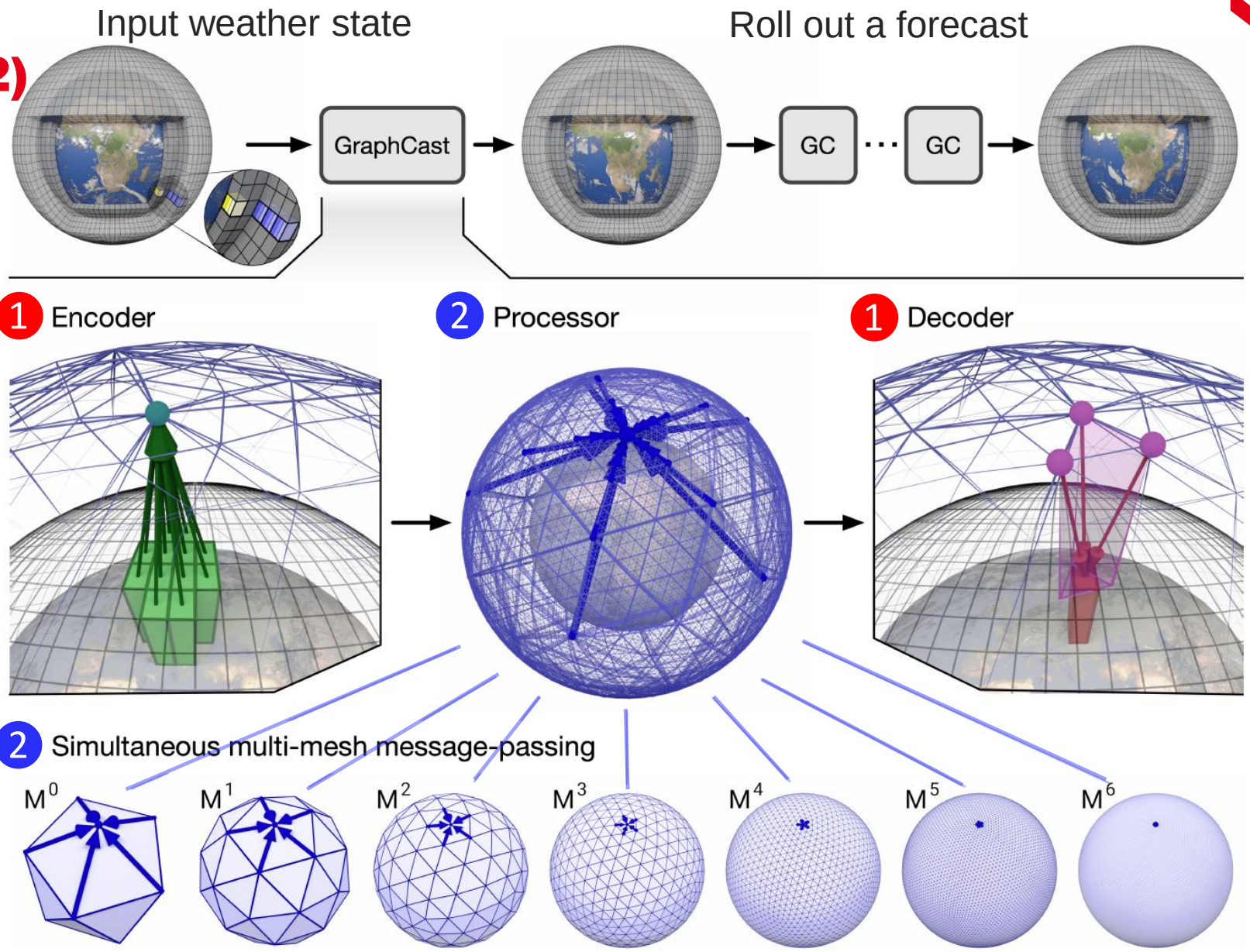
## ○○○● GraphCast* (2022)

### An example of GNN

- $36.7\ 10^6$ parameters.
- Multi-mesh derived from icosahedral meshes, from $\mathrm{M}^0$ to $\mathrm{M}^6$ (40962 nodes).
- Predicts $\sim 100s$ of weather variables over 10 days at 0.25° resolution, in $< 1$ min.

1. Encoding/decoding with a single layer.
2. MPNN: $L = 16$ layers to propagate information from local to global scale.

* arXiv 2212.12794



# 1. Preliminaries

## Laplacians, GFT, convolution, spectral and spatial approaches

# Preliminaries

## Basic definitions

- **Basic definitions**
  - *Graph*: $G = (V, E)$, $V$: set of **nodes** (vertices); $E \subset V \times V$: set of **edges**.
  - Edges can be directed or undirected.
  - A graph is *connected* when, for any two nodes $i$ and $j$, there exists a sequence of edges forming a path from $i$ to $j$.
  - **Adjacency** and **degree** matrices $\mathbf{A}, \mathbf{D} \in \mathbb{R}^{|V| \times |V|}$

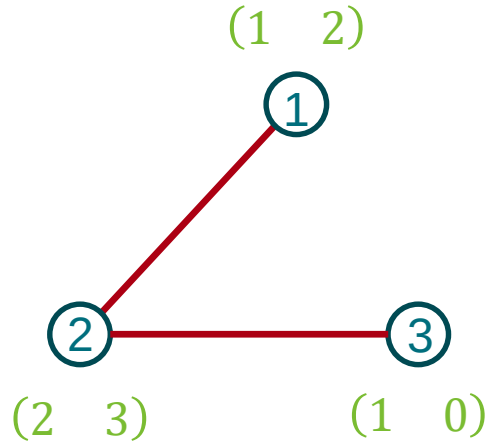
$$A_{ij} = \begin{cases} 1, & (v_i, v_j) \in E \\ 0, & (v_i, v_j) \notin E \end{cases} \text{ and } D_i = \sum_j A_{ij}.$$

  - For directed graphs, we have in-degree $D_i^{\text{in}}$ and out-degree $D_i^{\text{out}}$.
- **Learning tasks on graphs** $f: (G, \mathbf{X}) \to Y$, $\mathbf{X} \in \mathbb{R}^{|V| \times C}$ ($C$: feature dim.).

  **Node-level** app. (graph clustering, classification of web pages).

  **Edge-level** app. (traffic prediction, recommendation systems).

  **Graph-level** app. (molecule classification, point cloud analysis).




$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{D} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \mathbf{X} = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 1 & 0 \end{pmatrix}$$

# Preliminaries

## Combinatorial Laplacian

- **Definition and key properties**
  - Unnormalized Laplacian:

$$\mathbf{L} = \mathbf{D} - \mathbf{A},$$

$$\text{For } \mathbf{X} \in \mathbb{R}^{|V|},\ (\mathbf{LX})_i = \sum_j A_{ij}(x_i - x_j).$$

  - Symmetric, positive semi-definite:

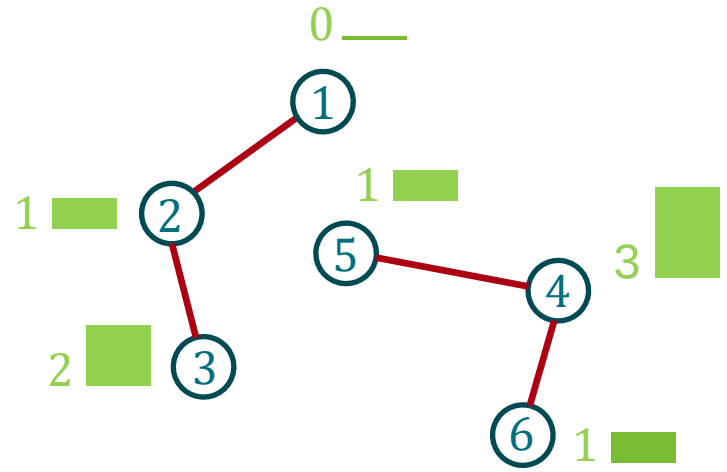
$$\mathbf{X}^{\mathrm{T}}\mathbf{LX} = \frac{1}{2}\sum_i \sum_j A_{ij}\left(x_i - x_j\right)^2 \geq 0.$$

    $\mathbf{X}^{\mathrm{T}}\mathbf{LX} = 0$ (total variation = 0) characterizes signals with zero variation (see ❷) across the graph, i.e. $x_i = x_j$ for any $(i, j)$.

  - Diagonalization: $\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^*$, with $\boldsymbol{\Lambda} = \mathrm{diag}\left(\lambda_1, \ldots, \lambda_{|V|}\right)$, $\lambda_i \in \mathbb{R}$.

    $\mathbf{L1} = \mathbf{0} \Rightarrow$ multiplicity $m_g$ of $\lambda = 0$ ($\dim(\ker \mathbf{L})$) equals the number of connected components (see ❶).

    An intuitive approach: $\mathbf{U} = \text{nodes} \times \text{spectral modes}$




❶

$$\mathrm{L} = \begin{pmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} \end{pmatrix};\ \mathbf{Q} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

$$\chi_{\mathbf{Q}}(\lambda) = \lambda(\lambda - 1)(\lambda - 3)$$

$$\Rightarrow m_g(0) = \mathbf{2}$$

❷

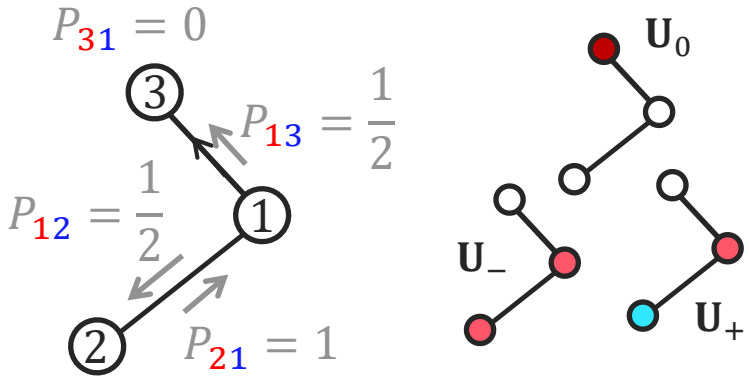
$$\mathbf{X}^{\mathrm{T}}\mathbf{LX} = \underbrace{\frac{1}{2} \times 4}_{G_1} + \underbrace{\frac{1}{2} \times 16}_{G_2} = 10$$

# Preliminaries

## Other definitions

$\sigma = \{0, 1 \pm 1/\sqrt{2}\}$

- **Symmetric normalized Laplacian**
  - $\mathbf{L}$ gives more weight to high-degree nodes (hubs) in the energy computation → **topological bias.**
  - Def.: $\mathbf{L}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$.
    - Counteract topol. bias by scaling **sender** and **receiver** contributions.
    - $\sigma(\mathbf{L}) \subset \left[0, \max_i D_i\right] \Rightarrow \sigma(\mathbf{L}_{\text{sym}}) \subseteq [0,2]$, i.e. $\lambda_{\max} \leq 2$.
  - Applications: spectral filtering, GCNs, Laplacian Encodings, ...
- **Random walk (rw) Laplacian**
  - Def.: $\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$.
    - Normalizes only the **sender**, consistent with a rw's view of the graph (each node distributes its information over its neighbors) ⇒ $\sigma(\mathbf{L}_{rw}) \subseteq [0,2]$.
    - Asym. operator that models **directional diffusion**, where information flows from $v_i$ to its neighbors $v_j$ with a **Markov transition matrix** $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$.
  - Applications: diffusion models, APPNP, Personalized PageRank, ...




$$\mathbf{L}_{\text{sym}} = \frac{1}{\sqrt{2}}\begin{pmatrix} \sqrt{2} & -1 & 0 \\ -1 & \sqrt{2} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{L}_{\text{rw}} = \frac{1}{2}\begin{pmatrix} 2 & -1 & -1 \\ -2 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

**Fig.** Spectral and probabilistic interpretations of Laplacian for a small directed graph. Rand. walker located at $v_i$ moves to $v_j$ with $P_{ij} = \Pr(i \to j)$.

# Preliminaries

## A simple example (1/2)

- **A (undirected) circular graph $G = (V, E)$**
  - Each $v_i$ is connected to $v_{i+1}$ and $v_{i-1}$ and

$$E = \{(i, i + 1 \bmod |V|)\}.$$

  - $\mathbf{D} = 2\mathbf{I}$ (each node is connected to 2 nodes), $\mathbf{A} = \mathbf{J} + \mathbf{J}^{N-1}$, where $\mathbf{J}$: circular shift matrix of order $N = |V|$, i.e.

$$J_{ij} = \delta_{i,(j+1) \bmod |V|}.$$

    - For $N = 4$:

$$\mathbf{J} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \text{ and } \mathbf{J}^N = \mathbf{I}.$$

- **Use in energy-based models**
  - Circular graphs are used in energy-based models such as Convolutional or Temporal Restricted Boltzmann Machines.
  - These architectures leverage circular connectivity to implement shared filters and translation invariance.

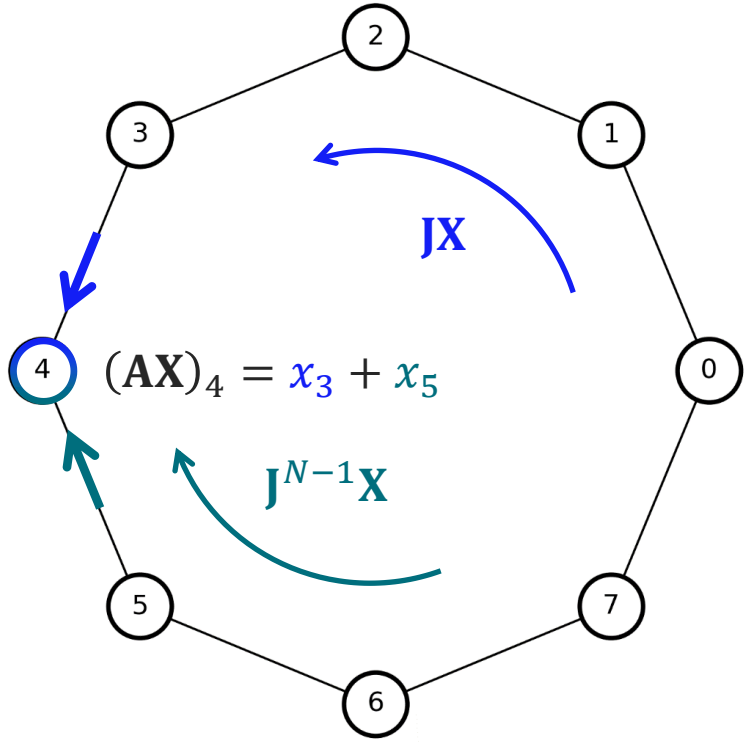



**Fig. :** Action of $\mathbf{A}$ on a circular graph. The operation aggregates forward and backward shifts of a signal, yielding a discrete circular convolution with kernel $[1\ 0\ 1]$ and reflecting the graph's rotational invariance..

# Preliminaries

## A simple example (2/2)

- **Spectral properties of the circular graph**
  - Since eigenvalues of $\mathbf{J}$ are the $N$-th roots of unity ($e^{2i\pi k/N}$), the eigenpairs of $\mathbf{L}$ are ($\mathbf{L}\mathbf{U}_k = \lambda_k \mathbf{U}_k$):

$$\lambda_k = 2[1 - \cos(2k\pi/N)] \in \mathbb{R},$$

$$(\mathbf{U}_k)_j = \frac{1}{\sqrt{N}} \exp\left(\frac{2k\pi ij}{N}\right).$$

  - Each $\mathbf{U}_k \in \mathbb{C}^{|V|}$ $(k = 0, \dots, N-1)$ represents a **harmonic mode** over $G$ and $\lambda_k \in \mathbb{R}$ measures how much the mode oscillates.
    - All $\lambda_k$ $(k \neq 0, N/2)$ appear in conjugate pairs, hence multiplicity 2 in the real spectrum (related to the graph's rotational symmetry).
  - Signal **reconstruction**, for $\mathbf{X} \in \mathbb{R}^{|V|}$:

$$\mathbf{X} = \underbrace{\sum_{k=0}^{N-1} \hat{X}_k \mathbf{U}_k}_{\mathbf{U}\hat{\mathbf{X}}} \text{ with } \hat{X}_k = \mathbf{U}_k^* \mathbf{X},$$

    Using $\mathbf{U} = (\mathbf{U}_1 \cdots \mathbf{U}_{|V|})$ gives: $\hat{\mathbf{X}} = \mathbf{U}^* \mathbf{X}$.

    ⇨ $\hat{\mathbf{X}}$ captures the signal's variation at different graph frequencies.

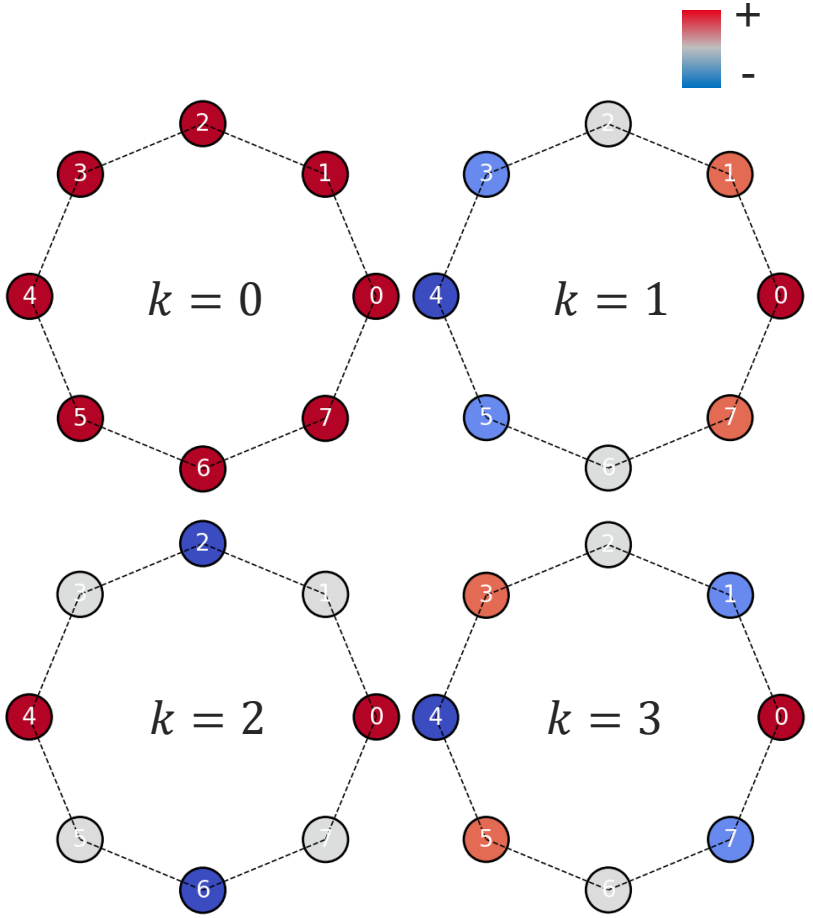
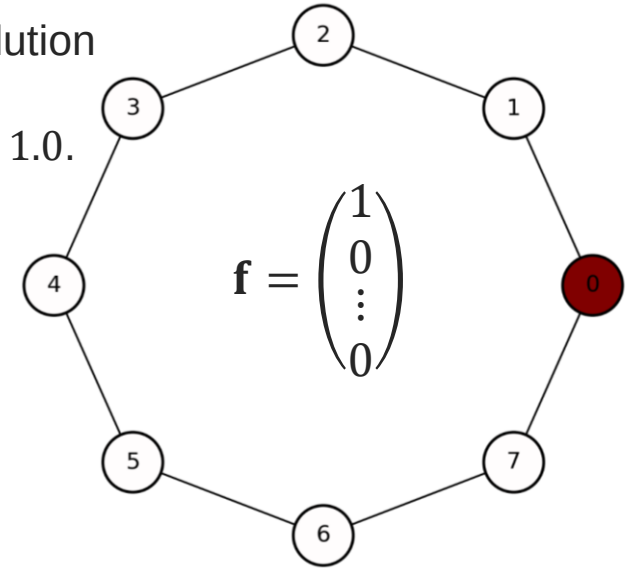



**Fig. :** Frequency modes for a circular graph with 8 nodes.

# Preliminaries

## GFT and convolution

**Fig. :** Graph convolution with a heat kernel $\hat{g}_k = e^{-\tau\lambda_k}$ and $\tau = 1.0$.




- **Definition of the Graph Fourier Transform (GFT)**
  - GFT allows to express $\widehat{\mathbf{X}} \in \mathbb{R}^{|V|}$ in terms of modes on the graph:

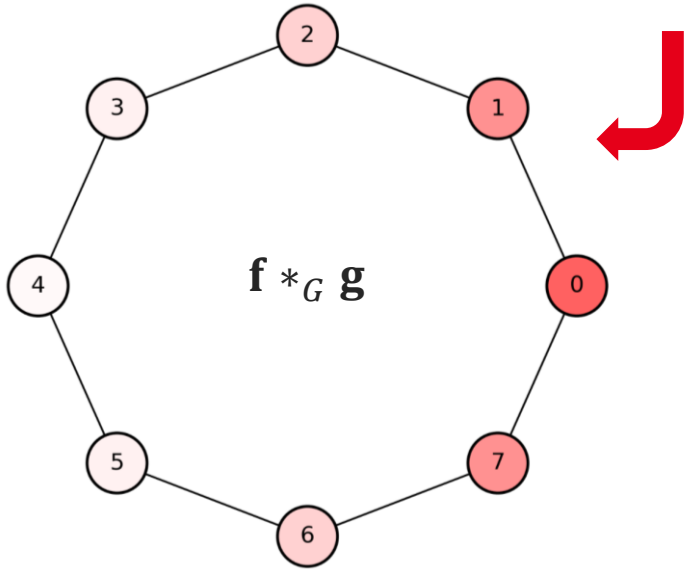
$$\text{GFT: } \widehat{\mathbf{X}} = \mathbf{U}^*\mathbf{X} \text{ and iGFT: } \mathbf{X} = \mathbf{U}\widehat{\mathbf{X}}.$$

  - Total variation: or $\mathbf{X}^{\mathrm{T}}\mathbf{L}\mathbf{X} = \sum_{k=0}^{N-1} \lambda_k \left|\hat{X}_k\right|^2$.
  - Parseval's identity: $\mathbf{X}^{\mathrm{T}}\mathbf{X} = \widehat{\mathbf{X}}^*\widehat{\mathbf{X}} = \sum_{k=0}^{N-1} \left|\hat{X}_k\right|^2$.

