

EECE 576L

Advanced Topics in Deep Learning Lecture 3: Graph Neural Networks

Renjie Liao

University of British Columbia

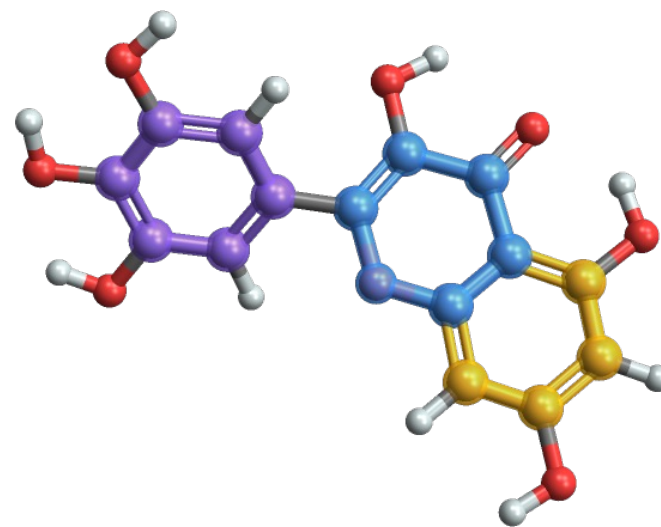
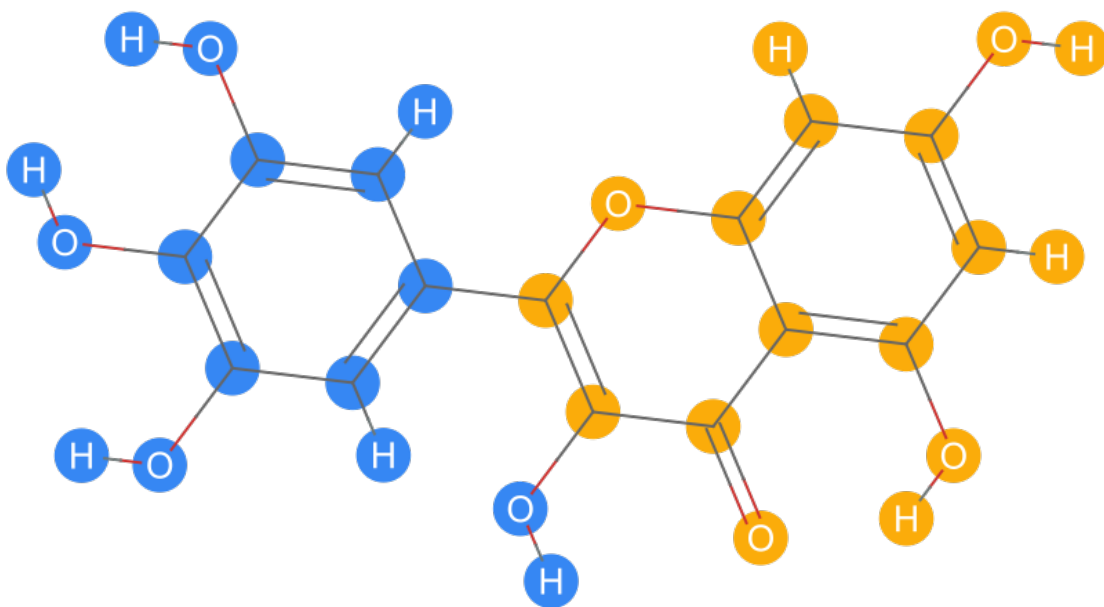
2026/27 Winter Session, Term 1 (Fall 2026)

Outline

- **Graphs and permutation symmetry**
 - Representations, node features, and relabeling
- Message passing neural networks
 - Messages, aggregation, updates, and graph prediction
- Graph convolutional networks (GCNs)
 - Graph signals, Laplacian bases, Fourier transforms, and convolution
 - Polynomial and Chebyshev filters
 - Deriving the GCN layer
- Expressivity and modern graph architectures
 - The WL test, depth, attention, and spectral features

Why Graphs?

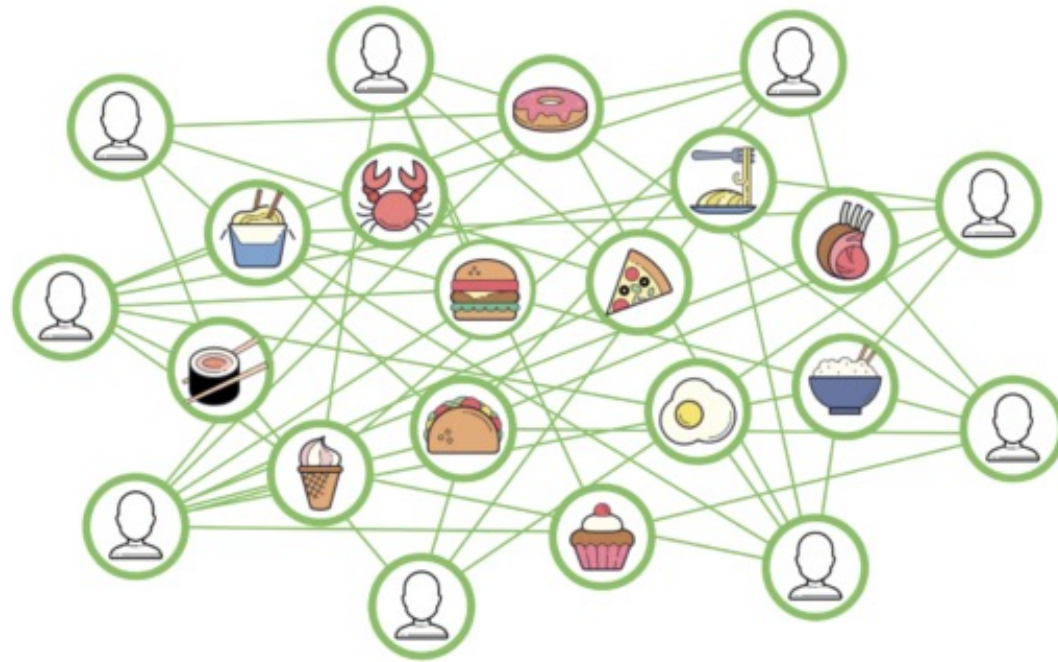
Molecules combine entity attributes with relationships.



- Nodes: atoms. Edges: bonds and bond types.
- Predict an atom, a bond, or a property of the whole molecule.

Recommendation Systems

Uber Eats: dish recommendations from order histories.

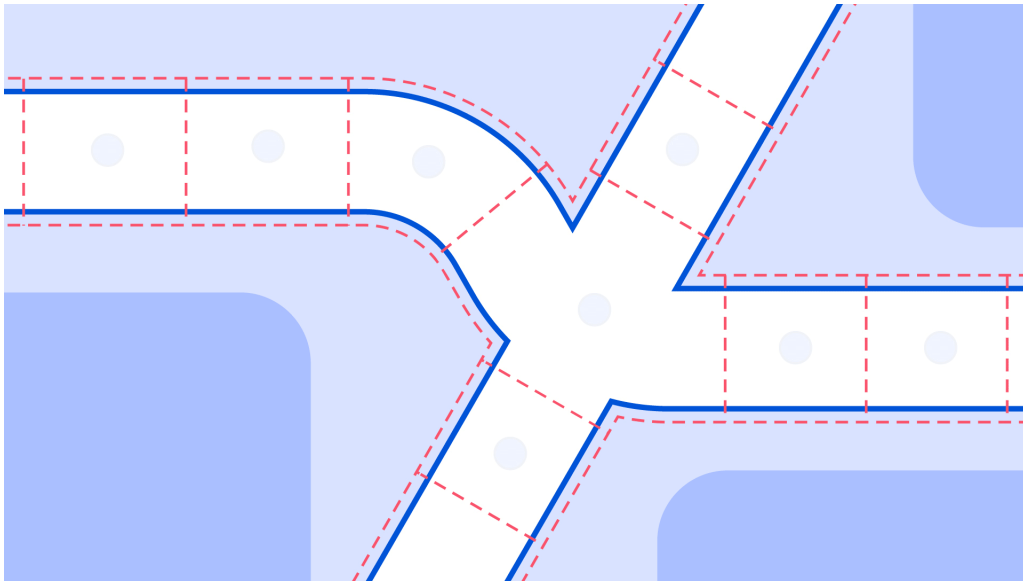


- Nodes: users and dishes. Edge weights: order counts.
- Graph representations help rank dishes for each user.

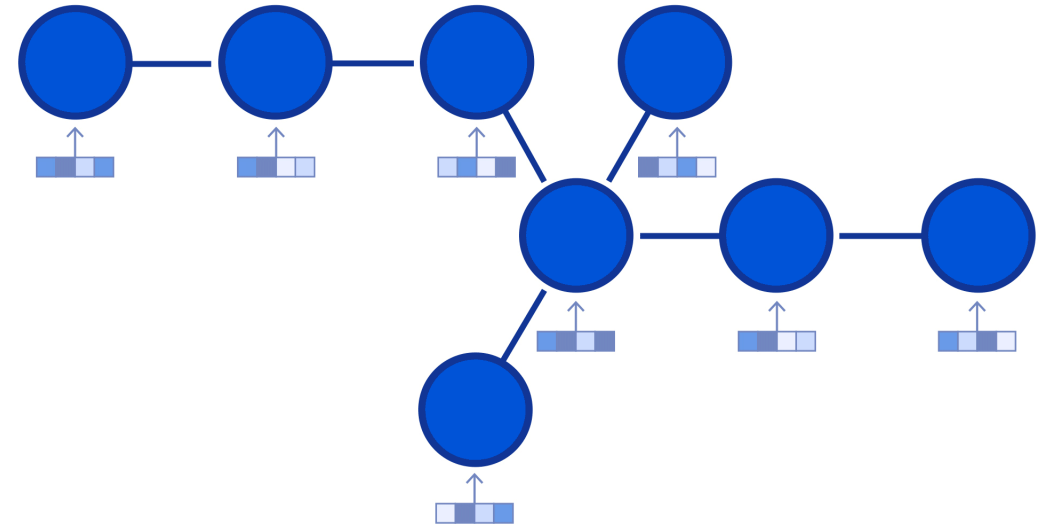
Traffic Prediction

Google Maps: travel-time prediction from live and historical traffic.

Road segments



Graph with traffic features



- Nodes: road segments. Edges: connected road segments.
- Predict travel time for a connected group of road segments.

Related work: Liang et al., Learning Lane Graph Representations for Motion Forecasting (LaneGCN), ECCV 2020.

Source: DeepMind (2020). Figures: Paulo Estriga & Adam Cain.

Sets and Graph Neighborhoods

Lecture 2: aggregate an unordered collection.

$$z = \rho \left(\sum_j \phi(x_j) \right)$$

A graph supplies a different neighborhood for each node.

