SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

ICLR 2017 February 22, 2017

Thomas N.Kipf, University of Amsterdam

Max Welling, University of Amsterdam

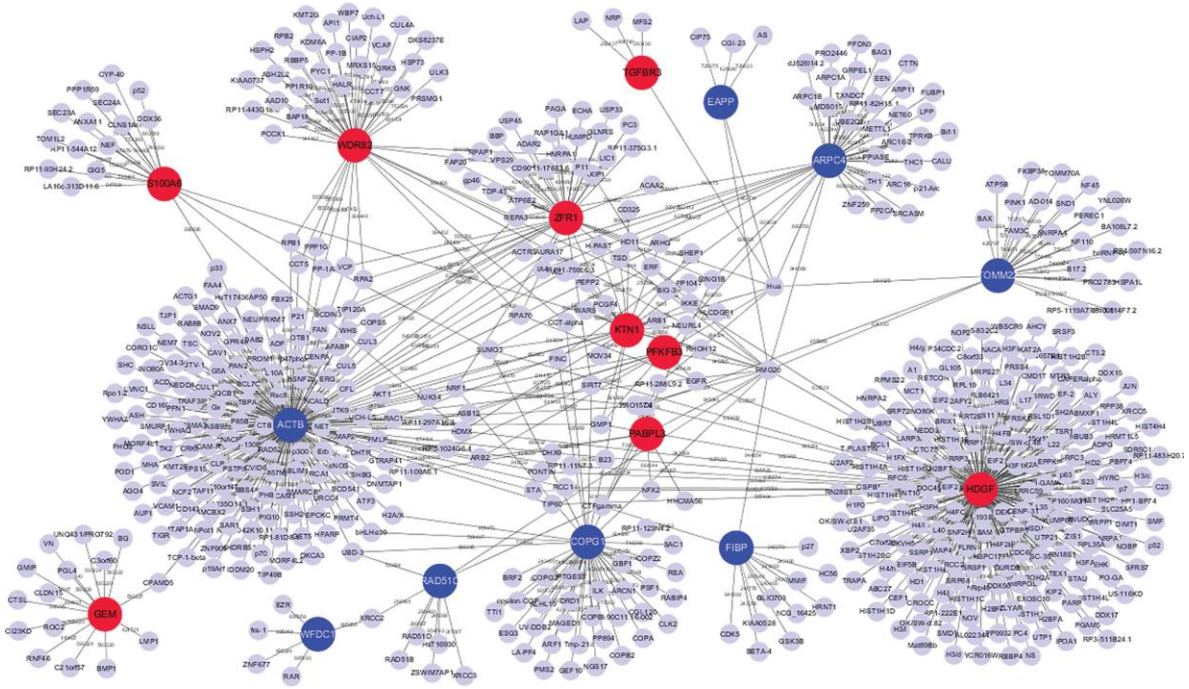
Slides Presented by Fatir Qureshi

Introduction

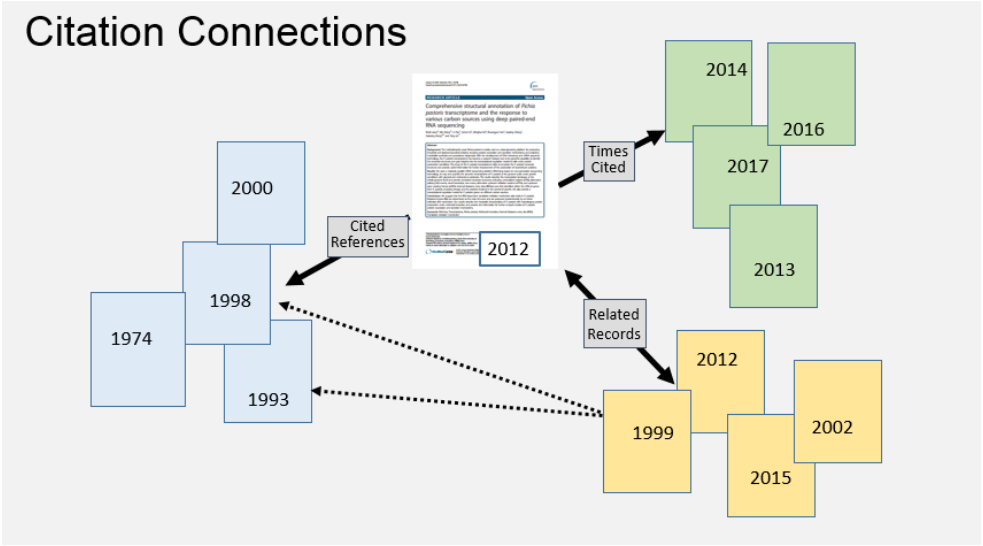
- Background
- Motivation
- Fast approximate convolutions on graphs
- Semi-supervised node classification
- Experiments
- Conclusions

