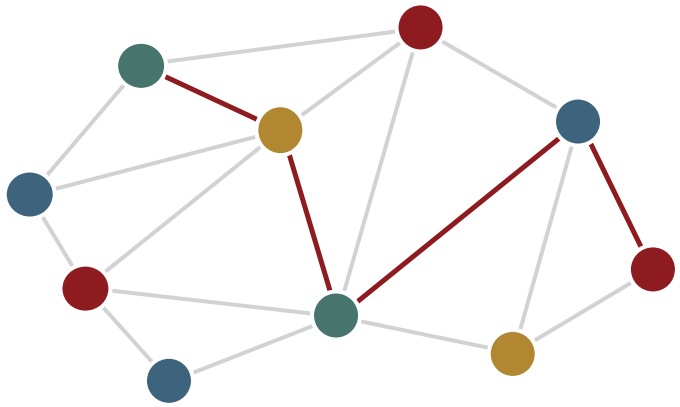
Message passing is graph convolution.



September 22

Zhiyang Wang · Electrical & Systems Engineering · WashU

Class 9 · In-depth study of Message Passing GNNs, GCN, ChebNet, GraphSAGE

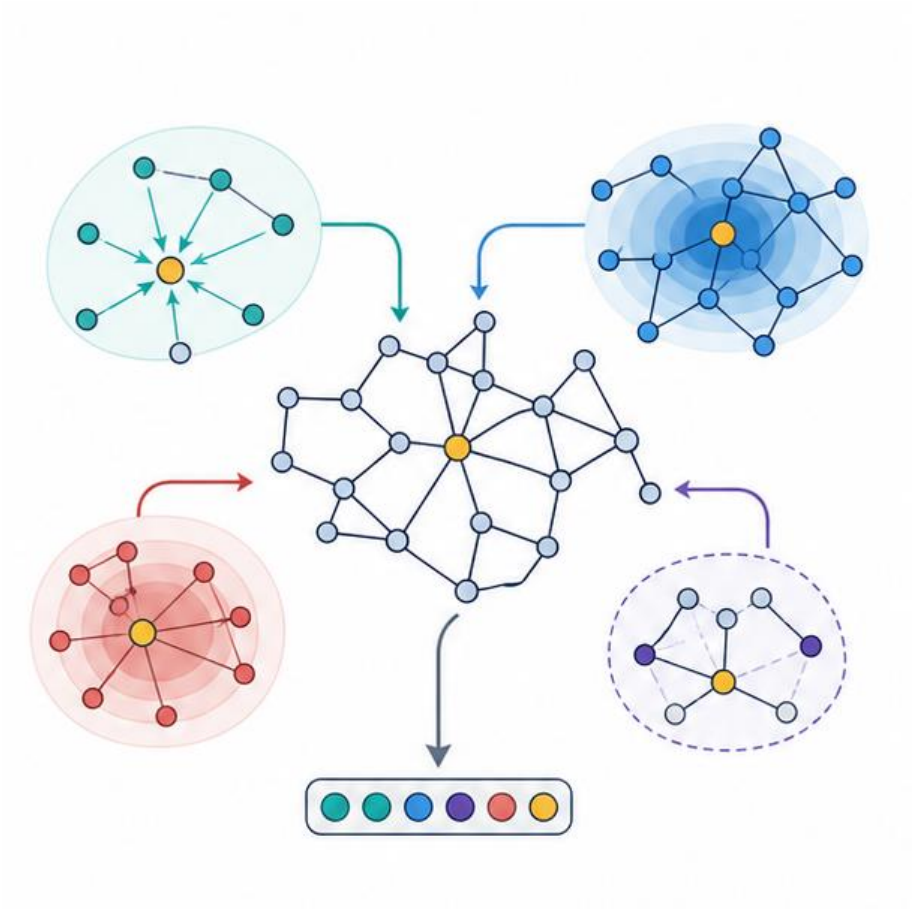
Message passing is graph convolution.

01 Build layers from local messages

02 Normalize one-hop diffusion in GCN

03 Learn multi-hop spectral filters in ChebNet

04 Sample neighborhoods for inductive GraphSAGE



Propagation and aggregation create each model's inductive bias.

Three questions guide us

01 **WHAT IS A MESSAGE-PASSING LAYER?**

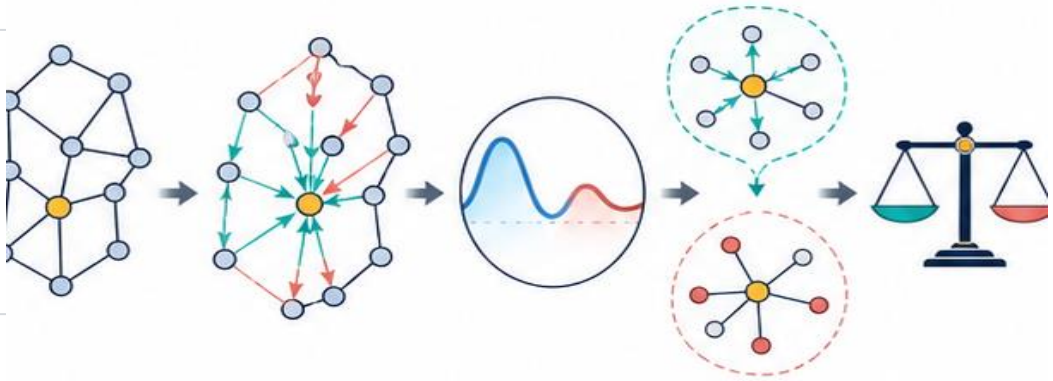
Separate edge messages, permutation-invariant aggregation, and node updates.

02 **HOW DO FILTERS DEFINE THE MODEL?**

Derive normalized GCN propagation and multi-order ChebNet filtering.

03 **HOW DO WE SCALE AND GENERALIZE?**

Use GraphSAGE aggregation and neighbor sampling for unseen nodes.



