Fusion Graph Convolutional Networks

Priyesh Vijayan
Robert Bosch Centre for Data Science and AI
Indian Institute of Technology Madras
priyesh@cse.iitm.ac.in

Yash Chandak
University of Massachusetts Amherst
ychandak@cs.umass.edu

Mitesh Khapra
Robert Bosch Centre for Data Science and AI
Indian Institute of Technology Madras
miteshk@cse.iitm.ac.in

Balaraman Ravindran
Robert Bosch Centre for Data Science and AI
Indian Institute of Technology Madras
ravi@cse.iitm.ac.in

ABSTRACT

Semi-supervised node classification involves learning to classify unlabelled nodes given a partially labeled graph. In transductive learning, all unlabelled nodes to be classified are observed during training and in inductive learning, the goal is to predict labels for nodes are not seen at training. In this paper, we focus on both the settings for node classification on attributed graphs, i.e., graphs in which nodes have additional features. State-of-the-art models for node classification on such attributed graphs use differentiable recursive functions. These differentiable recursive functions enable aggregation and filtering of neighborhood information from multiple hops (depths). These end-to-end learnable multi-hop graph functions primarily vary in their neighborhood aggregation functions as well as in the form of the weighted combinations of the node and its neighbors' information at each hop. Despite being powerful, these variants are limited in their ability to combine information from different hops efficiently. In this work, we analyze this limitation of recursive graph functions in effectively capturing multi-hop neighborhood information. Further, we provide a fusion component which is mathematically well motivated to address this limitation and improve the existing models to explicitly learn the importance of information from different hops. This proposed mechanism is shown to improve over existing methods across 8 popular datasets from different domains. Specifically, our model improves the Graph Convolutional Network (GCN) and a variant of Graph SAGE by a significant margin providing highly competitive state-of-the-art results.

ACM Reference Format:
Priyesh Vijayan, Yash Chandak, Mitesh Khapra, and Balaraman Ravindran. 2018. Fusion Graph Convolutional Networks. In *Proceedings of Mining and Learning with Graphs (MLG'18).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Many real-life datasets have an underlying graph structure, where along with the attributes of an entity a rich information is contained in the relationships between them. Such datasets are widely popular across various fields of science and engineering, like genomics, computer vision, social networks, neuroscience, molecular chemistry etc. Semantically categorizing such entities requires optimally extracting information from it's multi-hop neighborhood and combining it efficiently with its own features. Thus defining and finding the significance of neighborhood information over multiple hops becomes an important aspect of the problem.

Recent deep learning approaches that attempt to learn relational representation of nodes can be broadly categorized based on their usage of structural information into methods that either learn to aggregate neighbors' information [9], [11] or into methods that learn node representation with additional structural regularization constraints [12], [14], [16]. The latter is limited to work only in networks which exhibit high homophily, as they enforce the representation of a node to be similar to its neighboring nodes. Former methods avoid any such explicit assumption on homophily of the network data.

Various attempts [4], [9] have been made to generalize the famous convolutional neural networks to be invariant to spatial transformations such that they work on domains that have an inherent graphical structure. But, an unrestricted number of neighbors makes the problem more challenging. Graphs have widely varying network topology within and across different domain. Transfer or Inductive inference tasks on such graphs requires robust models which are invariant to such changes.

To deal with the graph's topological structure, [2] defined convolutional operations in the spectral domain for graph classification tasks, but required computationally expensive eigendecomposition of the graph Laplacian. To reduce this requirement, [4] approximated the higher order relational feature computation with first order Chebyshev polynomials defined on the graph Laplacian. Graph Convolutional Networks (GCNs) [9] adapted them to semi-supervised node level classification tasks. GCN simplified Chebyshev Nets by recursively convolving one-hop neighborhood information with a symmetric graph Laplacian. Recently, [6] proposed a generic framework called GraphSAGE with multiple neighborhood aggregator functions. GraphSAGE works with a partial (fixed number) neighborhood of nodes to scale to large graphs. GCN and GraphSAGE are the current state-of-the-art approaches for transductive and inductive node classification tasks in graphs with node features. These end-to-end differentiable methods provide impressive results besides being efficient in terms of memory and computational requirements.

Despite the impressive results, these models cannot efficiently summarize relevant information from multi-hop neighborhood as they cannot flexibly regulate the importance of information from

MLG'18, Aug'18, London, UK

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

different hops. Herein, we analyze the limitations of these models and provide a solution to the same. The proposed solution significantly boosts the performance of GCN and GraphSAGE model to achieve new state-of-the-art results.