- **Convolution on a graph $G$**:
  - No abelian group structure: unlike $\mathbb{R}^n$ or $\mathbb{Z}^n$ a general graph lacks translation invariance ⇨ no canonical convolution theorem.
  - **Convolution** between $\mathbf{f} \in \mathbb{R}^{|V|}$ and $\mathbf{g} \in \mathbb{R}^{|V|}$

$$\mathbf{f} *_G \mathbf{g} = \mathbf{U}(\underbrace{\widehat{\mathbf{f}} \odot \widehat{\mathbf{g}}}_{\mathrm{diag}(\hat{\mathbf{g}})\hat{\mathbf{f}}}) \Rightarrow \mathbf{f} *_G \mathbf{g} = \mathbf{U}\mathrm{diag}(\hat{\mathbf{g}})\mathbf{U}^*\mathbf{f}.$$

  - **Spectral filtering**: we reinterpret $\hat{\mathbf{g}}$ as the sampling of a function on the spectrum, i.e. $\hat{g}_k = h(\lambda_k) \Rightarrow \mathbf{f} *_G \mathbf{g} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^*\mathbf{f}$.



# Preliminaries

## Limitations: cospectrality

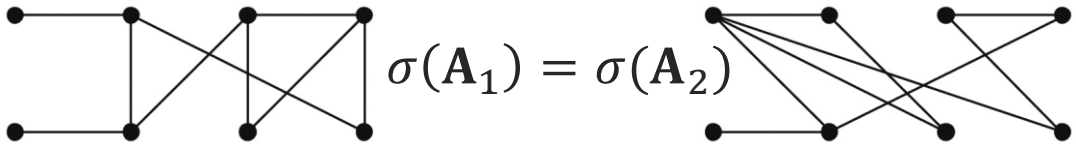
### Spectral graph convolution

- Requires **eigendecomposition** of the graph Laplacian:

$$\mathbf{f} *_G \mathbf{g} = \mathbf{U} h(\mathbf{\Lambda}) \mathbf{U}^* \mathbf{f}.$$

⇨ Impractical for large or changing graphs.

- **High computational cost**: requires full eigendecomposition of the Laplacian $O(|V|^3)$ → Lapack + machine-specific optimisations or Lanczos method for sparse networks.
- **Graph-specific basis**: eigenvectors $\mathbf{U}$ depend on graph topology ⇒ not transferable across graphs.
- **Incompatible with dynamic graphs**: any change in topology alters $\sigma(\mathbf{A})$; same $\sigma(\mathbf{A})$ does no imply same graph ⇨ **cospectrality**.




- **No locality**: spectral filters are global by construction — they blend all node information through $\mathbf{U}$.

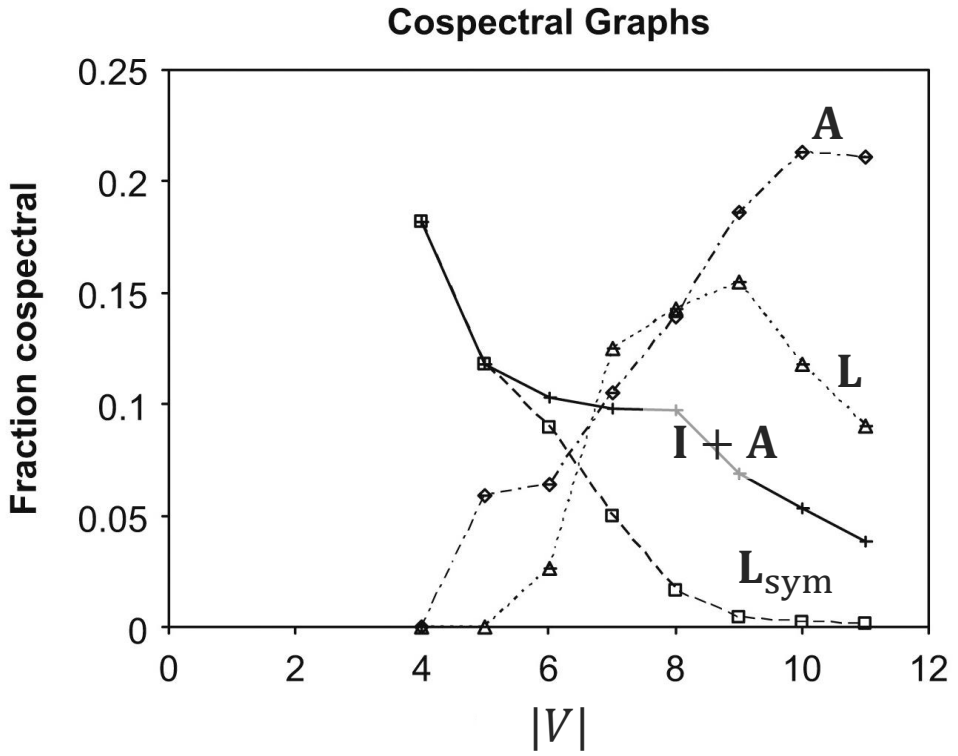
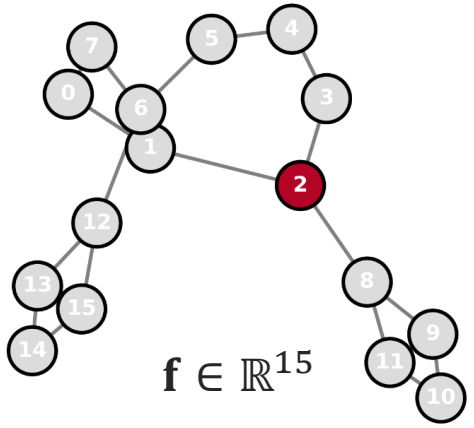



**Fig. :** Fraction of **non-isomorphic tree** (connected and acyclic) pairs that share the same $\sigma$, as a function of graph size, for different matrix representations [1].

[1] Wilson & Zhu, Pattern Recognition, 2008.

# Preliminaries

## From spectral to spatial theory




- **Polynomial approximation and locality**
  - $h$ can be approximated by a polynomial, leading to localized filters:

$$h(\lambda) \simeq \sum_{k=0}^{K} \theta_k \lambda^k \Rightarrow \mathbf{f} *_G \mathbf{g} = \sum_{k=0}^{K} \theta_k \mathbf{L}^k \mathbf{f},$$

where $\mathbf{L}^k$ captures information from the $k$-hop neighborhood.

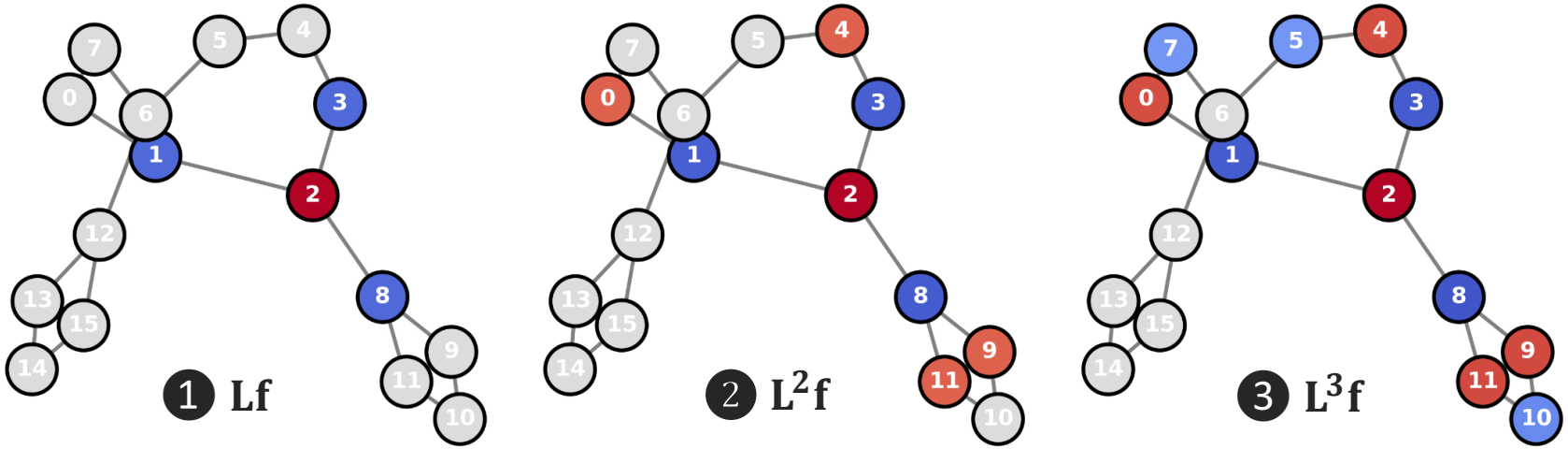



**Fig. :** Graph diffusion under powers of the Laplacian, starting from a Dirac signal at node 2

But this is still fixed (non-learnable structure, no adaptation to the task).

- **Idea:** replace **global spectral** filtering with **local spatial** message passing: information is propagated locally via learnable operations — a key idea behind Message Passing Neural Networks.

# 2. Graph Neural Networks

## MPNN, GCN, over-smoothing/squashing, expressiveness, GAT, Graph Transformers

# Graph neural networks

## GNN layer

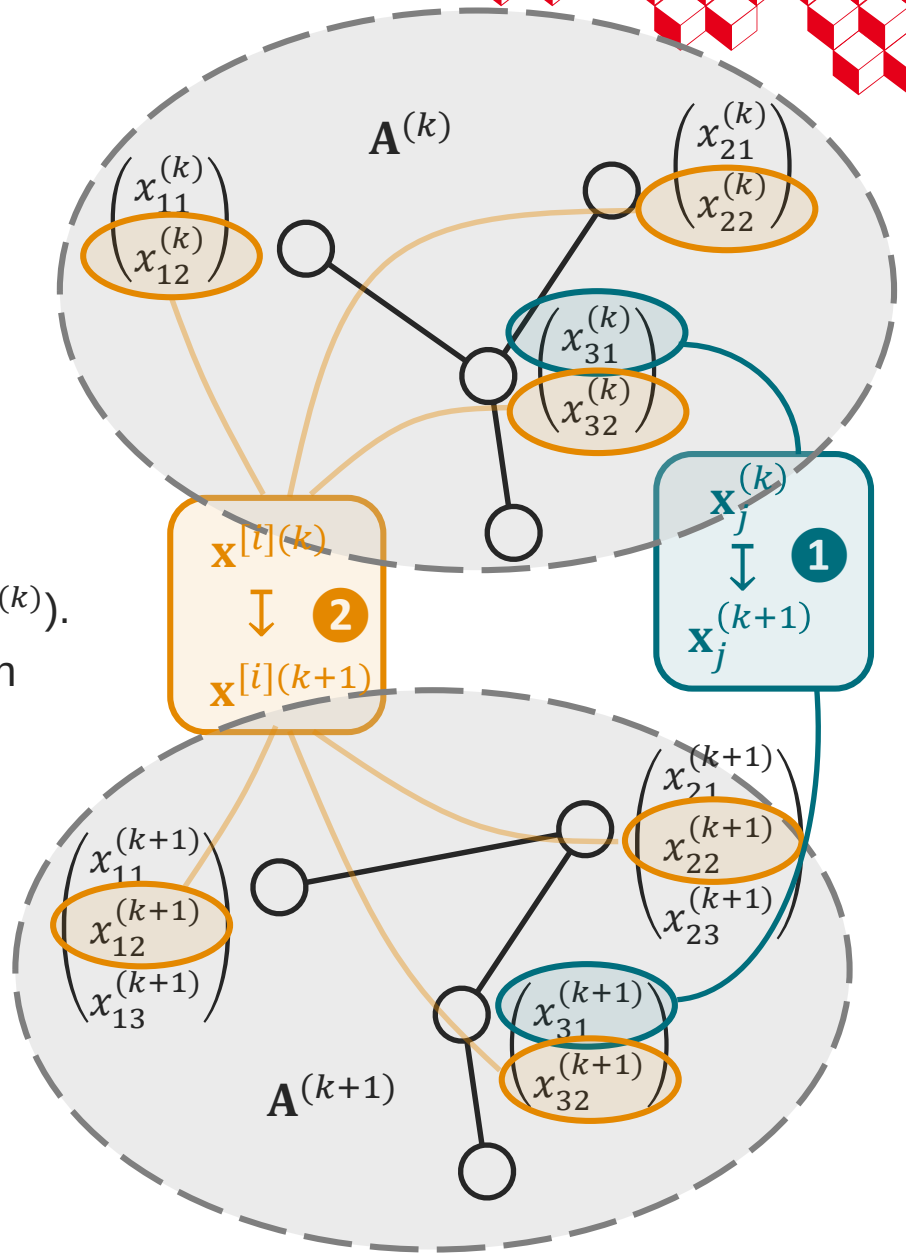
- **GNN layer**
  - The basic idea is to update the state of each node through embedding in a $C_{k+1}$-dimensional space at each GNN layer:

$$\underset{\mathbf{X}^{(k)}}{\mathbb{R}^{|V|\times C_k}} \xrightarrow[\text{layer } k]{} \underset{\mathbf{X}^{(k+1)}}{\mathbb{R}^{|V|\times C_{k+1}}}$$

    - $\mathbf{x}_j^{(k)} \in \mathbb{R}^{C_k}$: denotes the feature vector at node $j$ (i.e., the $j$-th row of $\mathbf{X}^{(k)}$).
    - $\mathbf{x}^{[i](k)} \in \mathbb{R}^{|V|}$: denotes the $i$-th channel of the graph signal, i.e., the $i$-th column of $\mathbf{X}^{(k)}$.

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \\ x_{41} & x_{42} \end{pmatrix} \quad \begin{matrix} \mathbf{x}^{[1]} \\ \mathbf{x}_1 \end{matrix}$$

- **Spatial vs spectral**
  - **MPN (1)**: feature vector at each node is updated using messages from neighbors.
  - **GCN (2)**: each signal channel is transformed using graph convolution.



# Graph neural networks

## Message Passing Network (1)

- **AGGregate-UPdate operation**
  - The MP framework defines a GNN layer as:

$$\mathbf{h}_j^{(k)} = \mathrm{AGG}^{(k)}(\{\mathbf{x}_i^{(k)} | v_i \in \mathcal{N}(v_j)\}),$$

$$\mathbf{x}_j^{(k+1)} = \mathrm{UP}^{(k)}\left(\mathbf{x}_j^{(k)}, \mathbf{h}_j^{(k)}\right).$$

    - The AGGregate step can be either a fixed permutation-invariant operation (e.g., $\Sigma_j$, avg, mean) or a learnable function.
    - The UPdate step refines the aggregated message by combining it with the node's current state, allowing gating mechanisms (GRU) to control information flow.
  - By adding self-loops in the graph:

$$\mathbf{x}_j^{(k+1)} = \sigma\Big(\bigoplus_{i|v_i \in \mathcal{N}(v_j)} f\left(\mathbf{x}_i^{(k)}, \mathbf{x}_j^{(k)}; \boldsymbol{\theta}^{(k)}\right)\Big)$$

    - $\sigma$: non-linear activation and $\boldsymbol{\theta}^{(k)}$ are trainable parameters.
    - $\bigoplus$ is a symmetric operator: $\sum$, avg or max.

Google Research Blog. 2024

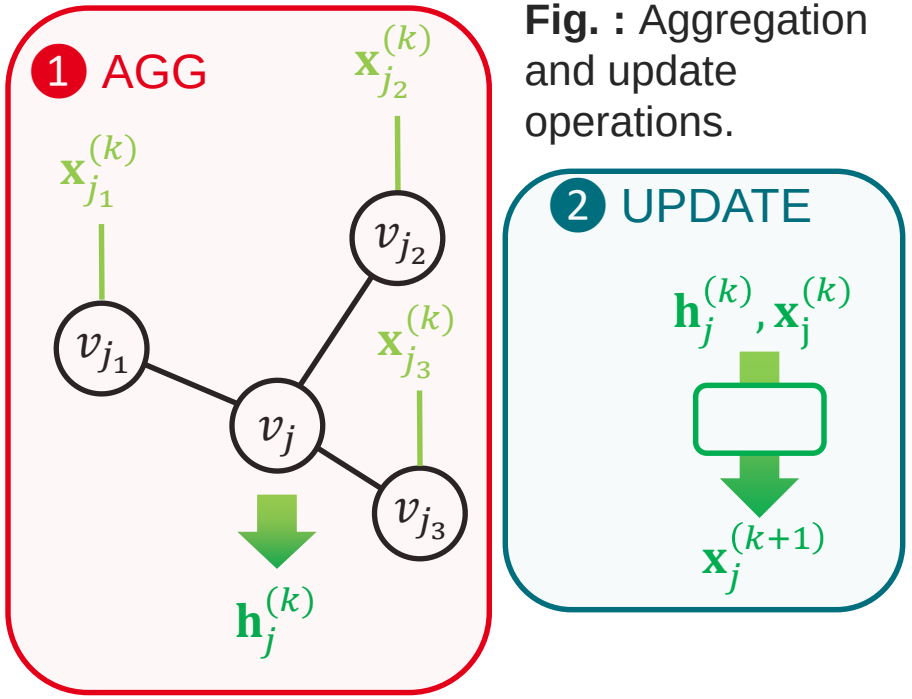
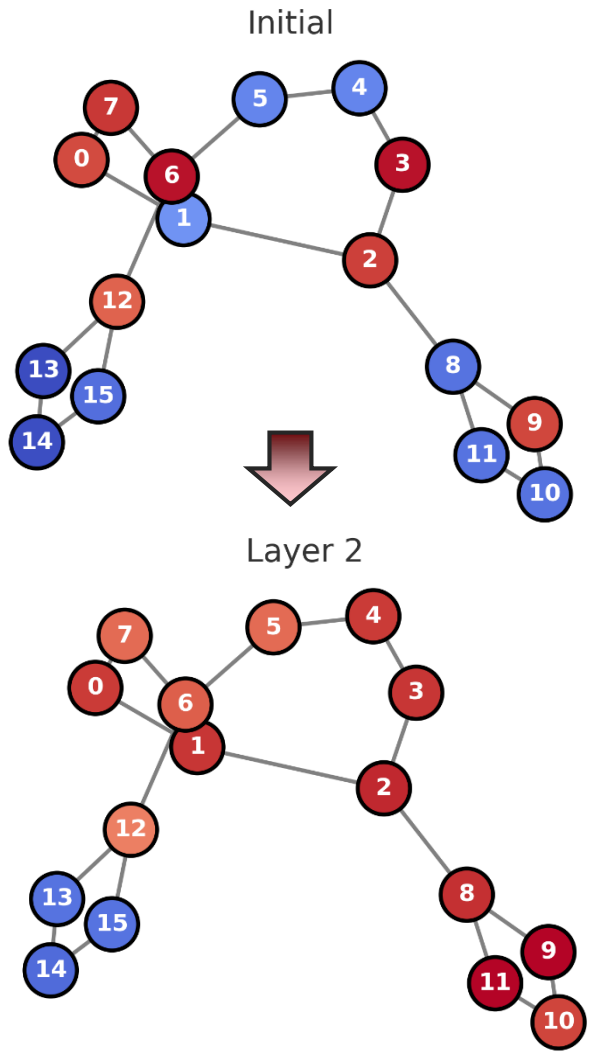



**Fig. :** Aggregation and update operations.

# Graph neural networks

## Aggregate, Update and Pooling

**Fig. :** $\mathbf{x} \sim \mathrm{U}([-1,1])$, $\bigoplus = \sum$ (+ norm.) 3-hop neighborhood, with $\boldsymbol{\theta} = \mathbf{I}$ . Neighborhood aggregation leads to **feature homogenization** at depth.