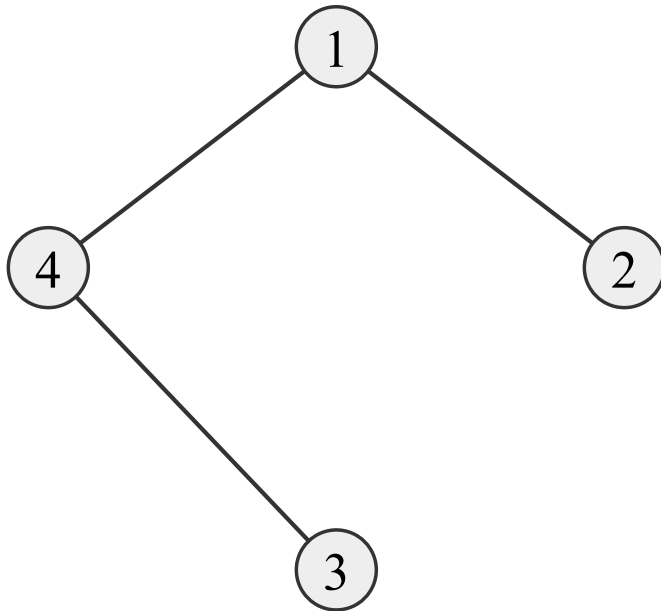
$$h_i^{(\ell+1)} = \psi_\ell \left(h_i^{(\ell)}, \sum_{j \in \mathcal{N}(i)} \phi_\ell(h_i^{(\ell)}, h_j^{(\ell)}, e_{ji}) \right)$$

- Reuse the same functions across all nodes and edges.
- The graph restricts who exchanges information.

Graph Representations

$$G = (V, E), \quad V = \{1, 2, 3, 4\}$$

$$E = \{\{1, 2\}, \{1, 4\}, \{3, 4\}\}$$



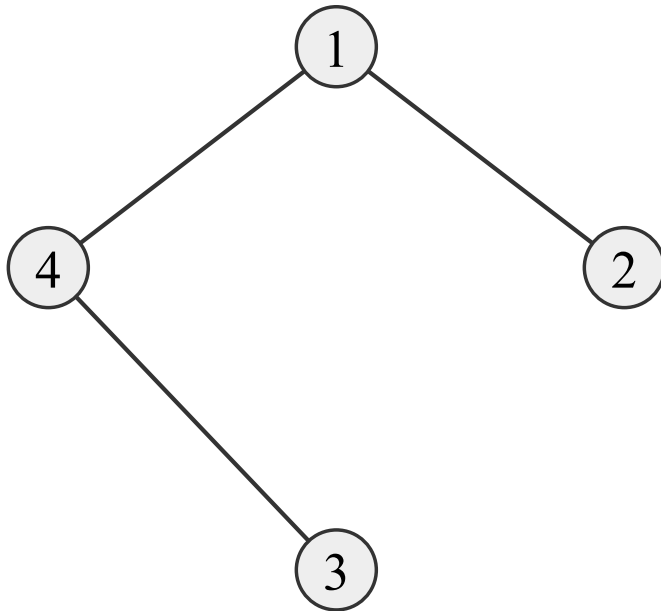
$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

$$X \in \mathbb{R}^{n \times d}, \quad e_{ij} \in \mathbb{R}^{d_e}$$

One feature vector per node, with optional edge attributes.

Relabeling a Graph

Node IDs identify positions in a representation.

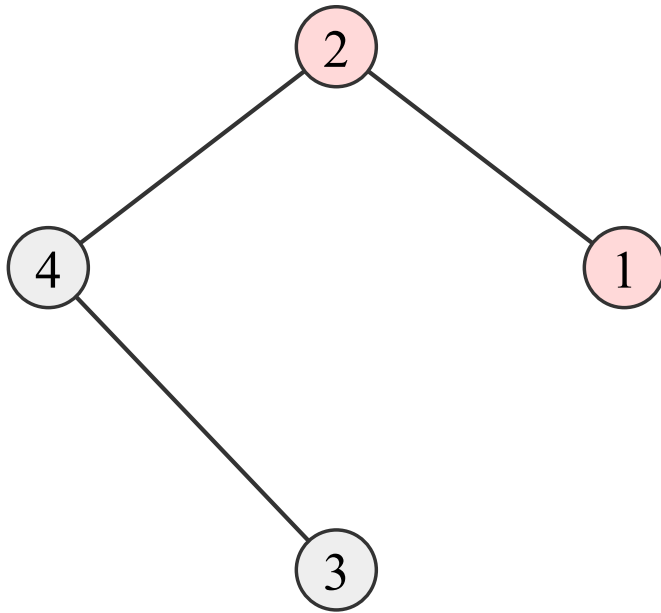


$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Swap labels 1 and 2. Keep the physical graph unchanged.

Relabeling a Graph

Connectivity and features must follow the same relabeling.



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

$$A' = PAP^{\top}, \quad X' = PX$$

Permutation Symmetry

Node predictions follow the nodes.

$$F(PAP^\top, PX) = PF(A, X)$$

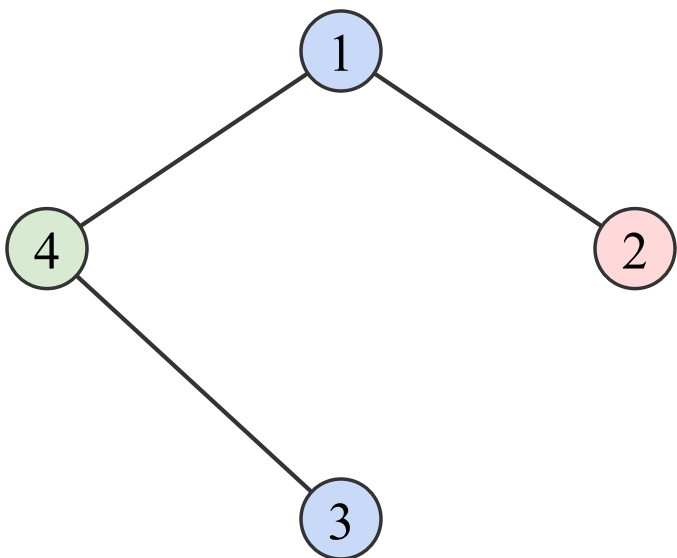
A graph-level prediction is independent of node order.

$$g(PAP^\top, PX) = g(A, X)$$

This is the same invariance / equivariance distinction as Lecture 2.

Node Features

Colors represent node attributes. Numbers are storage IDs.

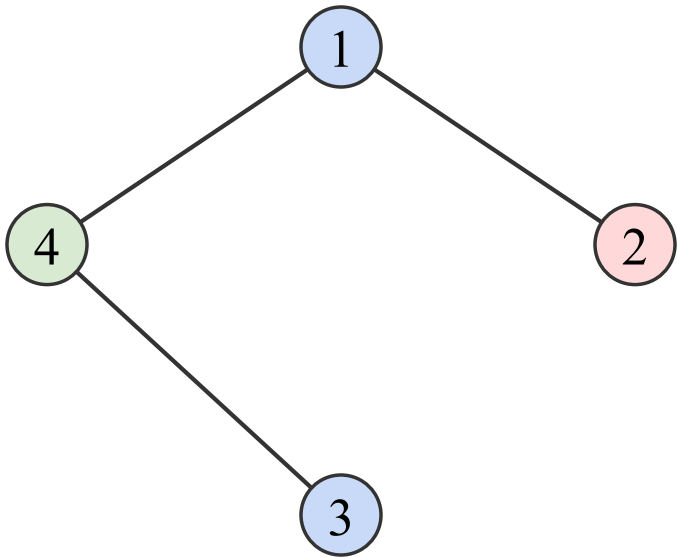


Original labeling

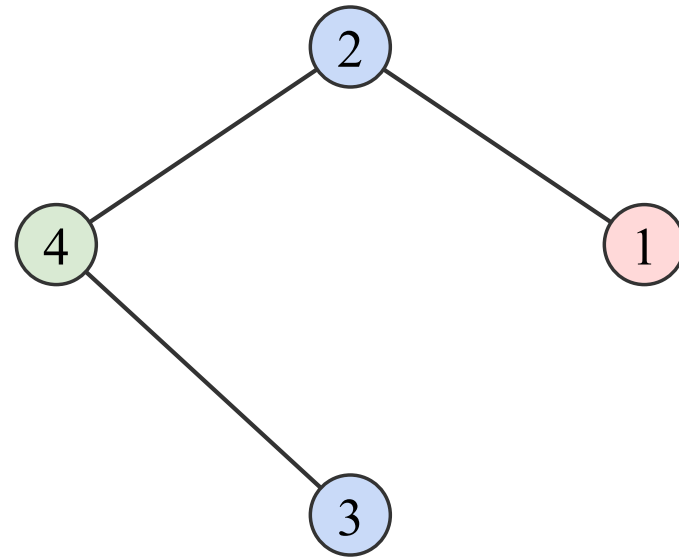
- **Intrinsic attributes**
 - Atom type, text, or measurements
- **When attributes are unavailable**
 - Use constant features or node degrees

Node Features

Relabel the nodes. Preserve their attributes and connections.



Original labeling



IDs 1 and 2 exchanged

- Features follow the nodes. Arbitrary IDs do not describe the graph.

Outline

- Graphs and permutation symmetry
 - Representations, node features, and relabeling
- **Message passing neural networks**
 - Messages, aggregation, updates, and graph prediction
- Graph convolutional networks (GCNs)
 - Graph signals, Laplacian bases, Fourier transforms, and convolution
 - Polynomial and Chebyshev filters
 - Deriving the GCN layer
- Expressivity and modern graph architectures
 - The WL test, depth, attention, and spectral features

Message Passing: A Shared Idea

- **Early GNNs (Scarselli et al., 2009)**
 - Iterate local updates to a fixed point.
- **Neural fingerprints (Duvenaud et al., 2015)**
 - Learn molecular neighborhood representations.
- **Gated graph networks (Li et al., 2016)**
 - Use gated recurrent updates on graph states.
- **Interaction networks (Battaglia et al., 2016)**
 - Model object–relation interactions.
- **MPNN framework (Gilmer et al., 2017)**
 - Unify message, update, and readout functions.

Message passing gives us a common language for these models.

Message Passing: One Layer

- 1. Compute a message on each incoming edge.**
- 2. Aggregate the unordered multiset of messages.**
- 3. Update the node representation.**

Every node uses the same functions at a given layer.

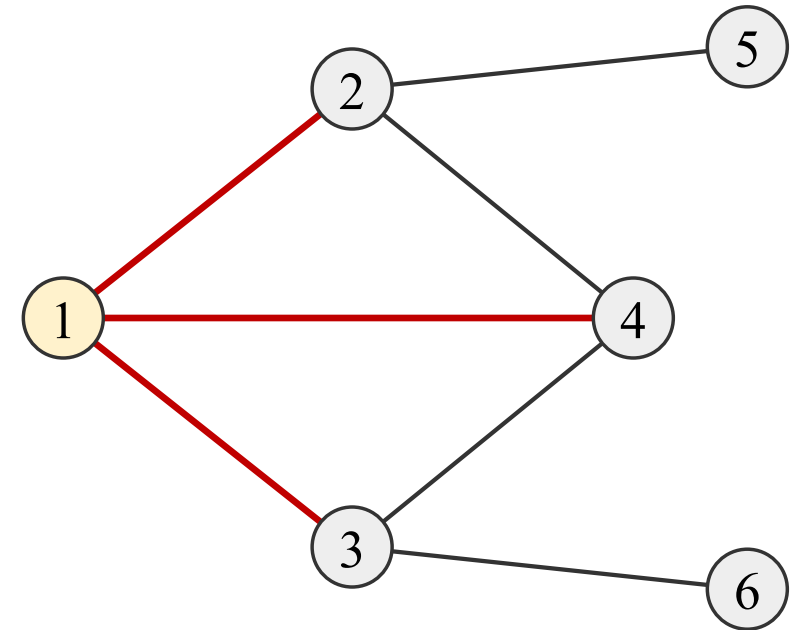
Message Passing: One Layer

1. Message

2. Aggregate

3. Update

$$m_{j \rightarrow i}^{(\ell)} = M_{\ell}(h_i^{(\ell)}, h_j^{(\ell)}, e_{ji})$$



Each neighbor sends a message to the highlighted node.