Below, we list out our contributions:

- We show that the current state-of-the-art graph convolutional models capture K-hop higher order neighborhood information by a K^{th} order binomial.
- On analysis, we show that this binomial formulation assigns a prior on the importance of different hops and enforces a recursive weight dependency which restricts the model from independently regulating information from different hops.
- We overcome this problem by introducing a minimal fusion component, which can flexibly regulate information from individual hops by learning to linearly combine the different binomial bases corresponding to different hops.
- We present our results on 8 datasets from different domains. Our proposed model F-GCN (GCN with fusion), outperforms the previous state of the art models on 6 datasets and hence provides the best overall results.

2 BACKGROUND

2.1 Notations

Let $G = (V, E)$ denote a graph comprising of vertices, V, and edges, E with $|V| = N$ respectively. Let $X \in \mathbb{R}^{N\times F}$ denote the nodes' features and $Y \in \mathbb{B}^{N\times L}$ denote the nodes' labels with F and L referring to the number of features and labels, respectively. Let $A \in \mathbb{R}^{N\times N}$ denote the adjacency matrix representation of the set of edges, E and let $D \in \mathbb{R}^{N\times N}$ denote the diagonal degree matrix defined as $D_{ii} = \sum_j A_{i,j}$. Let $L = I - D^{-\frac{1}{2}}(A)D^{-\frac{1}{2}}$ denote the normalized graph Laplacian and $\hat{L} = (D+I)^{-\frac{1}{2}}(A+I)(D+I)^{-\frac{1}{2}}$ denote the re-normalized Laplacian [9].

In this paper, a Graph Convolutional Network defined to capture K-hop information will have K graph convolutional layers with d dimensional outputs, h_k and an final label layer denoted by h_L. $h_L = h_K$ if the last convolution is considered as the label layer otherwise, $h_L = h_{K+1}$. Let W_k denote the weights associated with the layer, k where $W_1 \in \mathbb{R}^{F\times d}$ is the first hidden layer's weights, $W_L \in \mathbb{R}^{d\times L}$ is the label layer's weights and $W_k \in \mathbb{R}^{d\times d}$ is the intermediate layer's weights. Let σ_k define the activation function associated with layer, k.

2.2 Graph Convolutional Networks

Graph Convolutional Network (GCN), introduced in [9], is a multi-layer convolutional neural network where the convolutions are defined on a graph structure for the problem of semi-supervised node classification. The conventional two-layer GCN which captures information up to the 2nd hop neighborhood of a node can be reformulated to capture information up to any arbitrary hop, K as given below in Eqn: 1.

$$\begin{aligned} h_0 &= X \\ h_k &= \sigma_k(\hat{L}h_{k-1}W_k), \quad \forall\, k \in [1, K-1]. \\ Y &= \sigma_K(\hat{L}h_{K-1}W_L) \end{aligned} \tag{1}$$

GCN was used for multi-class classification task with RELU activation function, σ_k = ReLU $\forall k \in [1, K-1]$ and a softmax label layer, $\sigma_K = softmax$. We can rewrite the GCN model in terms of $(K-1)^{\text{th}}$ hop node and neighbor features as below by factoring $\hat{L}$.

$$\begin{aligned} h_k &= \sigma((\hat{D}^{-\frac{1}{2}}I\hat{D}^{-\frac{1}{2}} + \hat{D}^{-\frac{1}{2}}A\hat{D}^{-\frac{1}{2}})h_{k-1}W_k) \\ &= \sigma(SUM(\hat{D}^{-1}h_{k-1},\ \hat{D}^{-\frac{1}{2}}A\hat{D}^{-\frac{1}{2}}h_{k-1})W_k) \end{aligned} \tag{2}$$

2.3 GraphSAGE

Graph Sample and Aggregator (GraphSAGE) proposed in [6] consists of 3 models made up of different differentiable neighborhood aggregator functions. GraphSAGE models were defined for multi-label semi-supervised inductive learning task, *i.e* generalizing to unseen nodes during training. Let the function $Aggregate()$ abstractly denote the different aggregator functions in GraphSAGE, specifically Aggregate $\in$ {mean, max pooling, LSTM} and we will refer to these models as GS-MEAN, GS-MAX and GS-LSTM, respectively. Similar to GCN, GraphSAGE models also recursively combine neighborhood information at each layer of the Neural Network. GraphSAGE has an additional label layer unlike GCN, i. e., $h_L = h_{K+1}$. Hence, σ_k = RELU$\forall k \in [1, K]$ and σ_L = sigmoid. Here, the weights $W_{\hat{k}} \in \mathbb{R}^{2d\times d}$.