- **Choice of $f$**
  - $f$ can be given by:

$$f(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\theta}) = \boldsymbol{\theta}\mathbf{x}_i$$
$$f(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\theta}) = \boldsymbol{\theta}(\mathbf{x}_i - \mathbf{x}_j) \quad \text{or } \mathrm{MLP}(\mathbf{x}_i, \mathbf{x}_i - \mathbf{x}_j).$$

    - For $\boldsymbol{\theta} = \mathbf{I}$ this resembles the action of the Laplacian: $\mathbf{x}_j^{(k+1)} = -\left(\mathbf{L}\mathbf{X}^{(k)}\right)_j$.
    - Aggregation over $\kappa$-hop neighbors introduces $\kappa$ as a structural parameter. This controls the receptive field at each layer.
    - Normalization with respect to $\mathcal{N}(v_j)$ is often required to mitigate sensitivity to varying neighborhood sizes and to improve stability during training.
- **Graph pooling**: to obtain a graph-level representation $\mathbf{y} \in \mathbb{R}^C$:

$$\mathbf{y} = \bigoplus_{i|v_i \in V} \mathbf{x}_i,$$

  $\bigoplus$: denotes a permutation-invariant operation, used to compute a global graph representation for tasks like molecule classification or event detection.
  - Other approaches: graph coarsening, learned pooling (DiffPool, SAGPool).

# Graph neural networks

## Graph Convolutional Networks (2)

- **Learning Convolutional Filters**
  - Each channel CNN

$$\mathbf{x}^{[j](k+1)} = g\left(\sum_{i=1}^{C_k} \mathbf{x}^{[i](k)} *_G \mathbf{w}_{ij}^{(k)}\right),$$

where filters $\mathbf{w}_{ij}^{(k)} \in \mathbb{R}^{|V|\times|V|}$ are approximated using Chebyshev polynomials, i.e. $w(\lambda) = \sum_m \theta_m \mathrm{T}_m(\lambda)$

$$\mathbf{x}^{[j](k+1)} = g\left(\sum_{i=1}^{C_k} \sum_{m=0}^{K} \theta_{ijm}^{(k)} \mathrm{T}_m(\tilde{\mathbf{L}}_{\mathrm{sym}})\, \mathbf{x}^{[i](k)}\right).$$

  - Using $\mathrm{T}_0(\tilde{\mathbf{L}}_{\mathrm{sym}}) = \mathbf{I}$ and $\mathrm{T}_1(\tilde{\mathbf{L}}_{\mathrm{sym}}) = \tilde{\mathbf{L}}_{\mathrm{sym}}$ and keeping $m = 0{,}1$ gives the GCN approximation

$$\mathbf{x}^{[j](k+1)} = g\left(\sum_{i=1}^{C_k} \theta_{ij0}^{(k)} \underbrace{\left(\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)}_{\tilde{\mathbf{A}}} \mathbf{x}^{[i](k)}\right).$$

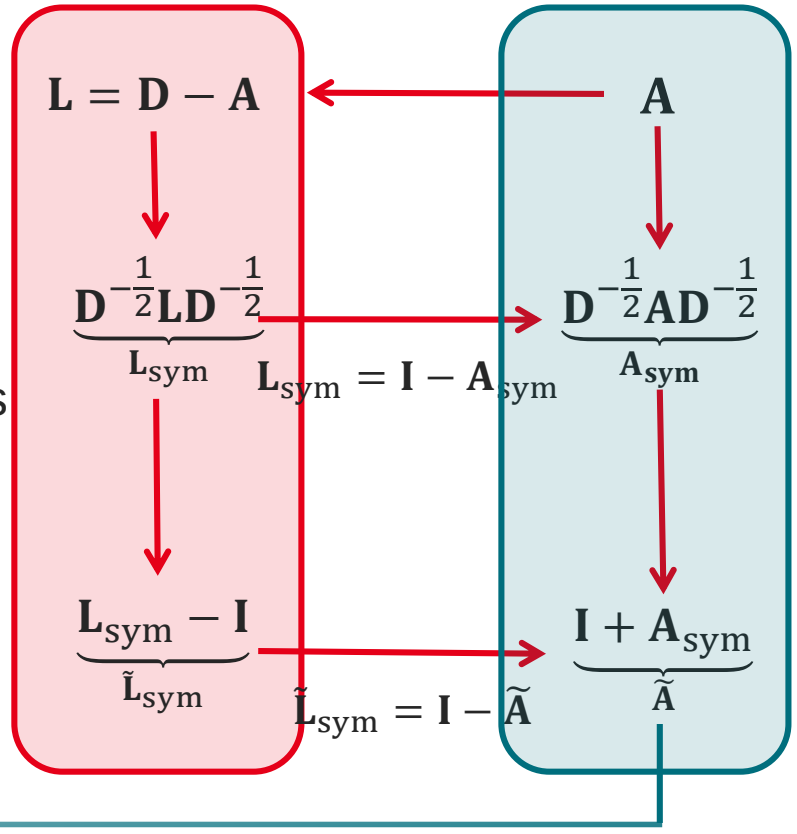
In matrix form, the update rule $h_{\boldsymbol{\Theta}^{(k)}}^{(\mathrm{k})}: \mathbb{R}^{|V|\times C_k} \to \mathbb{R}^{|V|\times C_{k+1}}$ is given by

$$\mathbf{X}^{(k+1)} = h_{\boldsymbol{\Theta}^{(k)}}^{(\mathrm{k})}\left(\mathbf{X}^{(k)}\right) = g\left(\left(\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)\mathbf{X}^{(k)}\boldsymbol{\Theta}^{(k)}\right).$$

spectral locality ⇒ **spatial** locality

$\tilde{\mathbf{L}}_{\mathrm{sym}} = 2\lambda_{\max}^{-1}\mathbf{L}_{\mathbf{sym}} - \mathbf{I} \Rightarrow \sigma\left(\tilde{\mathbf{L}}_{\mathrm{sym}}\right) \subset [-1,1].$

Approx. $\lambda_{\max} = 2$ and $\theta_{ij0}^{(k)} = -\theta_{ij1}^{(k)}$.



# Graph neural networks

## MPNN and GCN drawbacks

[1] Rusch, Bronstein, Mishra, arxiv abs/2303.10993, 2023.
[2] Li, Han & Wu, AIAA Conf on AI, 2018.
[3] Xu et al., arxiv: abs/1806.03536, 2018.

- **Over-smoothing** [1,2]
  - For a GCN:

$$\mathbf{X}^{(k+1)} = \sigma(\widetilde{\mathbf{A}}\mathbf{X}^{(k)}\mathbf{W}),$$

    After $L$ layers ($\sigma = \mathrm{id}$):

$$\mathbf{X}^{(K)} = \widetilde{\mathbf{A}}^K\mathbf{X}^{(0)}\prod_{k=0}^{K-1}\mathbf{W}^k.$$

  - Repeated applications of the symmetrically normalized adjacency matrix $\widetilde{\mathbf{A}}$ cause spectral filtering with decaying modes

$$\widetilde{\mathbf{A}}^K\mathbf{X}^{(0)} = \mathbf{U}\boldsymbol{\Lambda}^K\mathbf{U}^*\mathbf{X}^{(0)} \xrightarrow[K\to\infty]{} \mathbf{U}_1\mathbf{U}_1^*\mathbf{X}^{(0)}$$

    - Only the top eigenvector $\mathbf{U}_1$ (often const.) survives ($\sigma(\widetilde{\mathbf{A}}) \subset [-1,1]$), leading to identical features for all nodes → **over-smoothing.**

- **Influence distribution of the nodes** [3]
  - Influence score $I_u^{(k)}(v) = \mathrm{Tr}(\partial_{\mathbf{x}_v^{(0)}}\mathbf{x}_u^{(k)})$.
    - For a standard MPNN or GCN, $I_u^{(k)}(v) \sim \mathbb{E}(\mathbf{P}^k\mathbf{x}_u)$.
    - This links MP to **stochastic diffusion** on the graph and explains the progressive loss of locality in deeper layers.

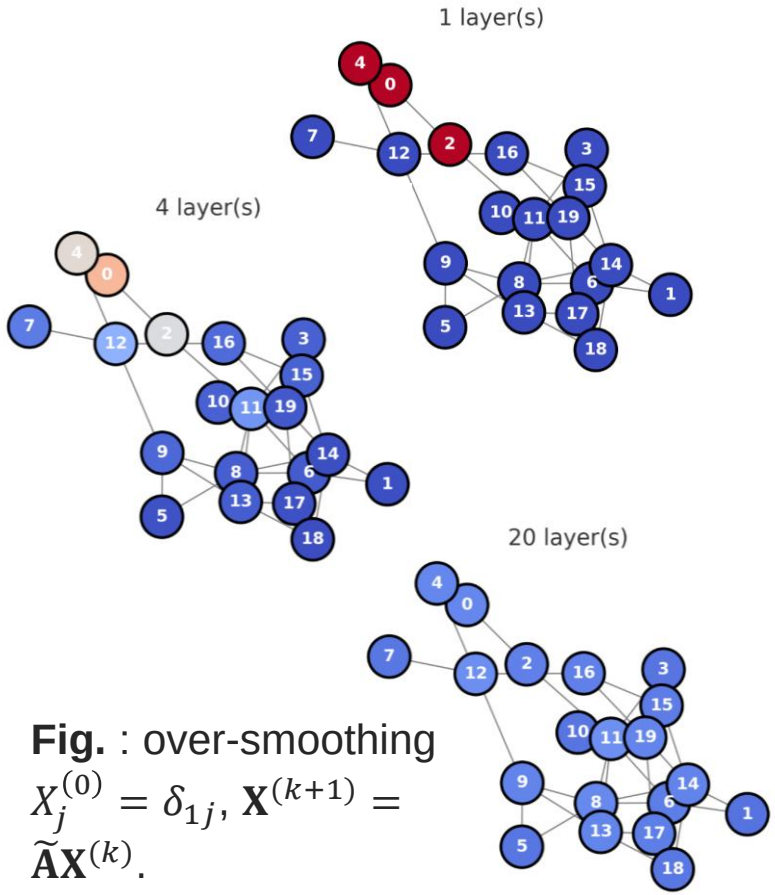



**Fig.** : over-smoothing $X_j^{(0)} = \delta_{1j}$, $\mathbf{X}^{(k+1)} = \widetilde{\mathbf{A}}\mathbf{X}^{(k)}$.

# Graph neural networks

## MPNN and GCN drawbacks

[1] Alon & Yahav, arxiv abs/2006.05205, 2020.

- **Over-squashing** [1]
  - Inability of GNNs to capture long-range dependencies due to compression of messages through paths $\gamma$ of $G$.
  - From update to paths ($X \in \mathbb{R}^{|V| \times C}$)

$$X_j^{(K)} = \sum_{j=1}^{|V|} \underbrace{\left(\tilde{\mathbf{A}}^K\right)_{ij}}_{\sum_{\gamma \in \Gamma_K^{ij}} \prod_{(u,v)\in\gamma} \tilde{\mathbf{A}}_{uv}} X_j^{(0)},$$

    - $\Gamma_K^{ij}$: set of all paths of length $K$ from node $i$ to node $j$.
    - $\tilde{\mathbf{A}}^K$: aggregates the contributions of **all sequences of $K$ hops** connecting $j$ to $i$, each weighted by the product of edge weights along $\gamma$.
  - **Bottlenecks** (Alon & Yahav, 2020)
    - When a large number of paths from distant nodes must pass through **a** limited set of intermediate nodes, these nodes are unable to carry sufficient information due to the fixed $C \rightarrow$ **over-squashing.**

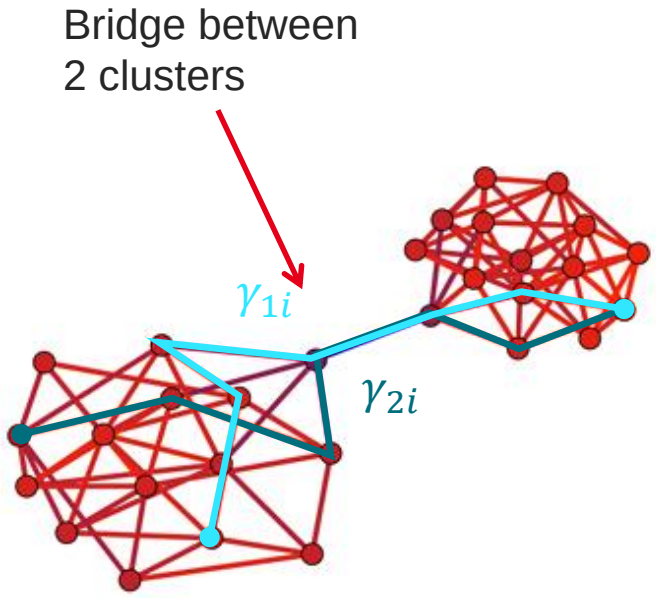


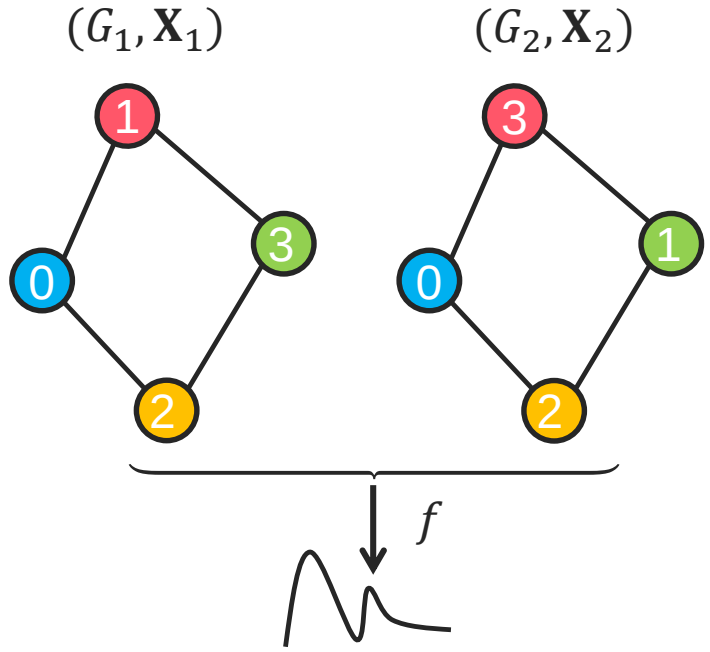

**Fig.** : A graph and multiple paths to connect $v_i$ ($i = 1, \ldots, |V|$) and $v_j$.

# Graph Neural Networks

## GNNs expressiveness

[1] László Babai, arXiv:1512.03547, 2015.
[2] Hamilton, Synthesis Lectures on AI & ML, 2022.

- **Permutation invariance**
  - A model $f(\mathbf{A}, \mathbf{X})$ is permutation invariant if for all $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$, all feature matrix $\mathbf{X} \in \mathbb{R}^{|V|\times C}$, and all permutation matrix $\mathbf{P} \in \mathbb{R}^{|V|\times|V|}$:
  $$f\left(\mathbf{PAP}^{\mathrm{T}}, \mathbf{PX}\right) = f(\mathbf{A}, \mathbf{X})$$
  Permutation equivariance: $f\left(\mathbf{PAP}^{\mathrm{T}}, \mathbf{PX}\right) = \mathbf{P}f(\mathbf{A}, \mathbf{X})$.
- **Graph isomorphism**
  - Two graphs $G_1$ and $G_2$ are isomorphic if $\exists\, \mathbf{P} \in \mathbb{R}^{|V|\times|V|}$ s.t. $\mathbf{PA}_1\mathbf{P}^{\mathrm{T}} = \mathbf{A}_2$.
  Extension to tuples $(G_1, \mathbf{X}_1)$ and $(G_2, \mathbf{X}_2)$:
  $$\mathbf{PA}_1\mathbf{P}^{\mathrm{T}} = \mathbf{A}_2 \text{ and } \mathbf{PX}_1 = \mathbf{X}_2.$$
    - $f$ permutation invariant and $(G_1, \mathbf{X}_1) \sim (G_2, \mathbf{X}_2) \Rightarrow f(\mathbf{A}_1, \mathbf{X}_1) = f(\mathbf{A}_2, \mathbf{X}_2)$.
    - Graph isomorphism testing is NP-indeterminate problem [1].
    - Heuristic: Weisfeler-Lehmann (**1**-WL) [2].
  - Expressive power of many MPNN-based GNNs is upper-bounded by the **1**-WL test — a heuristic for distinguishing non-isomorphic graphs.




$\mathrm{Expr}(f) < \mathbf{1}$-WL
**1** t-uple

# Graph Neural Networks

## Graph Attention Networks


[1] Velickovic et al., arxiv:1710.10903, 2017.
[2] Vaswani et al., NeurIPS, 2017.
[3] Brody et al., arxiv: 2105.14491, 2022.
[4] Xu et al., Arxiv:1810.00826, 2018


- **Back to the aggregation step**
  - The GAT framework [1] defines a GAT layer as:

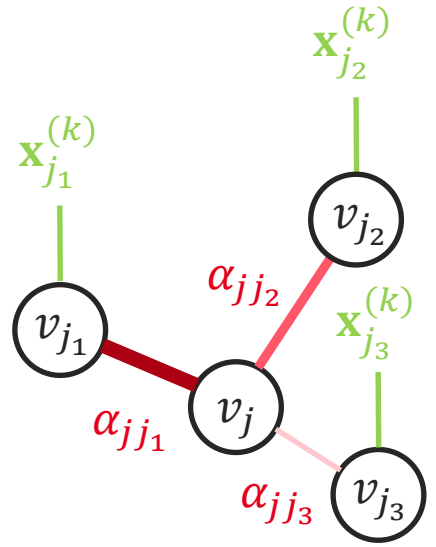
$$\mathbf{x}_j^{(k+1)} = \sigma\left[\sum_{i|v_i\in\mathcal{N}(v_j)} \alpha_{ij}\boldsymbol{\Theta}\mathbf{x}_i^{(k)}\right],$$

    or $\mathbf{X}^{(k+1)} = \sigma(\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{X}^{(k)}\boldsymbol{\Theta}^{(k)\mathrm{T}})$, where $\alpha_{ij} = \mathrm{softmax}(e_{ij})$

$$e_{ij} = a\big(\mathbf{W}\mathbf{x}_i, \mathbf{W}\mathbf{x}_j\big)1_{v_j\in\mathcal{N}(v_i)}.$$

    - $a\colon \mathbb{R}^{C_l}\times\mathbb{R}^{C_l}\to\mathbb{R}$: 1-layer MLP (+ LeakyReLU) or inner product/cos. similarity.
    - Multi-head attention [2] introduces $M$ coefficients $\alpha_{ij}^m$ and $\mathbf{x}_j^{(k+1)} = \underset{m}{||}\,\sigma(.)$.
    - **Close to GCNs in form**: the GAT layer can be seen as a GCN where $\widetilde{\mathbf{A}}$ is replaced by a learnable attention matrix $\boldsymbol{\alpha}^{\mathrm{T}}$.

**Fig.** : attention-based message passing for $v_j$. Thicker arrows indicate higher importance.




$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k\in\mathcal{N}(v_i)}\exp(e_{ik})}$$

- **GAT and Beyond**
  - GAT layers remain permutation invariant (as GCNs).
  - GATv2 [3] resolves expressiveness issues by allowing $e_{ij} = \mathbf{a}^{\mathrm{T}}\sigma\big(\mathbf{W}\big[\mathbf{x}_i||\mathbf{x}_j\big]\big)$.
  - **Limitations**: like MPNNs, standard GATs are still limited by the 1-WL expressiveness [4].

# Graph Neural Networks

## Graph Transformers

[1] Dwivedi et al., arxiv:2003.00982, 2023.
[2] Vaswani et al., NeurIPS, 2017.
[3] Cite GPS, etc.