Message Passing: One Layer

1. Message

2. Aggregate

3. Update

Example: every neighbor sends the scalar 2.

$$m_{j \rightarrow i}^{(\ell)} = M_{\ell}(h_i^{(\ell)}, h_j^{(\ell)}, e_{ji})$$

$$a_i^{(\ell)} = \sum_{j \in \mathcal{N}(i)} m_{j \rightarrow i}^{(\ell)}$$

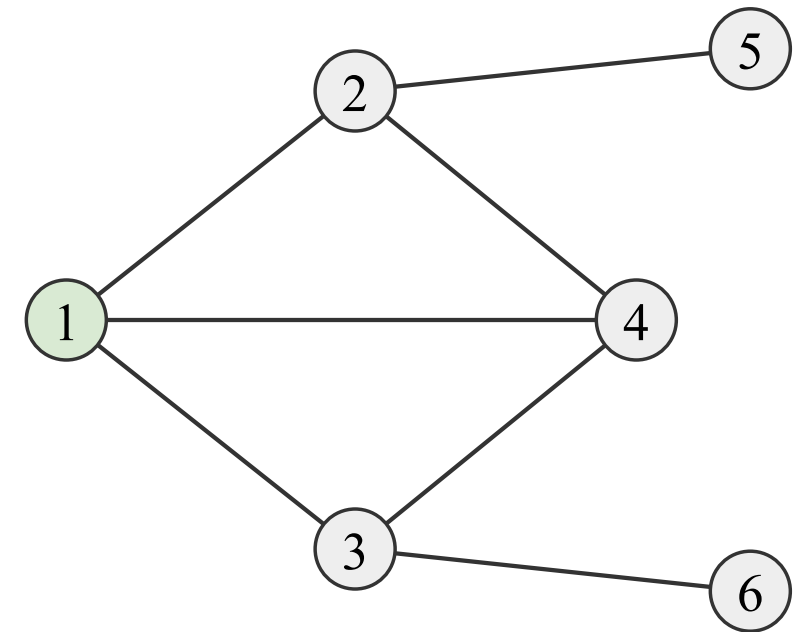
Node 1: neighbors 2, 3, 4
 $\{\{2, 2, 2\}\}$

Node 5: neighbor 2
 $\{\{2\}\}$

Mean: 2 for both. Max: 2 for both.

Sum: 6 at node 1; 2 at node 5.

The choice of aggregator determines which information survives.



Message Passing: One Layer

1. Message

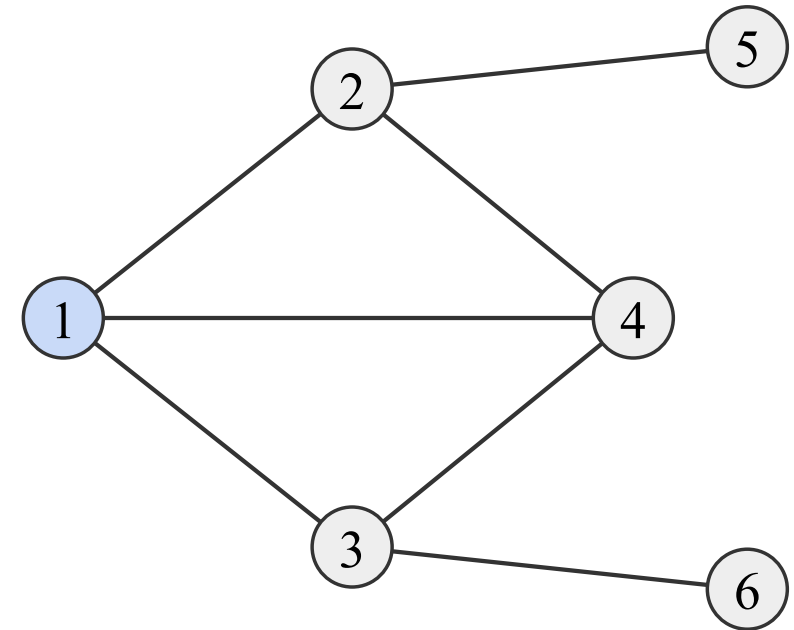
2. Aggregate

3. Update

$$m_{j \rightarrow i}^{(\ell)} = M_{\ell}(h_i^{(\ell)}, h_j^{(\ell)}, e_{ji})$$

$$a_i^{(\ell)} = \sum_{j \in \mathcal{N}(i)} m_{j \rightarrow i}^{(\ell)}$$

$$h_i^{(\ell+1)} = U_{\ell}(h_i^{(\ell)}, a_i^{(\ell)})$$



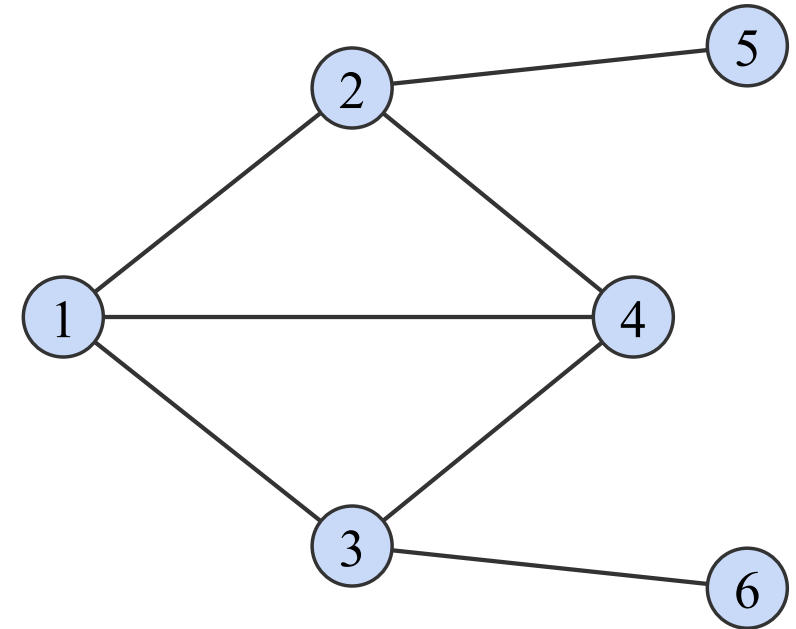
The update combines the old state and aggregated messages.

All Nodes Update in Parallel

Read the old states.

Compute messages on all edges.

Aggregate, then write the new states.

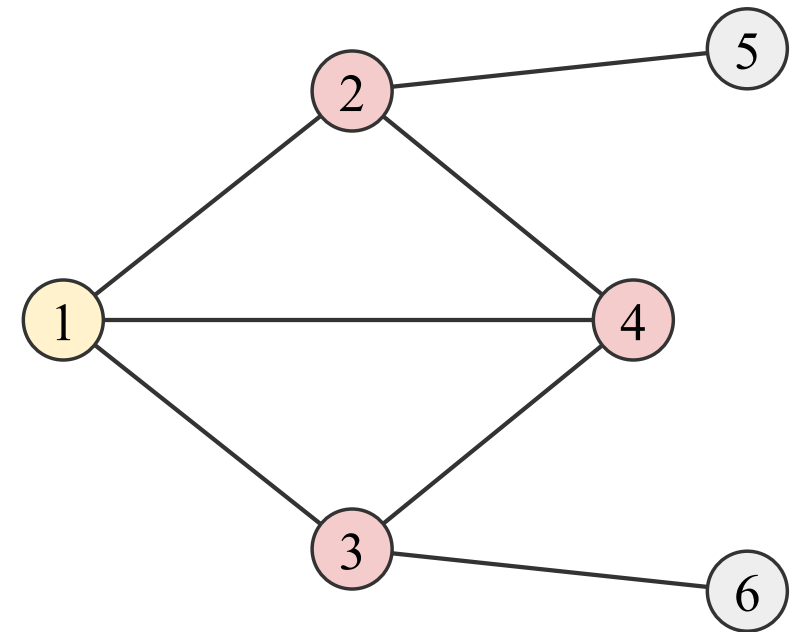


$$H^{(0)} = X, \quad H^{(\ell+1)} = \mathcal{F}_\ell(A, H^{(\ell)})$$

Depth and Receptive Fields

One layer

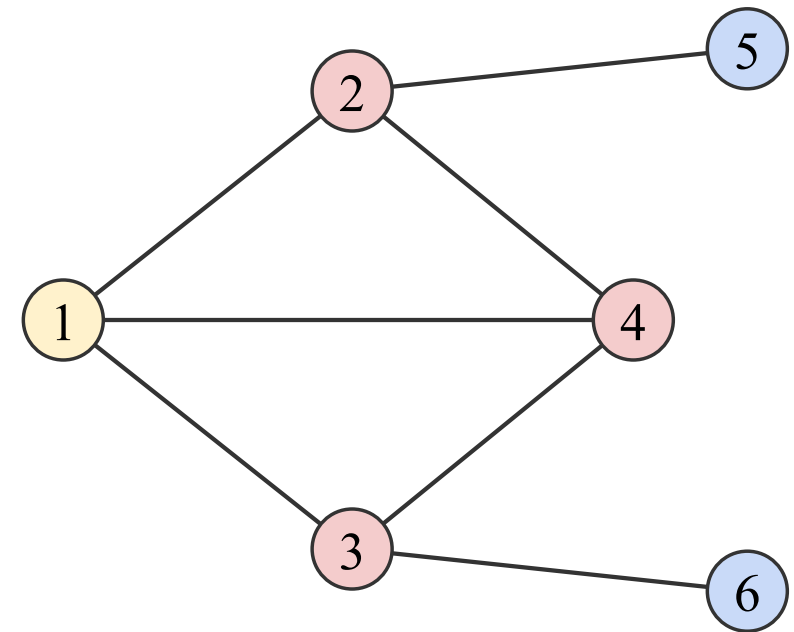
Node 1 receives information from nodes 2, 3, and 4.



Depth and Receptive Fields

Two layers

Node 1 can also depend on nodes 5 and 6.

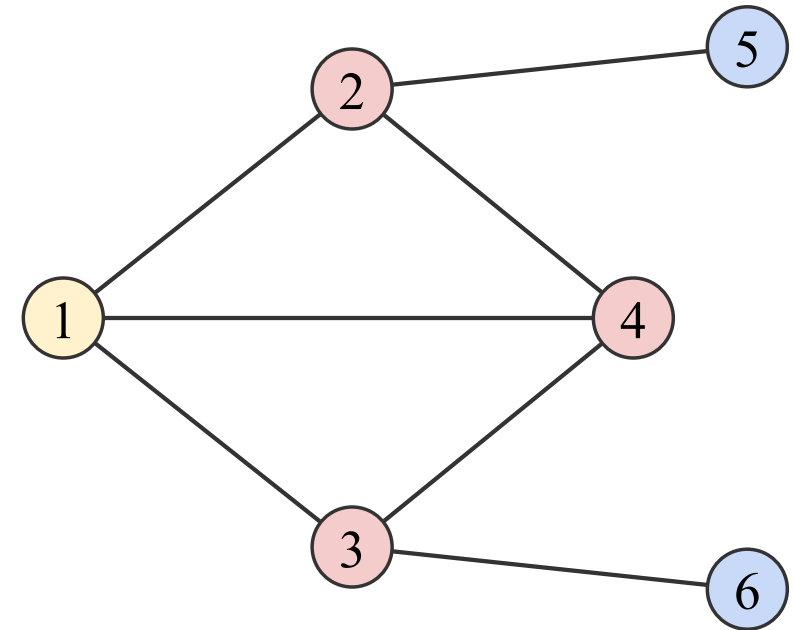


Depth and Receptive Fields

Two layers

Node 1 can also depend on nodes 5 and 6.