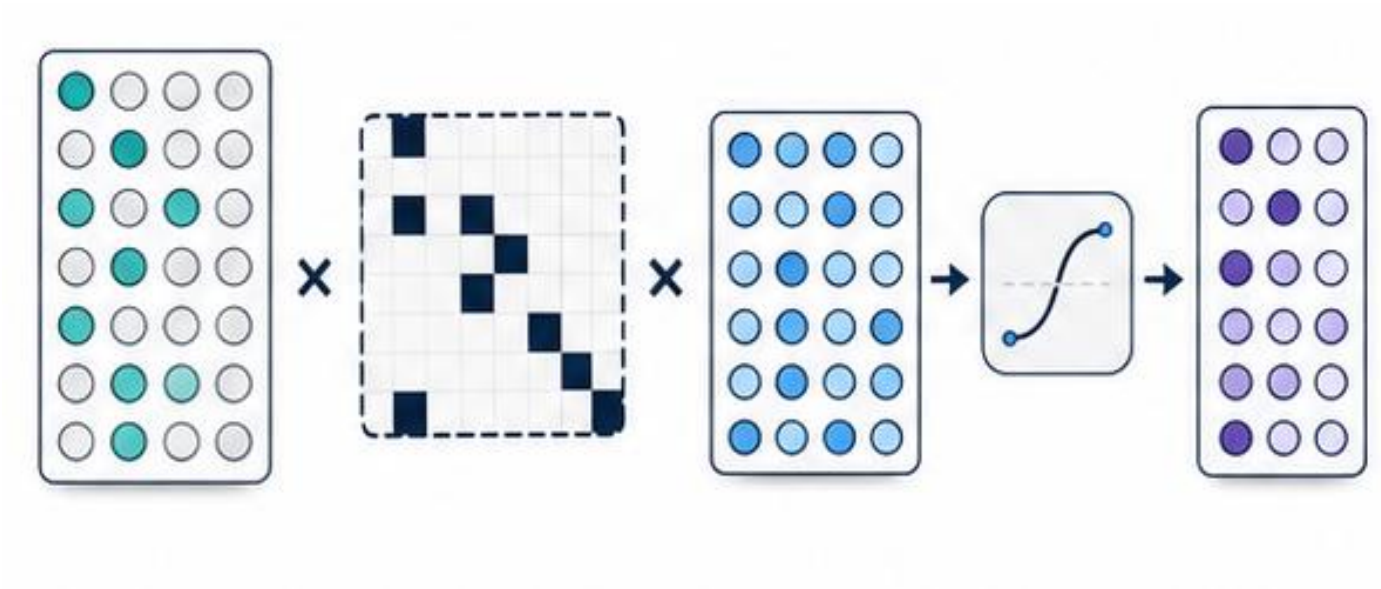
One layer maps features through a graph operator

$$\boldsymbol{X}_0 = \boldsymbol{X}, \quad \boldsymbol{S} \in \mathbb{R}^{n \times n}, \quad \boldsymbol{X}_l \in \mathbb{R}^{n \times d_l}$$

CONVOLUTIONAL LAYER

$$\boldsymbol{X_l} = \sigma\left(\sum_k \boldsymbol{S}^k \boldsymbol{X}_{l-1} \boldsymbol{H}_{l,k}\right)$$

$\boldsymbol{S}$ moves information across nodes;
$\boldsymbol{H}_{l,k}$ mixes feature channels;
σ acts independently on each node row.



Input: graph operator $\boldsymbol{S}$ and node features $\boldsymbol{X}$.
Output: one embedding row per node.

Topology moves information, trainable matrices mix channels.

A polynomial filter is a localized convolution

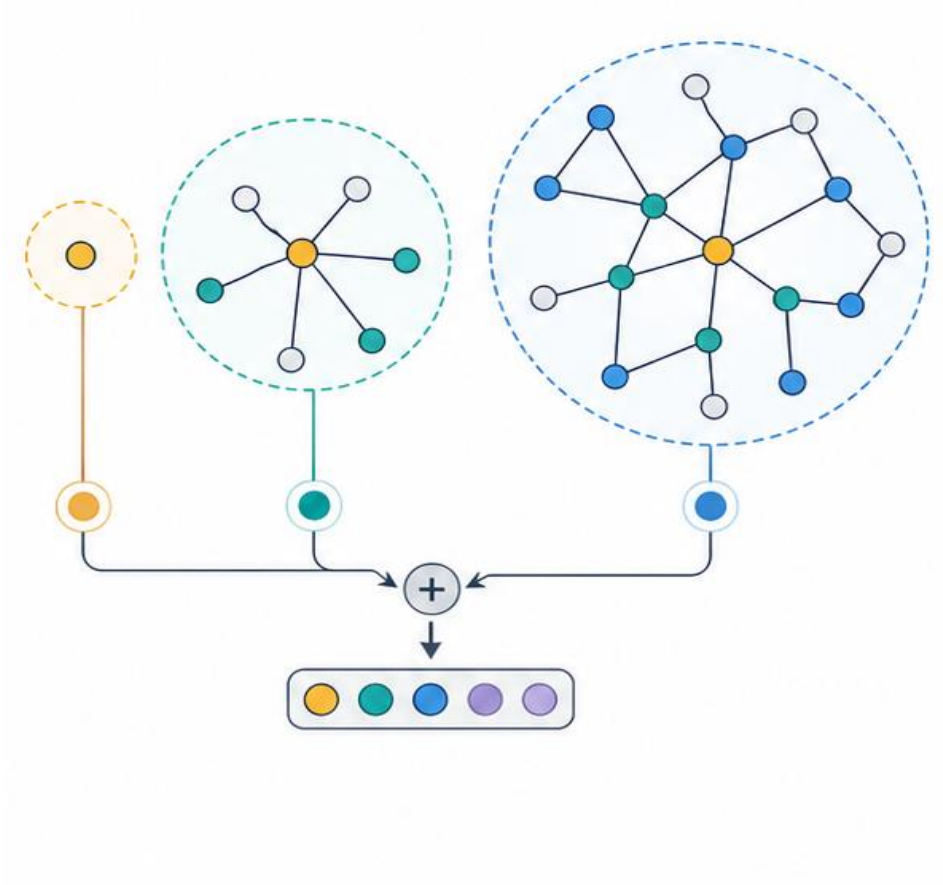
SPATIAL VIEW

$\boldsymbol{S}^{\mathrm{k}}\boldsymbol{X}$ *collects information from paths of length* k.

K *taps reach at most* $K-1$ *hops.*

SPECTRAL VIEW

$h(\lambda)=\sum_k h_k\lambda^k$ *sets the response on each graph frequency.*



Multi-hop locality and spectral shaping are two views of the same operator.