- **From local to global attention**
  - Laplacian **Positional Encoding** [1,2],

$$\text{for } v_i \text{ we extract } \mathbf{p}_j = (U_{i1} \cdots U_{ik}) \in \mathbb{R}^k$$

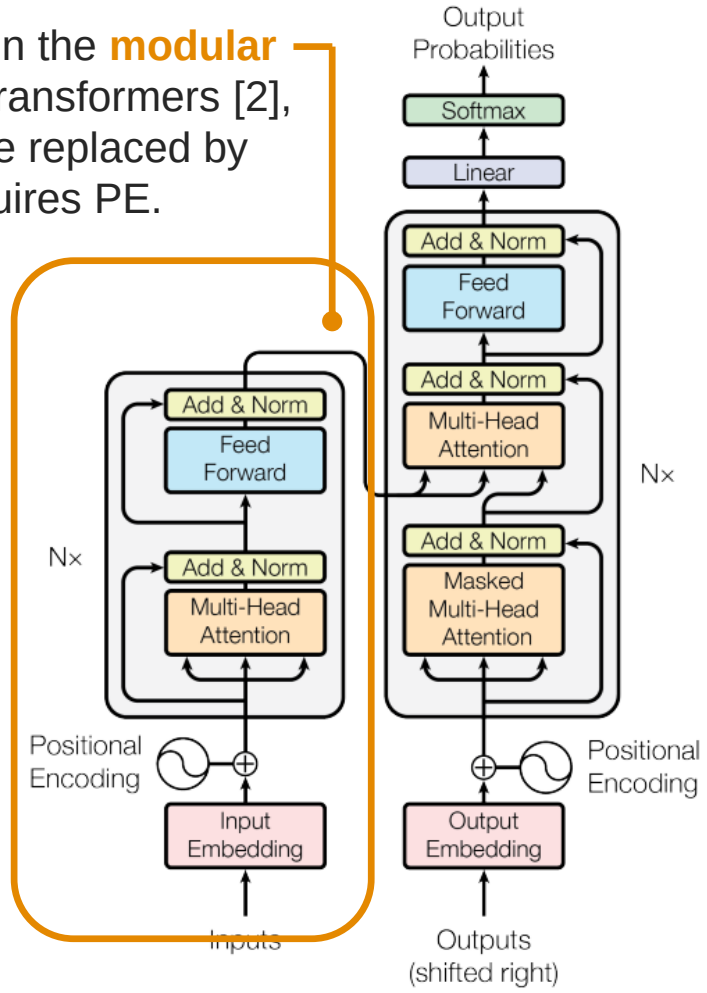
    - Serve as **structural anchors** for each node in the attention space, enabling the model to reason about graph topology.
    - **Allow the model to distinguish nodes with similar features** but different topological roles, which is not possible using node features alone.
  - Graph attention layers are defined by

$$\mathbf{h}_j^{(l+1)} = \Big\|_{k=1}^{K} \Big[ \sum_{i|v_i \in \mathcal{N}(v_j)} \alpha_{ij}^{lk} \boldsymbol{\theta}^{lk} \tilde{\mathbf{x}}^{(l)} \Big],$$

$$\mathbf{x}_j^{(l+1)} = \mathrm{No} \circ f \circ \mathrm{No}(\mathbf{x}_j^{(l)} + \mathbf{h}_j^{(l)}),$$

    - No: BatchNorm or LayerNorm; $f$: (nonlinear) MLP, $\mathbf{x}_i^{(0)} = \mathbf{x}_i + \mathbf{W}\mathbf{p}_i$; attention coefficients $\alpha_{ij}^{lk}$ are obtained from $e_{ij}^{kl} = a(\mathbf{W}\mathbf{x}_i, \mathbf{W}\mathbf{x}_j)/\sqrt{C_l}$.
    - Global attention if $\mathcal{N}(v_j) \to V$.

**Fig.** : GT retain the **modular structure** of transformers [2], sequences are replaced by graphs ⇒ requires PE.



# Graph Neural Networks

## Improving Expressivity: what's beyond MPNNs?

- **Add Features: enrich node/edge descriptors**
  - Add domain knowledge or structural information (degrees, distances, orbits, etc.).
  - Random initial features to break symmetry [Sato et al., 2021].
  - Subgraph-based features: counts, motifs, positional encodings → **GSN**: Substruct. Net. [Bouritsas et al., 2022].
  - Effective resistance & hitting times [Topping et al., 2022].
  - **Limitations:** not always transferable; handcrafted choices may lack generality.
- **Modulate Message Passing: adapt interactions**
  - Use edge-dependent weights, attention, anisotropy → **GAT**: Graph Attention Networks [Velickovic et al., 2018].
  - Identity-aware aggregation [You et al., 2021].
  - Directional aggregation from Laplacian eigenflows → **DGN**: Directional GNNs [Beaini et al., 2021].
  - Can capture fine-grained node distinctions & directionality.
  - **Tradeoff:** increased complexity, may amplify overfitting.
- **Modify the Graph: act on the computation structure**
  - Rewire the graph to enhance connectivity → DropEdge, digraph rewiring, adding virtual nodes.
  - Use high-order structures → k-WL GNNs [Morris et al., 2019], Ring-GNN [Chen et al., 2019].
  - Subgraph GNN [Zhang et al., 2021].
  - Encode hierarchy and locality → Nested GNNs, Hierarchical Pooling.
  - **Drawback:** cost scales with higher-order terms.

# 3. Graph Neural Operators

## NO, GNO and Spatio-Spectral GNO

# Graph Neural Operators

## ●○○○ Neural Operators

* Zongyi Li, Kovachki *et al*., 2021.

- **Intuition of NO ($t$ is ignored for simplicity)**
  - If $G$ is the Green function of PDE $\mathcal{L}(a)x = f$, then $x = G * f$.
  - $G$ is modelled as a kernel $\kappa_{\boldsymbol{\theta}}$: $G(\mathbf{p}, \mathbf{q}) \simeq \kappa_{\boldsymbol{\theta}}\big(\mathbf{p}, \mathbf{q}, a(\mathbf{p}), a(\mathbf{q})\big)$ and for any $\mathbf{x}: \mathbb{R}^d \to \mathbb{R}^m$, a NO is defined by

$$\mathbf{K}_{\boldsymbol{\theta}}\mathbf{x}(\mathbf{p}) = \int \kappa_{\boldsymbol{\theta}}\big(\mathbf{p}, \mathbf{q}, a(\mathbf{p}), a(\mathbf{q})\big)\mathbf{x}(\mathbf{q})\mathrm{d}\mathbf{q}\,.$$

    - FNO $\kappa_{\boldsymbol{\theta}}\big(\mathbf{p}, \mathbf{q}, c(\mathbf{p}), c(\mathbf{q})\big) = \kappa_{\boldsymbol{\theta}}(\mathbf{p} - \mathbf{q}) \Rightarrow \mathbf{K}_{\boldsymbol{\theta}}\mathbf{x}(\mathbf{p}) = \kappa_{\boldsymbol{\theta}} * \mathbf{x}$.
- **Architecture of NOs**
  - The mapping is learnt iteratively:

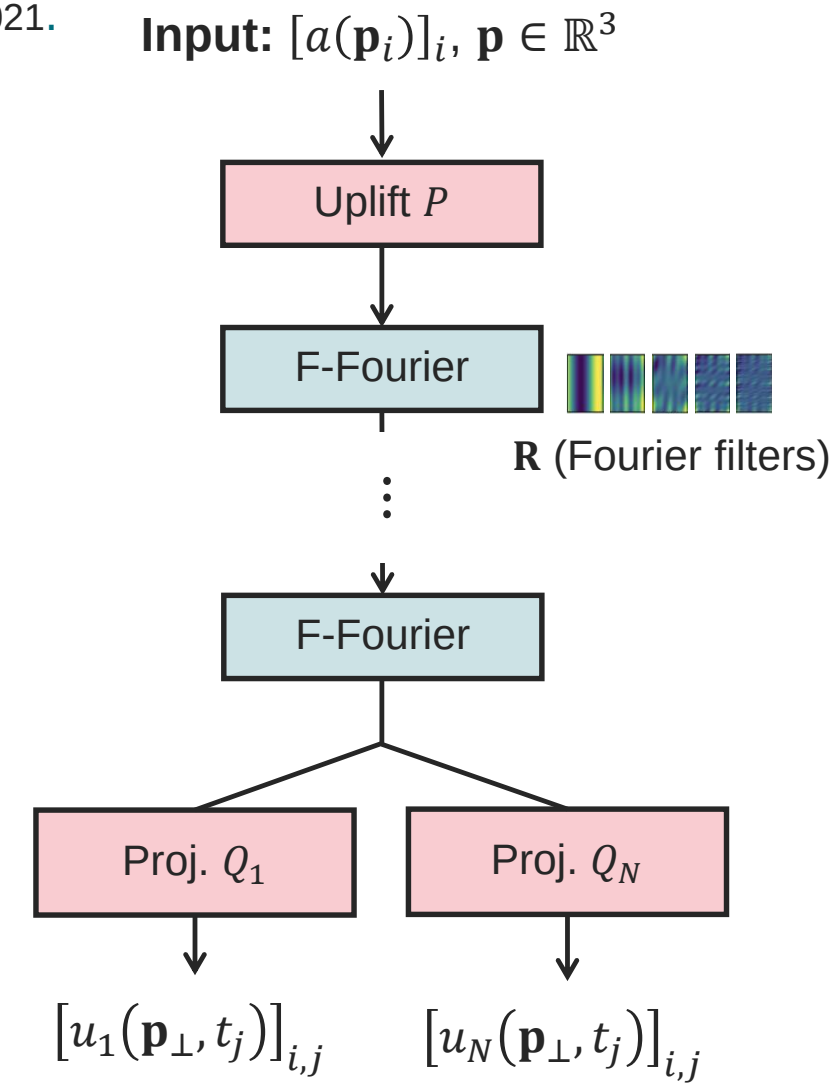
$$(a(\mathbf{p}), \mathbf{p}) \underset{P}{\to} \mathbf{X}^{(0)} \underset{F_1}{\to} \cdots \underset{F_L}{\to} \mathbf{X}^{(L)} \underset{Q}{\to} u(\mathbf{p}),$$

$$\mathbf{X}^{(l)} = \sigma_l\big([\mathbf{W}^{(l)} + \mathbf{K}(a)]\mathbf{X}^{(l-1)} + \mathbf{b}^{(l)}\big).$$

    - $P$: uplift layer, $Q$: projection layer.
  - If $\mathbf{K}$ is a convolution kernel, the convolution theorem leads to:

$$\mathbf{K}(a)\mathbf{x} = \mathcal{F}^{-1}(\underbrace{\mathcal{F}(\boldsymbol{\kappa})}_{\mathbf{R}}\mathcal{F}(\mathbf{x}))$$

    - The weights $\mathbf{R}$ are learnt inside each layer.



# Graph Neural Operators

## ○●○ From FNOs to GNOs

- **Low-frequency truncation**
  - In practice $\mathbf{R}$ is truncated to the first $m$ modes to obtain a **low-rank operator** acting on $\widehat{\mathbf{X}}^{(\mathbf{k})}$.
  - Convolution in a FNO can be re-written as:

$$\mathbf{X}^{(k+1)} = \sigma(\underbrace{\mathcal{F}^{-1}}_{\downarrow\ \mathbf{U}}(\mathbf{R}\,\underbrace{\mathcal{F}(\mathbf{X}^{(k)})}_{\downarrow\ \mathbf{U}^{\mathrm{T}}\mathbf{X}^{(k)}}))$$

    Analogy GCN:

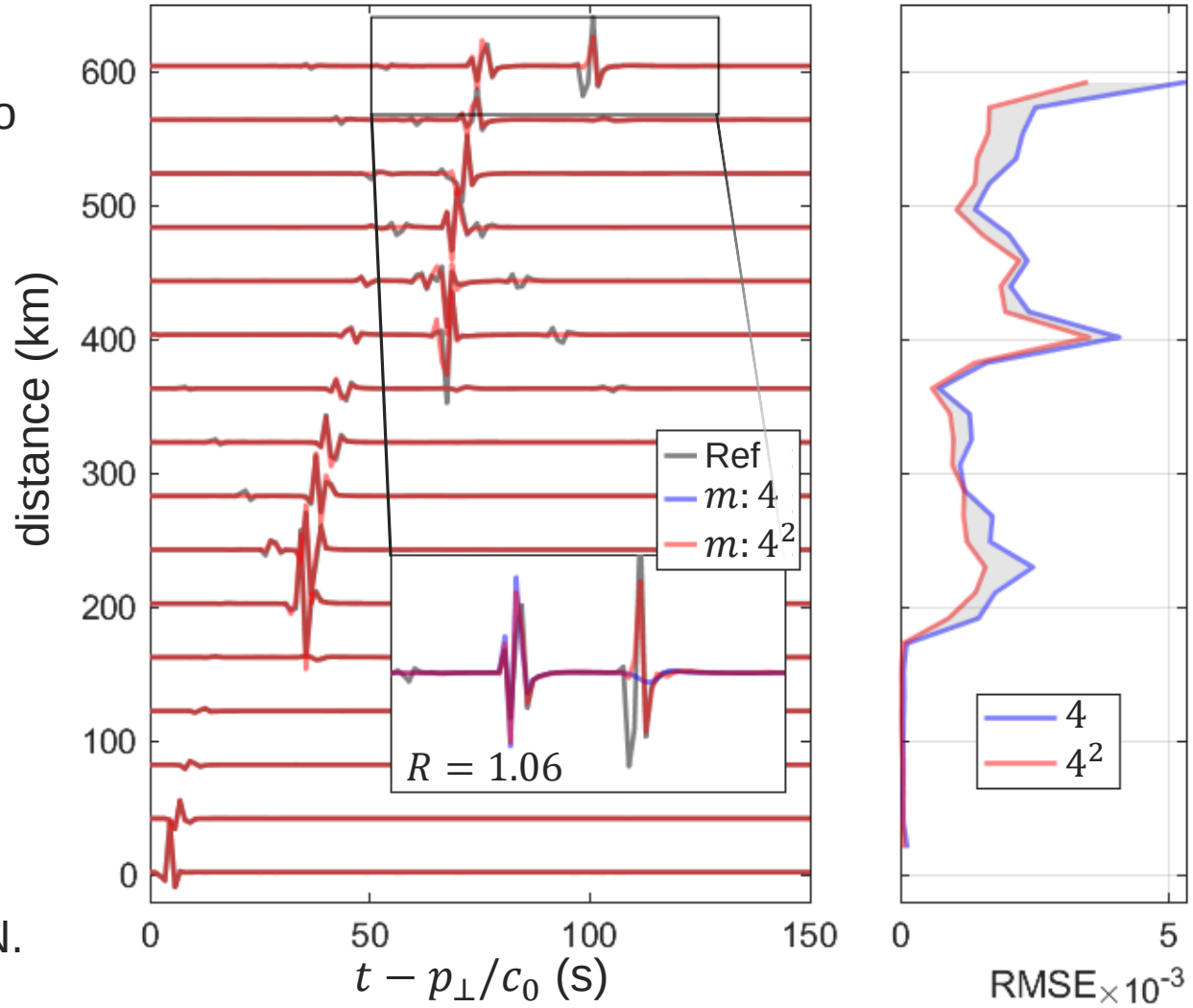
    - $\mathbf{X}^{(k)} \in \mathbb{R}^{N\times C_k}$ multi-channel field, $\mathbf{R} \in \mathbb{R}^{m\times m}$.

- **Extension to graphs**
  - The Fourier basis (FNO) can be replaced with the **graph Laplacian eigenbasis $\mathbf{U_m}$**:

$$\mathbf{X}^{(k+1)} = \mathbf{U}_m\mathbf{R} \times_1 \mathbf{U}_m^{\mathsf{T}}\mathbf{X}^{(k)}.$$

    - $\mathbf{U} \in \mathbb{R}^{|V|\times m}$, $\mathbf{R} \in \mathbb{R}^{m\times C_k\times C_{k+1}}$ and $\mathbf{X}^{(k)} \in \mathbb{R}^{|V|\times C_k}$.
    - $\times_1$ refers to a mode-1 tensor contraction.
  - The MLP in a FNO can be replaced by a MPNN.

**Fig.** : 2D wave propagation in waveguides. **Input**: sound speed $c\colon \mathbb{R}^2 \to \mathbb{R}$, **output**: waveforms $u\colon \mathbb{R}^2 \to \mathbb{R}$; 4 layers, 48 neurons/layer, $m$ is varied.



# Graph Neural Operators

## ○○● Spatio-Spectral GNO

* Sarkar & Chakraborty, 2409.00604, 2024.

- **Global Spectral Graph Convolution**
  - General form of graph convolution:

$$\mathbf{x}^{[j](k+1)} = \sigma\left(\sum_{i=1}^{C_k} \mathbf{x}^{[i](k)} *_G \mathbf{w}_{ij}^{(k)}\right),$$

    - $*_G$: Graph convolution, defined as filtering via $\Lambda$, i.e. $\mathbf{x} *_G \mathbf{w} = \mathbf{U} g_\theta(\mathbf{\Lambda}) \mathbf{U}^{\mathrm{T}} \mathbf{x}$.
    - $\mathbf{w}_{ij}^{(k)}$: learnable spectral filters between channel $i$ and $j$. For multi-valued $\mathbf{X} \in \mathbb{R}^{|V| \times C_k}$, we introduce $\mathbf{R} \in \mathbb{R}^{|V| \times C_k \times C_{k+1}}$

$$\mathbf{X}^{(k+1)} = \sigma\left(\mathbf{U}_m\left(\mathbf{R} \times_1 \left(\mathbf{U}_{\mathrm{m}}^{\mathrm{T}} \mathbf{X}^{(k)}\right)\right)\right).$$

    - $\mathbf{U}_m$: first $m$ eigenvectors of $\mathbf{\Lambda}$ computed using Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) algorithm → computational cost $O(m\kappa|V|)$ if $G$ is obtained using the $\kappa$-NN.
- **Local Convolution with Spatial GNN**
  - Aggregation is limited to the 1-hop direct neighbors $\mathbf{x}_i^{(k)} \in \mathbb{R}^{C_k}$

$$\mathbf{x}_j^{(k+1)} = \sum_{i | v_i \in \mathcal{N}(v_j)} \alpha_{ji} \mathbf{W} \mathbf{x}_i^{(k)}.$$

    - $\alpha_{ij} = \sigma_2(\mathbf{W}_3 \sigma_1(\mathbf{W}_1[\mathbf{h}_j || \mathbf{h}_i || \mathbf{W}_2 w_{ij}]))$: edge-weights are learned within a gating mechanism, with $w_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|$, where $\mathbf{p}_i$ is the positions of node $i$ and $\mathbf{h}_i$ is its Lipschitz embedding*.

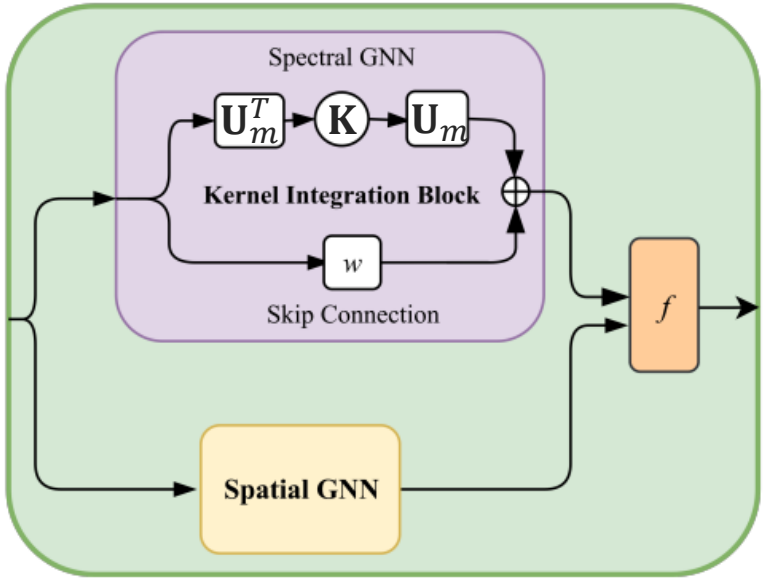



**Fig.** : Neural architecture of a block. It exploits spectral and spatial GNN to formulate the kernel integration operator [1].

$f$ concatenates the outputs of Spectral and Spatial GNNs.