More layers expand the receptive field.



More reach does not guarantee better information preservation.

Equivariance of Message Passing

Assume corresponding input states agree after relabeling.

The same shared functions act on corresponding neighborhoods.

$$\mathcal{N}'(\pi(i)) = \{\pi(j) : j \in \mathcal{N}(i)\}$$

$$m_{\pi(j) \rightarrow \pi(i)}^{(\ell)} = m_{j \rightarrow i}^{(\ell)} \implies a_{\pi(i)}^{(\ell)} = a_i^{(\ell)}$$

$$h_{\pi(i)}^{(\ell+1)} = U_\ell(h_i^{(\ell)}, a_i^{(\ell)}) = h_i^{(\ell+1)}$$

Shared functions and order-independent aggregation are the key.

Node, Edge, and Graph Readout

Node prediction $y_i = \rho_{\text{node}}(h_i^{(L)})$

Edge prediction $y_{ij} = \rho_{\text{edge}}([h_i + h_j, h_i \odot h_j, e_{ij}])$

Graph prediction $y_G = \rho\left(\sum_{i \in V} \phi(h_i^{(L)})\right)$

The output level determines which permutation symmetry is required.

Outline

- Graphs and permutation symmetry
 - Representations, node features, and relabeling
- Message passing neural networks
 - Messages, aggregation, updates, and graph prediction
- Graph convolutional networks (GCNs)
 - **Graph signals, Laplacian bases, Fourier transforms, and convolution**
 - Polynomial and Chebyshev filters
 - Deriving the GCN layer
- Expressivity and modern graph architectures
 - The WL test, depth, attention, and spectral features

Fourier Modes and the Laplacian

Complex exponentials form the classical Fourier basis.

$$\phi_\omega(t) = e^{i\omega t}, \quad \mathcal{L} = -\frac{d^2}{dt^2}$$

$$\mathcal{L}\phi_\omega(t) = \omega^2\phi_\omega(t)$$

The Fourier transform gives the coefficients in this basis.

$$\hat{x}(\omega) = \int_{\mathbb{R}} x(t)e^{-i\omega t} dt$$

Fourier modes are eigenfunctions of the Laplacian.

Convolution in the Fourier Domain

Classical convolution

$$y(t) = (h * x)(t) = \int_{\mathbb{R}} h(\tau)x(t - \tau) d\tau$$

The convolution theorem

$$\hat{y}(\omega) = \hat{h}(\omega)\hat{x}(\omega), \quad y = \mathcal{F}^{-1}\left(\hat{h}\hat{x}\right)$$

Transform the signal, scale its Fourier coefficients, then reconstruct.

Can the graph Laplacian supply an analogous basis?

The Graph Laplacian

$$L_c = D - A$$



$$D = \text{diag}(1, 2, 1)$$

$$L_c = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

Diagonal entries record degree. Off-diagonals encode edges.

Graph Signals and Patterns

A graph signal assigns one scalar value to each node.

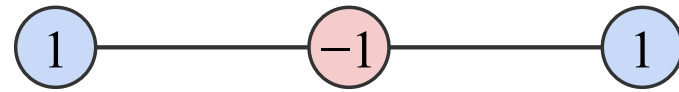
$$x = [x_1, \dots, x_N]^\top \in \mathbb{R}^N$$

Constant signal



$$x = [1, 1, 1]^\top$$

Alternating signal



$$x = [1, -1, 1]^\top$$

The same graph can carry different patterns of node values.

At the center, the Laplacian compares the value to both neighbors.

$$(L_c x)_2 = 2x_2 - x_1 - x_3$$

Smoothness of a Graph Signal

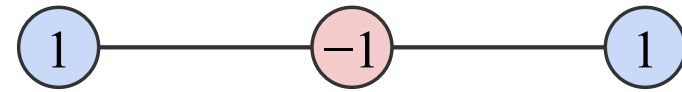
$$x^\top L_c x = \frac{1}{2} \sum_{i,j} A_{ij} (x_i - x_j)^2 \geq 0$$

Constant signal



Energy: 0

Alternating signal



Energy: 8

$$x^\top L_c x = (x_1 - x_2)^2 + (x_2 - x_3)^2$$

Larger differences across edges give larger energy.

Graph Fourier Modes: Definition

A mode is a basic pattern of values over the nodes.

1. Modes are eigenvectors of the graph Laplacian

$$L_c u_k = \lambda_k u_k, \quad \|u_k\|_2 = 1$$

Applying the Laplacian preserves this pattern and scales it.

2. Its eigenvalue measures variation across edges

$$\lambda_k = u_k^\top L_c u_k = \frac{1}{2} \sum_{i,j} A_{ij} (u_k(i) - u_k(j))^2$$

Small eigenvalue: less variation across edges.

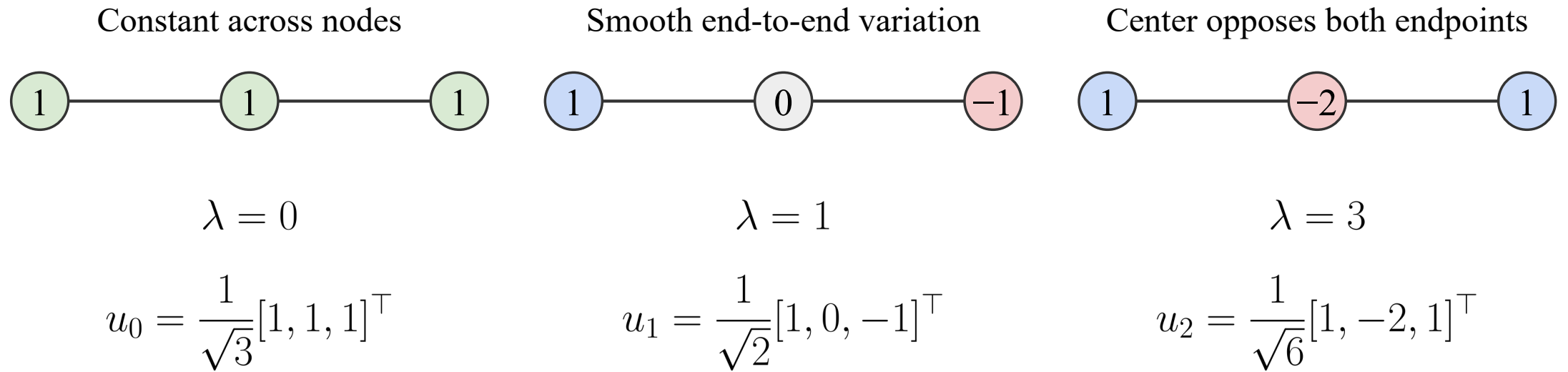
Large eigenvalue: more variation across edges.

Graph modes play the role of sinusoids in ordinary Fourier analysis.

Undirected graph, nonnegative edge weights. Shuman et al. (2013).

Graph Fourier Modes: Example

Each column shows one eigenvector of the path Laplacian.



Node labels show unnormalized pattern values.

The three independent modes form a complete basis for this path.

Next: why completeness holds for every undirected graph.

A Basis for Every Graph Signal

1. The signal can be arbitrary

Assign one value to every node.

$$x \in \mathbb{R}^N$$

2. The graph determines a complete basis

An undirected graph has a real symmetric Laplacian.

The spectral theorem provides a complete orthonormal eigenbasis.

$$U = [u_0, \dots, u_{N-1}], \quad U^\top U = U U^\top = I$$

A Basis for Every Graph Signal

1. The signal can be arbitrary

Assign one value to every node.

$$x \in \mathbb{R}^N$$

2. The graph determines a complete basis

An undirected graph has a real symmetric Laplacian.

The spectral theorem provides a complete orthonormal eigenbasis.

$$U = [u_0, \dots, u_{N-1}], \quad U^\top U = U U^\top = I$$

3. Change coordinates and reconstruct

$$x = U U^\top x = \sum_{k=0}^{N-1} \hat{x}_k u_k, \quad \hat{x}_k = u_k^\top x$$

All modes are used. No smoothness assumption is needed.

Graph Fourier Transform

Represent the same signal using the complete Laplacian basis.

1. Forward transform: measure each mode's coefficient

$$\hat{x} = U^\top x, \quad \hat{x}_k = u_k^\top x$$

2. Inverse transform: recover the node values

$$x = U\hat{x} = \sum_{k=0}^{N-1} \hat{x}_k u_k$$

For the three-node path:

$$x = [2, 1, 0]^\top \quad \longleftrightarrow \quad \hat{x} = [\sqrt{3}, \sqrt{2}, 0]^\top$$

All coefficients are retained, so the transform preserves all information.

Why Use the Laplacian Basis?

The graph gives the coordinates a meaning: variation across edges.

$$x^\top L_c x = \frac{1}{2} \sum_{i,j} A_{ij} (x_i - x_j)^2 = \sum_k \lambda_k \hat{x}_k^2$$

1. The eigenvalues weight variation

For equal coefficient sizes, higher-frequency modes contribute more.

2. The coordinates depend on the graph

Changing the graph can change the modes and coefficients.

Every signal still has an expansion, including random noise.

Nonnegative edge weights give a nonnegative variation energy.

Keeping Only Some Modes

Keeping fewer modes gives an approximation in general.

$$x_K = \sum_{k=0}^{K-1} \hat{x}_k u_k, \quad 1 \leq K < N$$

1. Exact reconstruction

Requires zero coefficients on all omitted modes (a bandlimited signal).

$$\|x - x_K\|_2^2 = \sum_{k=K}^{N-1} \hat{x}_k^2$$

2. Accurate approximation

Requires little energy in the omitted modes.

Smoothness can support an approximation using low-frequency modes.

When Degree Should Matter

Each center is one degree warmer than its neighbors.

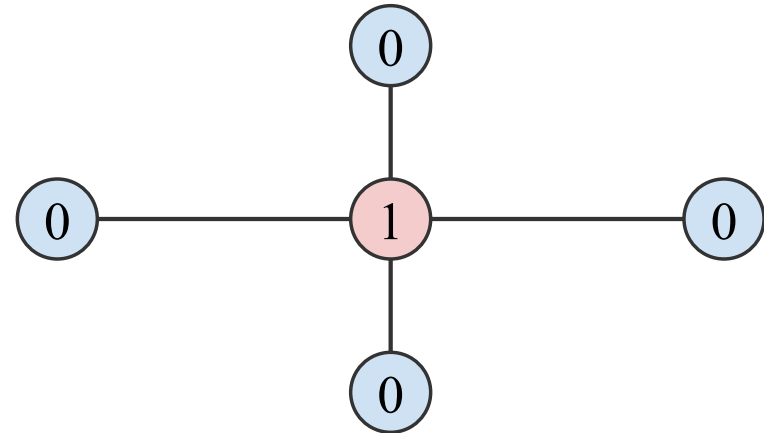
$$(L_c x)_i = \sum_j A_{ij}(x_i - x_j)$$

One connection



$$d_i = 1, \quad (L_c x)_i = 1$$

Four connections



$$d_i = 4, \quad (L_c x)_i = 4$$

More connections give greater total heat flow.