$$\begin{aligned} h_0 &= X \\ h_k &= \sigma_k(CONCAT(h_{k-1}, Aggregate(h_{k-1}, Neighbors)W_{\hat{k}}) \\ &\quad \forall k \in [1, K] \\ Y &= \sigma_{K+1}(h_K W_L) \end{aligned} \tag{3}$$

GraphSAGE models, unlike GCN, are defined to work with partial neighborhood information. For each node these models randomly sample and use only a subset of neighbors ($Neighbors$) from different hops. This choice to work with partial neighborhood information allows them to scale to large graphs but restricts them from capturing the complete neighborhood information. Rather than viewing it as a choice it can also be seen as restriction imposed by the use of Max Pool and LSTM aggregator functions which require fixed input lengths to compute efficiently. Hence, GraphSAGE constraints the neighborhood subgraph of a node to contain fixed number of neighbors at each hop.

3 ANALYSIS OF RECURSIVE PROPAGATION GRAPH MODELS

In this section, we first provide a unified formulation of GCN and GraphSAGE as recursive graph propagation models. Then, we point out two limitations of these models that can potentially restrict their capacity to combine information from multiple hops.

3.1 Unified Recursive Graph Propagation Kernel

GCN and GraphSAGE differ from each other in terms of their node features, the neighborhood features and the combination function. These differences can be abstracted to provide a unified formulation as below.

$$\begin{aligned} h_k &= \sigma_k(combine(\Omega_k, \Psi_k)W_k) \\ \Omega_k &= \alpha h_{k-1} \\ \Psi_k &= F(A)h_{k-1} \end{aligned} \tag{4}$$

where Ω_k and Ψ_k denote the $(k-1)^{\text{th}}$ hop node and neighbor features respectively, α denotes the scaling factor for node features, $F(A)$ denotes the neighbors' weights and *combine* denotes the mode of combination of node and neighbor features. For brevity, we have made the neighbor's weightage function to be independent of h_{k-1}.

We can view GCN in terms of Eqn: 4 of node features, $\Omega_k = \alpha \cdot h_{k-1}$ with $\alpha = \hat{D}^{-1}$, neighbor features, $\Psi_k = F(A)h_{k-1}W_k$ with $F(A) = \hat{D}^{-\frac{1}{2}}I\hat{D}^{-\frac{1}{2}}h_{k-1}$ and combining by summation, $combine = SUM$. Similarly, GraphSAGE can be viewed in terms of of Eqn: 4 with node features, $\Omega_k = \alpha \cdot h_{k-1}$ with $\alpha = I$ and neighbor features, $\Psi_k = F(A)h_{k-1}W_k$ with concatenation, $combine = CONCAT$. Different neighborhood aggregators of GraphSAGE yield different $F(A)$s. Specifically, $F(A) = D^{-1}A$ for GS-MEAN, $F(A) = concat(C_iA)\forall i$ for GS-MAX where C_i is a one hot vector with 1 in the position of the node with the maximum value for the i^{th} feature and for GS-LSTM, $F(A)$ is defined by the LSTM gates which randomly orders neighbors and gives weightage for a neighbor in terms of the neighbors seen before.

The concatenation (denoted by square braces below) can also be expressed in terms of a summation of node and neighbors features with different weight matrices, $W_k^{\omega}, W_k^{\psi} \in \mathbb{R}^{d\times d}$ respectively by appropriately padding zero matrices, ($\mathbf{0}$) as shown below.

$$\begin{aligned} h_k &= \sigma_k([\alpha h_{k-1}, F(A)h_{k-1}][W_k^{\omega}, W_k^{\psi}]) \\ h_k &= \sigma_k([\alpha \cdot h_{k-1}W_k^{\omega}, \mathbf{0}] + [\mathbf{0}, F(A)h_{k-1}W_k^{\psi}]) \end{aligned} \tag{5}$$

The Ω_k and Ψ_k terms for CONCAT and SUMMATION combinations are similar if weights are shared in the CONCAT formulation as shown in Eqn: 6 and Eqn: 7, respectively. Weight sharing refers to $W_k^{\omega} = W_k^{\psi} = W_k$.