# 4. Dynamic GNN for seismology

## Architectures, performance, interpretability, over-smoothing, practice

# Dynamic GNN for Seismology

## A practical example

### Goal of the approach:

- Learn a parametric model

$$f_{\boldsymbol{\theta}}: \mathbb{S} \to \mathbb{R}^4, \quad \mathbf{S} \mapsto \mathbf{y}$$

  - **Input:** $|\mathrm{V}| \times d$ signals recorded by $|V|$ stations and sampled using $P$ time steps. $d$: NS, EW, Up-Down components.
  - **Output:** characteristics of events.
- **Loss** function:

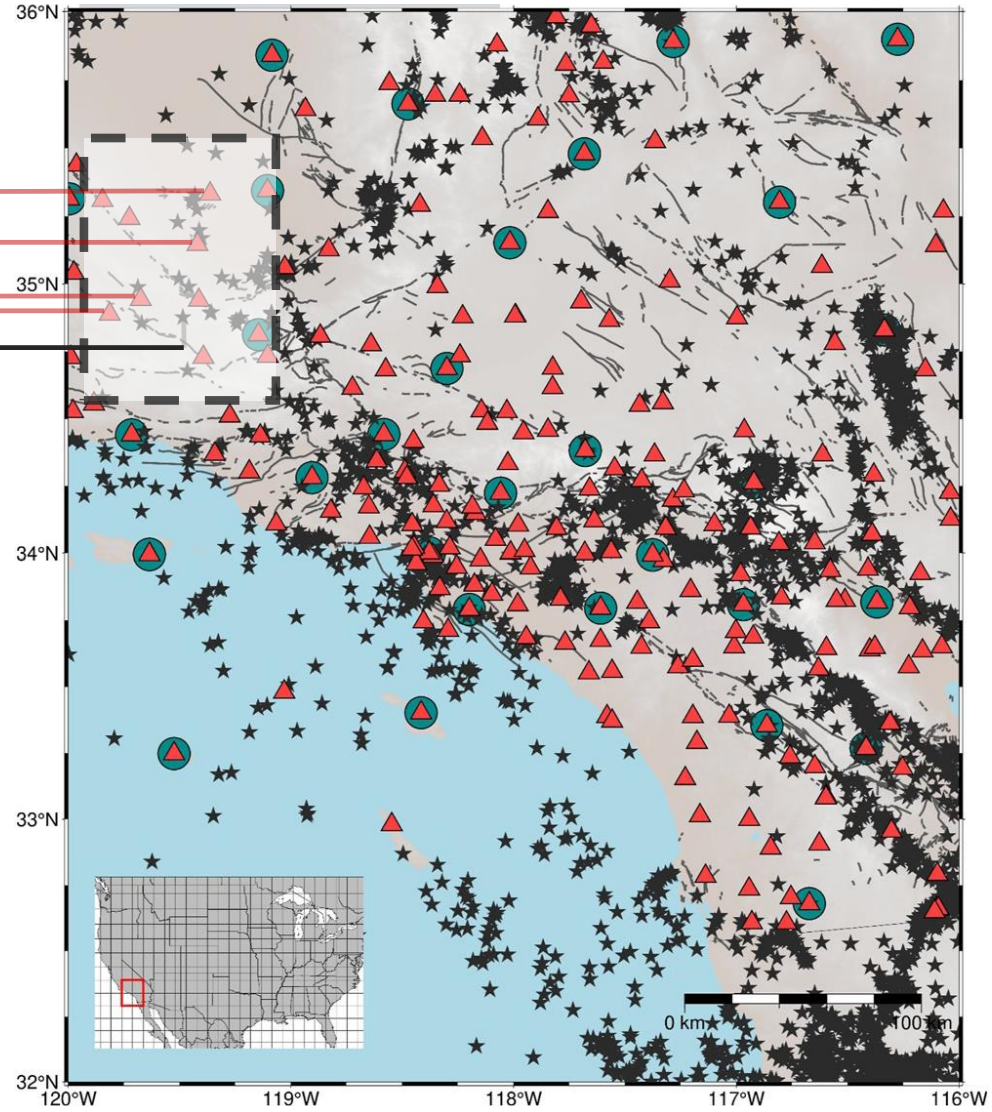
$$\mathcal{L} = \frac{1}{4n} \sum_{k=1}^{n} \sum_{j=1}^{4} |\hat{y}_{kj} - y_{kj}|.$$

  - $y_{kj}$: latitude, longitude, depth, or magnitude of event $k$.

### Graph-related issues

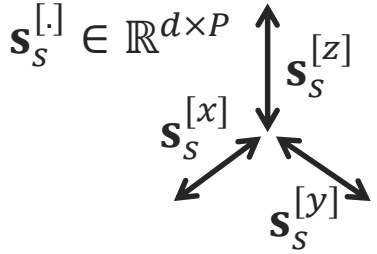
- $\mathbb{S}$ is unstructured and evolve over time (lack of data, temporary stations ▲ vs fixed stations ▲).
- The graph is not specified and $\mathbf{A}$ is *a priori* not known.

**Fig.** Southern California Seismic Network



# Dynamic GNN for Seismology

## Pipeline




### 1 Convolutional Encoding $h^{(\mathrm{I})}_{\boldsymbol{\theta}^{(\mathrm{I})}} : \mathbb{R}^{3C_0} \mapsto \mathbb{R}^{d^{(\mathrm{I})}}$

- Extraction of temporal features (CNN-based encoder):

$$x_s^{[i](l)}[n] = \mathrm{ReLu}(b_{\mathrm{i}}^{(\mathrm{l})} + \sum_{j=1}^{d_{l-1}} \underbrace{\mathbf{W}_{ji}^{(l)} * x_s^{[j](l-1)}[n]}_{\sum_{k=0}^{4} w_{jik}^{(l)} x_s^{[j](l-1)}[n+k]} ).$$

  - $x_s^{[i](l)}[n]$: value of the $i$-th feature channel at time index $n$ for station $s$ at layer $l$.
  - Valid output indices: $n < m_{l-1} - 4 \Rightarrow$ temporal dimension shrinks accross layers.

### 2 Spatial information integration $h^{(\mathrm{II})}_{\boldsymbol{\theta}^{(\mathrm{II})}} : \mathbb{R}^{(d^{(\mathrm{I})}+2)\times|V|} \to \mathbb{R}^{d^{(\mathrm{II})}\times|V|}$

### 3 Aggregation Pool: $\mathbb{R}^{d^{(\mathrm{II})}\times|V|} \to \mathbb{R}^{d^{(\mathrm{II})}}$

### 4 Prediction $h^{(\mathrm{III})}_{\boldsymbol{\theta}^{(\mathrm{III})}} : \mathbb{R}^{d^{(\mathrm{II})}} \to \mathbb{R}^{4}$

- Extraction of graph-level information:

$$\mathbf{y} = \sigma_2^{(\mathrm{III})}(\mathbf{W}_2^{(\mathrm{III})} \sigma_1^{(\mathrm{III})} \left(\mathbf{W}_1^{(\mathrm{III})} \mathbf{z} + \mathbf{b}_1^{(\mathrm{III})}\right) + \mathbf{b}_2^{(\mathrm{III})})$$

  - $\mathbf{z} \in \mathbb{R}^{d^{(\mathrm{II})}}$ is derived by aggregating the features $\mathbf{X} \in \mathbb{R}^{d^{(\mathrm{II})}\times|V|}$ using $\mathrm{Pool}(\mathrm{h}^{(\mathrm{II})}_{\boldsymbol{\theta}^{(\mathrm{II})}}(.))$.

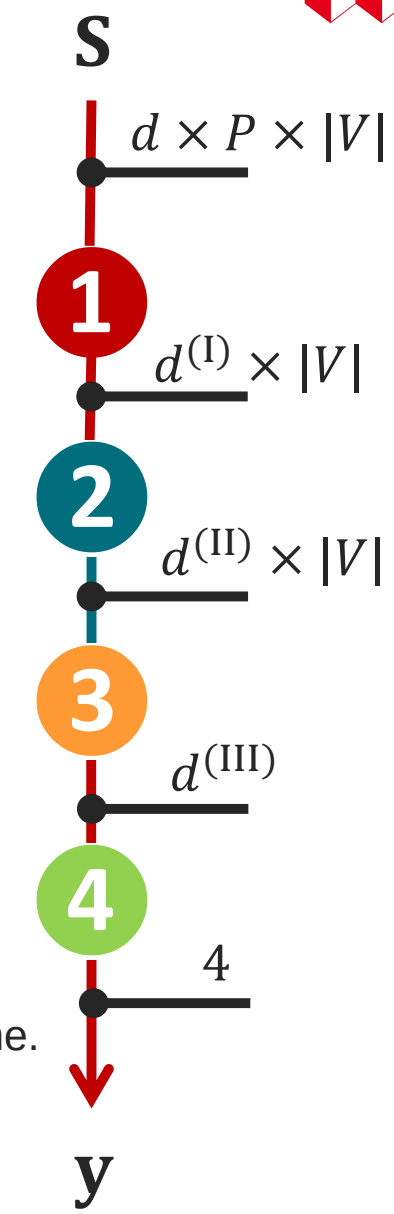



**Fig.** : Pipeline.

# Dynamic GNN for Seismology

## Three GNN blocks

### 2 Spatial information integration $h^{(\mathrm{II})}_{\boldsymbol{\theta}^{(\mathrm{II})}}: \mathbb{R}^{(d^{(\mathrm{I})}+2)\times|V|} \to \mathbb{R}^{d^{(\mathrm{II})}\times|V|}$

- Concatenating and "mixing" → *Edgeless*-**GNN**

$$\mathbf{x}_s^{(\mathrm{II})} = \sigma_2^{(\mathrm{II})}(\mathbf{W}_2^{(\mathrm{II})}\sigma_1^{(\mathrm{II})}\left(\mathbf{W}_1^{(\mathrm{II})}\tilde{\mathbf{x}}_s^{(\mathrm{I})} + \mathbf{b}_1^{(\mathrm{II})}\right) + \mathbf{b}_2^{(\mathrm{II})}),$$

  - $\tilde{\mathbf{x}}_s^{(\mathrm{I})} = \mathbf{x}_s^{(\mathrm{I})}||\mathbf{p}_s$, $\mathbf{p}_s$ is the geographic position of station $s$ and $\mathbf{x}_s^{(\mathrm{I})} = h^{(\mathrm{I})}_{\boldsymbol{\theta}^{(\mathrm{I})}}(\mathbf{x}_s)$ the temporal feature extraction of waveforms ($\mathbf{S} \in \mathbb{R}^{d\times P\times|V|}$).

- Using a local GCN → *Spectral*-**GNN**

$$\mathbf{x}_s^{(\mathrm{II})} = \sigma(\mathbf{x}_s\mathbf{W} + \textstyle\sum_{j|v_j\in\mathcal{N}_{\mathrm{geo}}(v_s)}\left(D_sD_j\right)^{-1/2}\mathbf{x}_j\mathbf{W}) \text{ or } \mathbf{X}'_s = \sigma(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}),$$

  - $D_i$: # of neighbors of $v_i$ (fixed graph), $\tilde{\mathbf{A}} = \mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\left(\mathbf{A} + \mathbf{A}^{\mathrm{T}}\right)\mathbf{D}^{-\frac{1}{2}}/2$ so that we obtain $D_i = \sum_i(A_{ij} + A_{ji})/2$.
  - $\mathcal{N}_{\mathrm{geo}}(v_s)$: neighborhood of node $v_s$ according to the geographic proximity.

- Using a MPNN → *Dynamic*-**GNN**

$$\mathbf{x}_s^{(\mathrm{II},l+1)} = \sum_{k\in\{1,2\}} \bigoplus_{j|v_j\in\mathcal{N}_k(v_s)} f_k^{(l)}(\mathbf{u}_s^{(\mathrm{II},l)}, \mathbf{u}_j^{(\mathrm{II},l)} - \mathbf{u}_s^{(\mathrm{II},l)})).$$

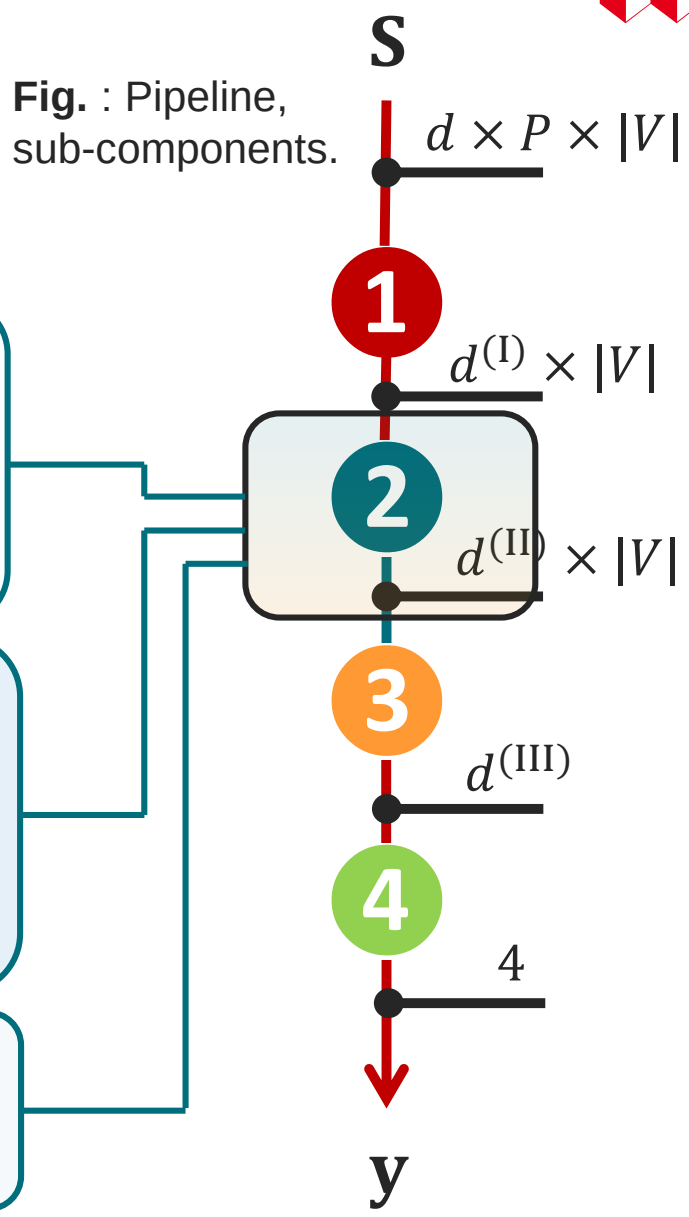



**Fig.** : Pipeline, sub-components.

# Dynamic GNN for Seismology

## Spatio-Temporal Graph Local Convolution

### ST-GC Block (based on MP → *local*)

- **Geographic distance**

$$\mathbf{x}_{\text{s,geo}}^{(l+1)} = \max_{j|v_j \in \mathcal{N}_{\text{geo}}(v_s)} f_{\text{geo}}^{(l)}(\mathbf{u}_s^{(,l)}, \mathbf{u}_j^{(l)} - \mathbf{u}_s^{(l)}),$$

  - $\mathbf{u}_s = \mathbf{x}_s || \mathbf{p}_s$ ($\mathbf{p}_s \in \mathbb{R}^2$) allow the model to distinguish nodes with similar features but different topological roles.
  - Asymmetric update $\mathbb{R}^{d_{\text{geo}}} \to \mathbb{R}^{d^{(1)}}$; $d^{(1)}$: dimension of the encoded features; input dim.: $2(d^{(1)} + 2)$.
  - $\mathcal{N}_{\text{geo}}^{(l)}$ : $\kappa$-NN of based on geographic positions $\|\mathbf{p}_i - \mathbf{p}_j\|_2$.

- **Feature similarity**

$$\mathbf{x}_{\text{s,dist}}^{(l+1)} = \max_{j|v_j \in \mathcal{N}_{\text{sig}}(v_s)} f_{\text{sig}}^{(l)}(\mathbf{x}_s^{(,l)}, \mathbf{x}_j^{(l)} - \mathbf{x}_s^{(l)}),$$

  - Asymmetric update $\mathbb{R}^{d_{\text{sig}}} \to \mathbb{R}^{d^{(1)}}$; input dim.: $2d^{(1)}$.
  - $\mathcal{N}_{\text{sig}}^{(l)}$ : $\kappa$-NN with $\|\mathbf{x}_i - \mathbf{x}_j\|^2$: combine $\mathbf{x}_i$'s from distant stations.

- **Multi-scale** embedding $f$: $\mathbb{R}^{(L+1)d^{(1)} \times |V|} \to \mathbb{R}^{d^{(1)} \times |V|}$

$$\mathbf{X}^{(0)} || \mathbf{X}^{(1)} || \mathbf{X}^{(2)} \cdots || \mathbf{X}^{(L)} \text{ where } \mathbf{X}^{(l)} = \mathbf{X}_{\text{geo}}^{(l)} \oplus \mathbf{X}_{\text{dist}}^{(l)}$$

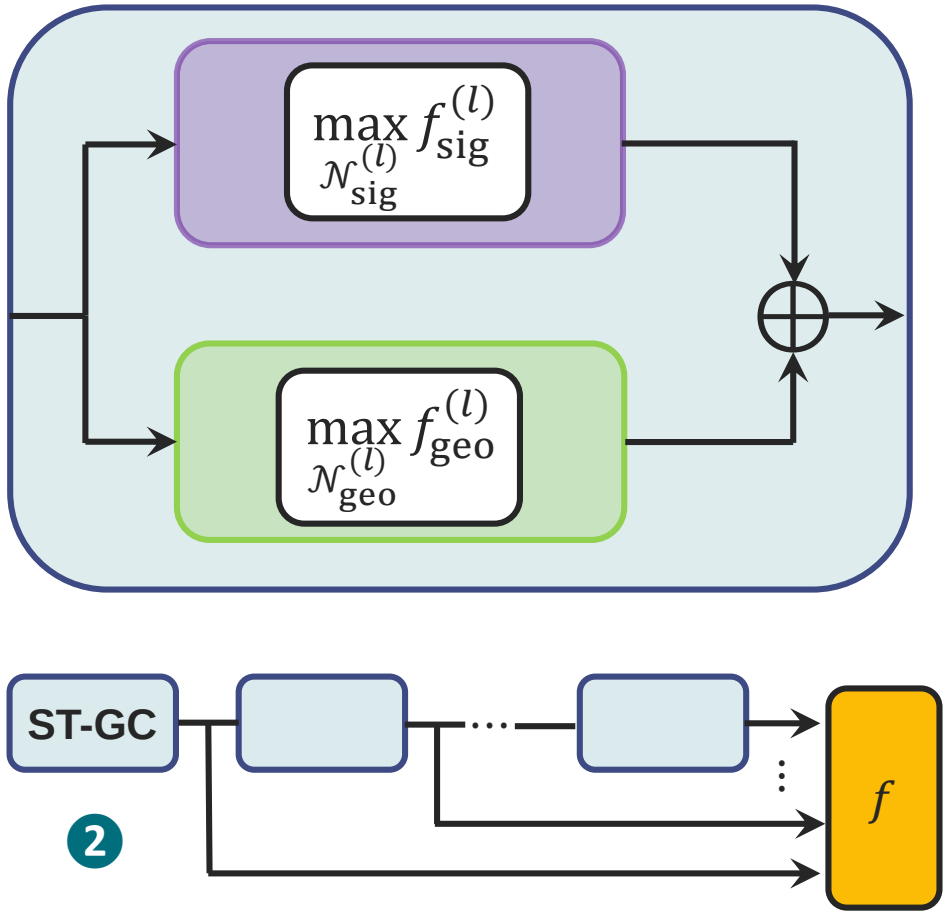



**Fig.** : ST-GC Block and multi-scale embedding.

# Dynamic GNN for Seismology

## Overview

### Convolutional encoding (shared)

- 12 convolutional layers (kernel size = 5).
- Max pooling applied every 3 layers (kernel size = 4).

### Spatial information integration

- *Edgeless*-**GNN** and *Spectral*-**GNN**
  - 2-layer MLP followed by max-pooling.
  - Hidden layer size: $2^7 = 128 \Rightarrow d^{(\mathrm{II})} = 128$.
  - For (local) GCN, $\kappa = 5$ spatial neigbors per node.
- *Dynamic*-**GNN**
  - ST-GC: 2-layer MLPs (one MLP per graph: geometric & feature-based). ST-GC block is repeated $L = 4$ times.
  - Hidden layer size: 32 per branch.
  - $\kappa = 5$, $L = 4$ layers $\Rightarrow d^{(\mathrm{III})} = (L+1)d^{(\mathrm{I})} = 5 \times 2^6 = 320$.

### Prediction

- 2-layer MLP with a hidden size of 128.

