Semi-Supervised Classification with Graph Convolutional Networks

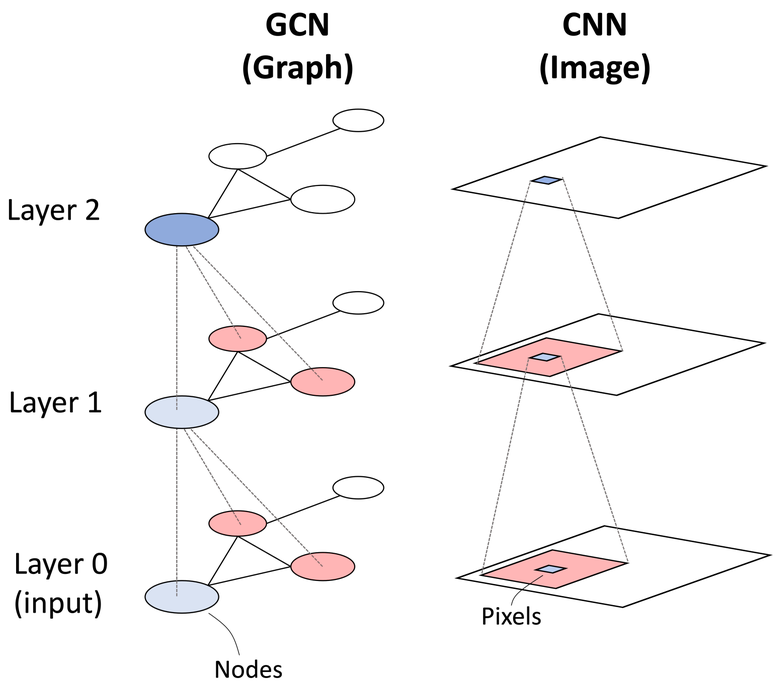
Kipf and Welling
University of Amsterdam

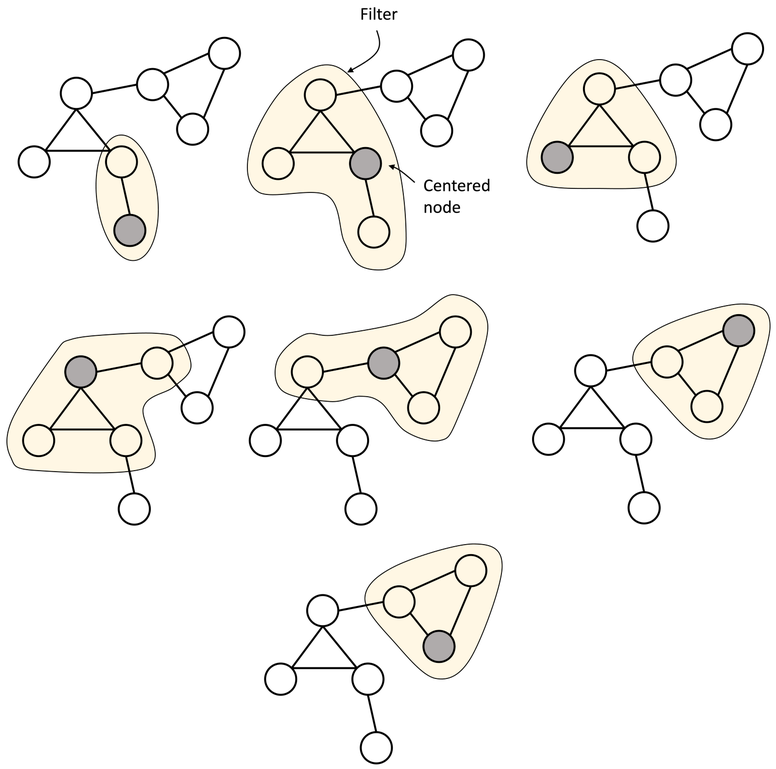
Sidharrth Nagappan (sn666)
University of Cambridge

How to perform semi-supervised learning on graph-structured data using both **node features** and **graph topology**?