Background

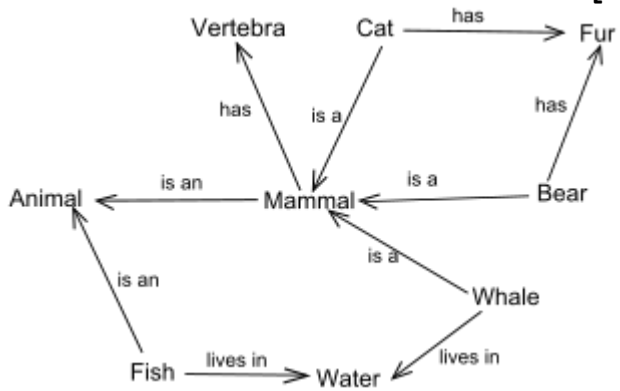
Graph structured data is a common model for visualizing many diverse types of information



Protein-Protein Interactions [1]



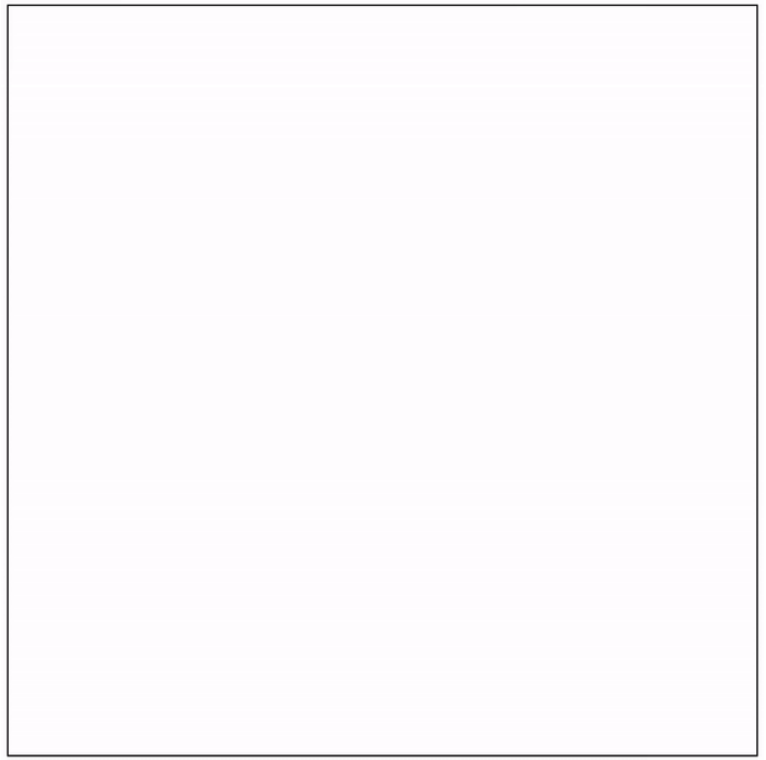
Citation network [3]



Semantic [2]

Background

- A large number of approaches for semi-supervised learning using graph representations were proposed prior to the time of this paper, most of which fell into two broad categories
  - Methods that use some form of explicit graph Laplacian regularization
  - Graph embedding-based approaches
- Developing a neural network based propagation rule on graphs has had a number of hurdles