Illustrative heat-flow model with unit conductance on every edge.

When Average Mismatch Matters

Compare each sensor with the average of its neighbors.

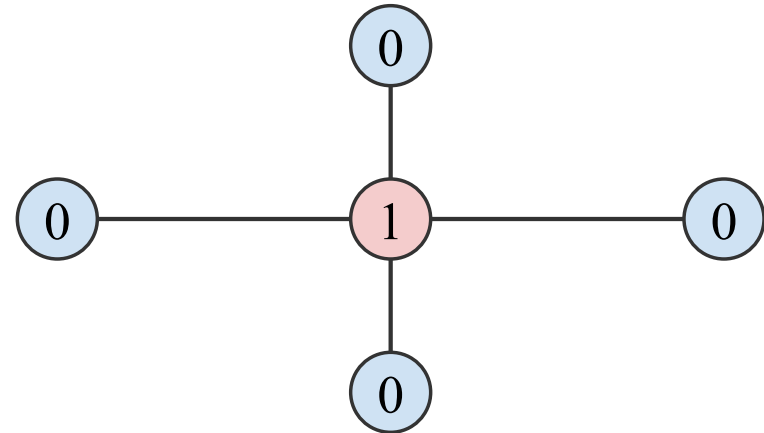
$$d_i^{-1}(L_c x)_i = x_i - d_i^{-1} \sum_j A_{ij} x_j$$

One connection



$$d_i = 1, \quad d_i^{-1}(L_c x)_i = 1$$

Four connections



$$d_i = 4, \quad d_i^{-1}(L_c x)_i = 1$$

Use the average when neighbor count should not set the score.

Symmetric scaling uses a different operator and keeps an orthonormal basis.

Symmetric Degree Normalization

Scale both endpoints while preserving symmetry.

$$R = D^{-1/2}AD^{-1/2}, \quad L = D^{-1/2}L_cD^{-1/2} = I - R$$

A degree-adjusted notion of smoothness

$$x^\top Lx = \frac{1}{2} \sum_{i,j} A_{ij} \left(\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}} \right)^2$$

Compare values after dividing by the square root of node degree.

Use the eigenbasis of this operator from here onward.

$$L = U\Lambda U^\top, \quad U^\top U = I, \quad 0 \leq \lambda_k \leq 2$$

This basis generally differs from the combinatorial Laplacian basis.

Graph Convolution

Generalize the convolution theorem by multiplying spectral coefficients.

1. Apply a response to each graph frequency

$$\hat{y}_k = g_\theta(\lambda_k)\hat{x}_k, \quad \hat{y} = g_\theta(\Lambda)\hat{x}$$

2. Transform back to node values

$$y = U\hat{y} = Ug_\theta(\Lambda)U^\top x = g_\theta(L)x$$

A low-pass response reduces rapidly varying components.

Other responses can preserve or emphasize high frequencies.

Learning the response gives a spectral graph convolutional filter.

Outline

- Graphs and permutation symmetry
 - Representations, node features, and relabeling
- Message passing neural networks
 - Messages, aggregation, updates, and graph prediction
- Graph convolutional networks (GCNs)
 - Graph signals, Laplacian bases, Fourier transforms, and convolution
 - **Polynomial and Chebyshev filters**
 - Deriving the GCN layer
- Expressivity and modern graph architectures
 - The WL test, depth, attention, and spectral features

Spectral Filters and Computation

An arbitrary spectral response can mix distant nodes.

$$y = U g_{\theta}(\Lambda) U^{\top} x$$

- A full dense eigenbasis is costly to construct and apply.
- Can we evaluate useful filters through local propagation?
- Choose a polynomial response.

Polynomial Filters and Locality

$$g_{\theta}(\lambda) = \sum_{k=0}^K \theta_k \lambda^k$$

$$U g_{\theta}(\Lambda) U^{\top} x = \sum_{k=0}^K \theta_k L^k x$$

$$\text{dist}(i, j) > k \implies (L^k)_{ij} = 0$$

- Polynomial degree bounds the number of propagation hops.
- This connects the spectral construction directly to message passing.

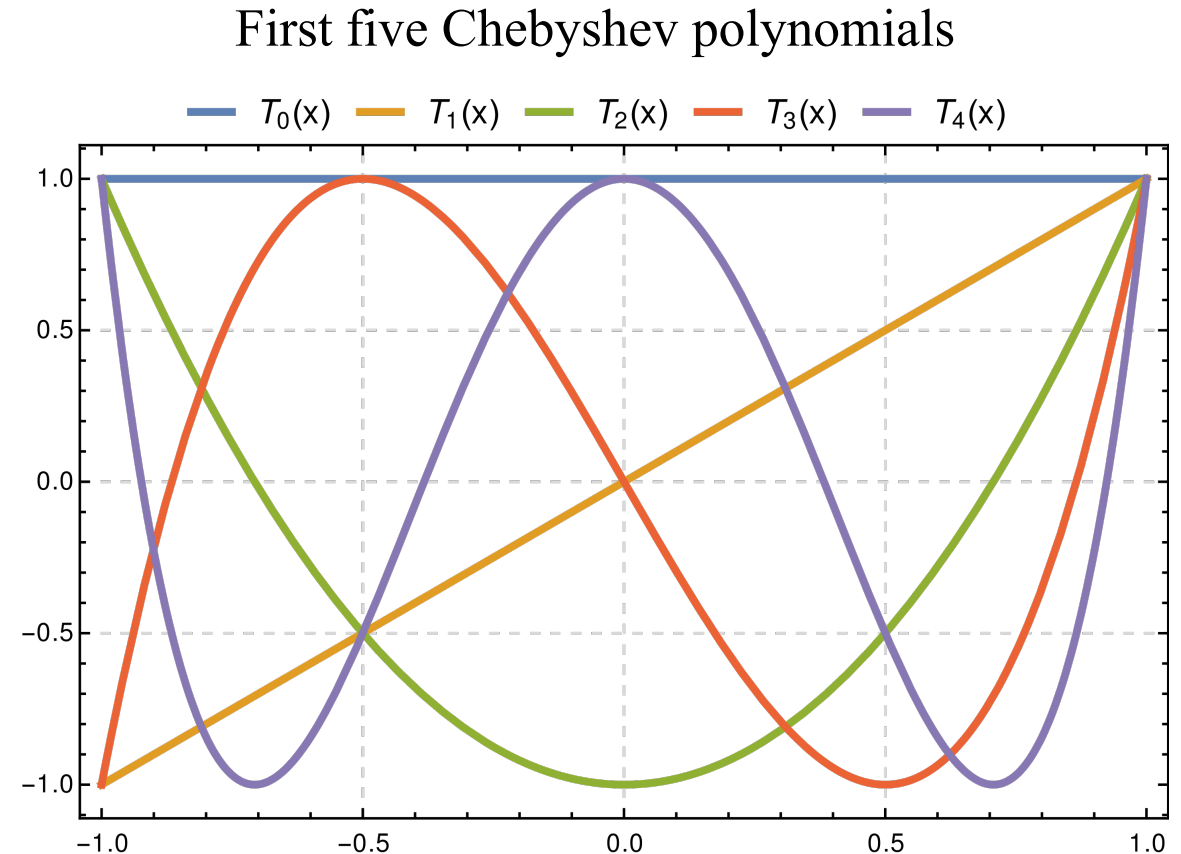
Chebyshev Polynomial Filters

$$\bar{L} = \frac{2}{\lambda_{\max}} L - I$$

$$T_0(t) = 1, \quad T_1(t) = t$$

$$T_k(t) = 2tT_{k-1}(t) - T_{k-2}(t)$$

$$g_{\theta}(L)x = \sum_{k=0}^K \theta_k T_k(\bar{L})x$$



The recurrence evaluates the polynomial without an eigenbasis.

Efficient Filtering by Recurrence

$$Z_0 = X, \quad Z_1 = \bar{L}X$$

$$Z_k = 2\bar{L}Z_{k-1} - Z_{k-2}$$

$$H' = \sigma \left(\sum_{k=0}^K Z_k W_k \right)$$

- Each step uses a sparse graph multiplication.
- Learn channel weights while avoiding the full eigendecomposition.

Outline

- Graphs and permutation symmetry
 - Representations, node features, and relabeling
- Message passing neural networks
 - Messages, aggregation, updates, and graph prediction
- Graph convolutional networks (GCNs)
 - Graph signals, Laplacian bases, Fourier transforms, and convolution
 - Polynomial and Chebyshev filters
 - **Deriving the GCN layer**
- Expressivity and modern graph architectures
 - The WL test, depth, attention, and spectral features

GCN: A First-Order Spectral Filter

1. Keep only degree 0 and degree 1 terms.

$$y = \theta_0 x + \theta_1 \bar{L}x$$

2. Approximate the largest eigenvalue by 2.

$$\bar{L} \approx L - I = -R, \quad y \approx \theta_0 x - \theta_1 R x$$

3. Tie the self and neighbor coefficients.

$$\theta_0 = -\theta_1 = \theta \implies y \approx \theta(I + R)x$$

First-order truncation and coefficient tying restrict the filter family.

GCN: Self-Loops and Renormalization

Add self-loops, then recompute degrees before normalizing.

$$\hat{A} = A + I, \quad \hat{d}_i = \sum_j \hat{A}_{ij}, \quad \hat{D} = \text{diag}(\hat{d}_i)$$

$$I + R \quad \longrightarrow \quad \hat{S} = \hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2}$$

This replaces the operator; the two matrices are generally different.

$$H^{(\ell+1)} = \sigma\left(\hat{S} H^{(\ell)} W_\ell\right)$$

This gives the GCN layer. Next, interpret its node update.

GCN as Message Passing

The derived GCN layer is a message-passing layer:

shared linear messages, a normalized sum, and an activation.

Add self-loops and compute the augmented degrees.

$$\hat{A} = A + I, \quad \hat{d}_i = \sum_j \hat{A}_{ij}$$

$$h_i^{(\ell+1)} = \sigma \left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{\hat{A}_{ij}}{\sqrt{\hat{d}_i \hat{d}_j}} h_j^{(\ell)} W_\ell \right)$$

Learn the feature transform. The graph determines the aggregation weights.