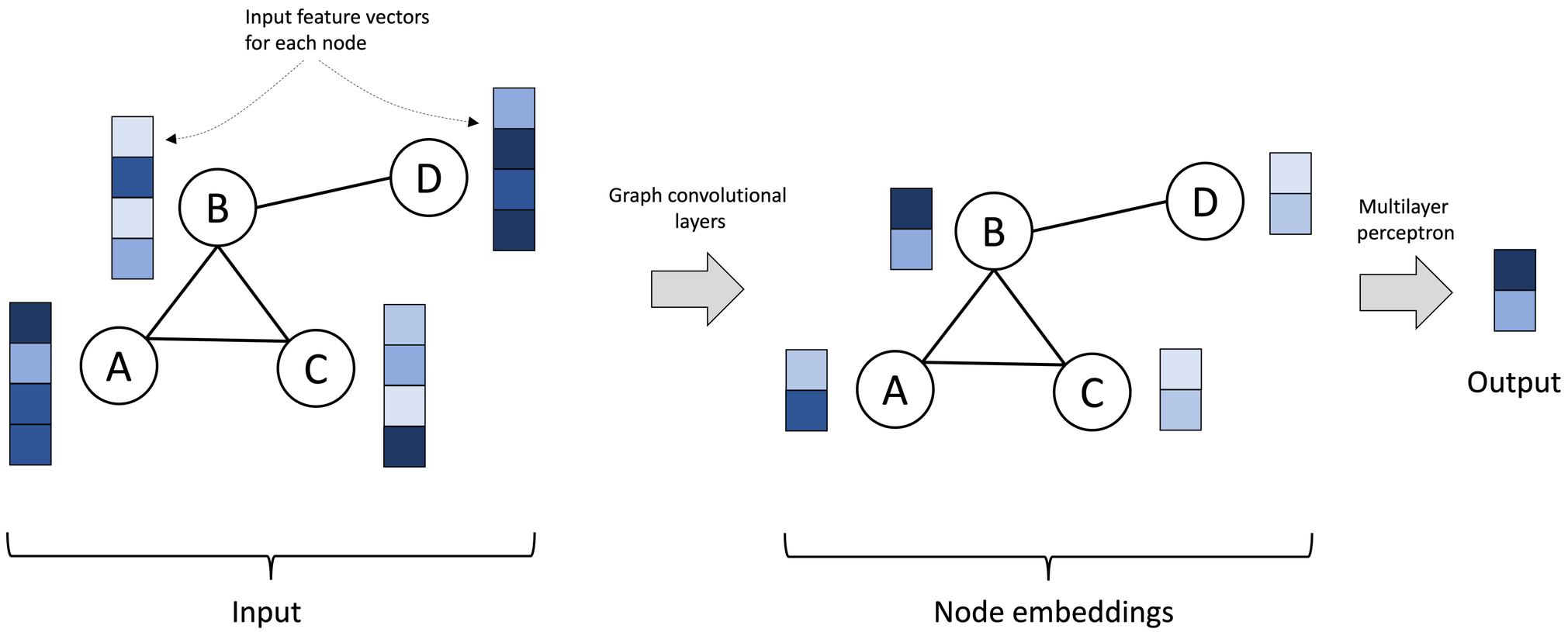
CNNs: Performant on grid-like data (e.g., images) by applying convolutional filters to capture local patterns.

GCNs: Extend this concept to graphs by defining convolution-like operations that aggregate information from a node's neighbors.