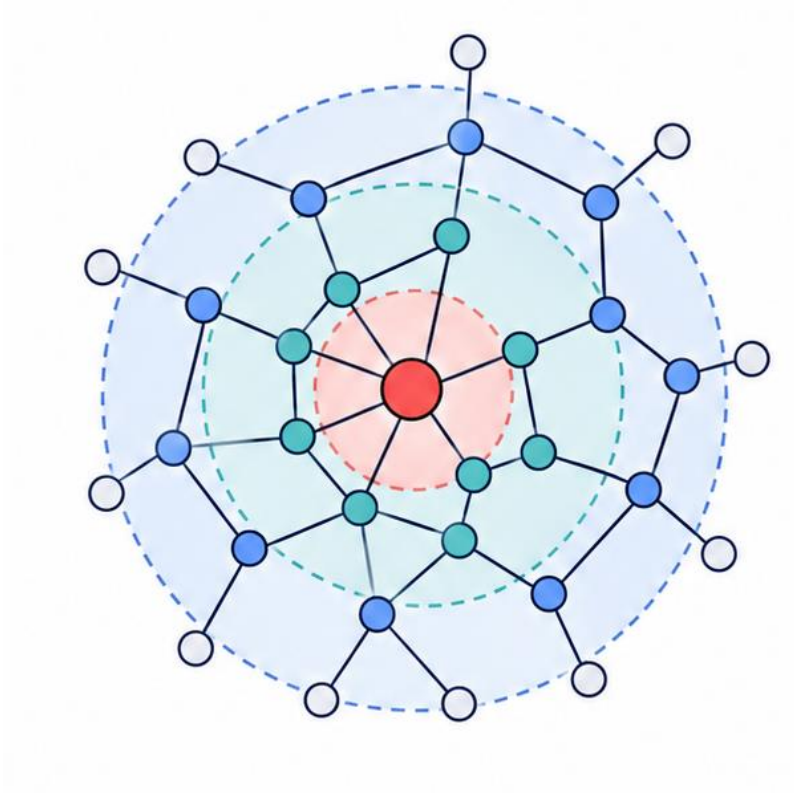
Depth expands the receptive field

SPATIAL REACH

Layer l can depend on nodes at distance at most l from the target.

COMPUTATION

Before overlap, a naive neighborhood can grow roughly as $\bar{d}^l$.



Deeper message passing increases both context and computation.

MPNNs separate messages from updates

Every node executes the same message, aggregation, and update functions at each layer.

MESSAGE FUNCTION

$$m_{ij}^{(l)} = M_l\left(x_i^{(l)}, x_j^{(l)}, e_{ij}\right)$$

AGGREGATION FUNCTION

$$m_i^{(l)} = AGG_{j\in\mathcal{N}_i}\left(m_{ij}^{(l)}\right)$$

UPDATE FUNCTION

$$x_i^{(l+1)} = U_l\left(x_i^{(l)}, m_i^{(l)}\right)$$

EXAMPLE 1 · GCN

message: $M_l(x_i, x_j, e_{ij}) = S_{ij}x_j$

aggregate: $m_i = \sum_j S_{ij}x_j$

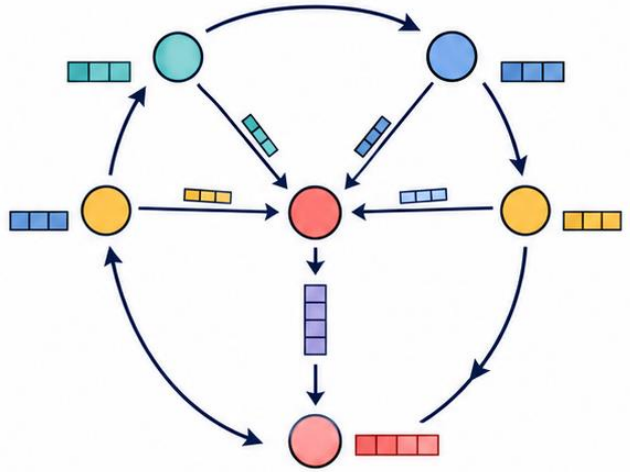
update: $x_i^+ = \sigma(m_i H_l)$

EXAMPLE 2 · GRAPHSAGE (MEAN)

message: $M_l(x_i, x_j) = x_j$

aggregate: $m_i = \mathrm{MEAN}_{j\in\widetilde{\mathcal{N}}_i} x_j$

update: $x_i^+ = \sigma([x_i \parallel m_i]\, H_l)$



Architecture choices enter through M_l, AGG_l, and U_l.

Aggregation must ignore neighbor order

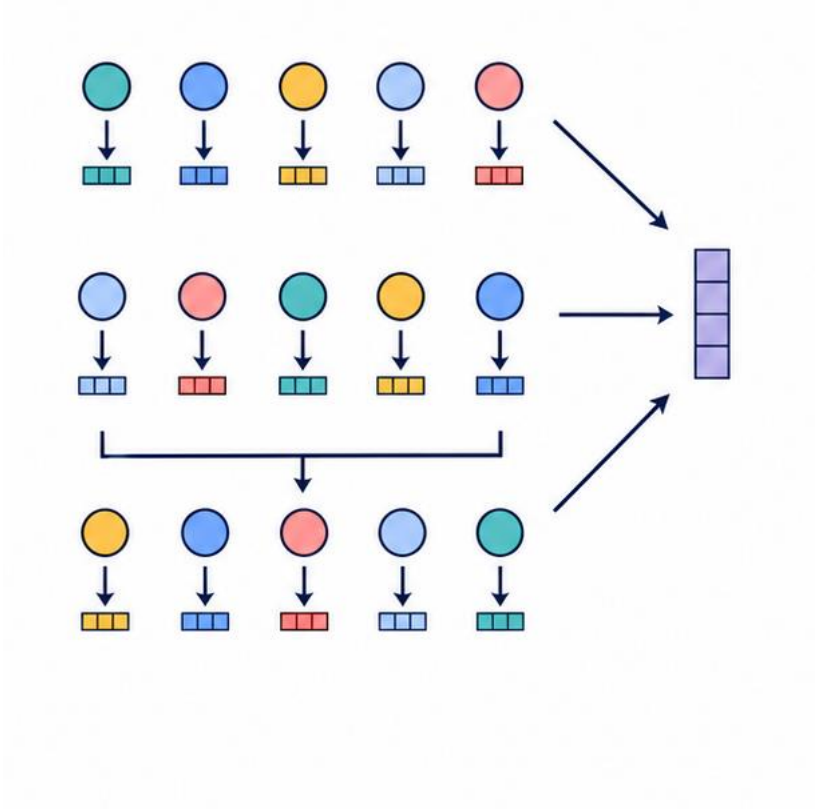
$$\mathrm{AGG}(\{a, b, c\}) = \mathrm{AGG}(\{c, a, b\})$$

A graph has no canonical ordering of $\mathcal{N}(i)$. Sum, mean, and max are permutation invariant.

