

Lecture 3: Graph Neural Networks

Lecturer: Renjie Liao

Date: Sep. 29, 2026

Scribes: GPT-6 Astra

Note: *LaTeX template courtesy of UC Berkeley EECS dept.***Disclaimer:** *These notes were revised with assistance from GPT-6 Astra.*

These notes develop the mathematical connections behind the lecture: shared local computation respects node relabeling, polynomial spectral filters implement local propagation, and a restricted filter construction yields the graph convolutional network (GCN). Worked examples explain the role of normalization and the limits of message passing. Slide references follow the 78-slide version of Lecture 3.

Conventions. A graph has N nodes. A scalar graph signal $x \in \mathbb{R}^N$ is a column vector, whereas a feature matrix $H \in \mathbb{R}^{N \times d}$ stores one row vector h_i per node. Unless stated otherwise, graphs are finite, undirected, with symmetric nonnegative adjacency A and no self-loops initially. Message passing also permits directed and attributed edges. The spectral arguments below specifically require symmetry. We write

$$d_i = \sum_j A_{ij}, \quad D = \text{diag}(d_1, \dots, d_N), \quad L_c = D - A.$$

Degree normalization assumes $d_i > 0$. Isolated nodes can be handled separately; the GCN construction later adds self-loops and hence has strictly positive augmented degrees.

3.1 Graphs and permutation symmetry

Companion to slides 3–12.

A graph describes relationships as well as individual entities. In a molecule, atom types alone omit chemical bonds. Recommendation graphs link users to items; road graphs encode connectivity between road segments. These choices determine what information can pass between entities. The graph may be observed, constructed from data, or learned. Its usefulness depends on whether these relationships help predict the target.

Node indices specify a storage order. For a permutation π , define P by $(Px)_{\pi(i)} = x_i$. Relabeling the same attributed graph gives

$$A' = PAP^\top, \quad X' = PX, \quad e'_{\pi(j)\pi(i)} = e_{ji}.$$

A node predictor F should be *permutation equivariant*, and a graph predictor g should be *permutation invariant*:

$$F(PAP^\top, PX) = PF(A, X), \quad g(PAP^\top, PX) = g(A, X). \quad (3.1)$$

Edge-attribute arguments are suppressed in (3.1). Attributes follow the nodes when their storage order changes. Assigning a fresh feature equal to each new storage index changes the attributed input; it does not represent a relabeling of the original input.

3.2 Message passing

Companion to slides 14–24.

Message passing covers a family of shared local updates, including neural fingerprints, gated graph networks, interaction networks, and the MPNN formulation [1, 2, 3, 4]. With $H^{(0)} = X$, one layer takes the form

$$\begin{aligned} m_{j \rightarrow i}^{(\ell)} &= M_\ell(h_i^{(\ell)}, h_j^{(\ell)}, e_{ji}), \\ a_i^{(\ell)} &= \text{AGG}_\ell\{\{m_{j \rightarrow i}^{(\ell)} : j \in \mathcal{N}(i)\}\}, \\ h_i^{(\ell+1)} &= \mathcal{U}_\ell(h_i^{(\ell)}, a_i^{(\ell)}). \end{aligned} \tag{3.2}$$

The functions are shared across nodes at a given layer. Parameters may differ between layers. All messages in a layer use the old states; the new states are written simultaneously.

Why a multiset? Neighbor order is irrelevant, but repetitions can matter. If every neighbor sends the scalar 2, the neighborhoods in slide 17 produce $\{\{2, 2, 2\}\}$ and $\{\{2\}\}$. Both have mean 2 and maximum 2, but their sums are 6 and 2. Mean discards multiplicity under uniform repetition; maximum discards counts altogether. A raw sum also loses information: $\{\{1, 3\}\}$ and $\{\{2, 2\}\}$ both sum to 4. The relevant choice is therefore the combination of message encoding and aggregation, not aggregation alone.

Proposition 3.1 (Equivariance of message passing) *Shared message and update functions, with an order-independent aggregator, make each layer in (3.2) equivariant to a consistent relabeling of nodes and edge attributes.*

Proof: Suppose $h'_{\pi(i)} = h_i$. Relabeling maps $\mathcal{N}(i)$ bijectively onto $\mathcal{N}'(\pi(i))$, and the shared message function gives $m'_{\pi(j) \rightarrow \pi(i)} = m_{j \rightarrow i}$. Corresponding nodes therefore receive identical multisets. Aggregation and the shared update produce $h'^{\ell+1}_{\pi(i)} = h_i^{\ell+1}$. Induction over layers proves the claim. ■

Depth and output. After T local layers, a node can depend only on inputs within T hops, assuming the initial features contain no information from outside that neighborhood. This describes information access, not whether the model preserves or uses all accessible information. A shared node readout $\rho_{\text{node}}(h_i^{(T)})$ remains equivariant. A graph readout such as

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

is invariant. For an undirected edge target, the endpoint readout must also ignore endpoint order. For example, $\rho_{\text{edge}}([h_i + h_j, h_i \odot h_j, e_{ij}])$ is symmetric when $e_{ij} = e_{ji}$. Directed targets need not have this symmetry.

3.3 Laplacians, graph signals, and Fourier coordinates

Companion to slides 26–37.