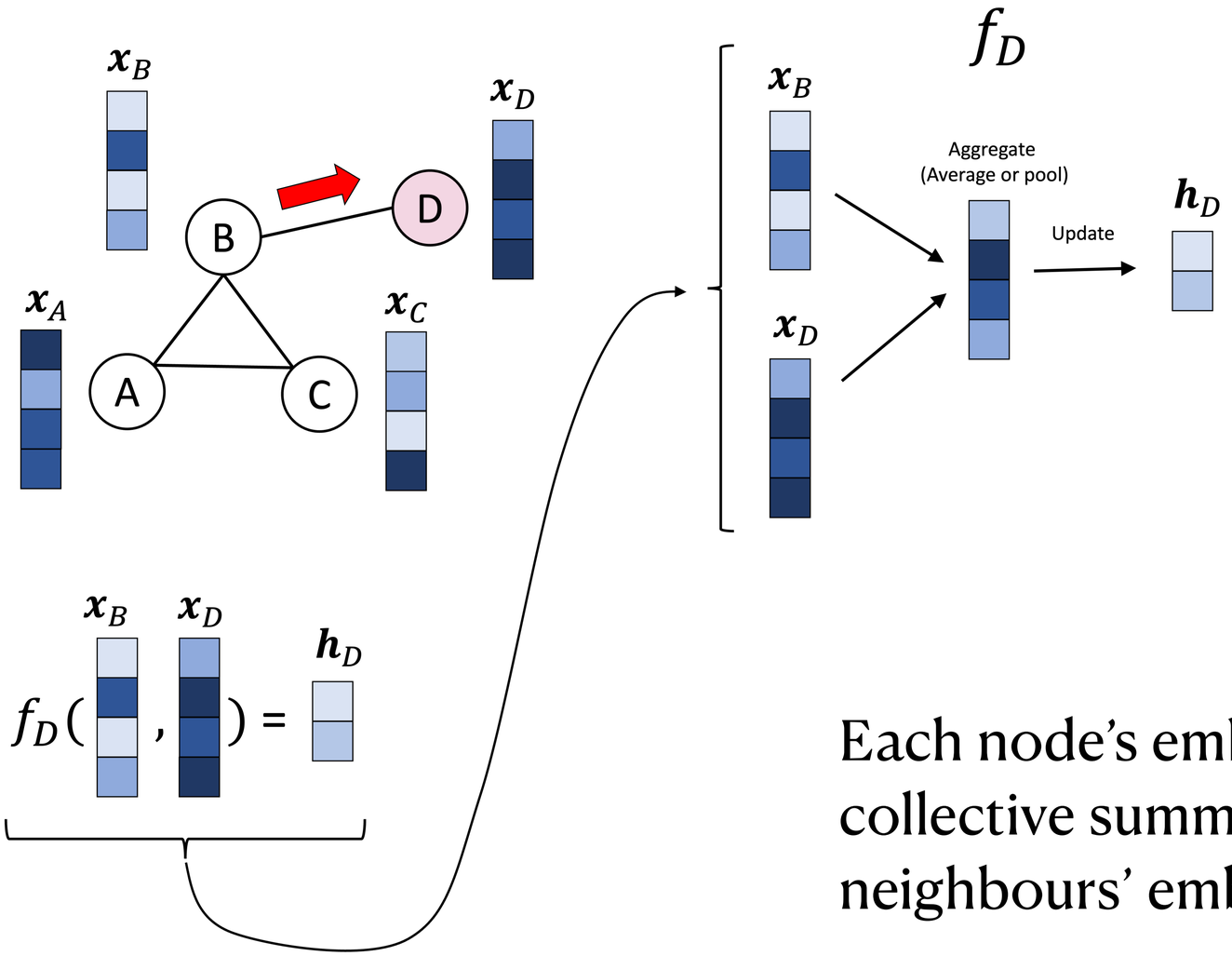
High-Level Graph Neural Network



$G = (V, E)$

Graph made up of
Vertices and Edges

Message Passing



Each node's embeddings is a collective summary of it's neighbours' embeddings.

Spectral and Spatial Graph Theory

Spectral Methods

Operate in the frequency domain

Leverage eigenvalues and eigenvectors of Graph Laplacian

Not localised unless specifically designed

Spatial Methods

Directly leverage graph's structure

Aggregate + Transform information from node neighbourhoods

Naturally local

This Paper

Started from spectral methods

Used first-order approximation of spectral convolution to "resemble" spatial method

Evolution of Graph Convolutions

Bruna et al. (2013) introduced Spectral Graph Convolutions using **Graph Fourier Transforms**

$O(N^3)$ where N is the number of nodes because of expensive Eigen-decomposition

↓

Defferrard et al. (2016) used **Chebyshev polynomials** to approximate spectral filters

O(K|E) — no need Eigen decomposition

↓

This paper **Kipf & Welling (2017)** set K=1

First-order approximation

O(1|E) — only consider immediate neighbours

By bridging spectral (operate in frequency domain) and spatial (operate directly on graph), this paper provides way to run direct convolutions without eigenvector computations.

Scales linearly with number of graph edges/nodes

Mathematical Formulation

$$f(X, A) = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}XW\right)$$

Scary GCN equation - let's break it down quickly

$\tilde{A} = A + I_n$ Adjacency matrix with self-loops

$\tilde{D}$ Degree matrix of A

X Input feature matrix

W Weight matrix

σ Non-linear activation function

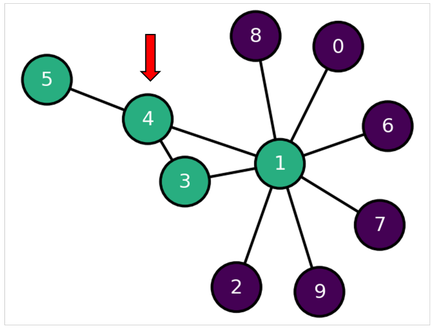
Bear with me, let’s break it down

Why Normalise the Adjacency Matrix?

