# Graph Convolutional Neural Network - Derivations & Proofs

Paul F. Roysdon, Ph.D.

## Contents



# 1 Mathematical Derivations & Proofs

## 1.1 Introduction

A Graph Convolutional Neural Network (GCN) generalizes convolution to signals defined on the vertices of a graph. Starting from the graph Laplacian and the Graph Fourier Transform, one derives spectral graph filters and then obtains a localized, linear-time approximation (Chebyshev polynomials). Specializing to a first-order approximation and a renormalized adjacency produces the widely-used propagation rule

$$\mathbf{H}^{(\ell+1)} \;=\; \sigma\big(\hat{\mathbf{A}}\,\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)} \;+\; \mathbf{1}\,(\mathbf{b}^{(\ell)})^{\top}\big), \quad \hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\,\tilde{\mathbf{D}}^{-1/2},\; \tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I},$$

which is *permutation equivariant* and implements *neighborhood averaging* (Laplacian smoothing) followed by a learnable feature mixing. We derive these results, prove locality and permutation equivariance, and provide full backpropagation formulas with all variables and dimensions explicit.

## 1.2 Data and Notation

Let $G = (V, E)$ be an undirected (possibly weighted) graph with $|V| = n$ nodes. Define:

- **Adjacency:** $\mathbf{A} \in \mathbb{R}^{n\times n}$, where $\mathbf{A}_{ij} \geq 0$ is the edge weight (zero if no edge). For self-loops we will use $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$.
- **Degree:** $\mathbf{D} \in \mathbb{R}^{n\times n}$ diagonal, $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$; likewise $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$.
- **(Normalized) Laplacian:** $\mathbf{L} = \mathbf{D} - \mathbf{A}$ and $\mathbf{L}_{\text{sym}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$.
- **Node features:** $\mathbf{X} \in \mathbb{R}^{n\times F_{\text{in}}}$, with $F_{\text{in}}$ input feature channels per node (rows index nodes).

- **Layer representations:** $\mathbf{H}^{(0)} \equiv \mathbf{X}$, $\mathbf{H}^{(\ell)} \in \mathbb{R}^{n \times F_\ell}$, $\ell = 0, \dots, L$.
- **Trainable weights/bias:** $\mathbf{W}^{(\ell)} \in \mathbb{R}^{F_\ell \times F_{\ell+1}}$, $\mathbf{b}^{(\ell)} \in \mathbb{R}^{F_{\ell+1}}$.

We use $\mathbf{1} \in \mathbb{R}^{n \times 1}$ for the all-ones column. Nonlinearities $\sigma(\cdot)$ act elementwise.

## 1.3 Model Formulation: Graph Convolution from Spectral First Principles

**Graph Fourier Transform (GFT).** Since $\mathbf{L}_{\mathrm{sym}}$ is real symmetric, it has an eigendecomposition

$$\mathbf{L}_{\mathrm{sym}} \;=\; \mathbf{U}\,\Lambda\,\mathbf{U}^\top, \qquad \mathbf{U} \in \mathbb{R}^{n \times n} \text{ orthonormal},\ \Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_n),\ \lambda_k \in [0, 2].$$

For a graph signal $\boldsymbol{f} \in \mathbb{R}^n$, the GFT is $\hat{\boldsymbol{f}} = \mathbf{U}^\top \boldsymbol{f}$ and inverse is $\boldsymbol{f} = \mathbf{U}\hat{\boldsymbol{f}}$.

**Spectral graph filtering.** A *spectral filter* with transfer function $g(\cdot)$ acts by

$$g \star \boldsymbol{f} \;=\; \mathbf{U}\,g(\Lambda)\,\mathbf{U}^\top \boldsymbol{f}, \qquad g(\Lambda) = \mathrm{diag}\big(g(\lambda_1), \dots, g(\lambda_n)\big). \tag{1}$$

For matrix-valued features $\mathbf{H} \in \mathbb{R}^{n \times F}$, apply Eqn. (1) columnwise.