$$h_k = \sigma_k(([\alpha h_{k-1}, 0] + [0, F(A)h_{k-1}])[W_k, W_k]) \tag{6}$$

$$h_k = \sigma_k(\alpha h_{k-1} + F(A)h_{k-1}W_k) \tag{7}$$

For brevity of analysis made henceforth, we only consider the summation model to discuss the limitations of the recursive propagation kernels without losing any generality on the deductions made. Further, we provide another abstraction to the summation formulation as in Eqn: 7 by Eqn: 8. Henceforth, we refer to Eqn: 8 as the generic recursive propagation kernel in the upcoming analysis.

$$\begin{aligned} \Phi &= (\alpha + F(A)) \\ h_k &= \sigma_k(\Phi h_{k-1}W_k) \end{aligned} \tag{8}$$

3.2 Limitations of recursive propagation models

In this section we highlight two important issues with the recursive propagation models: (i) No independent regulatory paths to different hops and (ii) Bias in the importance of different hops.

3.2.1 ***No independent regulatory paths to different hops***. Though these propagation models can combine information from multiple hops, they are restricted by their formulation from independently regulating them. This is a consequence of recursively computing K^{th} hop information in terms of $(K-1)^{\text{th}}$ hop information. This creates an interdependence among weights associated with the different hop information. We can view this problem by recursively expanding the unified formulation below:

$$h_K = \sigma(\Phi \cdot ...\sigma(\Phi \cdot (\sigma(\Phi \cdot h_0W_1)W_2)...W_K) \tag{9}$$

To better understand the problem with this recursive formulation, let's consider a 3-hop linear kernel with $K = 3$ and $\sigma_k = I$ which on expansion yields the following equation:

$$\begin{aligned} h_3 = \alpha^3 h_0 \prod_{k=1}^{k=3} W_k + 3\alpha^2 F(A)h_0 \prod_{k=1}^{k=3} W_k \\ + 3\alpha F(A)^2 h_0 \prod_{k=1}^{k=3} W_k + F(A)^3 h_0 \prod_{k=1}^{k=3} W_k \end{aligned} \tag{10}$$

This expansion makes it trivial to note that all the different hop information ($h_0, F(A)h_0, F(A)^2h_0$ and $F(A)^3h_0$) are influenced by all the weights in the model. For example, if we take the case where only first-hop information (only $F(A)$ term) is required then there exists no combination of W_ks that can provide it under the current model. It should be noted that we cannot obtain the 1st hop information alone by using a 1-hop kernel as that would also include 0-hop information, $F(A)^0h_0 = h_0$.

From the above analysis, we can say that the model cannot capture information from a particular subset of hops without including information from other hops. This lack of flexibility in these graph convolutional neural networks to regulate information from different hops independently is certainly undesirable. The limitation of these networks can be attributed to the specific formulation of recursion used to compute output at every layer. As with every layer, k of graph convolutional nets a new information about the k^{th} hop is introduced as $\Phi = I + \boldsymbol{F(A)}$ in $h_k = \sigma(\Phi h_{k-1}W_k)$.

The remainder of this subsection shows that adding conventional neural network components to the base model such as bias, skip connections and having different weights for node and neighbors does not fix the limitations completely.

Inclusion of bias: An inclusion of bias term in the recursive propagation kernel as in Eqn: 11 is not useful.

$$h_k = \sigma(\Phi \cdot h_{k-1}W_k + b_{k-1}) \tag{11}$$

The model still cannot regulate the relative magnitude of information independently as information at each hop, h_k (with the bias) is still computed recursively in terms of the previous hop. The resultant inter-dependency can be seen in Eqn: 12 with expansion of Eqn: 11. This was also observed empirically that the addition of bias did not result in any significant change in results (not reported here).

$$h_K = (\Phi)^K \Pi_{i=1}^K h_0 W_i + \sum_{j=2}^{K} (\Phi)^j (\Pi_{l=k-j+1}^K W_i) b_{k-j} \tag{12}$$

Inclusion of skip connections: A possible way to improve the flexibility of the model is by adding the popular skip connections [7] to these models as in Eqn: 13.

$$h_k = \sigma(\Phi \cdot h_{k-1} W_k) + h_{k-1} \tag{13}$$

On recursively expanding the above equation, it can be seen that adding skip connections to a layer, k results in directly adding information from all the lower hops, $i < k$ as shown in Eqn: 14.

$$\begin{aligned} h_k &= \sigma(\Phi \cdot h_{k-1} W_K) + \sigma(\Phi \cdot h_{k-2} W_{K-1}) + h_{k-2} \\ &= \sum_{i=1}^{k} \sigma(\Phi \cdot h_{i-1} W_i) \end{aligned} \tag{14}$$

Unlike Eqn: 8 where the output at each layer, k was only dependent on the previous layer, h_{k-1} accounting to only one computational path; now adding skip connections allows for multiple computational paths. As it can be seen that at the K^{th} layer the model has the flexibility to select output from any or all $h_k \forall k < K$.