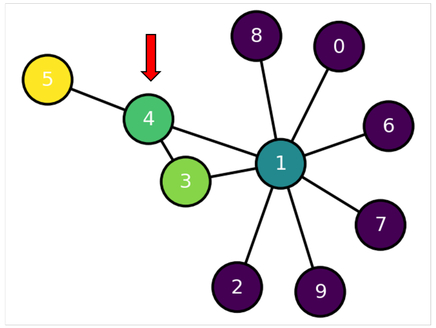
If we aggregate information from neighbours without normalisation, nodes with more neighbours will have HUGE feature values.

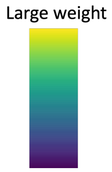
Numerical instability/gradient issues and explosion

$\boldsymbol{D^{-1}\widehat{A}}$



$\boldsymbol{D^{-1/2}\widehat{A}D^{-1/2}}$





Normalisation

Adjacency Matrix

$A_{ij} = 1$
$A_{ij} = 0$

Records link between nodes *i* and *j*.

Degree Matrix

$$D = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Each entry records degree of node *i*.

Proposed Normalisation

$A + I$ Add self-loops so each node is connected to itself and retains it's own features as well

$D^{-\frac{1}{2}}$ Each node I, scale down contribution from neighbours proportionally to degree of node and neighbours of node

$$\tilde{A} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}$$

Symmetrically multiplied on both sides:

Left-multiplying - scales down contribution based on node receiving information

Right-multiplying - scales down node influence based on it's own degree

Let's work through the maths

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

$$D = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \longrightarrow D^{-\frac{1}{2}} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\tilde{A} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

Stacking GCN Layers

First Layer

Aggregate information from immediate neighbours

Second Layer

Aggregate information from neighbours' neighbours

***L* Layer**

Aggregate information from nodes up to ***L*** steps away

Widening Receptive Field

Stacking layers results in successive multiplications by $\tilde{A}$

$$Z = \text{softmax}(\tilde{A}\,\sigma(\tilde{A}XW^{(0)})W^{(1)})$$

Equation of a 2-layer GCN

Results

Experimentation

GCNs tested on:

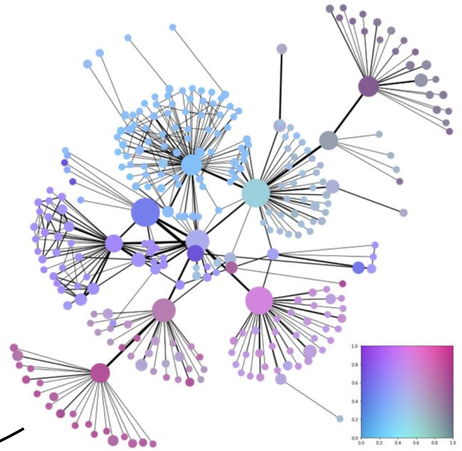
- Semi-supervised document **classification** in **citation networks**
- Entity classification in bipartite knowledge graphs
- Different graph propagation models
  - Recall the evolution of graph convolutions from earlier

Citeseer

Nodes - documents

Edges - citations

Only 3.6% of the nodes are labelled



Algorithms Outperformed

Label Propagation (LP)

DeepWalk

Planetoid

Semi-supervised Embedding (SemiEmb)

Iterative Classification Algorithm (ICA)

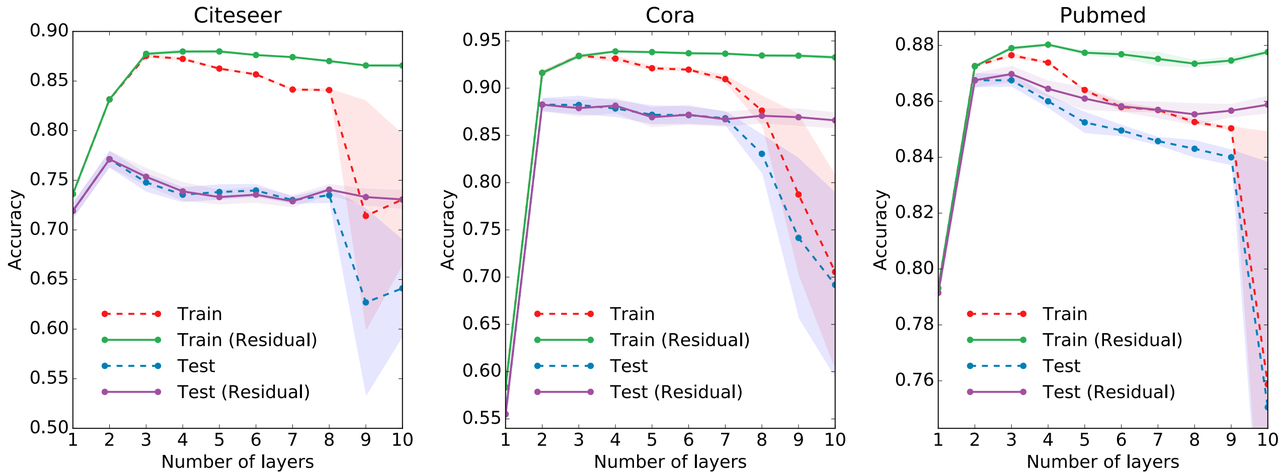
Method	**Citeseer**	**Cora**	**Pubmed**	**NELL**
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