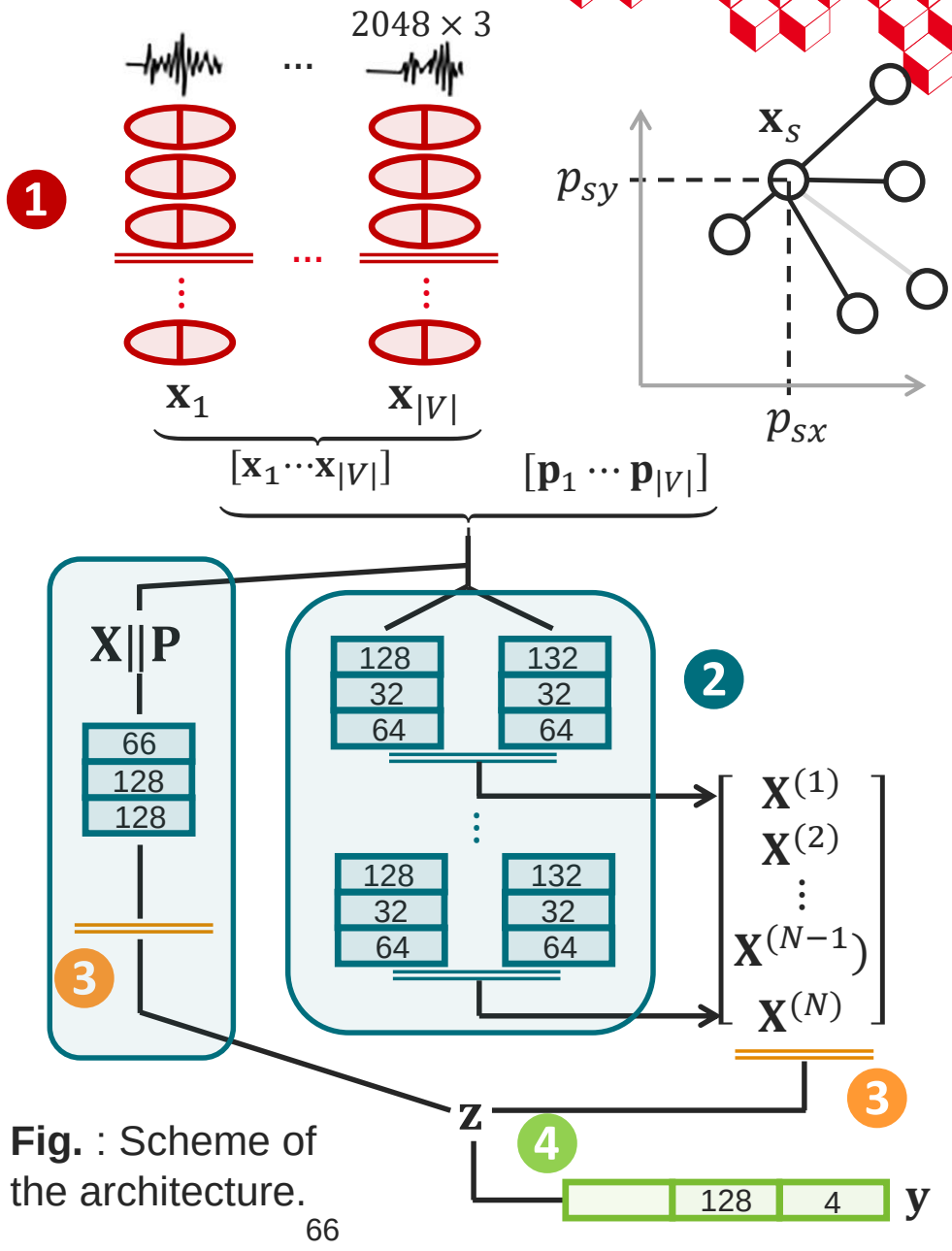


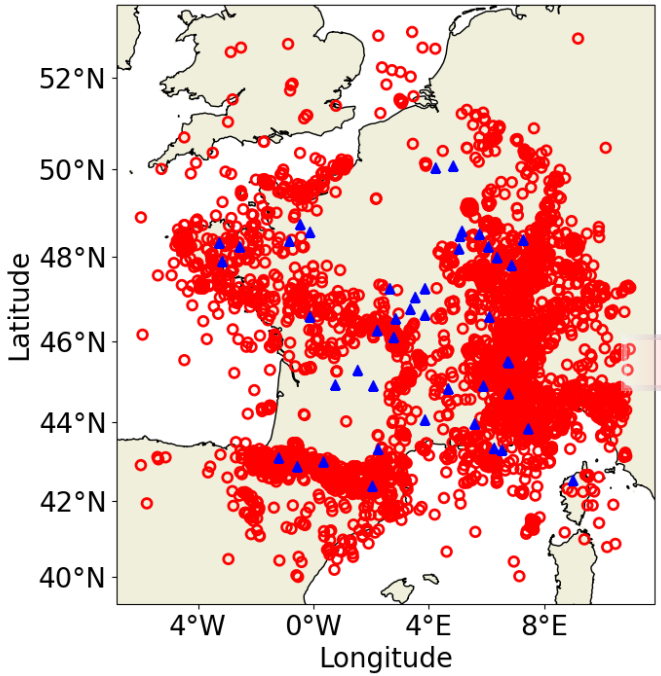

**Fig.** : Scheme of the architecture.

66

# Applications for seismology

## Datasets

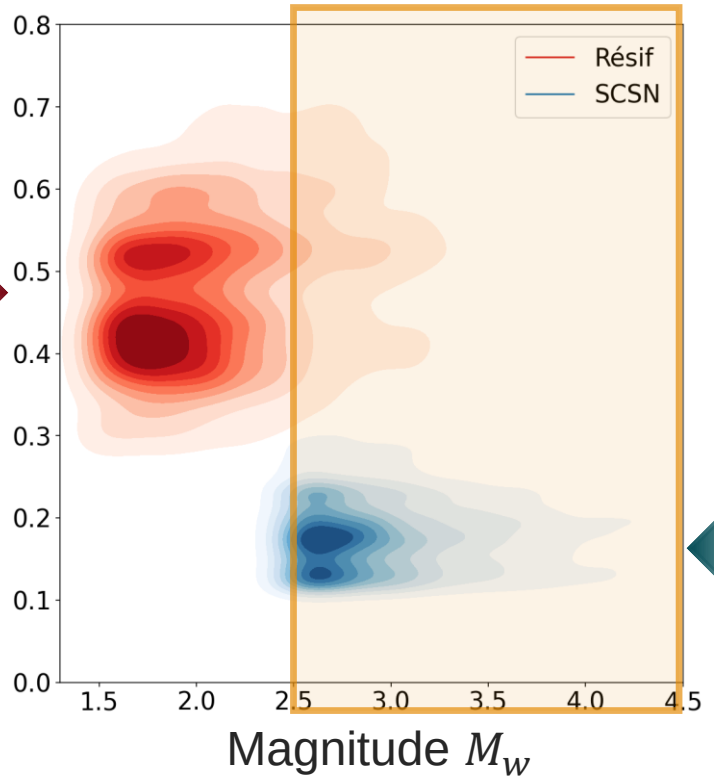
Resif[1] (France)




- 3 yrs (01/01/2019 – 12/31/2021).
- 13479 events ($12 \times 12$ deg).
- 3435 events with $M_l > 2$.
- 42 stations, 1 component.

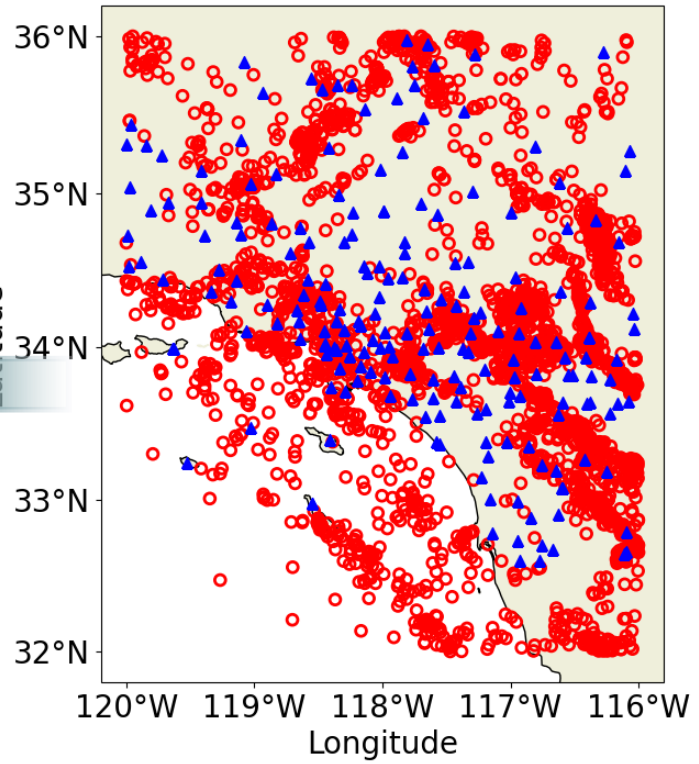
$\langle l \rangle$ ($\times 10^3$ km)

Resif$_{2.5}$
1011 events




- 20 yrs (01/01/2000 – 06/30/2019).
- 3493 events with $M_l > 2$.
- 72 stations, 3 components.

SCSN[2] (USA)




[1] Péguegnat *et al.*, doi: 10.1785/0220200392 (2021)

[2] Hutton, Woessner, Hauksson (2010), doi: 10.1785%2F0120090130 (2010)

# Applications for seismology

## Performance comparison

### Error metrics

- RMSE (for given $y \in \mathbb{R}$)

$$\bar{E} = \frac{1}{K}\sum_{k=1}^{R}\underbrace{\frac{1}{R}\sum_{r=1}^{R}\underbrace{\sqrt{\frac{1}{|\mathcal{D}_k|}\sum_{j\in\mathcal{D}_k}\epsilon_{jkr}^2}}_{E_{kr}}}_{\bar{E}_k}.$$

  - $\epsilon_{jkr} = y_j - \hat{y}_{jkr}$ with $y_j$: ground-truth for the $j$-th event in the test set $\mathcal{D}_k$ and $\hat{y}_{jkr}$: predicted value.
  - $E_{kr}$: RMSE for fold $k$ and seed $r$ ($R = 10$ and $K = 10$).
  - $\bar{E}_k$: average RMSE over the $R$ seeds, for each fold.
  - $\bar{E}$: global RMSE.

- Total variance

$$\sigma^2 = \frac{1}{KR}\sum_{k=1}^{K}\sum_{r=1}^{R}(E_{kr} - \bar{E})^2.$$

  - Variance to seed-level variability and that induced by differences between test-folds.

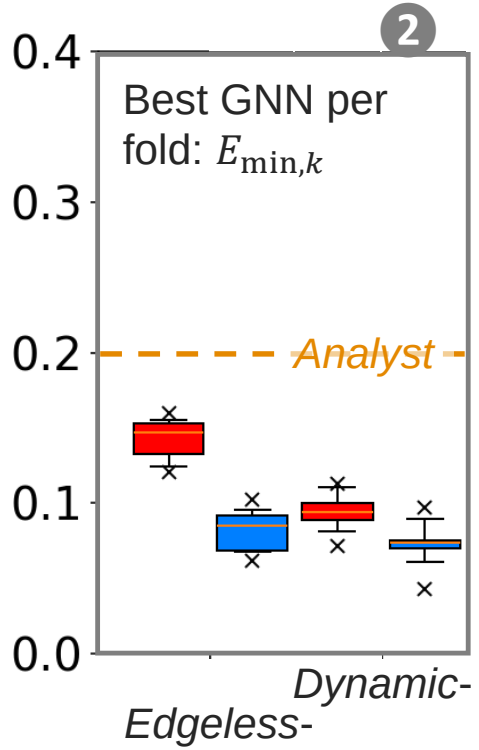
- Boxplots of $E_{kr}$ potential outliers and assess the stability/dispresion of model predictions.

| GNN (1) | $\bar{E} \pm \sigma$ (SCSN) $\mathbf{X} \in \mathbb{R}^{3\times P\times \lvert V\rvert}$ | $\bar{E} \pm \sigma$ (Resif$_{2.5}$) $\mathbf{X} \in \mathbb{R}^{P\times \lvert V\rvert}$ |
|---|---|---|
| *Edgeless-* | $0.175 \pm 0.019$ | $0.151 \pm 0.018$ |
| *Spectral-* | $0.149 \pm 0.020$ | $0.161 \pm 0.020$ |
| *Dynamic-* | $0.129 \pm 0.013$ | $0.141 \pm 0.020$ |

| $\lvert\mathcal{D}\rvert$ (3) | $\epsilon_{M_w}$ |
|---|---|
| 3000 | $0.136 \pm 0.017$ |
| 2000 | $0.136 \pm 0.016$ |
| 1000 | $0.148 \pm 0.018$ |

**Fig.** : (1) $\bar{E}$, $\sigma$; (2) boxplots for $E_{\min,k} = \min_r E_{kr}$; (3) impact $|\mathcal{D}|$.

(2) $\overline{E_{\min,k}}$[ ] $>$ $\overline{E_{\min,k}}$[ ]



# Applications for seismology

## Variance analysis

### Variance decomposition

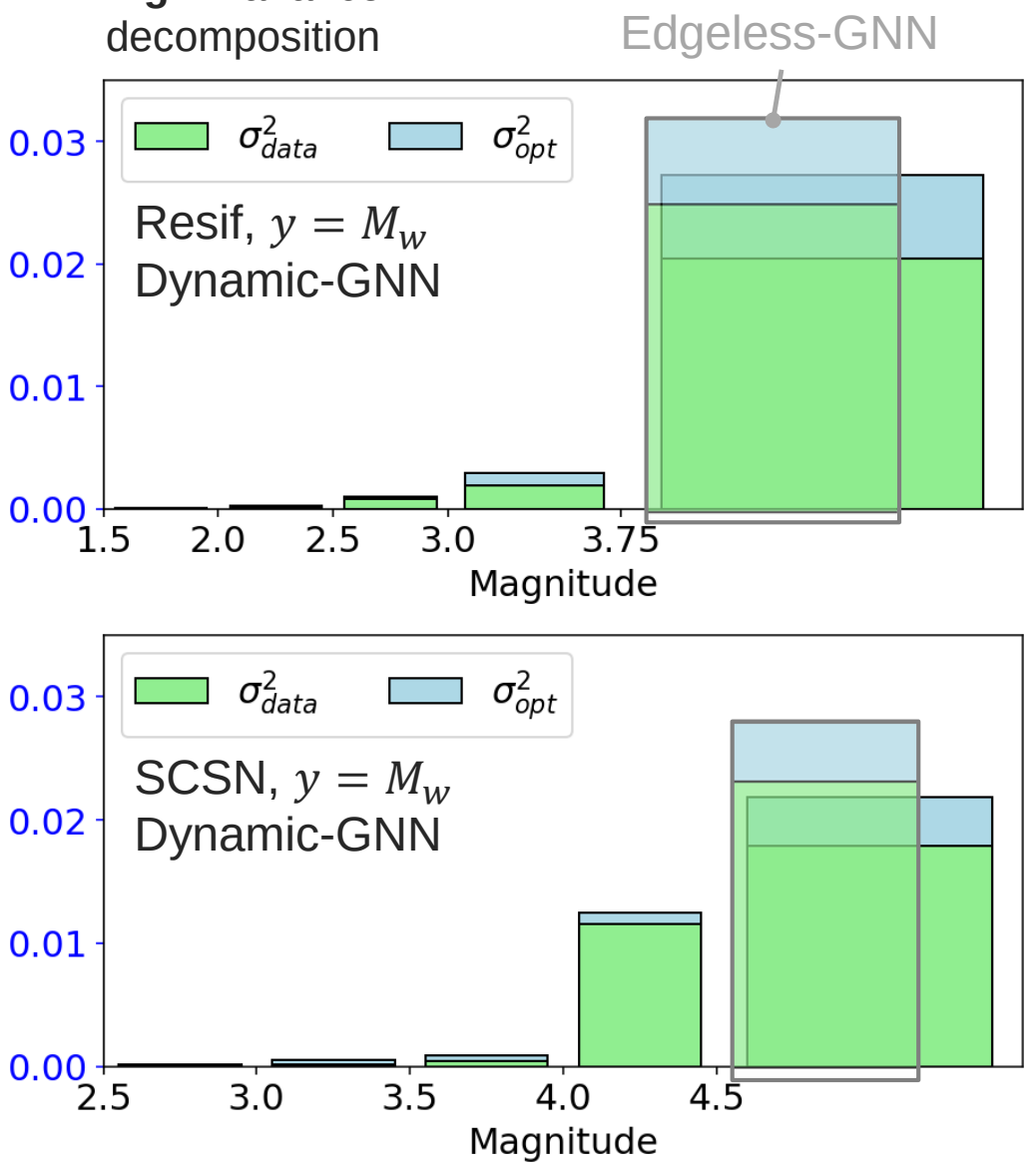
$$\sigma^2 = \underbrace{\frac{1}{KR}\sum_{k=1}^{K}\sum_{r=1}^{R}(\bar{E}_k - \bar{E}_k)^2}_{\sigma^2_{\text{opt}}} + \underbrace{\frac{1}{K}\sum_{k=1}^{K}(\bar{E}_k - \bar{E})^2}_{\sigma^2_{\text{dat}}}.$$

- $\sigma^2_{\text{opt}}$: seed-level variability.
- $\sigma^2_{\text{dat}}$: variability induced by differences between test folds.

### Main results *Edgeless* **vs** *Dynamic*

- **Observations**
  - $\sigma^2$ is primarly driven by $\sigma^2_{\text{dat}}$ as $M_w$ ↗.
  - $\sigma^2_{\text{dat}}$ ↘ with Dynamic-GNN.
- **Explanations**
  - **Architecture**: *Dynamic*-GNN leverages adaptive topology.
  - **Distribution**: More data reduces average RMSE, but imbalanced $M_w$ distribution ⇒ $\sigma^2$ ↗ for rare (large) events.
  - **Optimization**: seed-dependent variability accounts for 1/3 of $\sigma^2$; mainly affects low-$M_w$ events with weaker feature signals and higher SNR.

**Fig.** : Variance decomposition



# Applications for seismology

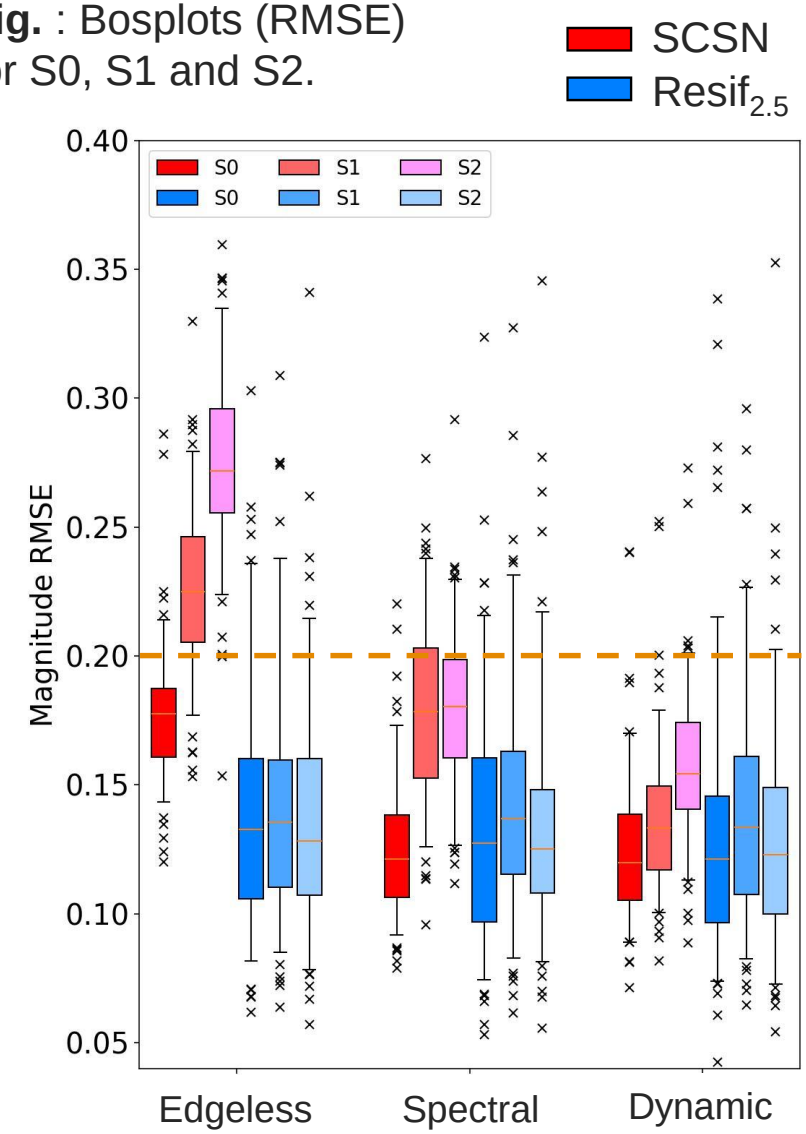
## Robustness to input signals

### Three strategies

$$\mathbf{X}_{S0} \rightarrow \mathbf{X}_{S1} \rightarrow \mathbf{X}_{S2}$$

- S0 (standard): **reference** configuration
  - Broadband filtering (1–8 Hz): removes LFs below the corner frequency ($\sim 0.2$ Hz), which are critical for seismic moment estimation.
  - Aligned time windows, based on origin time $t_0$ of each event.
- S1 (narrowband): **filtered** waveforms
  - Apply a bandpass filter (1–1.5 Hz).
  - Removes much of the spectral content related to rupture complexity (e.g., HF scattering outside the band).
- S2 (random sliding): **misaligned** windows
  - Apply a random shift ($10^2 \mathrm{U}([-1,1])$) to the start time of each window.
  - Add station-specific time shifts compare to published works*.
  - Breaks alignment to theoretical arrival times, simulating poor or missing phase picks (e.g., early warning, low-SNR environments).