3.3.1 The classical connection

For sufficiently integrable functions, ordinary convolution satisfies

$$y(t) = \int_{\mathbb{R}} h(\tau)x(t - \tau) d\tau, \quad \hat{y}(\omega) = \hat{h}(\omega)\hat{x}(\omega).$$

Thus convolution can be computed by changing to Fourier coordinates, multiplying each coordinate by a response, and transforming back. The complex exponentials also satisfy

$$-\frac{d^2}{dt^2} e^{i\omega t} = \omega^2 e^{i\omega t}.$$

They are generalized eigenfunctions on $\mathbb{R}$; on a periodic interval, the corresponding discrete Fourier modes form an orthogonal basis. This connection motivates using eigenvectors of a graph Laplacian as graph-dependent Fourier coordinates [5].

3.3.2 The combinatorial Laplacian

A graph signal is any vector $x \in \mathbb{R}^N$. No smoothness or generative assumption is part of this definition. The Laplacian acts as

$$(L_c x)_i = \sum_j A_{ij}(x_i - x_j). \quad (3.3)$$

It records signed disagreement with neighbors. Positive and negative disagreements can cancel, so a zero response at one node does not imply agreement with each neighbor.

Symmetry of A gives the energy identity

$$x^\top L_c x = \sum_i d_i x_i^2 - \sum_{i,j} A_{ij} x_i x_j = \frac{1}{2} \sum_{i,j} A_{ij} (x_i - x_j)^2 \geq 0. \quad (3.4)$$

The factor 1/2 avoids counting an undirected edge twice. Consequently L_c is positive semidefinite. Energy is zero exactly when the signal is constant on each connected component of the positive-weight graph.

Example 3.2 (Three-node path) For the path 1 – 2 – 3,

$$L_c = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}, \quad x^\top L_c x = (x_1 - x_2)^2 + (x_2 - x_3)^2.$$

The constant signal $(1, 1, 1)^\top$ has energy 0, whereas $(1, -1, 1)^\top$ has energy 8. An orthonormal eigenbasis is

$$\begin{aligned} u_0 &= \frac{(1, 1, 1)^\top}{\sqrt{3}} & u_1 &= \frac{(1, 0, -1)^\top}{\sqrt{2}} & u_2 &= \frac{(1, -2, 1)^\top}{\sqrt{6}} \\ \lambda_0 &= 0 & \lambda_1 &= 1 & \lambda_2 &= 3. \end{aligned} \quad (3.5)$$

These patterns progress from constant to more rapidly varying. For any unit eigenvector, $\lambda_k = u_k^\top L_c u_k$, so its eigenvalue measures its edge variation. On an irregular graph, a mode need not resemble a sinusoid.

3.3.3 Completeness does not assume a smooth signal

Proposition 3.3 (Complete graph Fourier coordinates) For every real symmetric L_c , there is an orthonormal matrix $U_c = [u_0, \dots, u_{N-1}]$ with $L_c = U_c \Lambda_c U_c^\top$. Every $x \in \mathbb{R}^N$ has the exact expansion

$$\hat{x} = U_c^\top x, \quad x = U_c \hat{x} = \sum_{k=0}^{N-1} (u_k^\top x) u_k. \quad (3.6)$$

Proof: The spectral theorem supplies the orthonormal eigenbasis. Then $U_c U_c^\top = I$, so $x = U_c U_c^\top x$. ■

The graph determines the basis, not which vectors are admissible signals. Even independent random node values have an exact expansion. Any orthonormal basis would give complete coordinates, but the Laplacian basis ties them to graph variation:

$$\|x\|_2^2 = \sum_k \hat{x}_k^2, \quad x^\top L_c x = \sum_k \lambda_k \hat{x}_k^2. \quad (3.7)$$

The graph Fourier transform changes coordinates without discarding information. For the path example, $x = (2, 1, 0)^\top$ has $\hat{x} = (\sqrt{3}, \sqrt{2}, 0)^\top$, and hence $x = \sqrt{3} u_0 + \sqrt{2} u_1$.

Repeated eigenvalues permit different orthonormal bases within the same eigenspace. Individual coefficients depend on this choice, but the projection onto the whole eigenspace does not.

Accurate truncation requires additional structure. Order eigenvalues increasingly and retain $r < N$ modes:

$$x_r = \sum_{k=0}^{r-1} \hat{x}_k u_k, \quad \|x - x_r\|_2^2 = \sum_{k=r}^{N-1} \hat{x}_k^2.$$

Exact recovery requires all omitted coefficients to vanish. If $\lambda_r > 0$, then

$$\|x - x_r\|_2^2 \leq \frac{x^\top L_c x}{\lambda_r}.$$

This follows by bounding each omitted energy weight below by λ_r . Thus low variation can justify approximation by low-frequency modes, although a small λ_r makes the bound weak. If omitted modes have zero eigenvalue, smoothness alone does not control their coefficients. A cutoff that splits a repeated eigenspace may also depend on its chosen basis; retaining complete eigenspaces avoids this ambiguity.

3.4 Choosing a degree normalization

Companion to slides 38–40.

Equation (3.3) can be written as

$$(L_c x)_i = d_i(x_i - \bar{x}_i), \quad \bar{x}_i = \frac{1}{d_i} \sum_j A_{ij} x_j.$$

The response contains both a difference from the neighbor mean and the total edge weight. Whether that degree factor is useful depends on the target.