EQUIVARIANCE

Relabel nodes → relabel output rows

A sequence aggregator needs explicit randomization or a consistent ordering rule.



Permutation symmetry is a design constraint, not a cosmetic choice.

Linear messages recover graph convolution

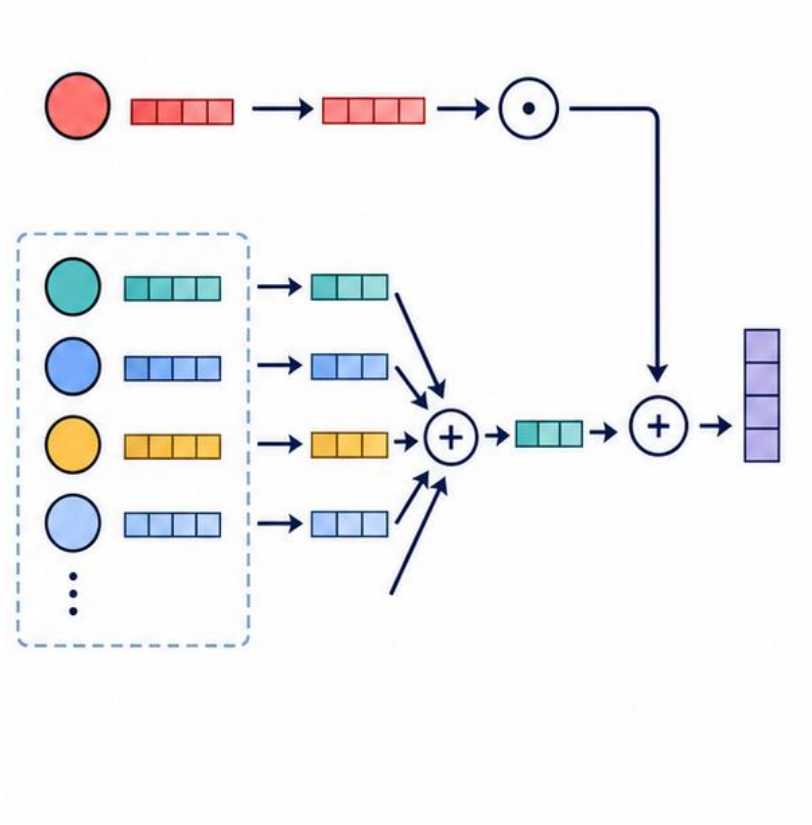
EDGE MESSAGE

$$M_l(x_i, x_j, e_{ij}) = \alpha_l x_i + \beta_l A_{ij} x_j$$

AGGREGATED MATRIX

$$\boldsymbol{m}^{(l)} = \alpha_l \boldsymbol{X} + \beta_l \boldsymbol{A}\boldsymbol{X}$$

—a degree-one polynomial in $\boldsymbol{A}$.



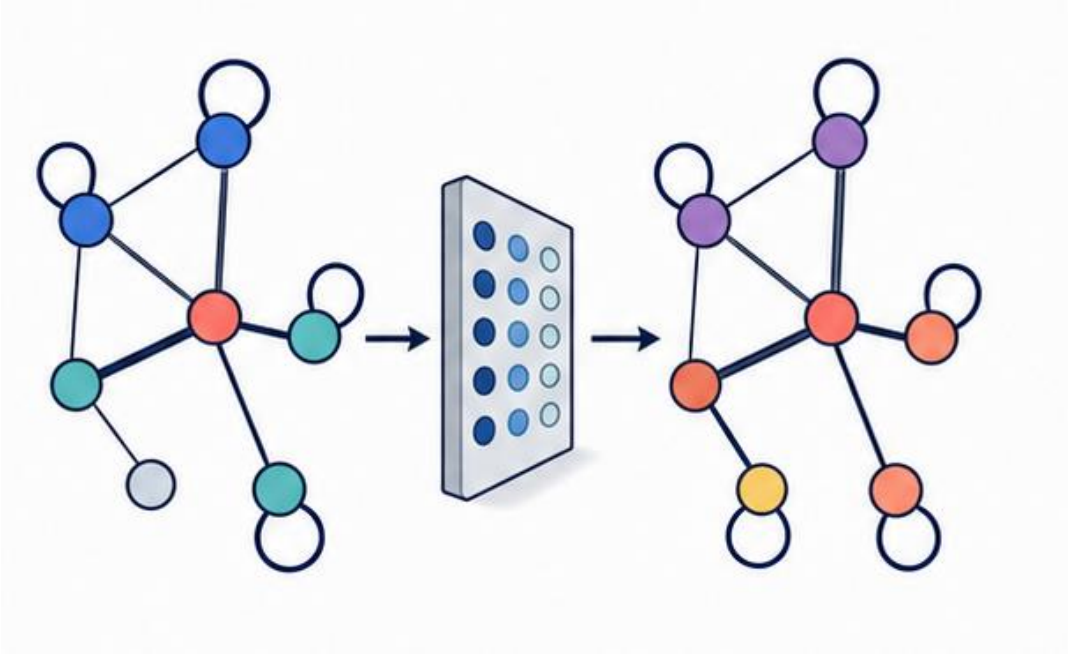
A pointwise nonlinear update turns this convolution into a GNN layer.

