Multi-hop Graph Convolutional Network with High-order Chebyshev Approximation for Text Reasoning

Shuoran Jiang, Qingcai Chen*, Xin Liu, Baotian Hu, Lisai Zhang
Harbin Institute of Technology, Shenzhen
{shuoran.chiang, hit.xinliu, lisaizhang2016}@gmail.com
{qingcai.chen, hubaotian}@hit.edu.cn

Abstract

Graph convolutional network (GCN) has become popular in various natural language processing (NLP) tasks with its superiority in long-term and non-consecutive word interactions. However, existing single-hop graph reasoning in GCN may miss some important non-consecutive dependencies. In this study, we define the spectral graph convolutional network with the high-order dynamic Chebyshev approximation (HDGCN), which augments the multi-hop graph reasoning by fusing messages aggregated from direct and long-term dependencies into one convolutional layer. To alleviate the over-smoothing in high-order Chebyshev approximation, a multi-vote based cross-attention (MVCAttn) with linear computation complexity is also proposed. The empirical results on four *transductive* and *inductive* NLP tasks and the ablation study verify the efficacy of the proposed model. Our source code is available at https://github.com/MathIsAll/HDGCN-pytorch.

1 Introduction

Graph neural networks (GNNs) are usually used to learn the node representations in Euclidean space from graph data, which have been developed to one of the hottest research topics in recent years (Zhang, 2020). The primitive GNNs relied on recursive propagation on graphs, which takes a long time to train (Zhang et al., 2019b). One major variant of GNNs, graph convolutional networks (GCNs) (Kipf and Welling, 2017; Yao et al., 2019), takes spectral filtering to replace recursive message passing and needs only a shallow network to convergent, which have been used in various NLP tasks. For example, Yao et al. (2019) constructed the text as a graph and input it to a GCN. This method achieved better results than conventional deep learning models in text classification. Afterward, the GCNs have became popular in more tasks, such as word embedding (Zhang et al., 2020b), semantic analysis (Zhang et al., 2019a), document summarization (Wang et al., 2020), knowledge graph (Wang et al., 2018), etc.

The spectral graph convolution in Yao's GCN is a localized first-order Chebyshev approximation. It is equal to a stack of 1-step Markov chain (MC) layer and fully connected (FC) layer. Unlike the multi-step Markov chains, the message propagation in vanilla GCN lacks the node probability transitions. As a result, the multi-hop graph reasoning is very tardy in GCN and easily causes the *suspended animation problem* (Zhang and Meng, 2019). However, the probability transition on the graph is useful to improve the efficiency in learning contextual dependencies. In many NLP tasks (like the question answering (QA) system and entity relation extraction), the features of the two nodes need to be aligned. As an example, Figure 1 shows a simple graph where the node $n4$ is a pronoun of node $n1$. In this example, the adjacency matrix is masked on nodes $n2$, $n3$, $n5$ to demonstrate the message passing between $n1$ and $n4$. Figure 1 (c) and (d) plot the processes of feature alignment on two nodes without and with probability transitions respectively. In this example, the feature alignment process without probability transition needs 10 more steps than which with probability transition. It is shown that encoding the multi-hop dependencies through the spectral graph filtering in GCN usually requires a deep network. However, as well known that the deep neural network (DNN) is tough to train and easily causes the over-fitting problem (Rong et al., 2019).

Some newest studies to improve the *multi-hop graph reasoning* include graph attention networks (GATs) (Veličković et al., 2018), graph residual neural network (GRESNET) (Zhang and Meng, 2019), graph diffusive neural network (DIFNET)

*corresponding author: Qingcai Chen

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 6563–6573
August 1–6, 2021.

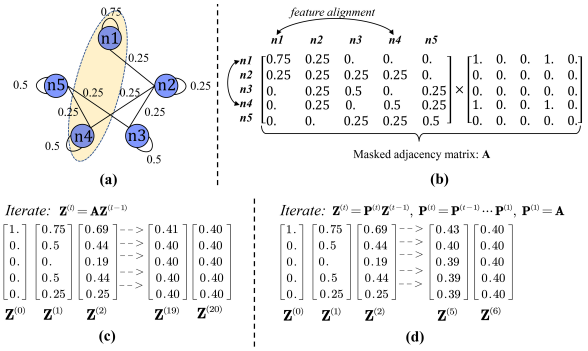

Figure 1: **(a):** A simple graph with 5 nodes and the weighted edges, in which the nodes $n4$ is a pronoun of $n1$ and the two nodes need to align features. **(b):** The masked adjacency matrix on this graph. **(c)** and **(d):** The processes of feature alignment on nodes $n1$ and $n4$ without transition probability and with transition probability respectively.