The Same Layer in Matrix Form

$$\hat{D} = \text{diag}(\hat{d}_1, \dots, \hat{d}_n)$$

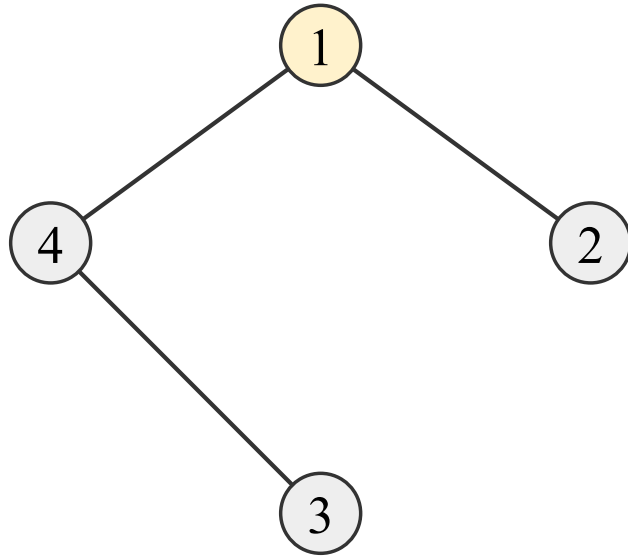
$$\hat{S} = \hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2}$$

$$H^{(\ell+1)} = \sigma(\hat{S} H^{(\ell)} W_\ell)$$

- Propagation mixes nodes. The shared weight matrix mixes channels.
- Node states are row vectors; the matrix stacks one state per row.

A Small GCN Calculation

One scalar channel, identity activation, unit feature weight.



$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \quad \hat{d} = \begin{bmatrix} 3 \\ 2 \\ 2 \\ 3 \end{bmatrix}$$

$$(\hat{S}x)_1 = \frac{1}{3} + \frac{2}{\sqrt{6}} + \frac{4}{3}$$

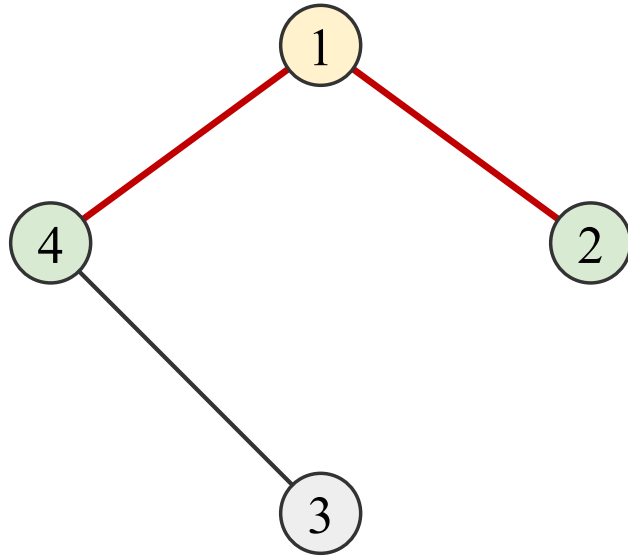
$$(\hat{S}x)_1 \approx 2.483$$

Original edges; self-loops are implicit.

Symmetric normalization is not the same as averaging neighbors.

Computing the GCN Update

Evaluate the same matrix rule using edges and self-loops.



Incoming connections to node 1

Self-loop, node 2, and node 4

1. Transform node features once.

$$Z = H^{(\ell)} W_{\ell}$$

2. Sum over neighbors and self-loops.

$$h_1^{(\ell+1)} = \sigma \left(\frac{z_1}{3} + \frac{z_2}{\sqrt{6}} + \frac{z_4}{3} \right)$$

3. Skip the missing connection.

$$\hat{S}_{13} = 0$$

Store and process only nonzero graph connections.

One GCN Layer, Two Views

$$H^{(\ell+1)} = \sigma(\hat{S}H^{(\ell)}W_\ell)$$

Message-passing view

Shared messages with fixed degree-normalized aggregation.

Spectral view

$$\hat{S} = I - \hat{L}$$

A first-degree graph filter before channel mixing and activation.

General nonlinear MPNNs need not be fixed spectral filters.

Checkpoint

A GCN is a normalized message-passing layer.

1. Which parts are learned?
2. Which parts are determined by the graph?
3. How did spectral filtering lead to this update?

Outline

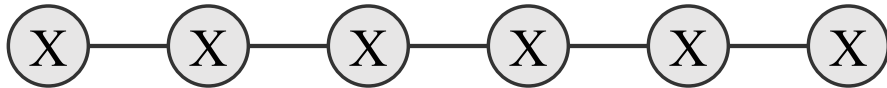
- Graphs and permutation symmetry
 - Representations, node features, and relabeling
- Message passing neural networks
 - Messages, aggregation, updates, and graph prediction
- Graph convolutional networks (GCNs)
 - Graph signals, Laplacian bases, Fourier transforms, and convolution
 - Polynomial and Chebyshev filters
 - Deriving the GCN layer
- **Expressivity and modern graph architectures**
 - The WL test, depth, attention, and spectral features

The Weisfeiler–Lehman Test

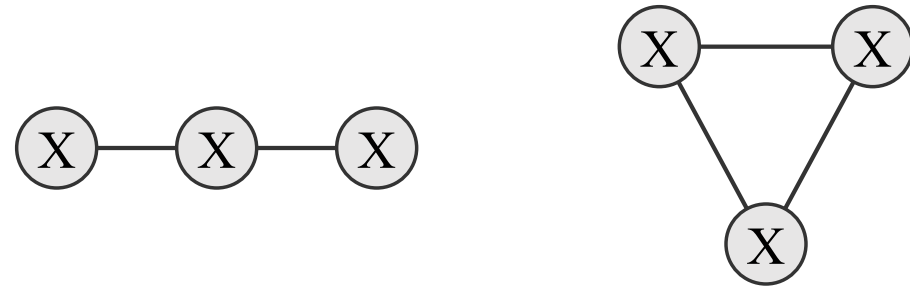
Color hashing assigns a label to each distinct local signature.

Initially, all nodes have the same label and the same color X.

Six-node path



Three-node path + triangle



A signature records the node's color and its neighbors' color multiset.

Same signature: same new color across both graphs.

Different signatures: different new colors.

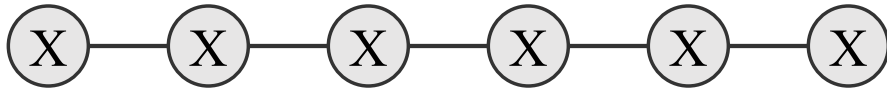
Update all nodes using only the previous round's colors.

The Weisfeiler–Lehman Test

Round 1: look up the old colors before changing any node.

One shared lookup table applies to both graphs.

Six-node path



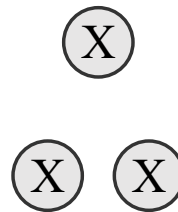
Three-node path + triangle



Own old color



Old neighbor colors (multiset)



New color



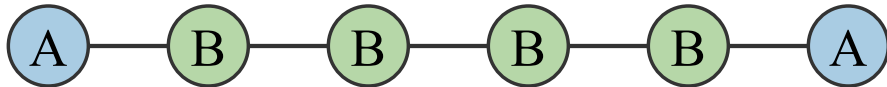
A multiset ignores order and keeps repeated colors.

The Weisfeiler–Lehman Test

Round 1: apply the shared lookup to every node together.

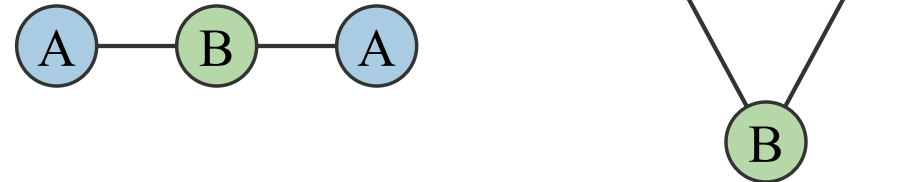
Every new color comes from the previous round's colors.

Six-node path



A: one X neighbor

Three-node path + triangle



B: two X neighbors

Both graphs: two A nodes and four B nodes.

The color counts match. Continue refining.

The Weisfeiler–Lehman Test

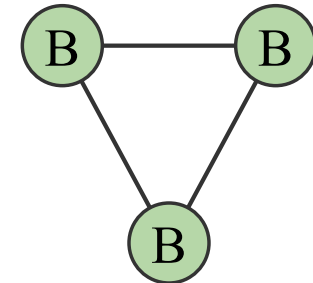
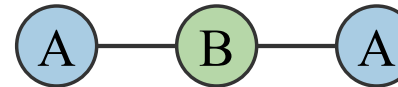
Round 2: use the old A/B colors to form new signatures.

The graphs below still show Round 1. Read these colors only.

Six-node path



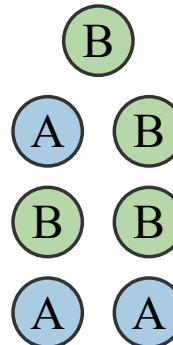
Three-node path + triangle



Own old color



Old neighbor colors (multiset)



New color

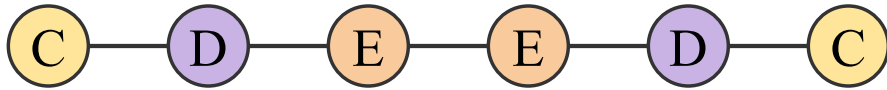


The Weisfeiler–Lehman Test

Round 2: apply the shared lookup to every node together.

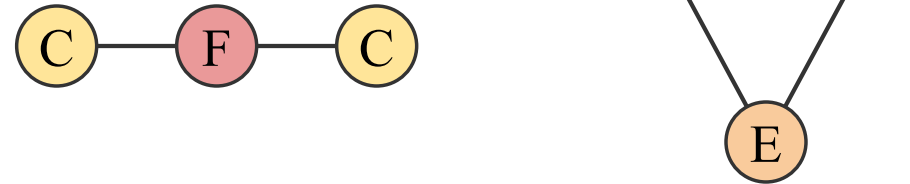
The graph positions and edges are unchanged. Only colors update.

Six-node path



Two C, two D, two E

Three-node path + triangle



Two C, one F, three E

The color counts differ after two rounds.

Therefore, the graphs are non-isomorphic.

What 1-WL Tells Us

Refinement summarizes increasingly large local neighborhoods.

$$c_v^{(k+1)} = \text{HASH}\left(c_v^{(k)}, \{\{c_u^{(k)} : u \in \mathcal{N}(v)\}\}\right)$$

Different color counts prove non-isomorphism.

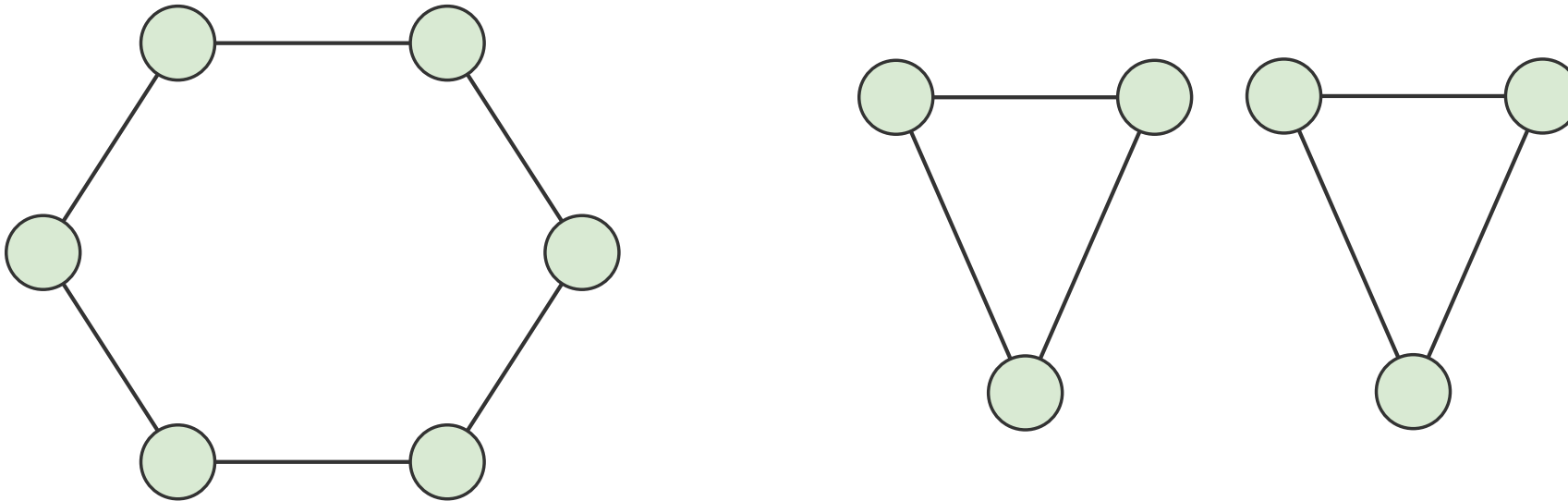
Matching counts at every round leave the test inconclusive.