Center value 1, all neighbor values 0	One neighbor	Four neighbors
Total mismatch $(L_c x)_i$	1	4
Average mismatch $d_i^{-1}(L_c x)_i$	1	1

If edges have unit thermal conductance and values are temperatures, $(L_c x)_i$ is total outward heat flow. Four connections should carry four times the flow. If the target is a sensor's discrepancy from its average neighbor reading, both configurations should have score 1. Neighbor count may still be useful for a separate confidence estimate.

Row normalization. The average-discrepancy operator is

$$L_{\text{rw}} = D^{-1} L_c = I - D^{-1} A, \quad (L_{\text{rw}} x)_i = x_i - \bar{x}_i.$$

Its neighbor-propagation matrix $D^{-1} A$ has rows summing to one. However, L_{rw} is generally not symmetric in the usual Euclidean inner product.

Symmetric normalization. For the remaining spectral construction, choose

$$R = D^{-1/2} A D^{-1/2}, \quad L = D^{-1/2} L_c D^{-1/2} = I - R. \quad (3.8)$$

This retains symmetry and a Euclidean orthonormal eigenbasis. It is related to row normalization through the change of coordinates

$$L = D^{1/2} L_{\text{rw}} D^{-1/2}, \quad x^\top L x = \frac{1}{2} \sum_{i,j} A_{ij} \left(\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}} \right)^2.$$

Thus normalized smoothness compares degree-scaled node values. For a connected graph, its zero mode is proportional to $(\sqrt{d_1}, \dots, \sqrt{d_N})^\top$, rather than generally constant. For instance, $L_{\text{rw}} \mathbf{1} = 0$, but on the three-node path

$$L \mathbf{1} = (1 - 1/\sqrt{2}, 1 - \sqrt{2}, 1 - 1/\sqrt{2})^\top \neq 0.$$

Symmetric normalization therefore does not compute the ordinary neighbor average at each node. Its eigenvalues lie in $[0, 2]$. From here onward write $L = U \Lambda U^\top$; this U generally differs from the combinatorial basis U_c .

3.5 Graph convolution and local spectral filters

Companion to slides 41–46.

3.5.1 Transform, filter, inverse transform

Using the chosen normalized Laplacian, the graph Fourier transform is $\hat{x} = U^\top x$. A spectral filter assigns a response $g_\theta(\lambda_k)$ to each frequency:

$$\hat{y}_k = g_\theta(\lambda_k)\hat{x}_k, \quad y = Ug_\theta(\Lambda)U^\top x =: g_\theta(L)x. \quad (3.9)$$

The first operation changes coordinates; the second changes the signal. For example, $g(\lambda) = e^{-t\lambda}$, $t > 0$, suppresses high-frequency components and yields diffusion under the chosen Laplacian. Other responses can emphasize high-frequency components.

Equation (3.9) generalizes the spectral multiplication in classical convolution. A general graph has no canonical translation operation, so this is an operator-based definition of graph filtering. Restricting the response to a function of the eigenvalue also gives the same response to an entire repeated eigenspace. Consequently the filter does not depend on arbitrary eigenvector signs or a basis choice inside that eigenspace. This restriction need not describe every classical convolution: opposite Fourier frequencies have the same Laplacian eigenvalue but can have different responses in an asymmetric filter.

The filter respects relabeling: if $L' = PLP^\top$, then $g_\theta(L') = Pg_\theta(L)P^\top$. An arbitrary dense spectral filter may mix distant nodes. Constructing a full dense eigenbasis also costs $O(N^3)$ in a standard dense eigendecomposition and $O(N^2)$ storage. Polynomial filters avoid that basis explicitly.

3.5.2 Polynomial degree gives a locality guarantee

For a degree- K response,

$$g_\theta(\lambda) = \sum_{k=0}^K \theta_k \lambda^k \implies g_\theta(L)x = \sum_{k=0}^K \theta_k L^k x. \quad (3.10)$$

There are $K + 1$ coefficients in this convention.

Proposition 3.4 (Polynomial locality) *If $\text{dist}(i, j) > K$, then $[g_\theta(L)]_{ij} = 0$.*

Proof: An entry of L^k is a sum of products along sequences of k matrix entries. Each nonzero entry of L either stays at a node or crosses one edge. Thus a nonzero product from j to i requires a path of at most k edges. Apply this argument to every term of (3.10). ■

The bound is *at most K hops*. Cancellations may give a smaller effective support. This is the precise link between polynomial spectral filtering and local propagation.

3.5.3 Chebyshev filters

ChebNet [6] parameterizes the spectral response using Chebyshev polynomials. The key connection is that evaluating a polynomial on the Laplacian has exactly the same effect as evaluating that polynomial on each eigenvalue in Fourier coordinates. We derive this connection before turning it into a graph convolution layer.

1. Start with scalar polynomials and rescale the spectrum. The scalar recurrence is

$$T_0(t) = 1, \quad T_1(t) = t, \quad T_k(t) = 2tT_{k-1}(t) - T_{k-2}(t), \quad k \geq 2.$$

The first terms are $1, t, 2t^2 - 1, 4t^3 - 3t, 8t^4 - 8t^2 + 1$. These polynomials form a basis for scalar polynomials: a degree- K response has $K + 1$ coefficients. They satisfy $T_k(\cos \alpha) = \cos(k\alpha)$, so $|T_k(t)| \leq 1$ on $[-1, 1]$. Their orthogonality uses the continuous weight $(1 - t^2)^{-1/2}$; it does not assert orthogonality over an arbitrary graph's finite collection of eigenvalues.