(Zhang, 2020), TGMC-S (Zhang et al., 2020c) and Graph Transformer Networks (Yun et al., 2019; Zhang and Zhang, 2020). GATs enhance the graph reasoning by implicitly re-defining the graph structure with the attention on the 1-hop neighbors, but there is equilibrial optimization on the whole graph. GRESNET solves the *suspended animation problem* by creating extensively connected highways to involve raw node features and intermediate representations throughout all the model layers. However, the multi-hop dependencies are still reasoned at a slow pace. DIFNET introduces a new neuron unit, i.e., GDU (gated diffusive unit), to model and update the hidden node states at each layer. DIFNET replaces the spectral filtering with a recursive module and realizes the neural gate learning and graph residual learning. But the time cost is aggravated in DIFNET compared with GCN. TGMC-S stacks GCN layers on adjacent matrices with different hops of traffic networks. Different from the ground-truth traffic network in TGMC-S, it is hard to construct the multi-hop word-word relationships objectively from the text. TGMC-S hadn't given a way to improve the multi-hop message passing in GCN.

Transformers (Vaswani et al., 2017) and corresponding pre-trained models (Xu et al., 2019) could be thought of as fully-connected graph neural networks that contain the multi-hop dependencies. They figure out the contextual dependencies on the fully-connected graph with the attention mechanism. The message propagation in transformers follows the relations self-adaptively learned from input sequence instead of the fixed graph structures. Publications have shown that transformers outperform GCNs in many NLP tasks. Graph Transformer (Dwivedi and Bresson, 2020) generalizes the Transformer to arbitrary graphs, and improves *inductive learning* from Laplacian eigenvectors on graph topology. However, due to the connections scale quadratically growth with node number N in graphs, things get out of hand for very large N. Additionally, the fully-connected graph is not an interpretable architecture in practical tasks. For example, whether Transformers are the best choice to bring the text in linguistic theory? [1]

To improve the efficiency and performance of *multi-hop graph reasoning* in spectral graph convolution, we proposed a new graph convolutional network with high-order dynamic Chebyshev approximation (HDGCN). A prime ChebNet and a high-order dynamic (HD) ChebNet are firstly applied to implement this Chebyshev approximation. These two sub-networks work like a trade-off on low-pass signals (direct dependencies) and high-pass signals (multi-hop dependencies) respectively. The prime ChebNet takes the same frame as the convolutional layer in vanilla GCN. It mainly extracts information from direct neighbors in local contexts. The HD-ChebNet aggregates messages from multi-hop neighbors following the transition direction adaptively learned by the attention mechanism. The standard self-attention (Vaswani et al., 2017) has a $\mathcal{O}\left(N^2\right)$ computation complexity and it is hard to be applied on long sequence. Even the existing sparsity attention methods, like the Star-Transformer (Guo et al., 2019) and Extended Transformer Construction (ETC) (Ainslie et al., 2020), have reduced the quadratic dependence limit of sequence length to linear dependence, but the fully-connected graph structure cannot be kept. We design a multi-vote-based cross-attention (MVCAttn) mechanism. The MVCAttn scales the computation complexity $\mathcal{O}(N^2)$ in self-attention to $\mathcal{O}(N)$.

The main contributions of this paper are listed below:

- To improve the efficiency and performance of multi-hop reasoning in spectral graph convolution, we propose a novel graph convolutional network with high-order dynamic Chebyshev Approximation (HDGCN).

[1] https://towardsdatascience.com/transformers-are-graph-neural-networks-bca9f75412aa

- To avoid the over-smoothing problem in HD-ChebNet, we propose a multi-vote based cross-attention (MVCAttn) mechanism, which adaptively learn the direction of node probability transition. MVCAttn is a variant of the attention mechanism with the property of linear computation complexity.

- The experimental results show that the proposed model outperforms compared SOTA models on four transductive and inductive NLP tasks.

2 Related Work

Our work draws supports from the vanilla GCN and the attention mechanism, so we first give a glance at the paradigm of these models in this section.

2.1 Graph Convolutional Network

The GCN model proposed by (Kipf and Welling, 2017) is the one we interested, and it is defined on graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set. The edge $(v_i, v_j) \in \mathcal{E}$ represents a link between nodes v_i and v_j. The graph signals are attributed as $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$, and the graph relations $\mathcal{E}$ can be defined as an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ (binary or weighted).