GCN uses normalized one-hop diffusion

Add self-loops, normalize by endpoint degrees, then apply one shared feature transform.

$$\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}, \qquad \widetilde{\boldsymbol{S}} = \widetilde{\boldsymbol{D}}^{-1/2}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-1/2}$$

$$\boldsymbol{X_l} = \sigma(\widetilde{\boldsymbol{S}}\, \boldsymbol{X_{l-1}}\, \boldsymbol{H_l})$$

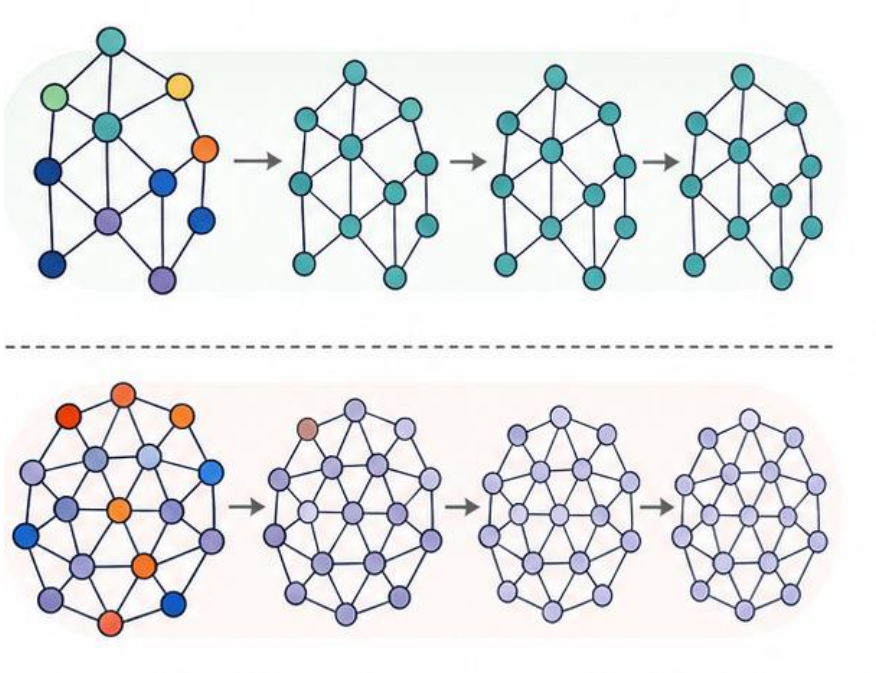


Each layer averages a node with its neighbors before mixing channels.

GCN trades flexibility for stability

The one-hop operator is simple, local, interpretable, and linear in the number of stored edges.

Only one fixed propagation operator: $\boldsymbol{S}$



More depth enlarges the receptive field but can oversmooth node distinctions.

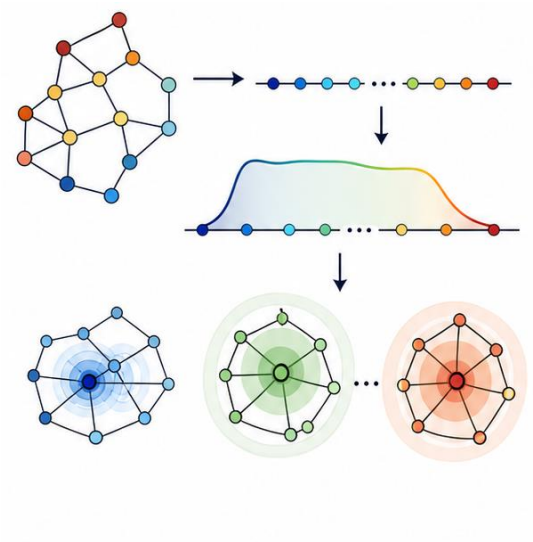
ChebNet learns a localized spectral response

TARGET FILTER

$h(\lambda)$ specifies which graph frequencies should be amplified or suppressed.

IMPLEMENTATION GOAL

Approximate $h(\boldsymbol{L})\boldsymbol{X}$ without eigenvectors or dense graph Fourier transforms.



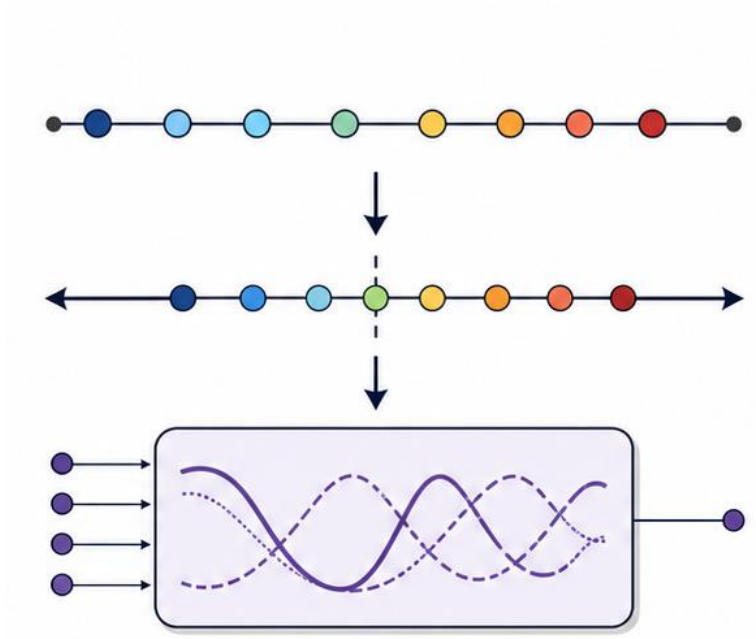
A K-term polynomial is K-hop localized and costs K sparse propagations.

Rescale the Laplacian before Chebyshev recursion

Map the spectrum to the Chebyshev interval

$$\tilde{\boldsymbol{L}} = \frac{2\boldsymbol{L}}{\lambda_{max}} - I, \qquad \mathrm{spec}(\tilde{\boldsymbol{L}}) \subset [-1,1]$$

- Choose $\boldsymbol{L}$ as a graph Laplacian.
- Estimate λ_{max} or use $\lambda_{max} \approx 2$ for the normalized Laplacian.



Apply the recurrence to $\tilde{L}$.
The rescaling keeps all basis functions in their stable domain.

Chebyshev polynomials are defined and controlled on $[-1,1]$.