To place the spectrum in this interval, choose $\lambda_* \geq \lambda_{\max}(L) > 0$ and define

$$\bar{L} = \frac{2}{\lambda_*} L - I = U \bar{\Lambda} U^\top, \quad \bar{\Lambda} = \frac{2}{\lambda_*} \Lambda - I, \quad \mu_i = \frac{2\lambda_i}{\lambda_*} - 1 \in [-1, 1]. \quad (3.11)$$

The eigenvectors stay the same; only the eigenvalues change. For the normalized Laplacian used here, $\lambda_* = 2$ is a valid upper-bound choice. This does not assume that the graph's largest eigenvalue actually equals 2.

2. Substitute a matrix using matrix multiplication. For any square matrix M , define a polynomial by

$$p(t) = \sum_{j=0}^m a_j t^j \implies p(M) = \sum_{j=0}^m a_j M^j, \quad M^0 = I.$$

The scalar constant 1 becomes the identity matrix, not an all-ones matrix. Powers use ordinary matrix multiplication, not entrywise powers. Consequently,

$$\begin{aligned} T_0(M) &= I, & T_1(M) &= M, \\ T_2(M) &= 2M^2 - I, & T_3(M) &= 4M^3 - 3M, \\ T_k(M) &= 2MT_{k-1}(M) - T_{k-2}(M), & k &\geq 2. \end{aligned} \quad (3.12)$$

For example, $[M^2]_{ij} = \sum_v M_{iv} M_{vj}$: multiplication combines two successive propagation steps. Evaluating T_2 on each entry of M separately would be a different operator.

Why does this substitution implement the intended spectral response? If $M = U \text{diag}(\mu_i) U^\top$, then

$$M^j = U \text{diag}(\mu_i^j) U^\top,$$

because every intermediate $U^\top U$ cancels to I . Applying this identity to each term of a polynomial gives

$$T_k(M) = U \text{diag}(T_k(\mu_i)) U^\top. \quad (3.13)$$

Thus $T_k(M)$ maps each eigenvector to a scalar multiple of itself, with multiplier $T_k(\mu_i)$. It need not leave the entries of a node-domain signal unchanged.

3. Derive the single-channel graph filter. Choose the scalar frequency response

$$g_\theta(\lambda) = \sum_{k=0}^K \theta_k T_k \left(\frac{2\lambda}{\lambda_*} - 1 \right).$$

Substitute it into the transform–filter–inverse-transform operation in (3.9), then use (3.13):

$$\begin{aligned} y &= U \left[\sum_{k=0}^K \theta_k T_k(\bar{\Lambda}) \right] U^\top x \\ &= \sum_{k=0}^K \theta_k [U T_k(\bar{\Lambda}) U^\top] x = \boxed{\sum_{k=0}^K \theta_k T_k(\bar{L}) x}. \end{aligned} \quad (3.14)$$

The last expression computes the same spectral filter directly in node coordinates. These are exact equalities for the chosen degree- K model. An approximation arises only when that finite model is used to fit some other desired frequency response.

We need only the filtered signals $z_k = T_k(\bar{L})x$, not the full matrices $T_k(\bar{L})$. Multiplying the matrix recurrence by x gives

$$z_0 = x, \quad z_1 = \bar{L}x, \quad z_k = 2\bar{L}z_{k-1} - z_{k-2}, \quad y = \sum_{k=0}^K \theta_k z_k.$$

Each step is one sparse matrix–vector multiplication and a subtraction. No eigenvectors or dense matrix powers need to be computed.

A three-node example. For the unweighted path $1 - 2 - 3$, write

$$R = D^{-1/2} A D^{-1/2} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad L = I - R.$$

With $\lambda_* = 2$, $\bar{L} = -R$. Therefore

$$T_2(\bar{L}) = 2R^2 - I = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

Apply the recurrence to an impulse $x = e_1 = (1, 0, 0)^\top$:

$$z_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad z_1 = \begin{pmatrix} 0 \\ -1/\sqrt{2} \\ 0 \end{pmatrix}, \quad z_2 = 2\bar{L}z_1 - z_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

The two-step multiplication reaches nodes 1 and 3; subtracting z_0 cancels the contribution at node 1. For $K = 2$, the learned output is $y = \theta_0 e_1 - \theta_1 e_2/\sqrt{2} + \theta_2 e_3$. This illustrates both the two-hop reach and the signed combinations in a Chebyshev filter. Such a filter need not be a neighbor average.