Each convolutional layer in GCN is a 1st Chebyshev approximation on spectral graph convolution, and its layer-wise propagation rule in neural network is defined as:

$$\begin{aligned} \mathbf{H}^{(l+1)} &= \sigma\left(\widetilde{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right), \quad L \geq l \geq 0 \\ \widetilde{\mathbf{A}} &= \left(\mathbf{D} + \mathbf{I}_N\right)^{-\frac{1}{2}} \left(\mathbf{A} + \mathbf{I}_N\right) \left(\mathbf{D} + \mathbf{I}_N\right)^{-\frac{1}{2}}, \end{aligned} \tag{1}$$

where $\mathbf{H}^{(0)} = \mathbf{X}$, $\widetilde{\mathbf{A}}$ is the normalized adjacency matrix and σ is a non-linear activation function.

The node embeddings output from the last convolutional layer are fed into a *softmax* classifier for node or graph classification, and the loss function $\mathcal{L}$ can be defined as the cross-entropy error. The weight set $\{\mathbf{W}^{(l)}\}_{l=0}^{L}$ can be jointly optimized by minimizing $\mathcal{L}$ via gradient descent.

2.2 Self-Attention Is a Dynamic GCN

The attention mechanism is an effective way to extract task-relevant features from inputs, and it helps the model to make better decisions (Lee et al., 2019). It has various approaches to compute the attention score from features, and the scaled dot-product attention proposed in Transformers (Vaswani et al., 2017) is the most popular one.

$$\mathbf{Z} = \underbrace{\text{softmax}\left(\frac{\mathbf{X}\mathbf{W}^q\mathbf{W}^k\mathbf{X}^{\mathrm{T}}}{\sqrt{d_k}}\right)}_{\mathbf{A}}\mathbf{X}\mathbf{W}^v \tag{2}$$

where $\mathbf{X} \in \mathbb{R}^{N \times d}$ is the input sequence, and weights $\mathbf{W}^q \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}^k \in \mathbb{R}^{d_k \times d}$, $\mathbf{W}^v \in \mathbb{R}^{d \times d_v}$ are used to transform sequence to queries, keys and values.

As showed in Equation 2, the attention scores $\mathbf{A}$ can be viewed as a dynamic adjacency matrix on sequence $\mathbf{X}$. This process in self-attention is similar to the graph convolutional layer defined in Equation 1. The only difference is that the adjacency matrix in Equation 2 is adaptively learned from input instead of prior graph structures.

3 Method

In our model, the input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ takes the same form as the one in GCN. The nodes are attributed as $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$, and the adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ (binary or weighted) is defined on graph edges $\mathcal{E}$.

The spectral graph convolution in Fourier domain is defined as,

$$\boldsymbol{g_\theta} \star \boldsymbol{x} = \mathbf{U}\boldsymbol{g_\theta}\left(\widetilde{\mathbf{\Lambda}}\right)\mathbf{U}^T\boldsymbol{x} \tag{3}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is the signal on a node, $\mathbf{U}$ is the matrix of eigenvectors on normalized graph Laplacian $\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, and the filter $\boldsymbol{g_\theta}(\widetilde{\mathbf{\Lambda}})$ is a function of the eigenvalues on normalized $\widetilde{\mathbf{L}}$ in Fourier domain.

The K-th ($K > 2$) order truncation of Chebyshev polynomials on this spectral graph convolution is,

$$\boldsymbol{g_\theta} \star \boldsymbol{x} \approx \sum_{i=0}^{K} \boldsymbol{\theta}_i \mathbf{U}\boldsymbol{T}_i\left(\widetilde{\mathbf{\Lambda}}\right)\mathbf{U}^T\boldsymbol{x} \tag{4}$$

where $\boldsymbol{T}_0\left(\widetilde{\mathbf{\Lambda}}\right) = \mathbf{I}$, $\boldsymbol{T}_1 = \widetilde{\mathbf{\Lambda}}$, $\boldsymbol{T}_{i>1}\left(\widetilde{\mathbf{\Lambda}}\right) = 2\widetilde{\mathbf{\Lambda}}\boldsymbol{T}_{i-1}\left(\widetilde{\mathbf{\Lambda}}\right) - \boldsymbol{T}_{i-2}\left(\widetilde{\mathbf{\Lambda}}\right)$.