Chebyshev bases are computed recursively

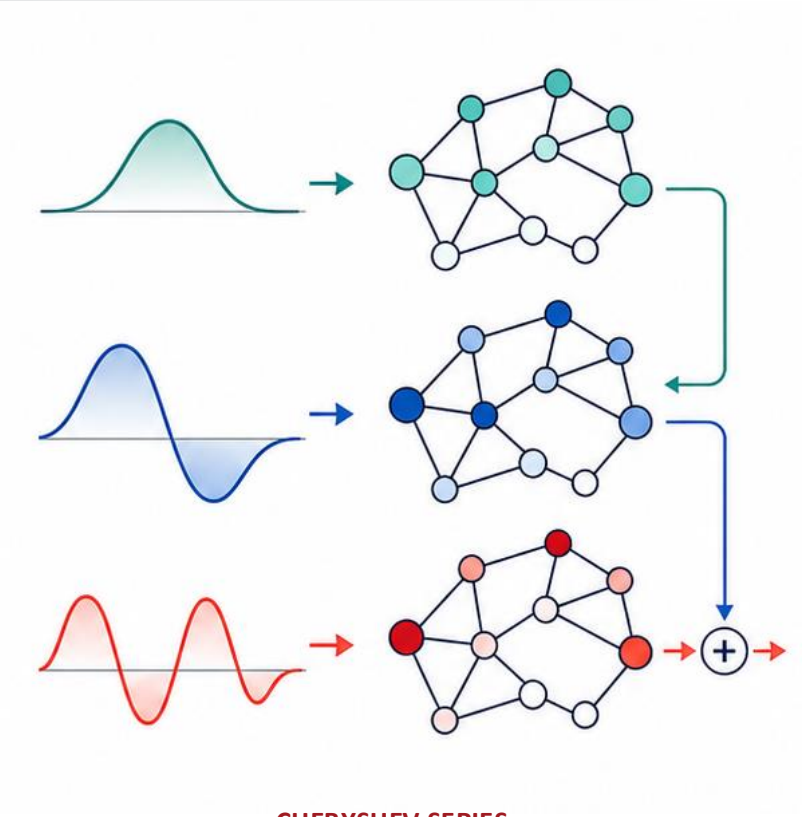
BASE CASES

$$T_0(\tilde{\boldsymbol{L}})\boldsymbol{X} = \boldsymbol{X}$$

$$T_1(\tilde{\boldsymbol{L}})\boldsymbol{X} = \tilde{\boldsymbol{L}}\boldsymbol{X}$$

NEXT ORDER

$$T_k(\tilde{\boldsymbol{L}})\boldsymbol{X} = 2\tilde{\boldsymbol{L}}T_{k-1}(\tilde{\boldsymbol{L}})\boldsymbol{X} - T_{k-2}(\tilde{\boldsymbol{L}})\boldsymbol{X}$$



POLYNOMIAL SERIES

Express same function class

Monomial basis is conditioned for learning

CHEBYSHEV SERIES

Express same function class

T_k are orthogonal on $[-1,1]$

No eigendecomposition is needed—only sparse matrix products.

ChebNet builds multi-hop bases

01 **RECURSE**

$Z_0 = X,\ Z_1 = \tilde{L}X,\ Z_k = 2\tilde{L}Z_{k-1} - Z_{k-2}.$

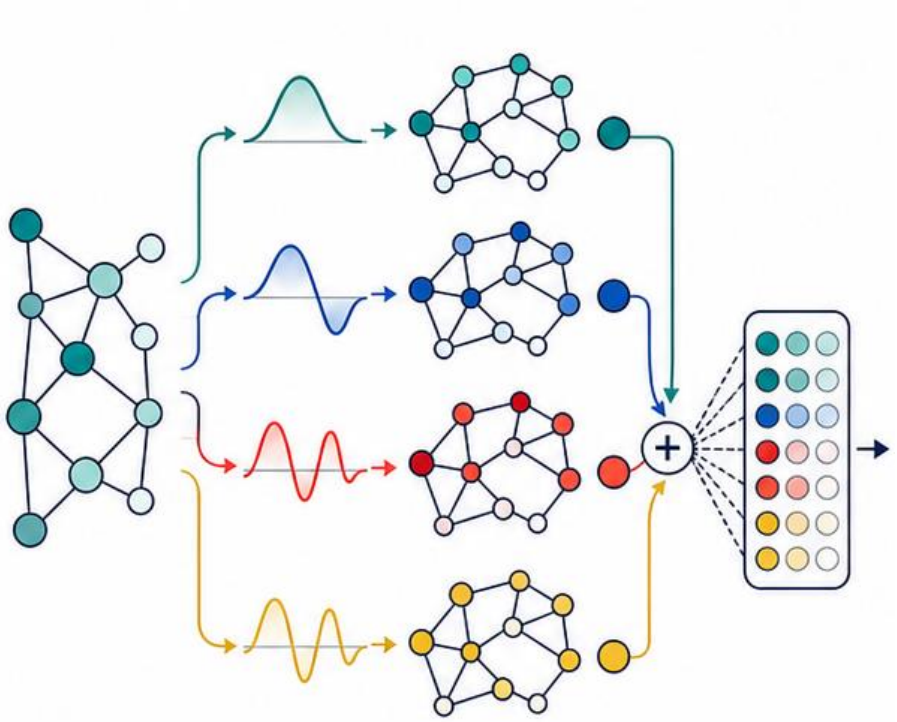
02 **WEIGHT**

Apply a learned matrix $H_{l,k}$ to each basis signal Z_k.

03 **SUM + ACTIVATE**

$X_l = \sigma(\sum_k Z_k H_{l,k}).$

K terms reach at most $K-1$ hops.



GCN is a first-order ChebNet approximation

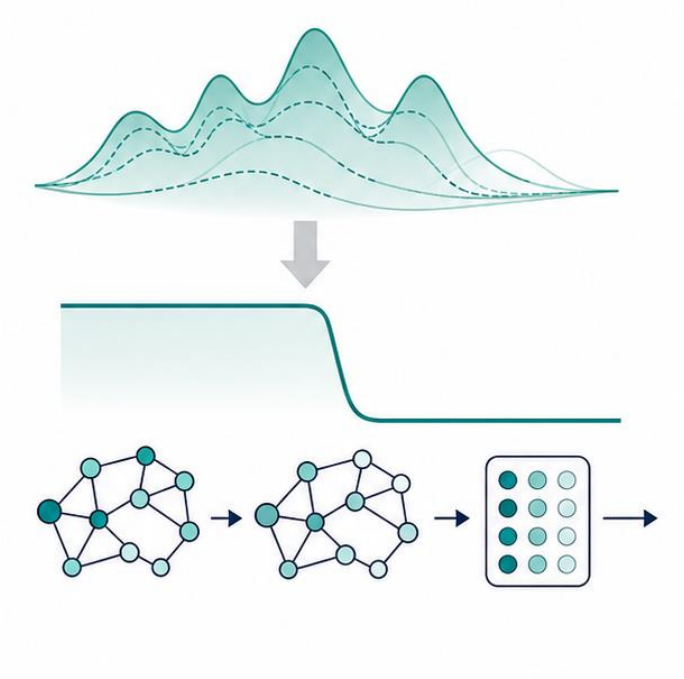
GCN is a $K = 2$ ChebNet with $\lambda_{max} \approx 2$, tied coefficients, and the renormalized operator.