Motivation

- Classification of unlabeled nodes in a network is a common problem with working with a graphical representation of data
  - Identifying relationships between individuals in a social network
  - Determining categories for unlabeled files

- This problem is approached as a semi-supervised learning problem and commonly was attempted by smoothing label information over graphs via graph-based regularization

  - e.g Laplacian regularization term

Motivation

Laplacian regularization term in the loss function commonly used for label smoothening

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}}\,, \quad \text{with} \quad \mathcal{L}_{\text{reg}} = \sum_{i,j} A_{ij} \|f(X_i) - f(X_j)\|^2 = f(X)^\top \Delta f(X)\,.$$

Here,L0 denotes the supervised loss with respect to the labeled part of the graph,f(·) can be a neural network like differentiable function, λ is a weighing factor and X is a matrix of node feature vectors Xi. Δ = D − A denotes the unnormalized graph Laplacian of an undirected graph G = (V,E)

Motivation

- This approach assumes that connected nodes are likely to share the same label, which is not always the case.
- Graph edges need not necessarily encode node similarity, but could contain additional information
- Subsequently, the proposal of this paper is to encode the graph structure directly using a neural network model f(X, A) and train on a supervised target L0 for all nodes with labels, thereby avoiding explicit graph-based regularization in the loss function

Graph Convolutional Networks

- Introduce a simple and well-behaved layer-wise propagation rule for neural network models which operate directly on graphs and show how it can be motivated from a first-order approximation of spectral graph convolutions
- Demonstrate how this form of a graph-based neural network model can be used for fast and scalable semi-supervised classification of nodes in a graph

Graph Convolutional Networks

For these models, the goal is to learn a function of signals/features on a graph G=(V,E) which takes as input:

- A feature description *xi* for every node i; summarized in a *N*×*D* feature matrix *X* (*N*: number of nodes, *D*: number of input features)
- A representative description of the graph structure in matrix form; typically in the form of an adjacency matrix *A* (or some function thereof)

Output of this model is a NxF matrix where F is the features per node and N is the nodes

Graph Convolutional Networks

- Basic propagation rule of all GCNs layers can be expressed as:

$$H^{(l+1)} = f(H^{(l)}, A) \quad \text{with } H^{(0)} = X$$

- For this paper, the following layer-wise propagation rule was investigated

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

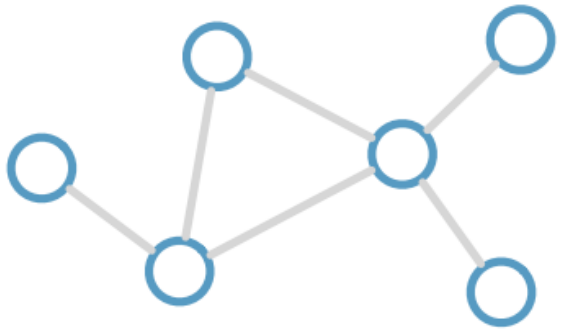
Graph Convolutional Networks

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

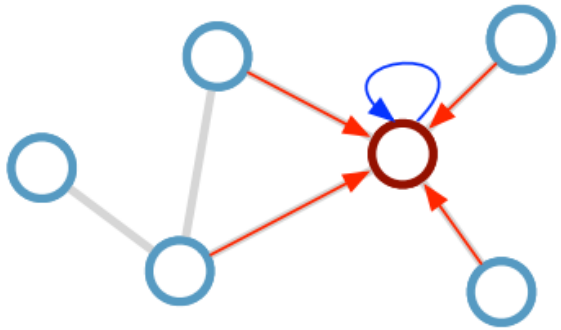
- $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph G with added self-connections
- $\tilde{D}_{ii} = \sum_j \tilde{A}_i$ and $W^{(l)}$ is a layer-specific trainable weight matrix
- σ(·) denotes an activation function, such as the ReLU(·) = max(0, ·)
- $H^{(l)}$ is the matrix of activations in the *l*th layer