Connection to message passing

- Local MPNNs cannot distinguish more graphs than 1-WL.
- Injective aggregation, updates, and readout can match 1-WL.
- GIN gives a sum-based construction under these conditions.

A Limitation of 1-WL and Message Passing

Start with identical features at every node.



Every node sees two identical neighbors at every layer.

Neither 1-WL nor standard local message passing separates these graphs.

Depth: Oversmoothing

Repeated propagation can erase distinctions between nodes.

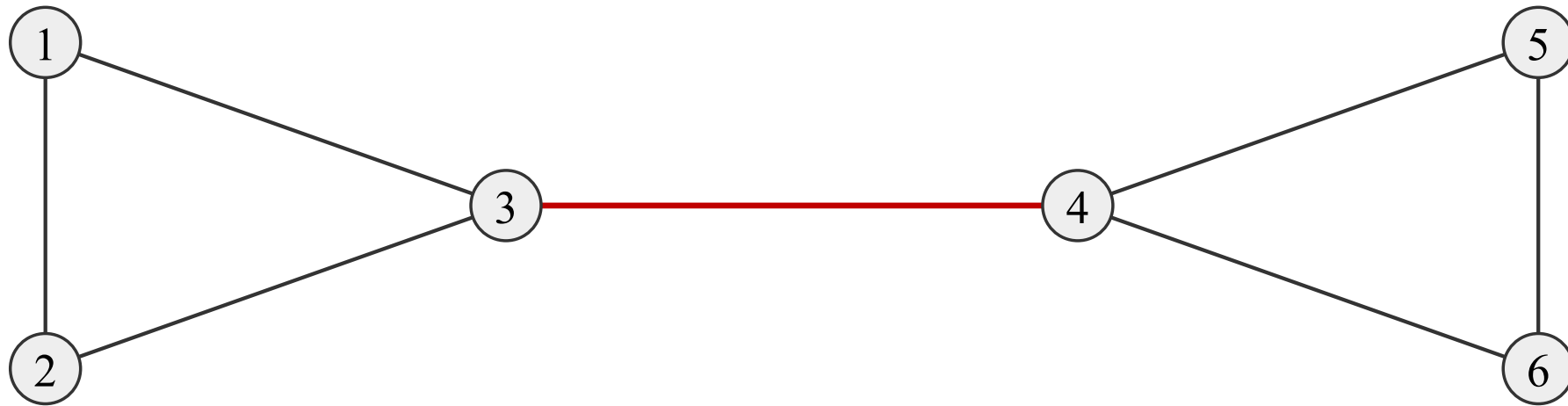
$$H^{(k)} = \hat{S}^k X \longrightarrow qq^\top X$$

$$q_i = \frac{\sqrt{\hat{d}_i}}{\sqrt{\sum_j \hat{d}_j}}$$

- Linear case: connected undirected graph, nonnegative weights, positive self-loops.
- Node states need not become identical. Residual connections change the limit.

Depth: Oversquashing

- Many distant signals compete for limited communication capacity.
- More layers alone need not resolve this bottleneck.
- Global edges or attention can help, at a computational and statistical cost.



Graph Attention

Replace fixed degree weights with learned, input-dependent weights.

$$s_{ij} = a_{\theta}(h_i W, h_j W)$$

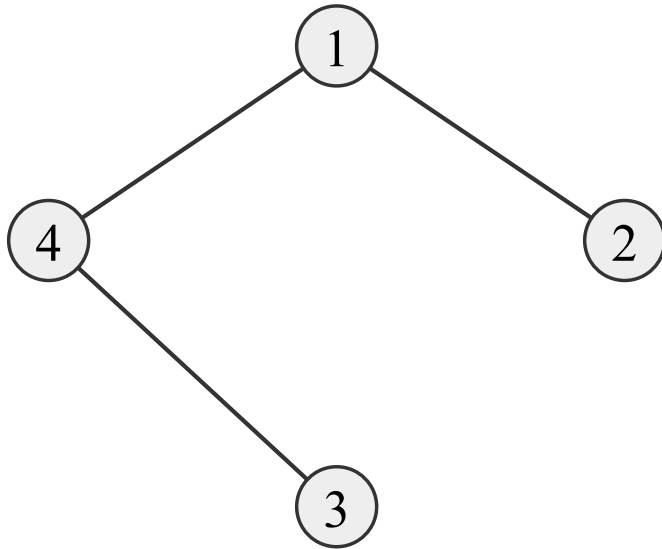
$$\alpha_{ij} = \frac{\exp(s_{ij})}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(s_{ik})}$$

$$h'_i = \sigma \left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij} h_j W \right)$$

- The original GAT uses an additive scorer; dot-product attention is another choice.

Graph Structure in Attention

$$\alpha_{ij} = \text{softmax}_j \left(\frac{q_i k_j^\top}{\sqrt{d}} + b_{ij} + M_{ij} \right)$$



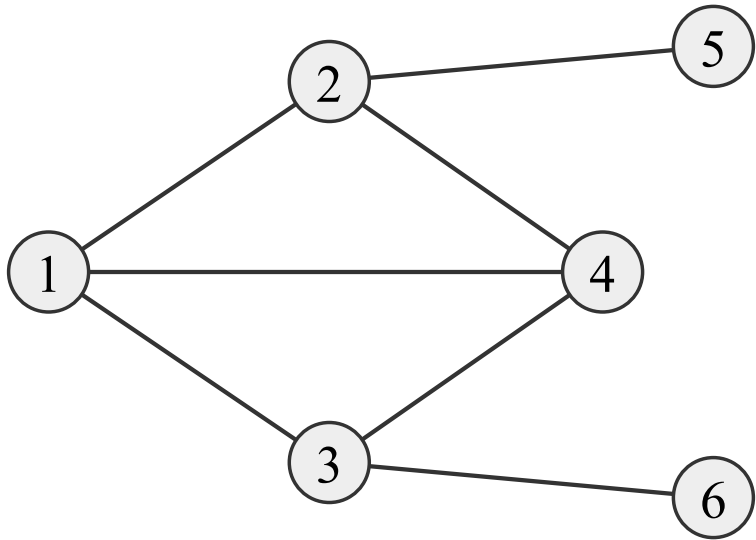
$$M = \begin{bmatrix} 0 & 0 & -\infty & 0 \\ 0 & 0 & -\infty & -\infty \\ -\infty & -\infty & 0 & 0 \\ 0 & -\infty & 0 & 0 \end{bmatrix}$$

Local mask shown above. Global attention instead allows every pair.

Supply connectivity through a mask, structural bias, or structural features.