ChebNet $(K = 2)$:

$\theta_0 X + \theta_1 \tilde{L} X = \theta(I + D^{-1/2} A D^{-1/2}) X$ with $\lambda_{max} = 2$, $\theta = \theta_0 = -\theta_1$

GCN:

$X^+ = \sigma(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} X H)$
renormalized: $\tilde{A} = A + I$, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $\theta \to H$



GCN compresses spectral flexibility into one stable one-hop operator.

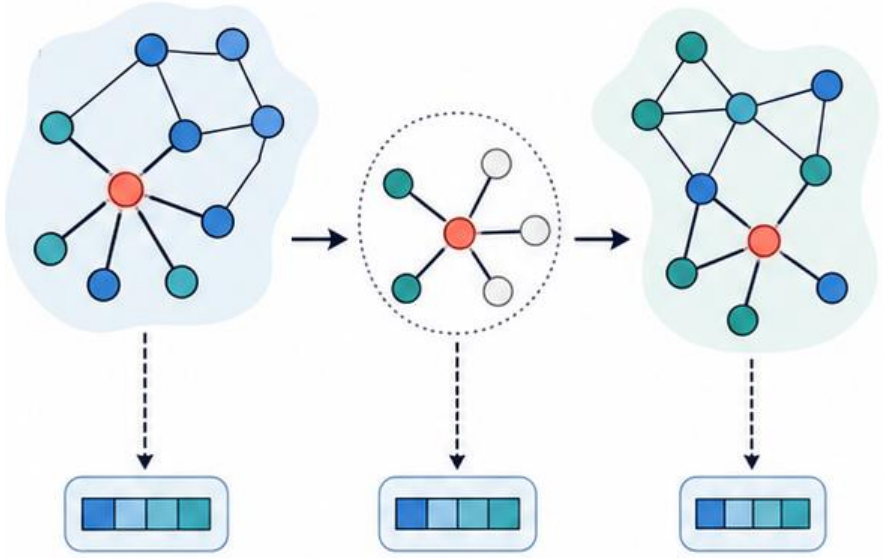
GraphSAGE learns an inductive aggregation rule

TRANSDUCTIVE EMBEDDING

A stored vector belongs to one specific training node.

INDUCTIVE FUNCTION

A shared aggregator maps features and neighborhoods to embeddings for unseen nodes.



GraphSAGE learns how to construct an embedding, not one embedding per node.

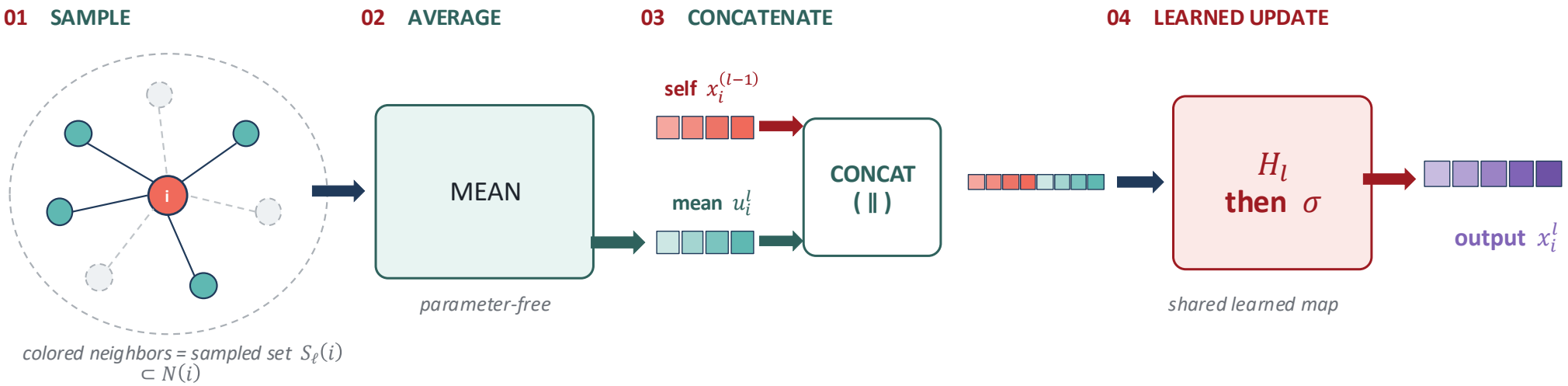
The mean aggregator is a sampled convolution

AGGREGATE

$$u_i^l = \text{MEAN}\{x_j^{l-1} : j \in S_l(i)\}$$

UPDATE

$$x_i^l = \sigma([x_i^{l-1} \parallel u_i^l]H_l)$$



Sampling changes evaluated neighbors, not the shared update rule.

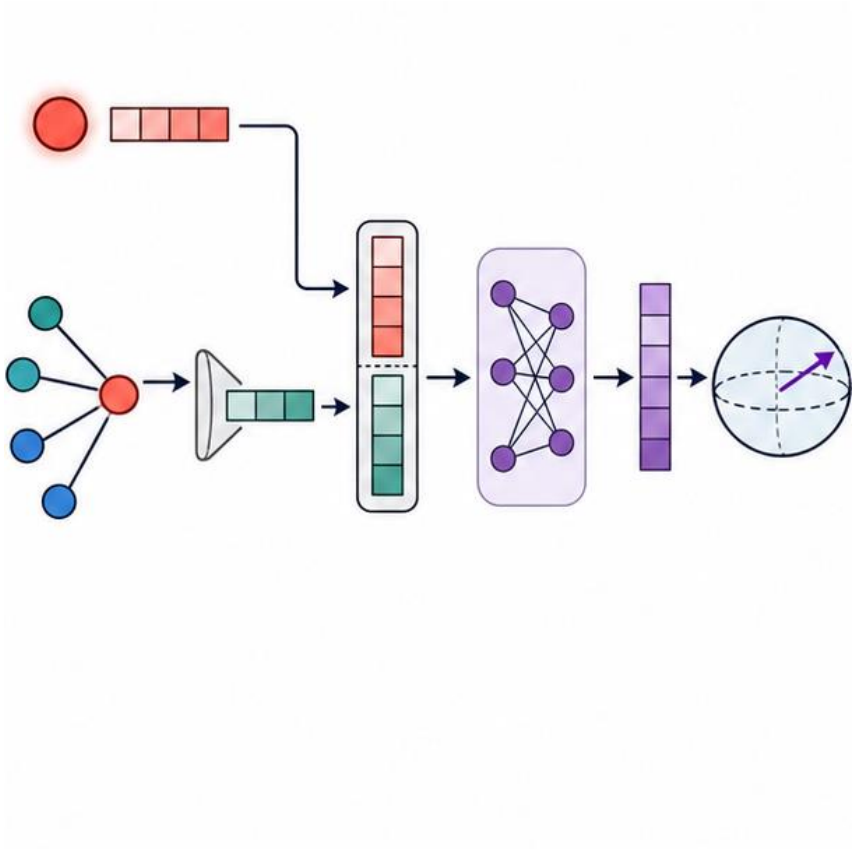
GraphSAGE separates self and neighbor channels

CONCATENATE

Keep x_i and u_i as distinct inputs so H_l can weight them differently.