**Polynomial filters $\Rightarrow$ spatial locality.** Let $g(\lambda) = \sum_{k=0}^{K} \theta_k\, T_k(\tilde{\lambda})$ be a degree-$K$ polynomial in the (rescaled) eigenvalue $\tilde{\lambda} \in [-1, 1]$ with Chebyshev polynomials $T_k$.[1] Then

$$g(\tilde{\mathbf{L}})\,\mathbf{H} \;=\; \sum_{k=0}^{K} \theta_k\, T_k(\tilde{\mathbf{L}})\,\mathbf{H}, \tag{2}$$

and $T_k(\tilde{\mathbf{L}})$ is a $k$-hop localized operator:

$$\big[T_k(\tilde{\mathbf{L}})\big]_{ij} = 0 \quad \text{whenever the graph distance } \mathrm{dist}(i,j) > k.$$

*Proof.* $T_0(\tilde{\mathbf{L}}) = \mathbf{I}$ and $T_1(\tilde{\mathbf{L}}) = \tilde{\mathbf{L}}$. The recurrence $T_{k+1}(\tilde{\mathbf{L}}) = 2\tilde{\mathbf{L}}T_k(\tilde{\mathbf{L}}) - T_{k-1}(\tilde{\mathbf{L}})$ yields a polynomial in $\tilde{\mathbf{L}}$ of total degree $k{+}1$. As $\tilde{\mathbf{L}}$ is a sparsified version of $\mathbf{L}_{\mathrm{sym}}$ with nonzeros only on edges and self-loops, any degree-$k$ polynomial connects at most $k$ hops. ∎

**First-order (K=1) approximation $\Rightarrow$ GCN propagation.** Take $K = 1$ and expand $g(\tilde{\mathbf{L}}) \approx \theta_0 T_0(\tilde{\mathbf{L}}) + \theta_1 T_1(\tilde{\mathbf{L}}) = \theta_0 \mathbf{I} + \theta_1 \tilde{\mathbf{L}}$. Undoing the rescaling gives (up to constants)

$$g(\mathbf{L}_{\mathrm{sym}})\,\mathbf{H} \;\approx\; \theta_0 \mathbf{I}\,\mathbf{H} \;-\; \theta_1\, \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\,\mathbf{H}. \tag{3}$$

Setting $\theta = \theta_0 = \theta_1$ and adding self-loops $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ (so that the identity term is absorbed into $\tilde{\mathbf{A}}$), one obtains the *renormalized* operator

$$\hat{\mathbf{A}} \;\triangleq\; \tilde{\mathbf{D}}^{-1/2}\,\tilde{\mathbf{A}}\,\tilde{\mathbf{D}}^{-1/2}, \qquad \tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}. \tag{4}$$

Finally, mix feature channels with a trainable matrix $\mathbf{W}$ and bias $\mathbf{b}$:

$$\boxed{\mathbf{Z}^{(\ell)} \;=\; \hat{\mathbf{A}}\,\mathbf{H}^{(\ell)}\,\mathbf{W}^{(\ell)} \;+\; \mathbf{1}\,(\mathbf{b}^{(\ell)})^\top, \qquad \mathbf{H}^{(\ell+1)} \;=\; \sigma\big(\mathbf{Z}^{(\ell)}\big).} \tag{5}$$

Thus a GCN layer performs *neighborhood averaging* (via $\hat{\mathbf{A}}$) followed by a per-node linear map.

[1] Use the rescaled Laplacian $\tilde{\mathbf{L}} = \frac{2}{\lambda_{\max}}\mathbf{L}_{\mathrm{sym}} - \mathbf{I}$ so that its spectrum lies in $[-1, 1]$.

**Interpretation (Laplacian smoothing).** For one feature channel ($F_\ell = 1$) and $\sigma = \mathrm{id}$, $\hat{\mathbf{A}}\mathbf{h}$ is a degree-normalized average of each node's own value and its neighbors', which is a one-step smoothing that reduces the Dirichlet energy $\mathbf{h}^\top \mathbf{L}_{\mathrm{sym}} \mathbf{h}$.