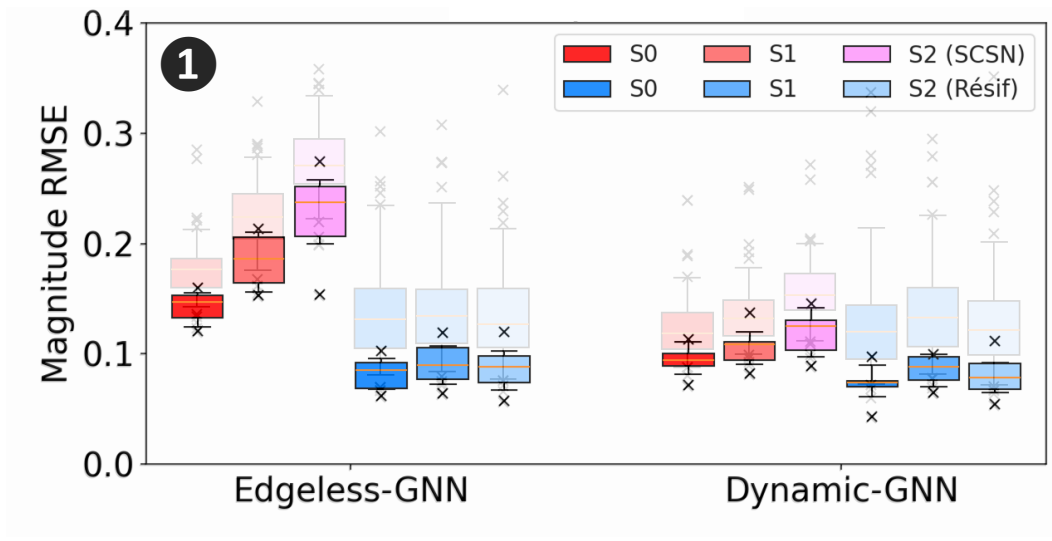
**Fig.** : Bosplots (RMSE) for S0, S1 and S2.

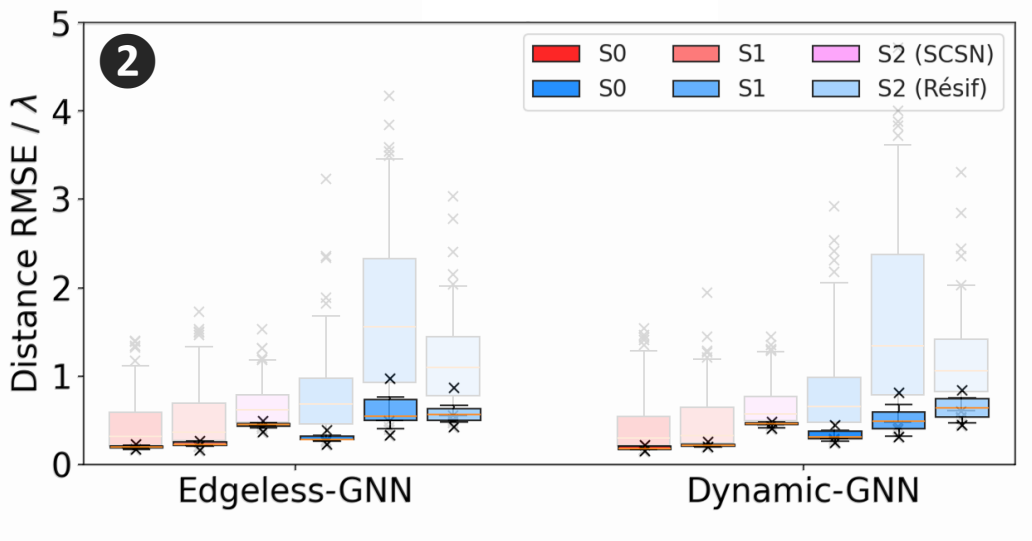



* van den Ende & Ampuero (2020), doi:10.1029/2020GL088690

# Applications for seismology

## Gains depend on graph and data quality




### Trends

- Why doesn't Dynamic-GNN outperform Edgeless-GNN on $\text{Resif}_{2.5}$ and S2? (1)
- S2 do not impact localization performance (2)
  - A proxy scale can be defined as:

$$\lambda = \frac{1}{N(\kappa-1)} \sum_{i=1}^{N} \sum_{j=1}^{\kappa-1} d_{\text{geo}}(v_i, n_j(v_i)).$$

    $d_{\text{geo}}$: geodesic distance to the neighbors $n_j(v_i)$
  - $\lambda \simeq 104$ km for $\text{Resif}_{2.5}$ and $\lambda \simeq 39$ km for SCSN.
  - GNN are able to resolve **sub-$\lambda$ spatial** differences.

- The retained events ($M_w > 2.5$) are less sensitive to SNR effects, making GNN models equally robust.
- The $\text{Resif}_{2.5}$ network is sparse and uneven, reducing the ability of dynamic graphs to extract meaningful spatial patterns.
- The limited number of training events (≈1000) may not support the higher model complexity of Dynamic-GNN.

# Applications for seismology

## Interpretability of graphs

### Sensitivity indices

- Contribution of node $s$ to the final prediction:

$$\left[S_s^{(L)}\right]^2 = \left\|\nabla_{\mathbf{x}_s^{(l)}} \epsilon\right\|_2^2.$$

Using $\mathbf{z} = \max_s \mathbf{x}_s$ leads to:

$$\left[S_s^{(L)}\right]^2 = \sum_{j=1}^{d_z} 1_{s=s_j^{(L)}} \left(\partial_{z_j} \epsilon\right)^2.$$

$S_s^{(L)}$ acts as a station-wise attribution **score**, highlighting which nodes contributed most to the prediction error $\epsilon$.

### Layer-Dependent Spatial Attention

- Deeper GNN layers focus on fewer, high-sensitivity stations $V^{(l)} \subset V$:

$$\left|V^{(l)}\right|/|V| \to h.$$

- Stations cluster near the epicenter, resembling a learned spatial **attention** mechanism.

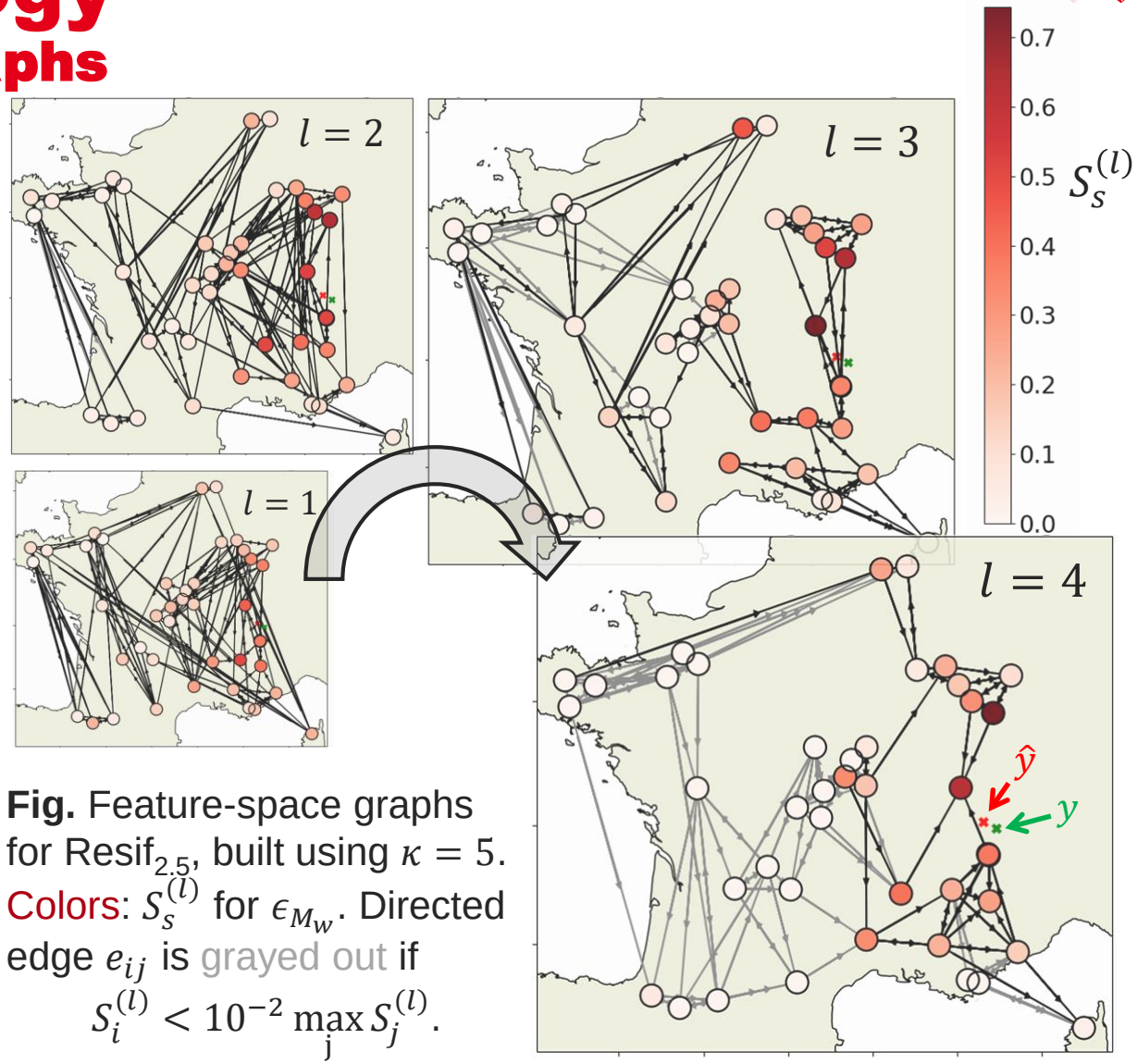



**Fig.** Feature-space graphs for Resif$_{2.5}$, built using $\kappa = 5$. Colors: $S_s^{(l)}$ for $\epsilon_{M_w}$. Directed edge $e_{ij}$ is grayed out if $S_i^{(l)} < 10^{-2} \max_j S_j^{(l)}$.

# Applications for seismology

## Oversmoothing

[1] Oono & Suzuki, arxiv 1905.10947, 2019
[2] Rusch, Bronstein & Mishra, arxiv 2303.10993, 2023

### Oversmoothing

- **Deeper GNNs tend to oversmooth**, i.e. make node embeddings indistinguishable across $G_{\text{geo}}$ and $G_{\text{sig}}$.
- For GCNs with fixed topology and linear propagation, $\nearrow L$ causes the node embeddings to converge towards a low-dim. invariant subspace: $d_{\mathcal{M}}(\mathbf{X}^{(l)}) \leq (s\lambda)^l d_{\mathcal{M}}(\mathbf{X}^{(0)})$ [1].

### Dirichlet energy $\text{Tr}(\mathbf{X}^{\text{T}}\mathbf{L}_{\text{sym}}\mathbf{X})$ [2]

- Measures local variation of node embeddings (low $\mathcal{E}$ = smoothness). For directed graphs:

$$\mathcal{E}(\mathbf{X}^{(l)}) = \sum_i \sum_{j|v_j \in \mathcal{N}(v_i)} \left\| \mathbf{e}_{ij}^{(l)} \right\|_2^2$$

  - $\mathbf{e}_{ij}^{(l)} = \mathbf{x}_i^{(l)} \left(d_i^{\text{in}}\right)^{-1/2} - \mathbf{x}_j^{(l)} \left(d_j^{\text{out}}\right)^{-1/2}$ with the dimensions defined as $d_i^{\text{in}} = \kappa$ and $d_j^{\text{out}} = \#\{v_i | v_j \in \mathcal{N}(v_i)\}$.
  - This double normalization ensures $\mathcal{E}$ remains **well-defined** when $\mathcal{N}(v_i)$'s are defined via **attention mechanisms.**

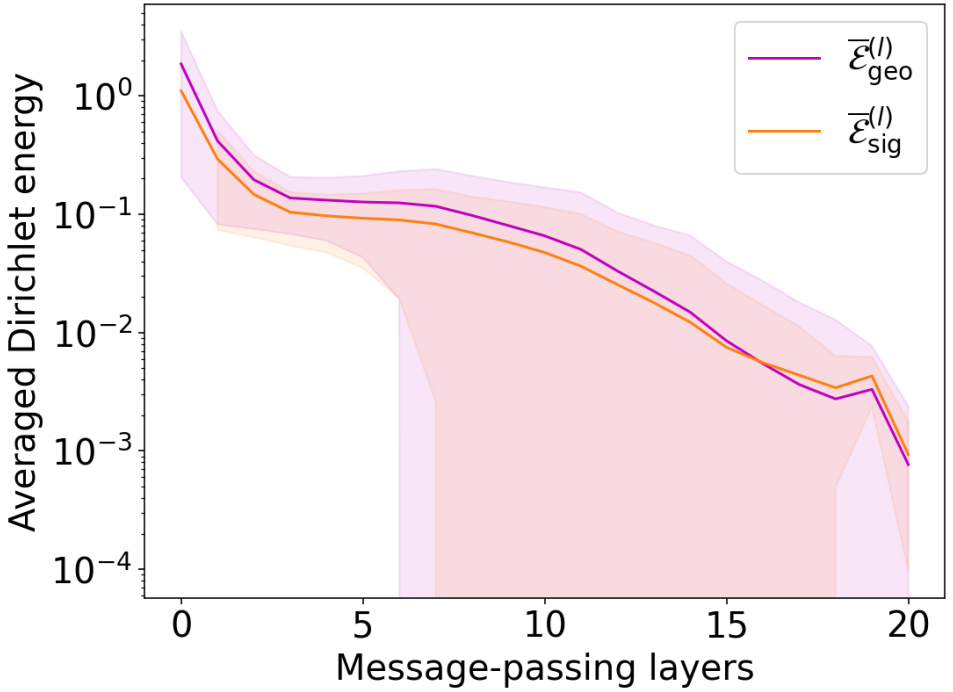



**Fig.** Averaged Dirichlet energies over all the events and $\pm\sigma$. Exponential decrease is satisfied which confirms oversmoothing. Resif$_{2.5}$.

# Applications for seismology

## Towards deeper GNNs

### Concat in ❷ mitigates oversmoothing

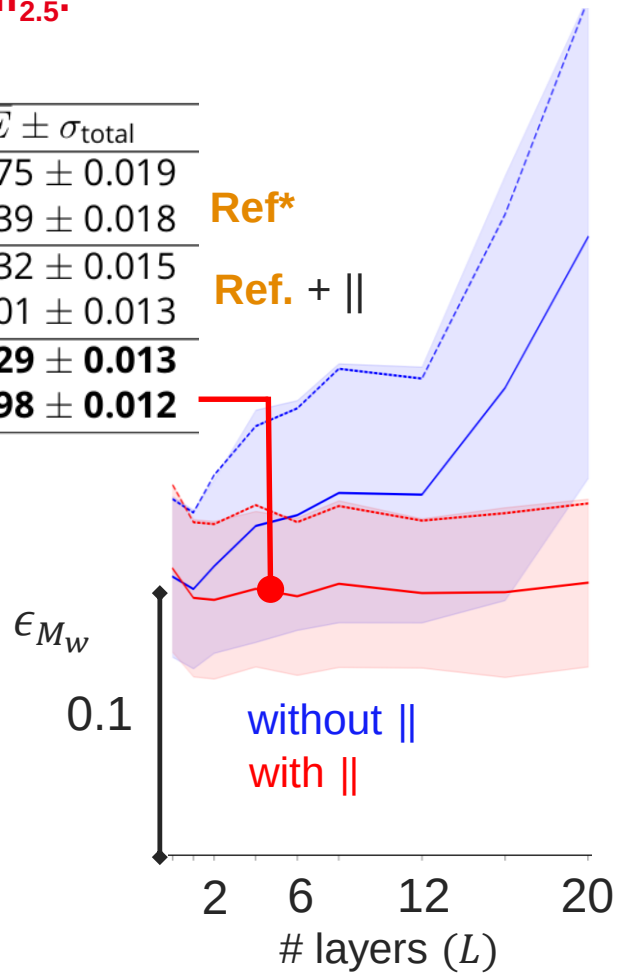
- $\kappa = |\mathcal{N}(v_i)|$ defines the local receptive field (= 5)
  - Stacking multiple layers means that the local neighborhoods are recursively propagated across layers.
  - $\kappa$ and $L$ should be **co-tuned** to avoid oversmoothing.
- || provides multi-scale agreggation (❸) that maps $\widetilde{\mathbf{X}}^{(L)} \in \mathbb{R}^{Ld}$ into a lower dimensional space

$$\underbrace{\mathbf{X}^{(0)} || \cdots || \mathbf{X}^{(L)}}_{\widetilde{\mathbf{X}}^{(L)}} \mapsto z = \mathrm{Pool}(\widetilde{\mathbf{X}}^{(L)}) \in \mathbb{R}^{d}.$$

  - $\mathbf{X}^{(0)} = \mathbf{X} || \mathbf{P}$ ($\mathbf{X}$: output of ❶) inspired by Positional Encodings.
  - || Plays a similar role to **multi-head attention** in Graph Transformers: aggregating multi-scale relational patterns into a unified representation
- || can be applied to large classes of GNN.
  - For **Edgeless***-GNN and “poor” datasets (**Resif$_{2.5}$**) → performance.
  - Preserve express. for GCNs (Chen *et al.*, arxiv 2007.02133, 2020).

**Fig.** Prediction error with *Dynamic*-GNN. $\overline{\mathrm{MAE}} \pm \sigma$ (solid lines); $\overline{\mathrm{RMSE}}$ (dashed line) **Resif$_{2.5}$.**

| Type | $\overline{E} \pm \sigma_{\text{total}}$ |
|---|---|
| RMSE | $0.175 \pm 0.019$ |
| MAE | $0.139 \pm 0.018$ |
| RMSE | $0.132 \pm 0.015$ |
| MAE | $0.101 \pm 0.013$ |
| RMSE | $\mathbf{0.129 \pm 0.013}$ |
| MAE | $\mathbf{0.098 \pm 0.012}$ |

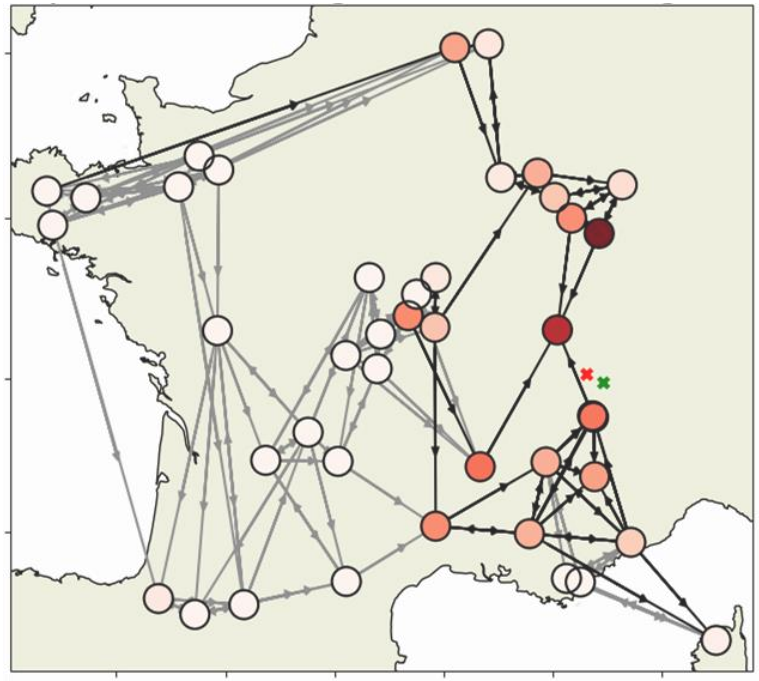


# Applications for seismology

## What have we learned?

### Dynamic Graphs: Physics-aware and interpretable

- The *Dynamic-GNN* learns **adaptive station-to-station connectivity**, reflecting **underlying physical processes** like wavefront propagation.
- **Sensitivity scores** reveal how information is routed, resembling a **spatial attention mechanism** focusing on informative stations near the epicenter.