4. Introduce input and output feature channels. Let $H \in \mathbb{R}^{N \times d_{\text{in}}}$. Denote input channel a by $H_{:a}$, and let output channel b sum filtered versions of all input channels. Give each channel pair its own response

$$g_{ab}(\lambda) = \sum_{k=0}^K \theta_{k,a,b} T_k \left(\frac{2\lambda}{\lambda_*} - 1 \right).$$

Before any activation, the output is

$$Y_{:b} = \sum_{a=1}^{d_{\text{in}}} g_{ab}(L) H_{:a} = \sum_{k=0}^K \sum_{a=1}^{d_{\text{in}}} \theta_{k,a,b} T_k(\bar{L}) H_{:a}. \quad (3.15)$$

Collect the learned coefficients into $W_k \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, with $[W_k]_{ab} = \theta_{k,a,b}$. Matrix multiplication now collects all output channels at once:

$$Y = \sum_{k=0}^K T_k(\bar{L}) H W_k = \sum_{k=0}^K Z_k W_k, \quad Z_k := T_k(\bar{L}) H. \quad (3.16)$$

Left multiplication by $T_k(\bar{L})$ mixes nodes; right multiplication by W_k mixes feature channels. In dimensions, $(N \times N)(N \times d_{\text{in}})(d_{\text{in}} \times d_{\text{out}})$ produces $N \times d_{\text{out}}$. The same W_k is shared across all nodes.

5. Evaluate the final layer by a sparse recurrence. Apply the scalar-signal recurrence to every feature channel simultaneously:

$$\begin{aligned} Z_0 &= H, & Z_1 &= \bar{L}H, \\ Z_k &= 2\bar{L}Z_{k-1} - Z_{k-2}, & 2 \leq k \leq K, \\ H' &= \sigma \left(\sum_{k=0}^K Z_k W_k \right). \end{aligned} \quad (3.17)$$

Use only Z_0 if $K = 0$. A shared bias $\mathbf{1}b^\top$, $b \in \mathbb{R}^{d_{\text{out}}}$, may be added before the activation. The graph fixes the propagation operator; the $K + 1$ matrices W_k are learned by backpropagation. Do not insert nonlinearities

into the recurrence for Z_k : doing so generally destroys its identity with the polynomial filter. The final activation makes the complete layer nonlinear.

For degree two, the form can be written out explicitly:

$$\begin{aligned} H' &= \sigma(HW_0 + \bar{L}HW_1 + (2\bar{L}^2 - I)HW_2) \\ &= \sigma(H(W_0 - W_2) - RHW_1 + 2R^2HW_2) \quad (\lambda_* = 2, L = I - R). \end{aligned}$$

The three terms combine zero-, one-, and two-step propagation with learned channel transforms. In general the degree- K layer has at most K -hop dependence, by polynomial locality; individual terms can cancel, as in the example above.

The sparse recurrence costs $O(K(|E|+N)d_{\text{in}})$, followed by channel transforms costing $O((K+1)Nd_{\text{in}}d_{\text{out}})$. Accumulate $\sum_k Z_k W_k$ while keeping only the two previous Z_k values for forward evaluation; training may retain additional intermediates for gradients. This avoids materializing dense polynomial matrices.

Finally, keeping only $K = 1$ gives $Y = HW_0 + \bar{L}HW_1 = HW_0 - RHW_1$ for $\lambda_* = 2$. Tying $W_0 = -W_1 = W$ yields $(I + R)HW$. Section 3.6 explains how a further change of the propagation normalization leads to the usual GCN layer.

3.6 Deriving and interpreting the GCN layer

Companion to slides 48–55.

3.6.1 The sequence of modeling choices

The GCN of Kipf and Welling [7] follows a restricted first-order spectral construction:

1. Keep degree 0 and 1: $y = \theta_0 x + \theta_1 \bar{L}x$.
2. Use $\lambda_{\text{max}} \approx 2$ in the rescaling. Then $\bar{L} \approx L - I = -R$, giving $y \approx \theta_0 x - \theta_1 Rx$.
3. Tie coefficients, $\theta_0 = -\theta_1 = \theta$, to obtain $y \approx \theta(I + R)x$.

Truncation and coefficient tying restrict the model; the rescaling specifies its parameterization. The largest eigenvalue need not equal 2; equivalently, one may choose 2 as a fixed rescaling bound. The coefficient tying leaves only one scalar response parameter.

Repeatedly applying $I + R$ can amplify its eigenvalue-2 direction. GCN replaces this operator by adding self-loops and normalizing again:

$$\hat{A} = A + I, \quad \hat{d}_i = \sum_j \hat{A}_{ij}, \quad \hat{D} = \text{diag}(\hat{d}_i), \quad \hat{S} = \hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2}. \quad (3.18)$$

This *renormalization* is a new operator, not an algebraic equality between $I + R$ and $\hat{S}$. The spectral radius of $\hat{S}$ is one. With a channel transform and pointwise activation, the layer is

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

(3.19)

Bias terms, when used, must be shared across nodes.

3.6.2 Message passing and a numerical example

Reading (3.19) one row at a time gives

$$h_i^{(\ell+1)} = \sigma \left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{\hat{A}_{ij}}{\sqrt{\hat{d}_i \hat{d}_j}} h_j^{(\ell)} W_\ell \right). \quad (3.20)$$

The feature transform W_ℓ is learned and shared. For a fixed graph, the aggregation coefficients are determined by edges and augmented degrees. They need not sum to one.