Comparing Propagation Models

			Performance on Citeseer
Make spectral convolution feasible for large graphs by using polynomials instead of full Eigen-decomposition of Laplacian	$\sum_{k=0}^{K} T_k(\tilde{L})X\Theta_k$	Chebyshev Filter with different values of K	69.8%
First-order approximation of Chebyshev Does not capture higher-order neighbourhood information	$X\Theta_0 + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta_1$	1st Order Model	68.3%
Use single parameter for the entire layer But treats self-loops and neighbours contribution uniformly	$\left(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right)X\Theta$	Single Parameter Model	69.3%
Add self-loops + symmetric normalisation Balance simplicity and efficiency	$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$	**Final Propagation Model**	**70%**

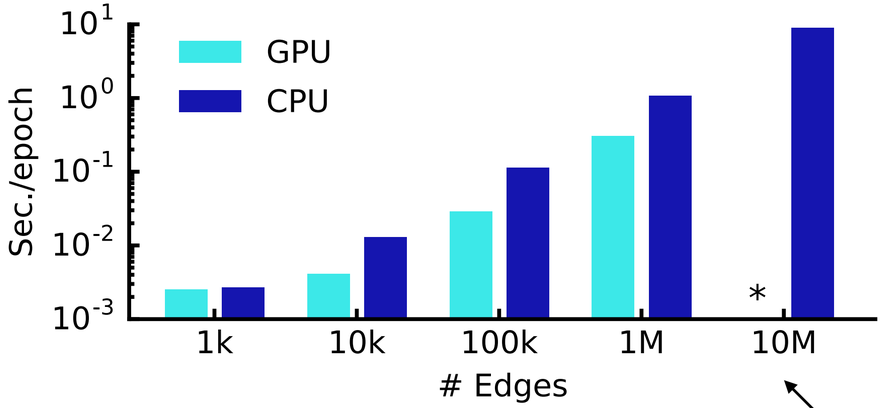
Spectral convolutions without computational overhead

Depth and Graph Scaling



Optimal between 2~3 layers

Not much benefit when increasing > 7 layers



Scaling is decent

Out of Memory!

Limitations, Thoughts and Future Work

Memory Requirement

Full-batch gradient descent requires entire graph in memory

Does not scale for large graphs

→ **Implement Mini-Batch Training**

Directed Graphs

GCNs in paper for undirected graphs only

→ **Support Directed and Weighted Edges**

Betting on Locality

Betting on local neighbourhoods being sufficient to learn node representations

Not suitable for long range dependencies

→ **Adaptive Neighbourhood Selection**

Deeper Models Overfitting

Deeper models suffer from numerical instability

Experiments show GCN best in shallow settings with less layers (like 2)

→ **Residual Connections to stabilise gradients**

Laying the Groundwork + My Thoughts

Good:

- Incredibly seminal paper with **38,333 citations**
- **Open-Source Code** available, unlike past theoretical work
- Laid the groundwork for various spatial methods later.
  - **GraphSAGE** - Hamilton et al. (2017) - aggregate information from local neighbours via mean/LSTM
  - **Relational GCNs** - Schlichtkrull et al. (2017)
  - **Graph Attention Networks (GATs)** - Velickovic et al. (2018) - attention mechanism used in neighbourhood aggregation, adaptive weighting

---

Bad *(minor though)*:

- Computational complexity reduction could have been made clearer
- Symmetric normalisation not as clearly explained — I needed to read several other articles to figure it out

Layer number comparison “conveniently” left in appendix

Graph Convolutional Networks (GCN) by Kipf and Welling **Spectral**

Graph Attention Networks (GAT) **Spatial**

Chebyshev Networks by Defferrard et al. **Spectral**

GraphSAGE **Spatial**

Graph Isomorphism Networks (GIN) **Spatial**

Spectral or Spatial Algorithms?

Thank you, questions?