### Concatenation mitigates oversmoothing

- Aggregating intermediate outputs provides **multi-scale information** and preserves node-wise diversity (conceptually aligned with **multi-head attention** in Transformers, where different attention heads specialize in different relational patterns).

### Robustness in low-data settings

- The Dynamic-GNN adapts its graph to the signal structure, improving generalization on **small or imbalanced datasets** (Resif$_{2.5}$ vs SCSN).
- Fixed-topology models struggle in such settings, as they cannot exploit data-specific spatial correlations.

# Open questions

## Over-SMoothing (OSM)

- **OSM is not always harmful**
  - OSM is not systematically detrimental to performance — it can help classification if it aligns with the task objective (**Fig.**).
  - Keriven et al. (2022) suggest: *some OSM is desired*, especially in homophilic graphs.
- **OSM happens non-uniformly**
  - Smoothing can occur faster in some embedding subspaces than others. If labels are aligned with these slow-smoothing subspaces → performance improves.
  - This leads to the idea of **task-oriented smoothing**.
- **OSM analysis in real-world GNNs**
  - Real GNNs may not suffer as much from OSM, thanks to residual connections, gated or relational variants, etc.
  - There is a need to extend OSM analysis to real-world architectures and not idealized deep GCNs.

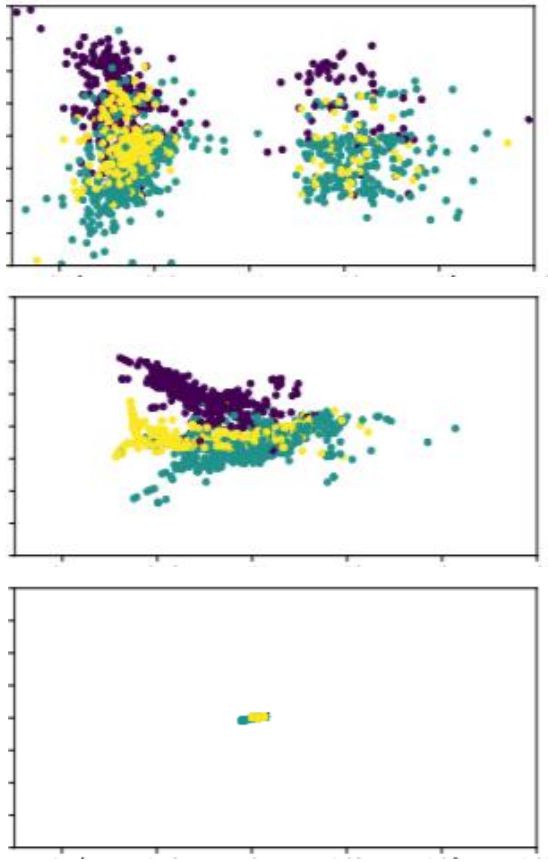
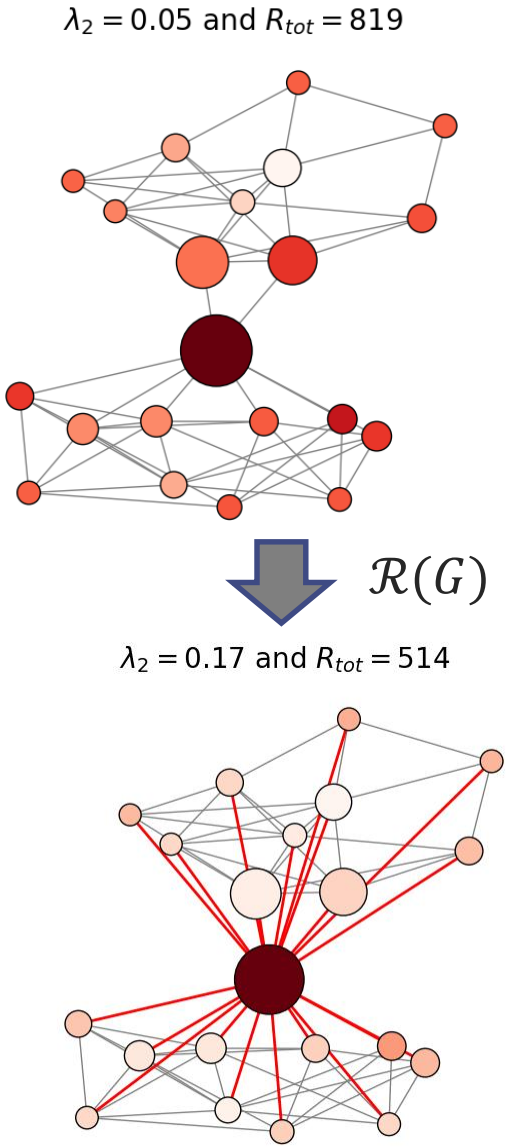


**Fig.** Evolution of node embeddings from a GCN under increasing message-passing depth, showing class separation collapse as smoothing increases.

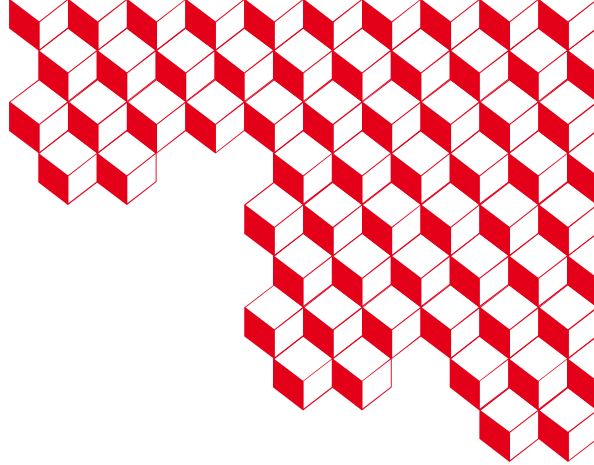
# Open questions

**Fig.** Rewiring the graph improves connectivity by reducing bottlenecks. The transformation $\mathcal{R}(G)$ increases $\lambda_2$, lowers resistance $R_{\text{tot}}$, and facilitates long-range message passing.




## Over-SQuashing (OSQ)

- **Compression vs. Bottlenecks**
  - OSQ arises from **exponential compression** of messages from many paths through a limited number of edges (bottlenecks).
  - Classical metrics ($\lambda_2$, $R_{\text{tot}}$) often used, but Structural metrics may fail to capture task-relevant information flow, especially when node labels or signals are misaligned with structural bottlenecks.
- **Bottleneck metrics vs. task sensitivity**
  - Metrics like **curvature**, **resistance distance** identify structural bottlenecks.
  - But they might not align with what the network is trying to learn → There's a **gap between structural measures and task relevance**.
- **Spectral perspective**
  - **Spectral decomposition** offers a way to measure how label information is distributed across frequency modes:
    - Homophilic labels: aligned with **low-frequency** modes.
    - Heterophilic labels: encoded in **high-frequency** components.



# Any questions now or later?

Christophe.millet@cea.fr

école normale supérieure paris–saclay

université PARIS-SACLAY

CENTRE BORELLI

# References


- Abboud, R., Dimitrov, R., & Ceylan, I. I. (2022, December). Shortest path networks for graph property prediction. In Learning on Graphs Conference (pp. 5-1). PMLR.
- Abu-El-Haija, Sami, et al. "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing." international conference on machine learning. PMLR, 2019.
- Alev, V. L., Anari, N., Lau, L. C., & Gharan, S. O. (2018). Graph clustering using effective resistance http://drops.dagstuhl.de/opus/volltexte/2018/8369.
- Alon, U., & Yahav, E. (2020). On the bottleneck of graph neural networks and its practical implications. In ICLR 2021.
- Arnaiz-Rodríguez, A., Begga, A., Escolano, F., & Oliver, N. (2022). Diffwire: Inductive graph rewiring via the Lovász bound. In the First Learning on graphs (LoG) Conference 2022.
- Arnaiz-Rodriguez, A., Curto, G., & Oliver, N. (2024). Structural Group Unfairness: Measurement and Mitigation by means of the Effective Resistance. In TrustLOG Workshop at WWW 2024.
- Banerjee, P. K., Karhadkar, K., Wang, Y. G., Alon, U., & Montúfar, G. (2022, September). Oversquashing in gnns through the lens of information contraction and graph expansion. In 2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton) (pp. 1-8). IEEE.
- Chamberlain, B., Rowbottom, J., Eynard, D., Di Giovanni, F., Dong, X., & Bronstein, M. (2021). Beltrami flow and neural diffusion on graphs. Advances in Neural Information Processing Systems, 34, 1594-1609.
- Bi, W., Du, L., Fu, Q., Wang, Y., Han, S., & Zhang, D. (2022). Make heterophily graphs better fit gnn: A graph rewiring approach. arXiv preprint arXiv:2209.08264.
- Black, M., Wan, Z., Nayyeri, A., & Wang, Y. (2023, July). Understanding oversquashing in gnns through the lens of effective resistance. In International Conference on Machine Learning (pp. 2528-2547). PMLR.

# References


- Bodnar, C., Di Giovanni, F., Chamberlain, B., Lio, P., & Bronstein, M. (2022). Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. Advances in Neural Information Processing Systems, 35, 18527-18541.
- Brüel-Gabrielsson, R., Yurochkin, M., & Solomon, J. (2022). Rewiring with positional encodings for graph neural networks. TMLR 2023
- Buchnik, E., & Cohen, E. (2018, June). Bootstrapped graph diffusions: Exposing the power of nonlinearity. In Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems (pp. 8-10).
- Cai, C., & Wang, Y. (2020). A note on over-smoothing for graph neural networks. In ICML 2020.
- Cai, C., Hy, T. S., Yu, R., & Wang, Y. (2023, July). On the connection between mpnn and graph transformer. In International Conference on Machine Learning (pp. 3408-3430). PMLR.
- Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M., Webb, S., & Rossi, E. (2021, July). Grand: Graph neural diffusion. In International conference on machine learning (pp. 1407-1418). PMLR.
- Chamberlain, B., Rowbottom, J., Eynard, D., Di Giovanni, F., Dong, X., & Bronstein, M. (2021b). Beltrami flow and neural diffusion on graphs. Advances in Neural Information Processing Systems, 34, 1594-1609.
- Chen, M., Wei, Z., Huang, Z., Ding, B., & Li, Y. (2020, November). Simple and deep graph convolutional networks. In International conference on machine learning (pp. 1725-1735). PMLR.
- Chen, T., Zhou, K., Duan, K., Zheng, W., Wang, P., Hu, X., & Wang, Z. (2022). Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study. IEEE TPAMI.
- Chung, F. R. (1997). Spectral graph theory (Vol. 92). American Mathematical Soc..
- Devriendt, K., & Lambiotte, R. (2022). Discrete curvature on graphs from the effective resistance. Journal of Physics: Complexity, 3(2), 025008.

# References


- Di Giovanni, F., Giusti, L., Barbero, F., Luise, G., Lio, P., & Bronstein, M. M. (2023, July). On over-squashing in message passing neural networks: The impact of width, depth, and topology. ICML
- Di Giovanni, F., Rowbottom, J., Chamberlain, B. P., Markovich, T., & Bronstein, M. M. (2023). Understanding convolution on graphs via energies. In TLMR.
- Di Giovanni, F., Rusch, T. K., Bronstein, M. M., Deac, A., Lackenby, M., Mishra, S., & Veličković, P. (2024). How does over-squashing affect the power of GNNs?. TMLR.
- Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., & Beaini, D. (2022). Long range graph benchmark. Advances in Neural Information Processing Systems, 35, 22326-22340.
- Eliasof, Moshe, Eldad Haber, and Eran Treister. "Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations." Advances in neural information processing systems 34 (2021): 3836-3849.
- Errica, F., Christiansen, H., Zaverkin, V., Maruyama, T., Niepert, M., & Alesiani, F. (2024). Adaptive Message Passing: A General Framework to Mitigate Oversmoothing, Oversquashing, and Underreaching. In JMLR 2024
- Fesser, L., & Weber, M. (2024, April). Mitigating over-smoothing and over-squashing using augmentations of Forman-Ricci curvature. In Learning on Graphs Conference (pp. 19-1). PMLR.
- Gasteiger, J., Weißenberger, S., & Günnemann, S. (2019). Diffusion improves graph learning. Advances in neural information processing systems, 32.
- Giraldo, J. H., Skianis, K., Bouwmans, T., & Malliaros, F. D. (2023, October). On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In ICKM.
- Gutteridge, B., Dong, X., Bronstein, M. M., & Di Giovanni, F. (2023, July). Drew: Dynamically rewired message passing with delay. In International Conference on Machine Learning (pp. 12252-12267). PMLR.

# References


- Hamilton, W. L. (2020). Graph representation learning. Morgan & Claypool Publishers.
- Hasanzadeh, A., Hajiramezanali, E., Boluki, S., Zhou, M., Duffield, N., Narayanan, K., & Qian, X. (2020, November). Bayesian graph neural networks with adaptive connection sampling. In ICML 2022.
- Huang, K., Wang, Y. G., & Li, M. (2024). How Universal Polynomial Bases Enhance Spectral Graph Neural Networks: Heterophily, Over-smoothing, and Over-squashing. arXiv preprint arXiv:2405.12474.
- Jamadandi, Adarsh, Celia Rubio-Madrigal, and Rebekka Burkholz. "Spectral Graph Pruning Against Over-Squashing and Over-Smoothing." arXiv preprint arXiv:2404.04612 (2024).
- Karhadkar, K., Banerjee, P. K., & Montúfar, G. (2022). FoSR: First-order spectral rewiring for addressing oversquashing in GNNs. In ICLR 2023
- Jiang, W., Liu, H., & Xiong, H. (2023). Survey on Trustworthy Graph Neural Networks: From A Causal Perspective. arXiv preprint arXiv:2312.12477.
- Keriven, N. Not too little, not too much: a theoretical analysis of graph (over) smoothing. NeurIPS 2022.
- Kondor, R. I., & Lafferty, J. (2002, July). Diffusion kernels on graphs and other discrete structures. In Proceedings of the 19th international conference on machine learning (Vol. 2002, pp. 315-322).
- Li, Q., Han, Z., & Wu, X. M. (2018, April). Deeper insights into graph convolutional networks for semi-supervised learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32, No. 1).
- Li, G., Muller, M., Thabet, A., & Ghanem, B. (2019). Deepgcns: Can gcns go as deep as cnns?. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 9267-9276).
- Liu, M., Gao, H., & Ji, S. (2020, August). Towards deeper graph neural networks. In Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 338-348).

# References


- Liu, Y., Zhou, C., Pan, S., Wu, J., Li, Z., Chen, H., & Zhang, P. (2023, April). Curvdrop: A ricci curvature based approach to prevent graph neural networks from over-smoothing and over-squashing. In WWW.
- Liu, Y., Zheng, Y., Zhang, D., Lee, V. C., & Pan, S. (2023b, June). Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating. AAAI.
- Lim, D., et al. "New benchmarks for learning on non-homophilous graphs“. In WWW Workshop on GLB, 2021.
- Lovász, L. (1993). Random walks on graphs. Combinatorics, Paul erdos is eighty, 2(1-46), 4.
- Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S., ... & Precup, D. (2022). Revisiting heterophily for graph neural networks. Advances in neural information processing systems, 35, 1362-1375.
- Luan, S. et al (2024). The Heterophilic Graph Learning Handbook: Benchmarks, Models, Theoretical Analysis, Applications and Challenges. Arxiv 2407.09618
- Ma, Y., Liu, X., Shah, N., & Tang, J. (2022). Is homophily a necessity for graph neural networks?. ICLR 2022.
- Maskey, S., Paolino, R., Bacho, A., & Kutyniok, G. (2024). A fractional graph laplacian approach to oversmoothing. Advances in Neural Information Processing Systems, 36.
- Newman, M. “Assortative mixing in networks”. Phys. Rev. Lett., 89, 2002.
- Nguyen, K., Hieu, N. M., Nguyen, V. D., Ho, N., Osher, S., & Nguyen, T. M. (2023, July). Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In ICML 2023.
- Oono, K., & Suzuki, T. (2019). Graph neural networks exponentially lose expressive power for node classification. In ICLR 2020.
- Pei, H. et al. “Geom-GCN: Geometric GCNs”. In ICLR, 2019.

# References


- Pham, T., Tran, T., Dam, H., & Venkatesh, S. (2017). Graph classification via deep learning with virtual nodes. arXiv preprint arXiv:1708.04357.
- Qiu, H., & Hancock, E. R. (2006). Graph embedding using commute time. In Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, SSPR 2006 and SPR 2006,
- Qian, Y., Expert, P., Rieu, T., Panzarasa, P., & Barahona, M. (2021). Quantifying the alignment of graph and features in deep learning. IEEE TNNLS
- Rong, Y., Huang, W., Xu, T., & Huang, J. (2020). Dropedge: Towards deep graph convolutional networks on node classification. In ICLR 2020.
- Rusch, T. K., Bronstein, M. M., & Mishra, S. (2023). A survey on oversmoothing in graph neural networks. arXiv preprint arXiv:2303.10993.
- Rusch, T. K., Chamberlain, B., Rowbottom, J., Mishra, S., & Bronstein, M. (2022, June). Graph-coupled oscillator networks. In International Conference on Machine Learning (pp. 18888-18909). PMLR.
- Southern, J., Di Giovanni, F., Bronstein, M., & Lutzeyer, J. F. (2024). Understanding Virtual Nodes: Oversmoothing, Oversquashing, and Node Heterogeneity. arXiv preprint arXiv:2405.13526.
- Shao, Z., Shi, D., Han, A., Guo, Y., Zhao, Q., & Gao, J. (2023). Unifying over-smoothing and over-squashing in graph neural networks: A physics informed approach and beyond. arXiv preprint arXiv:2309.02769.
- Spielman D. (2018). Spectral Graph Theory, Lecture 10: Random Walks on Graphs. Lecture at Yale https://www.cs.yale.edu/homes/spielman/561/lect10-18.pdf
- Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., & Bronstein, M. M. (2021). Understanding over-squashing and bottlenecks on graphs via curvature. ICLR 2022.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks.
- Xhonneux, L. P., Qu, M., & Tang, J. (2020, November). Continuous graph neural networks. In International conference on machine learning (pp. 10432-10441). PMLR.

# References


- Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K. I., & Jegelka, S. (2018, July). Representation learning on graphs with jumping knowledge networks. In International conference on machine learning (pp. 5453-5462). PMLR.
- Zheng, C., Zong, B., Cheng, W., Song, D., Ni, J., Yu, W., ... & Wang, W. (2020, November). Robust graph representation learning via neural sparsification. In International Conference on Machine Learning (pp. 11458-11468). PMLR.
- Zheng, Y., Luan, S., & Chen, L. (2024). What Is Missing In Homophily? Disentangling Graph Homophily For Graph Neural Networks. arXiv preprint arXiv:2406.18854.
- Zhao, L., & Akoglu, L. (2020). Pairnorm: Tackling oversmoothing in gnns. In ICLR 2020.
- Zhao, J., Dong, Y., Tang, J., Ding, M., & Wang, K. (2021). Generalizing graph convolutional networks via heat kernel.
- Zhou, K., Huang, X., Li, Y., Zha, D., Chen, R., & Hu, X. (2020). Towards deeper graph neural networks with differentiable group normalization. Advances in neural information processing systems, 33, 4917-4928.
- Zhou, K., Huang, X., Zha, D., Chen, R., Li, L., Choi, S. H., & Hu, X. (2021). Dirichlet energy constrained learning for deep graph neural networks. Advances in Neural Information Processing Systems, 34, 21834-21846.
- Zhou, K., Dong, Y., Wang, K., Lee, W. S., Hooi, B., Xu, H., & Feng, J. (2021b, October). Understanding and resolving performance degradation in deep graph convolutional networks. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management (pp. 2728-2737).
- Zhu, J., et al. “Beyond homophily in graph neural networks: Current limitations and effective designs”. in NeurIPS, 2020
- Zhu, Y., Xu, W., Zhang, J., Du, Y., Zhang, J., Liu, Q., ... & Wu, S. (2021). A survey on graph structure learning: Progress and opportunities. arXiv preprint arXiv:2103.03036.
- Zhu, Y., Du, Y., Wang, Y., Xu, Y., Zhang, J., Liu, Q., & Wu, S. (2022, December). A survey on deep graph generation: Methods and applications. In Learning on Graphs Conference (pp. 47-1). PMLR.