Though adding Skip connections improves the model, it should be noted that it allows to capture information only up to a particular hop, h_k and is still not sufficient to individually regulate the importance of information from each hop as all hops $i < k$ have inter-dependencies. Lets us consider the same example as earlier to capture information from 1^{st} hop alone ignoring the rest with a 3-hop model. The best, the 3-hop model with skip connection can do is to learn to ignore information from 2^{nd} and 3^{rd} hop by setting $W_2 = W_3 = 0$ and considering h_1 and h_0. It can be reasoned as before to see that W_0 cannot be set to 0 as h_1 depends on the result of h_0 thereby having no means to ignore information from h_0. This limits the expressive power to efficiently span the entire space of K^{th} order neighborhood information. To summarize, skip connections at best can obtain information up to a particular hop by ignoring information from later hops. CONCAT operation for combination can be perceived as linear skip connection as noted by the authors of GraphSAGE.

3.2.2 *Bias on the importance of each hop*

. The K-hop generic propagation kernel defined in Eqn: 8 with a linear activation function can be recursively rolled out and expressed as a binomial in terms of node and neighbor features raised to the K^{th} power as in Eqn: 15.

$$h_K = (\alpha I + F(A))^K h_0 \prod_{k=1}^{K} W_k \tag{15}$$

The higher order binomial term in Eqn: 15 when expanded assigns different weights to different $F(A)^k h_0$ terms. These weights correspond to the binomial coefficients of the binomial series, $(\alpha I + F(A))^K$. For example, refer to Eqns: 16 and 17 corresponding to a 2-hop and 3-hop kernel with $\alpha = I$ and $W_K = I$ for simplicity. It can be seen that for 2-hop kernel the weights are $[1, 2, 1]$ and for the 3-hop kernel it is $[1, 3, 3, 1]$. Thus, these recursive propagation kernels combines different hop information weighed by the binomial coefficients.

$$h_2 = h_0 + 2F(A)h_0 + F(A)^2 h_0 \tag{16}$$

$$h_3 = h_0 + 3F(A)h_0 + 3F(A)^2 h_0 + 3F(A)^3 h_0 \tag{17}$$

These weights induce a bias on the importance of each hops. Any such fixed bias over different hops cannot consistently provide good performance across numerous datasets. In the limit of infinite data, we can expect the W_k parameters to correct these scaling factors induced by these biases. But as with most graph based semi-supervised learning applications where the amount of labeled data is very less, an undesirable bias can result in a sub-optimal model. The experimental performance improvement and the theoretical advantage for re-normalized graph convolutional model over the mean model provided in [9] along with our results validates that these biases affect the learning.

Existing propagation kernels defined over K-hop information, extract relational information by performing convolution operations on different k-hop neighbors based on their respective K-th order binomial. As discussed earlier, biasing the importance of information along with recursive weight dependencies hinder the model from learning relevant information from different hops. These limitations constrain the expressive power of these models from spanning the entire space of K^{th} order neighborhood information. Hence, it is restricted to only a subspace of all possible K^{th} order polynomial defined on the neighborhood of nodes. Though the above analysis is done on a linear propagation model, the insight can be carried over to the non-linear models. The Empirical results in later section seem to support this.

4 PROPOSED MODEL

To overcome the limitations put forth in the analysis earlier, it is clear that the differentiable graph kernels should have the power to effectively span the entire space of a K^{th} order function. Thus to mitigate these issues with existing models we propose a minimalist additional component for these models, fusion component. This fusion component would consists of parameters to combine the existing binomial bases defined on different hop information to effectively scale the entire space of a K^{th} order neighborhood. We define the fusion component in Eqn: 18 as a linear weighted combination over K-hop neighborhood space spanned by the binomial bases, h_ks $([h_0, \Phi h_0, \Phi^2 h_0, \ldots, \Phi^K h_0]$ with K coefficients $[\theta_0, \theta_1, \theta_2, \ldots, \theta_K])$. The θ coefficients allow the neural network to explicitly learn the optimal combination of information from different hops. As the h_ks are binomials, a parameterized linear combination of these binomials can obtain any combination of the individual hop information. This can be verified by the fact that the coefficients of the binomial equation which form the Pascal matrix is a non-singular lower triangular matrix. Thus in the linear neural network case, we can see that the binomial system of equations can be solved to obtain any combination of $F(A)$ terms. Though the proposed component, in theory, is an optimal solution in the linear activation case, it seems to be experimentally useful with piece-wise linear RELU activation too as shown with the experiments.