Graph Convolutional Networks

Consider this undirected graph:



Calculate update for node in red:



Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

Spectral Graph Convolutions

- We consider spectral convolutions on graphs defined as the multiplication of a signal x (a scalar for every node) with a filter gθ = diag(θ)

$$g_\theta \star x = U g_\theta U^\top x$$

,where U is the matrix of eigenvectors of the normalized graph Laplacian (L= $U \Lambda U^\top$)

- gθ is thus a function of the eigenvalues of the Laplacian L, i.e. gθ(Λ)
- Evaluating this equation is computationally expensive, as multiplication with the eigenvector matrix U is a $O(N^2)$ problem

Spectral Graph Convolutions

- To reduce this complexity, it was suggested in Hammond et al. (2011) that gθ(Λ) can be well-approximated by a truncated expansion in terms of Chebyshev polynomials Tk(x)

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda})$$

Going back to our definition of a convolution of a signal x with a filter gθ’:

$$g_{\theta'} \star x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) x$$

Spectral Graph Convolutions

- Neural network model based on graph convolutions can be built by stacking multiple convolutional layers of the form:

$$g_{\theta'} \star x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) x$$

$\lambda_{\max} \approx 2$

$$g_{\theta'} \star x \approx \theta'_0 x + \theta'_1 (L - I_N) x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$$

$$I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

- Renormalization trick applied:

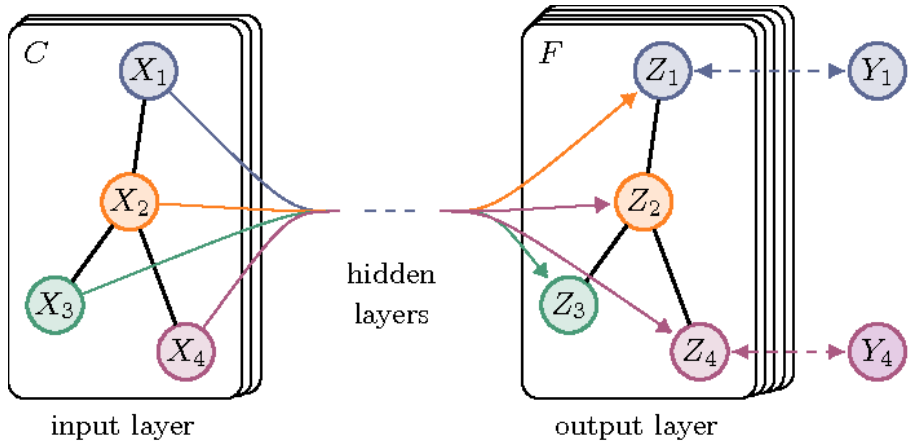
$$\tilde{A} = A + I_N \text{ and } \tilde{D}_{ii} = \sum_j \tilde{A}_{ij}.$$

$$I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$

$$g_\theta \star x \approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x$$

Semi-supervised Node Classification

f(X, A) provides an effective means for information propagation on graphs



(a) Graph Convolutional Network

(b) Hidden layer activations

Schematic depiction of multi-layer Graph Convolutional Network (GCN) for semi-supervised learning with C input channels and F feature maps in the output layer. The graph structure (edges shown as black lines) is shared over layers, labels are denoted by Yi .

Experimental Testing

The GCN proposed were evaluated on document classification in citation networks with low label rates

Experimental setup was based on approach taken by Yang et al. (2016), which involved the assessment of Citeseer, Cora and Pubmed

Two layer GCN was trained on 1000 labeled examples from each database and was applied to a validation set of 500 examples

Table 1: Dataset statistics, as reported in Yang et al. (2016).