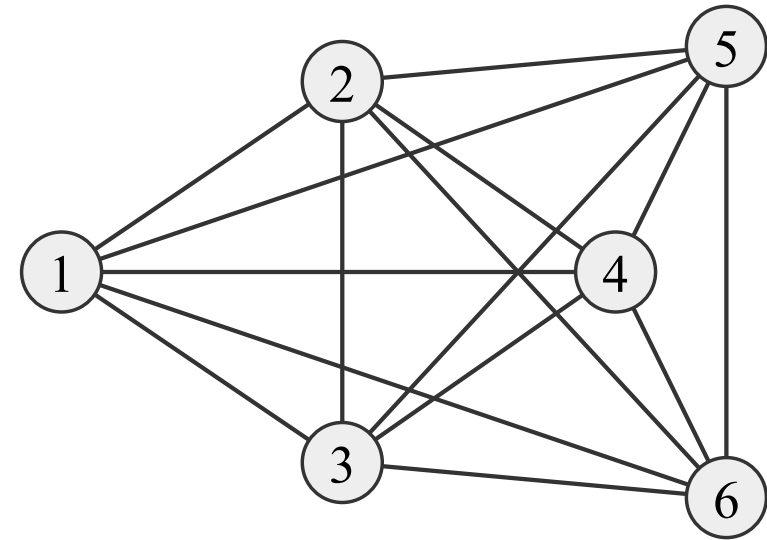
Local and Global Computation

Local message passing



Observed neighbors

Global attention

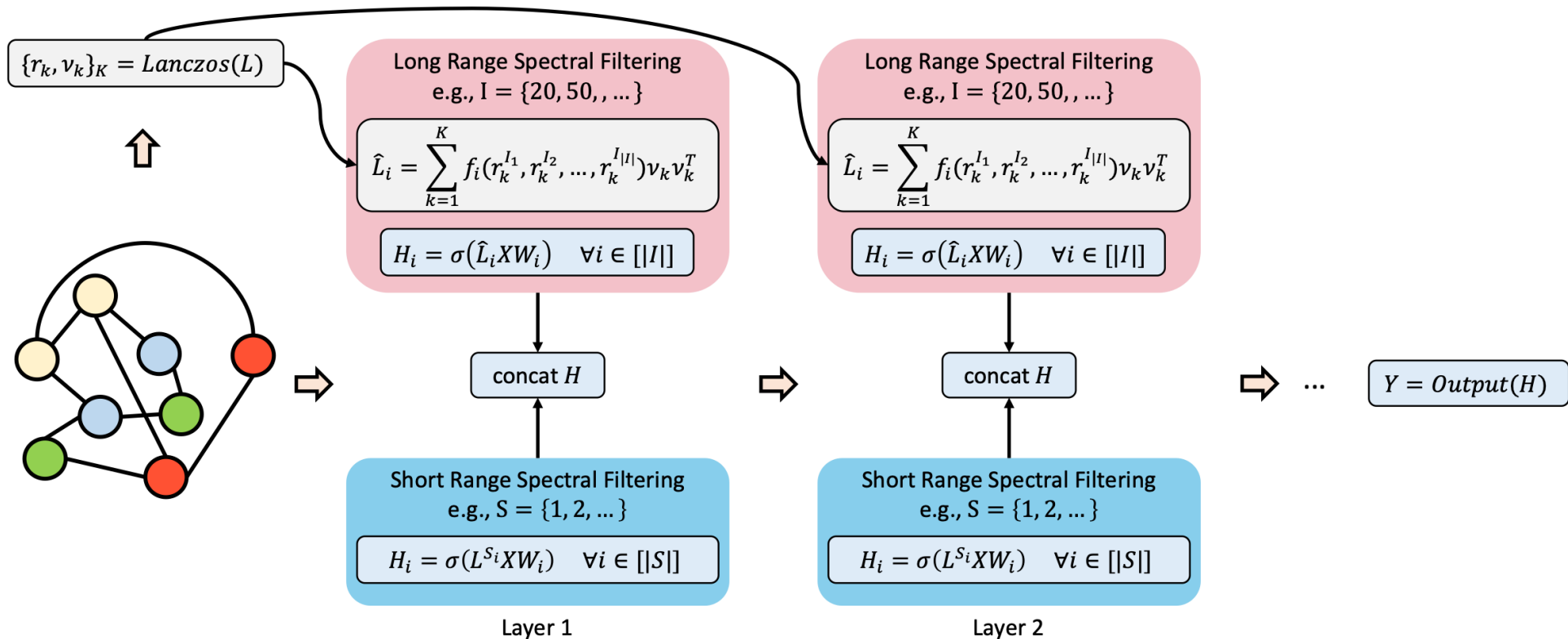


All node pairs

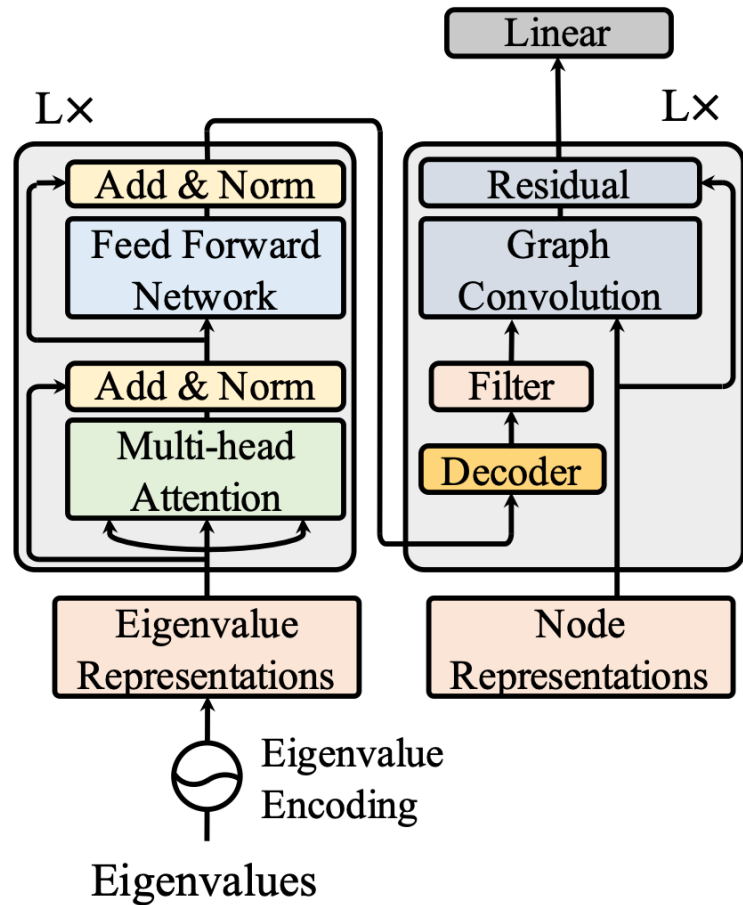
- GraphGPS combines both paths with structural encodings.

LanczosNet: Multiple Diffusion Scales

1. Short scales: propagate directly to nearby nodes.
2. Long scales: learn filters using a compact spectral approximation.



Specformer: Learning Spectral Responses



1. Encode each eigenvalue.
2. Attend across the spectrum.
3. Decode a response for each mode.

$$c = F_{\theta}(\lambda_1, \dots, \lambda_N)$$

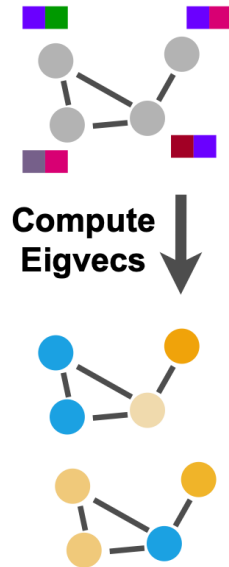
$$y = U \operatorname{diag}(c) U^{\top} x$$

Responses depend on the whole spectrum.

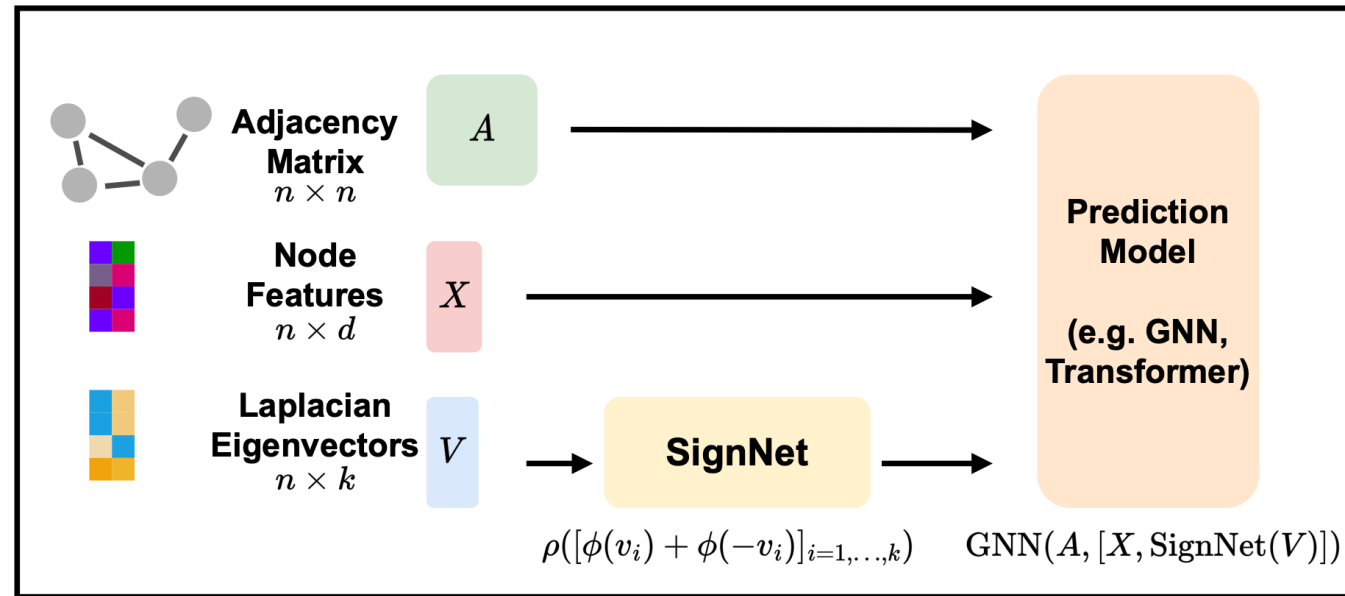
SignNet and BasisNet

Use Laplacian eigenvectors as additional node features.

Input Graph



Model



SignNet: remove sign ambiguity

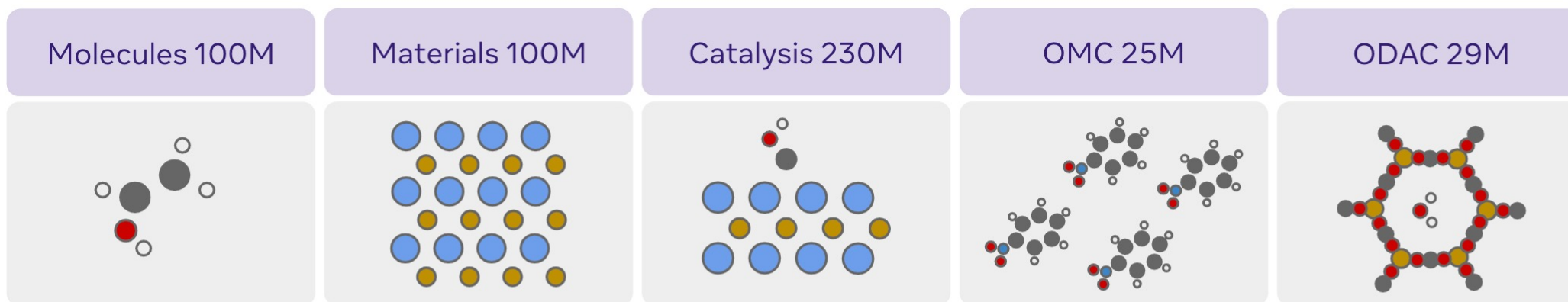
$$h(u) = \phi(u) + \phi(-u)$$

BasisNet: process eigenspace projectors

$$h(V) = \phi(VV^\top)$$

Geometric Graph Models at Scale

UMA: A Family of Universal Models for Atoms



- Inputs: atom types and 3D positions.
- Geometric interactions support atomistic prediction.
- Next lecture: symmetries of positions and directions.

Discussion: Which Structure Helps?

- **Information**
 - What does the graph add beyond node features?
- **Communication**
 - Would local messages, global attention, or both suit the target?
- **Evidence**
 - What changes if we remove or randomize the edges?
 - What result would falsify the claimed architectural advantage?

Summary

- **Symmetry**
 - Node representations follow a relabeling; graph predictions ignore it.
- **Computation**
 - Polynomial spectral filters connect convolution to local propagation.
 - GCNs are normalized message-passing layers.
- **Model choice**
 - Expressivity, information flow, and available data guide the design.

References: Message Passing

- [1] Scarselli et al. (2009). The Graph Neural Network Model. IEEE Transactions on Neural Networks.
- [2] Duvenaud et al. (2015). Convolutional Networks on Graphs for Learning Molecular Fingerprints. NeurIPS.
- [3] Li et al. (2016). Gated Graph Sequence Neural Networks. ICLR.
- [4] Battaglia et al. (2016). Interaction Networks for Learning about Objects, Relations and Physics. NeurIPS.
- [5] Gilmer et al. (2017). Neural Message Passing for Quantum Chemistry. ICML.

References: Convolution and Limits

- [6] Defferrard et al. (2016). Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. NeurIPS.
- [7] Kipf & Welling (2017). Semi-Supervised Classification with Graph Convolutional Networks. ICLR.
- [8] Xu et al. (2019). How Powerful Are Graph Neural Networks? ICLR.
- [9] Li et al. (2018). Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. AAAI.
- [10] Alon & Yahav (2021). On the Bottleneck of Graph Neural Networks and its Practical Implications. ICLR.
- [11] Veličković et al. (2018). Graph Attention Networks. ICLR.

References: Recent Connections

- [12] Rampášek et al. (2022). Recipe for a General, Powerful, Scalable Graph Transformer. NeurIPS.
- [13] Lim et al. (2023). Sign and Basis Invariant Networks for Spectral Graph Representation Learning. ICLR.
- [14] Liao et al. (2019). LanczosNet: Multi-Scale Deep Graph Convolutional Networks. ICLR.
- [15] Bo et al. (2023). Specformer: Spectral Graph Neural Networks Meet Transformers. ICLR.
- [16] Wood et al. (2025). UMA: A Family of Universal Models for Atoms. arXiv:2506.23971.

Implementations and practical guidance

PyTorch Geometric: Memory-Efficient Aggregations

GraphGPS official implementation

UMA / FAIR-Chem documentation

Questions?