$$y = \sum_{k=0}^{K} h_k \theta_k \tag{18}$$

4.1 Fusion Graph Convolutional Network

We propose a Fusion Graph Convolutional Network, F-GCN in equations in 19. F-GCN is a minimalist architecture that adds the fusion component defined in Eqn: 18 to GCN defined in Eqn: 1. It can be seen to combine different k^{th} hop information by the fusion component. The fusion component mentioned in the last two lines of equations 19 can be seen to fuse information at the label layer as label prediction scores from each propagation step of the GCN are linearly combined before being normalized.

$$\begin{aligned} h_0 &= \sigma_k(XW_1) \\ h_k &= \sigma_k(\hat{L}h_{k-1}W_k + h_{k-1}), \quad \forall\, k \in [2, K]. \\ y &= \sum_{k=0}^{K} h_k \theta_k \\ \mathcal{L} &= softmax(y) \ or \ sigmoid(y) \end{aligned} \tag{19}$$

The dimensions of h_k, θ_k, Y , W_0, W_k are in $\mathbb{R}^{N\times d}$, $\mathbb{R}^{d\times L}$, $\mathbb{R}^{N\times L}$, $\mathbb{R}^{F\times d}$, $\mathbb{R}^{d\times d}$, respectively. F-GCN uses ReLU activation layers and a softmax label layer accompanied by a multi-class cross entropy if it is a multi-class classification problem or a sigmoid layer followed by a binary cross entropy layer if it is a multi-label classification problem. Since predictions are obtained from every hop, we also subject h_0 to a non-linear activation function with weights same as W_1 from h_1.

This simple architectural design provides miscellaneous benefits besides explicitly allowing to capture different hop information. F-GCN has additional direct gradient flow paths to each of the propagation steps allowing it learn better discriminative features at the lower hops too and also improves its chances of mitigating vanishing gradient. F-GCN, is a multi-resolution architecture which simultaneously looks at information from different resolutions/hop and also models the correlations among them.

Our particular choice of this simple fusion architecture over a complex gated combination of hop information is primarily motivated by the reason that existing gating and attention mechanism are defined for positive combination of different information sources. As we have already motivated with the linear fusion component, we need a combination function that would combine different bases (h_k)s to obtain any combination of $F(A)^k$ components. Positive combination of h_ks cannot span the entire space of $F(A)^k h_0$ and hence are not suited directly. Besides this, scalar gating functions are also less flexible as it assumes that all labels have similar neighborhood dependency. Our preliminary experiments with scalar gating mechanism also backed this as we obtained inferior performances compared to the simple fusion component (not reported).

We also introduce a F-MEAN architecture which is essentially the Fusion component added to GraphSAGE's mean model, GS-MEAN. The fusion component is similar in spirit to the Chebyshev filters introduced in [4] for complete graph classification task. The primary difference is that the Chebyshev filters learn coefficients to combine Chebyshev polynomials defined over neighborhood information whereas in Fusion models the coefficients of the filter are used to combine different binomials defined by GCN or GraphSAGE's MEAN functions. And an additional difference is that the Chebyshev polynomial basis is not associated with weights W_k to filter k^{th} hop information which can potentially enable the model to learn complex non-linear feature basis. F-GCN also enjoys the benefit of the re-normalization trick of GCN that stabilizes the learning to diminish the effect of vanishing or exploding gradient problems associated with training Neural Networks.

5 EXPERIMENTS

We provide experimental results on a number of datasets from social, citation, movie, product and biological networks on semi-supervised multi-class and multi-label tasks.

5.1 Datasets

The description of the datasets we have used are provided below and their statistics are mentioned in Table 1.

Social networks: We use Facebook (FB) [11, 13] and BlogCatalog (BLOG) [15] dataset. In the Facebook dataset, the nodes are Facebook users and the task is to predict the political views of a user given the gender and religious view of the user as features. In the BlogCatalog dataset, the nodes are users of a social blog directory, the user's blog tags are treated as node features and edges correspond to friendship or fan following. The task here is to predict the interests of users.

Citation Networks: We use three citation graphs: Cora, Citeseer, and Cora_ML. In all the three datasets, the articles are the nodes and the edges denote citations. The bag-of-word representation of the article is used as node attributes. The task is to predict the research area of the article. Cora-ML is the multi-label classification dataset [1], unlike the others which are multi-class.