## 1.4 Permutation Equivariance (Graph Isomorphism Invariance)

Let $\mathbf{P} \in \mathbb{R}^{n \times n}$ be a permutation matrix (relabeling of nodes). Under relabeling,

$$\mathbf{A}' = \mathbf{P}\mathbf{A}\mathbf{P}^\top, \quad \tilde{\mathbf{A}}' = \mathbf{P}\tilde{\mathbf{A}}\mathbf{P}^\top, \quad \tilde{\mathbf{D}}' = \mathbf{P}\tilde{\mathbf{D}}\mathbf{P}^\top, \quad \hat{\mathbf{A}}' = \mathbf{P}\hat{\mathbf{A}}\mathbf{P}^\top, \quad \mathbf{H}'^{(\ell)} = \mathbf{P}\mathbf{H}^{(\ell)}.$$

**Claim.** The GCN layer Eqn. (5) is permutation equivariant:

$$\mathbf{H}'^{(\ell+1)} \;=\; \mathbf{P}\,\mathbf{H}^{(\ell+1)}.$$

*Proof.*

$$\mathbf{Z}'^{(\ell)} = \hat{\mathbf{A}}'\mathbf{H}'^{(\ell)}\mathbf{W}^{(\ell)} + \mathbf{1}\,(\mathbf{b}^{(\ell)})^\top = \mathbf{P}\hat{\mathbf{A}}\mathbf{P}^\top\mathbf{P}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)} + \mathbf{1}\,(\mathbf{b}^{(\ell)})^\top = \mathbf{P}\hat{\mathbf{A}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)} + \mathbf{1}\,(\mathbf{b}^{(\ell)})^\top.$$

Since $\mathbf{1}$ is invariant to permutation ($\mathbf{P}^\top\mathbf{1} = \mathbf{1}$), we can write $\mathbf{1}(\mathbf{b}^\top) = \mathbf{P}\,\mathbf{1}(\mathbf{b}^\top)$; hence $\mathbf{Z}'^{(\ell)} = \mathbf{P}\mathbf{Z}^{(\ell)}$ and by elementwise $\sigma$, $\mathbf{H}'^{(\ell+1)} = \mathbf{P}\mathbf{H}^{(\ell+1)}$. ■

## 1.5 Message-Passing View and $k$-Hop Locality

Equation (5) can be written nodewise:

$$\boldsymbol{z}_i^{(\ell)} \;=\; \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\tilde{d}_i\,\tilde{d}_j}}\,\boldsymbol{h}_j^{(\ell)}\,\mathbf{W}^{(\ell)} \;+\; \mathbf{b}^{(\ell)}, \quad \boldsymbol{h}_i^{(\ell+1)} = \sigma\big(\boldsymbol{z}_i^{(\ell)}\big), \tag{6}$$

where $\tilde{d}_i = \tilde{\mathbf{D}}_{ii}$ and $\mathcal{N}(i)$ are neighbors of $i$. Thus a single layer aggregates *1-hop* messages; composing $L$ layers yields $L$-hop receptive fields. More generally, polynomial filters of degree $K$ (Eqn. (2)) are $K$-hop localized.

## 1.6 Training Objective and Semi-Supervised Setting

Let $\mathcal{S} \subseteq \{1, \ldots, n\}$ be the index set of labeled nodes, $K$ classes, and final layer produce logits $\mathbf{R} \in \mathbb{R}^{n \times K}$ with row-softmax $\mathbf{P}$:

$$\mathbf{R} \;=\; \mathbf{H}^{(L)}\mathbf{W}^{(L)} + \mathbf{1}(\mathbf{b}^{(L)})^\top, \qquad \mathbf{P}_{ik} = \frac{\exp(\mathbf{R}_{ik})}{\sum_{t=1}^{K}\exp(\mathbf{R}_{it})}.$$

Using one-hot targets $\mathbf{Y} \in \{0,1\}^{n \times K}$, the cross-entropy loss restricted to $\mathcal{S}$ is

$$\mathcal{L} \;=\; -\sum_{i \in \mathcal{S}}\sum_{k=1}^{K} \mathbf{Y}_{ik}\,\log \mathbf{P}_{ik} \;+\; \sum_{\ell=0}^{L} \frac{\lambda_\ell}{2}\|\mathbf{W}^{(\ell)}\|_F^2 \quad \text{(optional } \ell_2 \text{ regularization)}. \tag{7}$$

## 1.7 Backpropagation Through a GCN Layer

Consider one layer (drop superscript $\ell$) with pre-activation

$$\mathbf{Z} \;=\; \hat{\mathbf{A}}\,\mathbf{H}\,\mathbf{W} \;+\; \mathbf{1}\,\mathbf{b}^\top, \qquad \mathbf{H}_{\mathrm{next}} = \sigma(\mathbf{Z}), \tag{8}$$

and let $\mathbf{G}_{\mathrm{next}} = \partial\mathcal{L}/\partial\mathbf{H}_{\mathrm{next}} \in \mathbb{R}^{n \times F_{\mathrm{out}}}$. Define $\mathbf{G}_Z = \mathbf{G}_{\mathrm{next}} \odot \sigma'(\mathbf{Z})$ (Hadamard product).