Dataset	Type	Nodes	Edges	Classes	Features	Label rate
Citeseer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
Pubmed	Citation network	19,717	44,338	3	500	0.003
NELL	Knowledge graph	65,755	266,144	210	5,414	0.001

Experimental Results

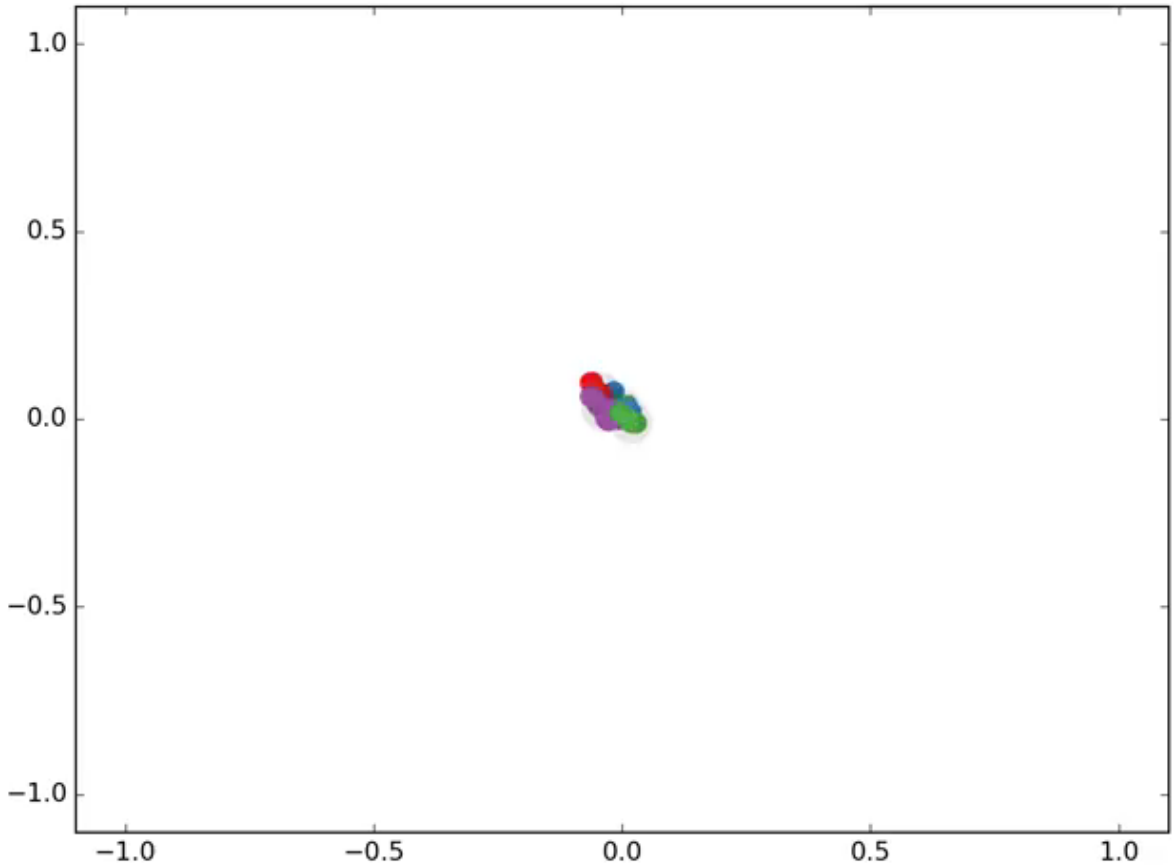
Method	**Citeseer**	**Cora**	**Pubmed**	**NELL**
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (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

Experiments on Model Depth

- Different variations of the per-layer propagation model was tested on all three of the citation networks to determine their respective ability to label documents
- Renormalization trick was compared to Chebyshev filter model, 1st order model and single parameter model

Description		Propagation model	Citeseer	Cora	Pubmed
Chebyshev filter (Eq. 5)	$K = 3$	$\sum_{k=0}^{K} T_k(\tilde{L})X\Theta_k$	69.8	79.5	74.4
	$K = 2$		69.6	81.2	73.8
1st-order model (Eq. 6)		$X\Theta_0 + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta_1$	68.3	80.0	77.5
Single parameter (Eq. 7)		$(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})X\Theta$	69.3	79.2	77.4
Renormalization trick (Eq. 8)		$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$	**70.3**	**81.5**	**79.0**
1st-order term only		$D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta$	68.7	80.5	77.8
Multi-layer perceptron		$X\Theta$	46.5	55.1	71.4