Biological network: We use the Human tissue protein-protein interaction (PPI) network introduced in GraphSAGE [6]. The dataset contains PPI from 24 human tissues, the task is to predict the gene's functional ontology. Positional gene sets, motif gene sets, and immunology signatures were considered as features.

Movie network: We construct a movie network from movielens-2k dataset available as a part of HetRec 2011 workshop [3]. The dataset is an extension of the MovieLens10M dataset with additional movie tags. The nodes are the movies and edges are created between movies if they share a common actor. The movie tags form the movie features. The task here is to predict genres of movies.

Product network: We constructed an Amazon DVD co-purchase network which is a subset of co-purchase data, Amazon_060 [10]. The network construction procedure is similar to the one created in [11]. The nodes correspond to DVDs and edges are constructed if two DVDs are co-purchased. The DVD genres are treated as DVD features. The task here is to predict whether a DVD will have Amazon sales $\leq$ 7500 or not.

5.2 Experiment Setup

We set the experiments in a semi-supervised manner with a random 20% of data set aside for testing. For training we only use 10% of the

Table 1: Dataset statistics

Dataset	Network	Nodes	Edges	Classes	Multi-label	Features
CORA	Citation	2708	5429	7	FALSE	1433
CITE	Citation	3312	4715	6	FALSE	3703
CORA_ML	Citation	11881	34648	79	TRUE	9568
HUMAN	Biology	56944	1612348	121	TRUE	50
BLOG	Social	69814	2810844	46	TRUE	5413
FB	Social	6302	73374	2	FALSE	2
AMAZON	Product	16553	76981	2	FALSE	30
MOVIE	Movie	7155	388404	20	TRUE	5297

labeled data. To have a realistic setup, we create the training sets by randomly sampling five sets of 10% nodes from the entire graph. Further, 20% of these training nodes are chosen as the validation set. We do not use the validation set for (re)training. It is ensured that these training samples are mutually exclusive from the held out test data. For all transductive experiments the reported results are an average over these 5 different training sets. For the Inductive learning experiment, we use the PPI network where we reuse the setup of GraphSAGE [6], where the test nodes and validation nodes have no path to the nodes in the training set.

The hyperparameters for the models are the number of layers of neural network or number of neighborhood hops, dimensions of the layers, dropouts for all layers and L2 regularization, similar to [9]. We set the same starting learning rate for all the models across all datasets. We train all the models for a maximum of 2000 epochs using Adam [8] with learning rate set to 1e-2. We use a variant of patience method with learning rate annealing for early stopping of the model. Specifically, we train the model for a minimum of 50 epochs and start with a patience of 30 epochs and drop the learning rate and patience by half when the patience runs out (i.e when the validation loss does not reduce within the patience window). We stop the training when the model consecutively loses patience for 2 turns. We added all these components to the baseline codes too. In fact, we observed an improvement of 25.91 percentage for GraphSage on their dataset.

For hyper-parameter selection, we search for optimal setting on a two-layer deep feedforward neural network with the node attributes (NODE) alone. We then use the same hyper-parameters across all the other models. We observed this way of hyper parameter search to be effective in terms of performance and compute power as node only classifier is simple and fast to run compared to other relational models. We row-normalize the node features and initialize the weights with [5]. Since the percentage of different labels in training samples can be significantly skewed, like [11] we weigh the loss for each label inversely proportional to its total fraction. We ensure that all models have the same setup in terms of the weighted cross entropy loss, the number of layers, dimensions, patience based stopping criteria and dropouts. The best results for models across multiple hops were reported, 3 hops for Amazon, 4 hops for Cora_ML and HUMAN and 2 hops for the remaining datasets.

Our implementation is mini-batch trainable, similar to GraphSAGE. We compare our models: F-GCN and F-mean against node only classifier, our mini-batch version of GCN along with Mean, Maxpool and LSTM models of GraphSAGE. We use skip connections for our model and GCN baseline and we use the CONCAT combination for GraphSAGE models.

5.3 Results

We report the performances in terms of Micro-F1 for transductive and inductive experiments in Tables 2 and 3, respectively. To compare the overall performance and measure robustness in term of consistent performance across datasets, we introduce a statistic, *penalty*, that is a measure of a model's absolute loss from the best performing model in terms of Micro-F1. Penalty is defined as, `Penalty[model] = mean(Best_results[dataset] - result[model][dataset])` , where `Best_result[dataset]` is the micro_f1 of the best performing model for the dataset and `result[model][dataset]` is the model's performance for that dataset. Lower penalty indicates more consistent performance overall.