**Bias gradient.**

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \Big(\mathbf{G}_Z\Big)^{\top}\mathbf{1} = \sum_{i=1}^{n} \mathbf{G}_Z(i,:) \in \mathbb{R}^{F_{\text{out}}}. \tag{9}$$

**Weight gradient.** Using $\mathbf{Z} = (\hat{\mathbf{A}}\mathbf{H})\mathbf{W} + \mathbf{1}\mathbf{b}^{\top}$,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \big(\hat{\mathbf{A}}\,\mathbf{H}\big)^{\top}\mathbf{G}_Z \in \mathbb{R}^{F_{\text{in}}\times F_{\text{out}}}. \tag{10}$$

If $\hat{\mathbf{A}}$ is symmetric (as in Eqn. (4)), we may also write $\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \mathbf{H}^{\top}\hat{\mathbf{A}}\,\mathbf{G}_Z$.

**Input gradient.**

$$\frac{\partial \mathcal{L}}{\partial \mathbf{H}} = \hat{\mathbf{A}}^{\top}\mathbf{G}_Z\,\mathbf{W}^{\top} = \hat{\mathbf{A}}\,\mathbf{G}_Z\,\mathbf{W}^{\top} \in \mathbb{R}^{n\times F_{\text{in}}}. \tag{11}$$

**(Optional) Gradient w.r.t. edge-weights.** If $\tilde{\mathbf{A}}$ (or $\hat{\mathbf{A}}$) is learnable, use $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{A}}} = \mathbf{G}_Z\,(\mathbf{H}\mathbf{W})^{\top}$ and chain-rule through $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ (*note*: this requires care with the degree normalization).

## 1.8 Computational Aspects

With sparse $\hat{\mathbf{A}}$, one forward layer costs

$$\mathcal{O}(|E|\,F_{\text{in}} + n\,F_{\text{in}}F_{\text{out}})$$

for computing $\hat{\mathbf{A}}\mathbf{H}$ (sparse-dense) and the dense mixing by $\mathbf{W}$. Training time/space scale linearly in $|E|$ for fixed feature widths. Mini-batching on large graphs typically uses neighborhood sampling (e.g., $k$-hop subgraphs) to bound computation.

## 1.9 Algorithm (Semi-Supervised Node Classification with GCN)

1. **Input:** graph $(\mathbf{A}, \mathbf{X})$, labeled-node set $\mathcal{S}$ with one-hot labels $\mathbf{Y}[\mathcal{S},:]$, depth $L$, hidden widths $\{F_\ell\}$, nonlinearity $\sigma$, learning rate $\eta$.
2. **Precompute:** $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$, $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ (sparse).
3. **Initialize:** $\mathbf{H}^{(0)} \leftarrow \mathbf{X}$, parameters $\{\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}\}$.
4. **For** epoch $= 1, \ldots$ until convergence:
   (a) **Forward:** For $\ell = 0, \ldots, L-1$, compute $\mathbf{Z}^{(\ell)} = \hat{\mathbf{A}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)} + \mathbf{1}(\mathbf{b}^{(\ell)})^{\top}$, $\mathbf{H}^{(\ell+1)} = \sigma(\mathbf{Z}^{(\ell)})$. For the final layer, apply softmax to logits.
   (b) **Loss:** Evaluate $\mathcal{L}$ in Eqn. (7) on $\mathcal{S}$.
   (c) **Backward:** Backpropagate using Eqns. (9)–(11).
   (d) **Update:** Apply an optimizer (SGD/Adam) to all $\{\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}\}$.