NORMALIZE

$x_i \leftarrow \frac{x_i}{\|x_i\|_2}$ *often stabilizes embedding scale after the update.*



The layer is a degree-one convolution with two independently weighted taps.

Aggregator choice changes the inductive bias

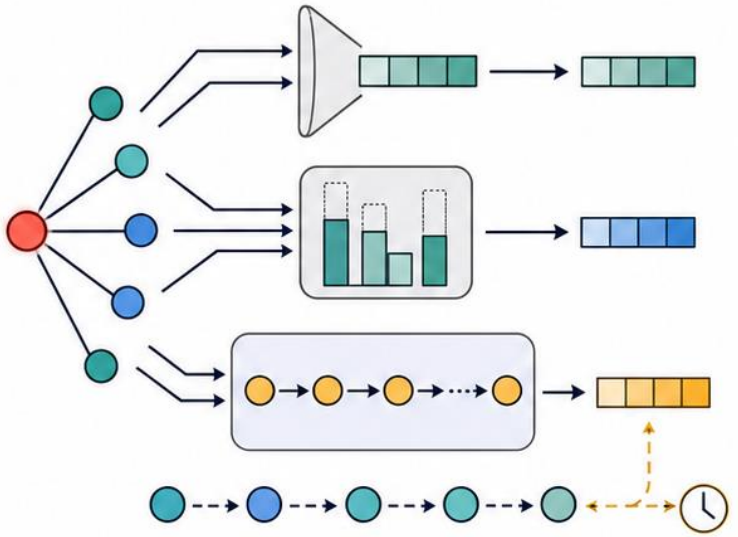
PERMUTATION-INVARIANT

Mean: smooth and degree normalized.

Max-pool: detects salient neighbor features.

SEQUENCE-BASED

LSTM can model interactions but depends on an imposed neighbor order.



Use symmetric aggregators when graph relabeling must not matter.

Choose the operator for the task

01 **PROPAGATION**

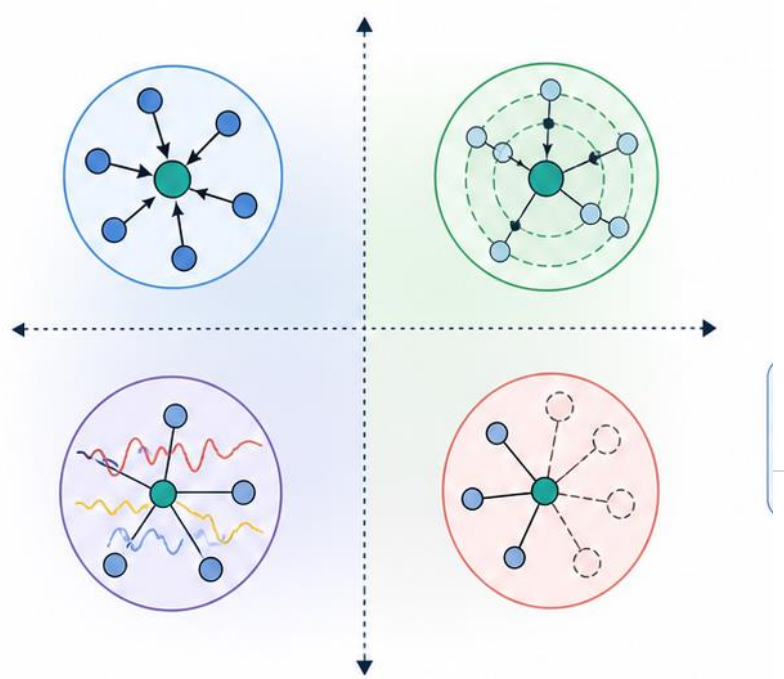
GCN: normalized one hop.
ChebNet: K-order polynomial.
GraphSAGE: sampled aggregator.

02 **GENERALIZATION**

GCN/ChebNet usually use full-graph operators; GraphSAGE applies a shared function to unseen nodes.

03 **COMPUTATION**

All use sparse neighborhood operations; sampling caps mini-batch expansion.



GCN vs. ChebNet vs. GraphSAGE

Criterion	GCN	ChebNet	GraphSAGE
Operator	SXH	$\Sigma_k T_k(\tilde{L}) X H_k$	$AGG\left(S_l(i)\right) + concat$
Per-layer reach	1 normalized hop	$K-1$ hops	1 sampled hop
Trainable parts	One shared H	One H_k per filter tap	Aggregator + self/neighbor weights
Generalization	Usually transductive	Usually transductive	Inductive; supports unseen nodes
Computation	$O(\|E\|d)$	$O(K\|E\|d)$	Mini-batches
Best fit	Stable local baseline	Multi-scale spectral control	Large or evolving graphs
Main tradeoff	Fixed smoothing; oversmoothing with depth	More filter freedom, K propagations	Sampling variance and estimator bias

TAKEAWAYS

One framework, three graph convolutions.

01 MPNN defines message, aggregate, and update

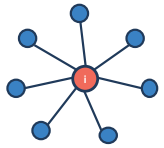
02 GCN fixes a normalized one-hop operator

03 ChebNet learns a K-hop spectral polynomial

04 GraphSAGE samples an inductive neighborhood

COMMON MESSAGE-PASSING LAYER

GCN



Full normalized one-hop neighborhood

fixed structural weights, learned channel mixing

CHEBNET

K polynomial taps

support reaches at most K−1 hops

GRAPHSAGE

Shared aggregator on sampled neighbors

sampling bounds fanout; features support unseen nodes

Start from the graph-learning task, then choose locality, normalization, and sampling.