Among the baselines, GraphSAGE models with more complex aggregator functions and no shared weights for node and neighborhood featues significantly outperform the GCN model with no shared weights and a limiting scaling factor, α. GCN model performs poorly in datasets where the number of edges is high. This is primarily due to the fact that, when the degree of a node is high the scaling factor associated with node features is heavily under weighed ($\alpha = \hat{D}^{-1}$) relative to the neighbors' information. This can be observed from the datasets: Blog, FB, Amazon, and Movie, as it performs poorly than the classifier which only uses the node attributes.

In the results for transductive experiments in Table 2, the proposed Fusion models outperform their base models (GCN and GS-MEAN) significantly. The F-GCN model has seem to have learned to avoid the bias induced by the scaling factor by learning to effectively combine the node features along with other hop information resulting in improved performances over GCN by up to an absolute $\approx$ 15%. Similarly, F-MEAN improves over GS-MEAN by up to $\approx$ 4% on datasets. The GCN model which was poorly under performing among the propagation kernels not only obtained significant boost in performance with addition of fusion component but also achieved best overall consistent score with a penalty as low as 0.241%. As it can be seen F-GCN performs best on 6 datasets while falling short by only 0.241% from the best on the other 2 datasets.

Table 2: Transductive Experiments: Micro-f1 scores for semi-supervised node classification

	NODE	GCN	Mean	Max	LSTM	F-MEAN	F-GCN
CORA	60.222	79.039	76.821	73.272	65.730	76.858	**79.039**
CITE	65.861	**72.991**	70.967	71.390	65.751	70.393	72.266
CORA_ML	40.311	63.848	62.800	53.476	OOM	62.064	**63.993**
HUMAN	41.459	62.057	63.753	65.068	64.231	65.308	**65.538**
BLOG	37.876	34.073	39.433	**40.275**	OOM	39.467	39.069
FB	64.683	49.762	64.127	64.571	64.619	63.714	**64.857**
AMAZON	63.710	61.777	68.266	70.302	68.024	70.163	**72.550**
MOVIE	50.712	39.059	50.557	50.569	OOM	51.450	**52.021**
Penalty	10.804	06.082	01.818	02.793	05.324	**01.481**	**00.241**

Table 3: Inductive learning task in PPI: Micro-f1 scores for semi-supervised node classification

NODE	GCN	Mean	Max	LSTM	Mean*	Max*	LSTM*	F-MEAN	F-GCN
44.644	85.708	79.634	78.054	87.111	59.800	60.000	61.200	79.049	**88.942**

Similarly, F-GCN performs best on the inductive learning task too where it improves over the base GCN by $\approx 3\%$ and beats all other complex aggregation methods of GraphSAGE.

To note: Our experiment setup for training improved the originally reported results for GraphSAGE models (Mean*, LSTM*, Max*) as mentioned in the table.

6 CONCLUSION

In this work, we have shown that the current state-of-the-art models for node classification can be viewed as higher order binomial combinations of node and neighborhood information. Further, we have analytically shown potential limitations of the current state-of-the-art methods in node classification due to their inability to effectively capture multi-hop information. To address this issue, we have proposed a simple fusion component that can be added to existing models. The fusion component linearly combines different binomial basis defined for different hop information. We empirically demonstrate that the proposed fusion component improves existing models by a significant margin providing highly competitive state-of-the-art results across 8 datasets from different domains.

REFERENCES

[1] 2001. CORA Research Paper Classification Datase. *https://people.cs.umass.edu/ mccallum/data.html* (2001).

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[3] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011). In *Proceedings of the 5th ACM conference on Recommender systems (RecSys 2011)*. ACM, New York, NY, USA.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*. 3844–3852.

[5] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 249–256.

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1025–1035.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[8] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[9] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[10] Jure Leskovec and Rok Sosič. 2016. SNAP: A General-Purpose Network Analysis and Graph-Mining Library. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 1 (2016), 1.

[11] John Moore and Jennifer Neville. 2017. Deep Collective Inference.. In *AAAI*. 2364–2372.

[12] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[13] Joseph J Pfeiffer III, Jennifer Neville, and Paul N Bennett. 2015. Overcoming relational learning biases to accurately predict preferences in large scale networks. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 853–863.

[14] Suhang Wang, Jiliang Tang, Charu Aggarwal, and Huan Liu. 2016. Linked document embedding for classification. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 115–124.

[15] Xufei Wang, Lei Tang, Huiji Gao, and Huan Liu. 2010. Discovering overlapping groups in social media. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 569–578.

[16] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861* (2016).