For the slide example, the unweighted edges are $\{1, 2\}, \{1, 4\}, \{3, 4\}$. Adding self-loops gives $\hat{d} = (3, 2, 2, 3)^\top$. With $x = (1, 2, 3, 4)^\top$, $W = 1$, and identity activation,

$$(\hat{S}x)_1 = \frac{1}{3} + \frac{2}{\sqrt{6}} + \frac{4}{3} \approx 2.48316.$$

Only node 1, node 2, and node 4 contribute. The corresponding arithmetic mean would be $7/3$, a different result.

For efficient evaluation, first compute $Z = HW$, then sum weighted rows of Z over the nonzero entries of $\hat{S}$. This costs $O(Nd_{\text{in}}d_{\text{out}} + (|E| + N)d_{\text{out}})$ using sparse storage, without materializing a dense $N \times N$ matrix.

Two views of the same layer. Equation (3.20) is normalized message passing. On the self-loop-augmented graph, $\hat{S} = I - \hat{L}$, where $\hat{L}$ is its normalized Laplacian, so the linear propagation is also a degree-one spectral filter. The subsequent channel mixing and nonlinearity complete the neural layer. General nonlinear MPNNs need not be fixed spectral filters.

3.7 The Weisfeiler–Lehman test and expressivity

Companion to slides 57–63.

3.7.1 Color refinement

For this section, consider simple unweighted graphs. Any initial node labels have consistent meanings across graphs. The one-dimensional Weisfeiler–Lehman test (1-WL) repeatedly refines categorical node colors:

$$c_v^{(t+1)} = \text{HASH}_t\left(c_v^{(t)}, \{\{c_u^{(t)} : u \in \mathcal{N}(v)\}\}\right). \quad (3.21)$$

The hash is an ideal *injective lookup*: identical complete signatures receive the same new color, and distinct signatures receive different colors. Both graphs use the same lookup. All updates read only the previous round’s colors. Colors are labels, not numerical magnitudes.

Compare color histograms after every round. Different counts certify non-isomorphism, because an isomorphism preserves every refinement step. Matching counts are inconclusive. Refinement stabilizes when no color class splits further; the names of the colors may still change.

3.7.2 A complete two-round example

Compare P_6 , the six-node path, with $P_3 \sqcup C_3$, a three-node path and a disjoint triangle. Both have six nodes, five edges, and degree multiset $\{1, 1, 2, 2, 2, 2\}$. Start all nodes at color X and apply the following shared lookup.

Round	Own old color	Neighbor old colors	New color
1	X	$\{\{X\}\}$	A
1	X	$\{\{X, X\}\}$	B
2	A	$\{\{B\}\}$	C
2	B	$\{\{A, B\}\}$	D
2	B	$\{\{B, B\}\}$	E
2	B	$\{\{A, A\}\}$	F

For example, the short-path center has old color B and two old A neighbors, so it becomes F . A long-path node next to an endpoint has old color B and neighbors A, B , so it becomes D . Figure 3.1 shows the simultaneous updates. The different round-2 histograms prove non-isomorphism.

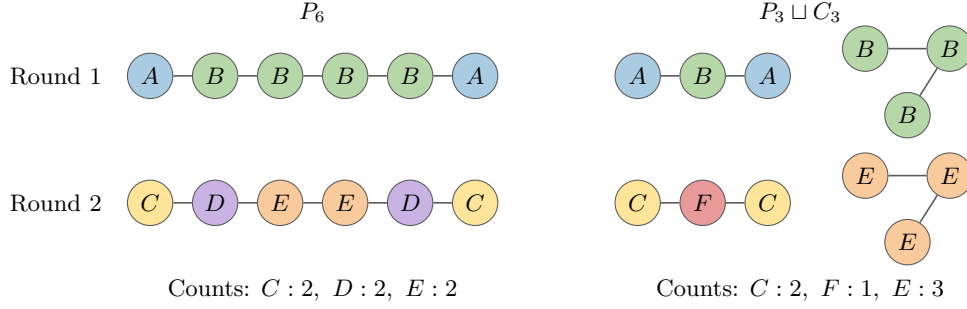


Figure 3.1: The same lookup acts on both graphs. The first round has matching counts ($A : 2, B : 4$); the second round distinguishes the graphs.

3.7.3 The MPNN upper bound

For a round-by-round comparison, assume shared deterministic updates

$$h_v^{(t+1)} = \mathcal{F}_t \left(h_v^{(t)}, \{ \{ h_u^{(t)} : u \in \mathcal{N}(v) \} \} \right),$$

and a graph readout that depends only on the multiset of node states. Initialize 1-WL with the same features as the MPNN, with no extra node identifiers or positional encodings on either side.

Proposition 3.5 (Standard message passing is bounded by 1-WL) *If two graphs have matching 1-WL color histograms through T rounds, a T -layer MPNN with these inputs and a permutation-invariant multiset readout cannot distinguish them [8].*

Proof: Inductively, equal WL colors imply equal hidden states. This holds at initialization. Equal colors at the next round mean equal old center colors and equal multisets of neighbor colors. By the induction hypothesis, these imply equal center states and neighbor-state multisets, hence equal outputs of the shared update. Matching histograms therefore imply identical multisets of final hidden states and identical graph readouts. ■

The upper bound does not require injective neural updates. Noninjective updates can discard distinctions that 1-WL preserves. Degree-dependent normalization can be covered by carrying degrees as fixed state components and including them in the initial WL colors. Without this augmentation, comparing layers to rounds can have an offset; the eventual 1-WL upper bound still applies to a standard GCN. For edge-attributed or directed message passing, the comparison must use a refinement rule that receives the corresponding edge labels and directions.