## 1.10 Proofs and Identities

**From spectral to spatial (locality).** *Proof.* A degree-$K$ polynomial $g(\mathbf{L}_{\text{sym}}) = \sum_{k=0}^{K} \theta_k \mathbf{L}_{\text{sym}}^k$ only mixes features along paths of length $\leq K$; therefore $g(\mathbf{L}_{\text{sym}})\mathbf{H}$ is $K$-hop localized. Combining with feature mixing $\mathbf{W}$ preserves locality. ■

**Renormalization trick.** *Proof.* Starting from Eqn. (3), absorbing the identity with self-loops $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and re-normalizing to $\hat{\mathbf{A}}$ in Eqn. (4) stabilizes the spectrum of the propagator ($\|\hat{\mathbf{A}}\|_2 \le 1$), mitigating exploding/vanishing over multiple layers. ∎

**GCN as neighborhood averaging + linear map.** *Proof.* Equation (6) shows $\hat{\mathbf{A}}$ performs a symmetric, degree-weighted average of neighbor features, which is exactly Laplacian smoothing. The learnable $\mathbf{W}$ then recombines channels; $\sigma$ adds nonlinearity. ∎

### 1.11 Extensions (Brief)

- **Chebyshev GCN (ChebNet).** Use Eqn. (2) with $K > 1$ to obtain $K$-hop localized filters without eigen-decomposition; compute $T_k(\tilde{\mathbf{L}})\mathbf{H}$ via the three-term recurrence.
- **Edge weights/directions.** For weighted graphs, $\mathbf{A}$ carries weights; for directed graphs, one may use symmetrization or separate in/out normalizations.
- **Dropout and residuals.** Apply dropout to $\mathbf{H}^{(\ell)}$ or edges (DropEdge) and add residual/skip connections to alleviate over-smoothing.

### 1.12 Summary of Variables and Their Dimensions

- $\mathbf{A} \in \mathbb{R}^{n\times n}$: adjacency (symmetric, nonnegative entries). $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ includes self-loops.
- $\mathbf{D}, \tilde{\mathbf{D}} \in \mathbb{R}^{n\times n}$: degree diagonals; $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$.
- $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2} \in \mathbb{R}^{n\times n}$: renormalized propagator.
- $\mathbf{L}_{\text{sym}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \in \mathbb{R}^{n\times n}$: normalized Laplacian.
- $\mathbf{X} = \mathbf{H}^{(0)} \in \mathbb{R}^{n\times F_0}$: input features; $\mathbf{H}^{(\ell)} \in \mathbb{R}^{n\times F_\ell}$ hidden representations.
- $\mathbf{W}^{(\ell)} \in \mathbb{R}^{F_\ell \times F_{\ell+1}}$, $\mathbf{b}^{(\ell)} \in \mathbb{R}^{F_{\ell+1}}$: layer parameters.
- $\mathbf{Z}^{(\ell)} \in \mathbb{R}^{n\times F_{\ell+1}}$: pre-activations; $\mathbf{H}^{(\ell+1)} = \sigma(\mathbf{Z}^{(\ell)})$.
- $\mathbf{R} \in \mathbb{R}^{n\times K}$: final logits; $\mathbf{P} \in \mathbb{R}^{n\times K}$: softmax probabilities.
- $\mathcal{S} \subseteq \{1, \ldots, n\}$: labeled-node index set.
- Gradients: $\partial\mathcal{L}/\partial\mathbf{b} \in \mathbb{R}^{F_{\text{out}}}$ (Eqn. (9)), $\partial\mathcal{L}/\partial\mathbf{W} \in \mathbb{R}^{F_{\text{in}}\times F_{\text{out}}}$ (Eqn. (10)), $\partial\mathcal{L}/\partial\mathbf{H} \in \mathbb{R}^{n\times F_{\text{in}}}$ (Eqn. (11)).

### 1.13 Summary

From first principles: define convolution on graphs via the Laplacian eigenbasis and spectral filtering Eqn. (1); enforce *locality* with polynomial filters Eqn. (2); specialize to a first-order approximation and absorb the identity with self-loops to obtain the *renormalized* propagator $\hat{\mathbf{A}}$ Eqn. (4); compose with a learnable feature mixer to yield the GCN layer Eqn. (5). The layer is permutation equivariant, $k$-hop localized, and implements neighborhood averaging (Laplacian smoothing). We provided full backpropagation Eqns. (9)–(11), training objective Eqn. (7), algorithmic steps, and variable dimensions for a complete, implementation-ready derivation.