Social Network Example

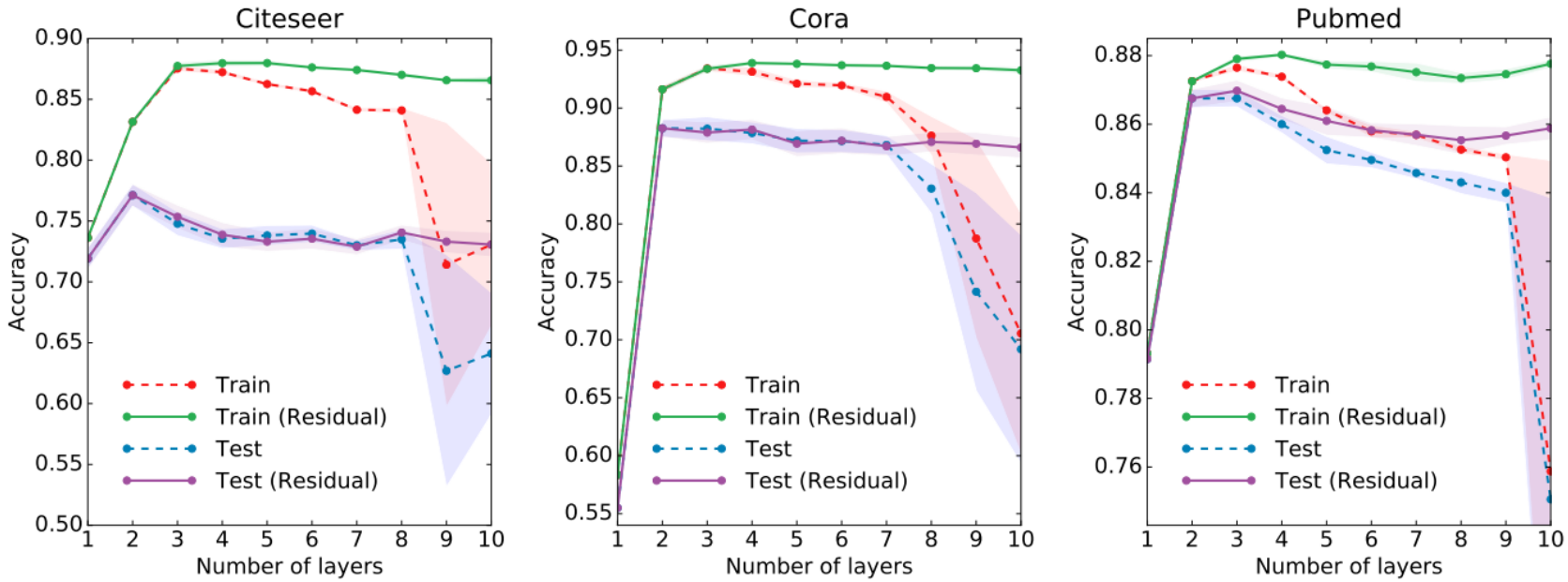


Experiments on Model Depth

- 5-fold cross-validation experiment on the Cora, Citeseer and Pubmed datasets was compared between models with different number of layers
- Basic GCN propagation was compared to model variant where residual connections between hidden layers facilitate training of deeper models by enabling the model to carry over information from the previous layer's input

Experiments on Model Depth

- The best results are obtained with a 2- or 3-layer model
- Models deeper than 7 layers become less accurate without residual connections



Limitations

- Memory requirements are an issue as it grows linearly with the size of the dataset
- Directed edges and edge feature are not naturally supported by the GCN framework presented
- Limiting assumptions regarding locality

Conclusions

- GCN model uses an efficient layer-wise propagation rule that is based on a first-order approximation of spectral convolutions on graphs
- Demonstrated viability by outperforming contemporary methods for semi-supervised graph node classification
- Learning on directed or relational graphs may be potential area of extension of this work

Questions ?