Conditions for matching 1-WL. Sufficient conditions are an injective encoding of each complete center-plus-neighbor-multiset signature and an injective graph readout on the inputs considered. Enough layers must reach a distinguishing round. Xu et al.’s sum-based construction assumes a countable feature universe and bounded multiset size [8]. A suitable embedding ϕ then makes $\sum_{x \in M} \phi(x)$ injective on those multisets. This motivates the graph isomorphism network (GIN):

$$h_v^{(t+1)} = \text{MLP}_t \left((1 + \epsilon_t) h_v^{(t)} + \sum_{u \in \mathcal{N}(v)} h_u^{(t)} \right).$$

The theorem is conditional on suitable encodings and functions. Arbitrary parameters or choices of ϵ_t do not guarantee injectivity. For example, $\epsilon_t = 0$ can lose the distinction between the center and a single neighbor when their states are exchanged. A finite trained network need not realize the ideal distinguishing functions.

A limitation. With uniform initial features, C_6 and $C_3 \sqcup C_3$ both have six nodes, each seeing two identical neighbors. All nodes therefore receive the same signature at every round. Neither 1-WL nor the corresponding standard MPNN separates them, even though one graph is connected and the other is not. Richer input features change this comparison. Expressive power specifies representable distinctions under stated inputs; it does not by itself determine test accuracy.

3.8 Depth and communication bottlenecks

Companion to slides 64–65.

Oversmoothing. Repeated propagation may remove distinctions among node representations. A precise example uses the linear iteration $H^{(k)} = \hat{S}^k X$. Assume a connected undirected graph with nonnegative weights and positive self-loops. Then

$$q_i = \frac{\sqrt{\hat{d}_i}}{\sqrt{\sum_j \hat{d}_j}}, \quad \hat{S}^k X \longrightarrow qq^\top X. \quad (3.22)$$

To see this, $\hat{S}q = q$, and $\hat{S}$ is similar to the transition matrix $\hat{D}^{-1}\hat{A}$. Connectedness and self-loops make this finite Markov chain irreducible and aperiodic. Every other eigenvalue has magnitude less than one, so its contribution vanishes as k grows. Rows of the limit are proportional to $\sqrt{\hat{d}_i}$, rather than necessarily identical. This linear result illustrates oversmoothing [9]; it does not prove convergence for every trained nonlinear GNN. Residual paths and normalization can change it.

Oversquashing. A different problem arises when many relevant distant signals must cross a small set of edges and be compressed into fixed-width states [10]. Increasing depth expands reach but does not by itself remove this communication bottleneck. Rewiring, larger states, or global attention can help, with additional computational or statistical costs. Oversmoothing concerns loss of distinction under mixing; oversquashing concerns restricted transmission of information. Either can occur without the other.

3.9 Architectures beyond the basic GCN

Companion to slides 66–72.

3.9.1 Attention and local–global models

Graph attention learns the aggregation weights from node states. A shared scorer and neighborhood softmax give

$$s_{ij} = a_\theta(h_i W, h_j W), \quad \alpha_{ij} = \frac{\exp(s_{ij})}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(s_{ik})}, \quad h'_i = \sigma \left(\sum_j \alpha_{ij} h_j W \right),$$

where the last sum uses the same neighborhood. The original GAT uses an additive scorer [11]. Learned weights still respect the chosen mask; they do not create communication beyond it. Shared functions and a consistently permuted mask preserve equivariance.

Global attention permits all node pairs. A common score is

$$\frac{q_i k_j^\top}{\sqrt{d}} + b_{ij} + M_{ij},$$

where b_{ij} is a structural bias and M_{ij} is 0 on allowed pairs and $-\infty$ otherwise. Equivariance requires these structural quantities to follow node relabeling. Attention on node features alone does not reveal which pairs are graph edges.

GraphGPS [12] combines local message passing, global attention, and positional or structural encodings. The local branch uses observed edges while the global branch provides longer-range communication. Dense attention has $O(N^2)$ pairwise cost; efficient attention variants can reduce it, while encoding precomputation has its own cost. Global access does not by itself guarantee useful structural information or better generalization.

3.9.2 LanczosNet and Specformer: learning spectral filters

LanczosNet. For symmetric normalized adjacency $S = D^{-1/2}AD^{-1/2}$, Lanczos iteration in exact arithmetic constructs an orthonormal basis $Q \in \mathbb{R}^{N \times r}$ for the Krylov subspace $\text{span}\{b, Sb, \dots, S^{r-1}b\}$, starting from a vector b , and a small tridiagonal matrix $T = Q^\top SQ$. The lifted approximation $QTQ^\top$ captures the action projected onto that subspace: with $\Pi = QQ^\top$, $QTQ^\top = \Pi S \Pi$. If $T = BRB^\top$, the columns of QB and the diagonal entries of $\mathcal{R}$ provide approximate spectral components. LanczosNet [13] combines direct short-scale propagation with learned filters using these components at longer diffusion scales. Rank r controls an approximation–cost tradeoff; a small rank need not approximate all signals well. A diffusion scale is not an exact hop radius. Here S denotes propagation, whereas the normalized Laplacian is $L = I - S$.

Specformer. Specformer [14] encodes eigenvalues as tokens, attends across them, and decodes spectral responses. A simplified single-response branch is

$$c = F_\theta(\lambda_0, \dots, \lambda_{N-1}), \quad y = U \text{diag}(c)U^\top x.$$

Each response can depend on the whole spectrum. Attention here operates on eigenvalue tokens, rather than node tokens. Reordering tokens must reorder responses. Equal eigenvalues must receive equal responses to preserve invariance within repeated eigenspaces; shared value encodings and permutation-equivariant attention, without token-index features, have this property. The full model combines multiple learned spectral bases. Eigendecomposition and spectral attention remain computational costs, and truncating the spectrum changes the information available.

3.9.3 SignNet and BasisNet: spectral features

Eigenvectors can instead be used as additional node features. Their ambiguity matters: u and $-u$ represent the same mode. SignNet [15] uses shared processing of both signs, with a building block

$$h(u) = \phi(u) + \phi(-u), \quad h(-u) = h(u).$$

For node features, choosing $\phi(Pu) = P\phi(u)$ also ensures $h(Pu) = Ph(u)$; sign sharing alone does not guarantee this equivariance. For a repeated eigenvalue, an orthonormal eigenspace basis V may also be replaced by VQ for any orthogonal Q . Its projector satisfies

$$(VQ)(VQ)^\top = VV^\top.$$

BasisNet processes such projectors with permutation-equivariant networks. Sign invariance alone does not handle arbitrary basis rotations in a repeated eigenspace. These constructions remove spectral ambiguities; node relabeling must still permute the resulting node features. Full models combine several eigenvector or eigenspace blocks.

3.9.4 Geometric models

Atomistic graphs add positions to node types and connectivity. A rigid rotation should preserve a predicted scalar energy and rotate predicted forces. For a differentiable invariant energy, forces obtained as its negative positional gradient have the corresponding equivariance and conservative structure. UMA [16] illustrates large-scale equivariant atomistic modeling. It is a geometric GNN, not a vanilla GCN. Permutation symmetry therefore leads naturally to the next lecture’s rotation and translation symmetries.

3.10 Choosing and evaluating the inductive bias

Companion to slides 73–74.

The useful question is which structural assumptions fit the prediction problem. A molecular target can depend on bonds even when atom types are unchanged. Conversely, a noisy or irrelevant graph can harm a model that assumes local neighbors should mix. Long-range targets may require more direct communication or appropriate spatial edges.

A focused evaluation compares a well-tuned feature-only baseline, a local graph model, and a relevant alternative under comparable feature access, data splits, and tuning effort. Edge removal or degree-preserving randomization can probe whether specific connectivity matters. Retrain under each modified input condition: randomizing edges only at test time measures robustness to a distribution shift. Learning curves test a claim about limited-data efficiency; size-shift experiments test a claim about generalization to larger graphs. Predictive gains alone do not identify their mechanism.

References

- [1] D. Duvenaud et al. *Convolutional Networks on Graphs for Learning Molecular Fingerprints*. NeurIPS, 2015. arXiv:1509.09292.
- [2] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. *Gated Graph Sequence Neural Networks*. ICLR, 2016. arXiv:1511.05493.
- [3] P. Battaglia et al. *Interaction Networks for Learning about Objects, Relations and Physics*. NeurIPS, 2016. arXiv:1612.00222.
- [4] J. Gilmer et al. *Neural Message Passing for Quantum Chemistry*. ICML, 2017. PMLR 70:1263–1272.
- [5] D. I. Shuman et al. *The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains*. IEEE Signal Processing Magazine, 30(3):83–98, 2013. arXiv:1211.0053.
- [6] M. Defferrard, X. Bresson, and P. Vandergheynst. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*. NeurIPS, 2016. arXiv:1606.09375.
- [7] T. N. Kipf and M. Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. ICLR, 2017. arXiv:1609.02907.
- [8] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. *How Powerful Are Graph Neural Networks?* ICLR, 2019. arXiv:1810.00826.
- [9] Q. Li, Z. Han, and X.-M. Wu. *Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning*. AAAI, 2018. arXiv:1801.07606.
- [10] U. Alon and E. Yahav. *On the Bottleneck of Graph Neural Networks and its Practical Implications*. ICLR, 2021. arXiv:2006.05205.
- [11] P. Veličković et al. *Graph Attention Networks*. ICLR, 2018. arXiv:1710.10903.
- [12] L. Rampášek et al. *Recipe for a General, Powerful, Scalable Graph Transformer*. NeurIPS, 2022. arXiv:2205.12454.
- [13] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel. *LanczosNet: Multi-Scale Deep Graph Convolutional Networks*. ICLR, 2019. arXiv:1901.01484.
- [14] D. Bo, C. Shi, L. Wang, and R. Liao. *Specformer: Spectral Graph Neural Networks Meet Transformers*. ICLR, 2023. arXiv:2303.01028.
- [15] D. Lim et al. *Sign and Basis Invariant Networks for Spectral Graph Representation Learning*. ICLR, 2023. arXiv:2202.13013.
- [16] B. M. Wood et al. *UMA: A Family of Universal Models for Atoms*. 2025. arXiv:2506.23971.