To replace the parameters $\{\boldsymbol{\theta}_i\}_{i=1}^{K}$ with another parameter set $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^{K/2}$, the Kth-order Chebyshev polynomials in Equation 4 are approximated as:

$$\begin{aligned} \boldsymbol{g_\theta} \star \boldsymbol{x} &\approx \sum_{k=0}^{K/2}\left(\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^T\right)^{2k}\left(\mathbf{I} - \mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^T\right)\boldsymbol{x}\boldsymbol{\theta}^{(k)} \\ &\approx \sum_{k=1}^{K/2}\widetilde{\mathbf{A}}^{2k}\widetilde{\mathbf{A}}\boldsymbol{x}\boldsymbol{\theta}^{(i)} \end{aligned} \tag{5}$$

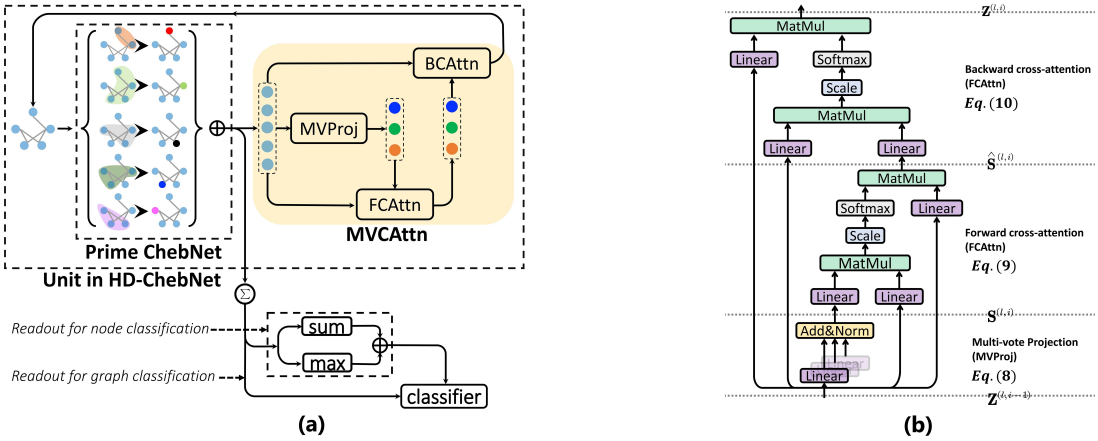

Figure 2: **(a):** The architecture of HDGCN taking the simple graph in Figure 1 as an example. **(b):** The schematics of the multi-vote based cross-attention (MVCAttn) in every unit in HD-ChebNet.

where the $\widetilde{\mathbf{A}}$ is normalized adjacency matrix (as defined in Equation 1). As the node state transition $\widetilde{\mathbf{A}}^{2k}$ causes the over-smoothing problem (Li et al., 2018; Nt and Maehara, 2019), we take the dynamic pairwise relationship $\mathbf{A}_d$ self-adaptively learned by the attention mechanism to turn the direction of node state transition.

The powers of adjacency matrix $\widetilde{\mathbf{A}}^{2k}$ in Equation 5 can cause the over smoothing problem, we replace the $\widetilde{\mathbf{A}}^{2k}$ with $\widetilde{\mathbf{A}}^{k}\mathbf{A}_d^k$.

In our implementation, the first-order and higher-order Chebyshev polynomials in Equation 5 is approximated with a prime Chebyshev network (ChebNet) and high-order dynamic Chebyshev networks (HD-ChebNets) respectively. We generalize the graph convolution on Kth-order dynamic Chebyshev approximation (Equation 5) to the layer-wise propagation as follows,

$$\begin{aligned}
\mathbf{H} &\approx \sum_{k=0}^{K/2} \mathbf{Z}^{(k)}, \\
\mathbf{Z}^{(0)} &= \underbrace{\sigma\left(\widetilde{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}\right)}_{\text{Prime ChebNet}}, \\
\mathbf{Z}^{(k)} &= \underbrace{\sigma\left(\widetilde{\mathbf{A}}\left(\mathbf{A}_d^{(k)}\mathbf{Z}^{(k)}\mathbf{W}_d^{(k)}\right)\mathbf{W}^{(k)}\right)}_{\text{Unit in HD-ChebNet}},
\end{aligned} \tag{6}$$

where k is the order and $\mathbf{W}^{(0)}$, $\mathbf{W}^{(k)}$, $\mathbf{W}_d^{(k)}$ are nonlinear filters on node signals. For the convenience of writing, we just define the first layer of HDGCN.

3.1 Prime ChebNet

We consider the same convolutional architecture as the one in GCN to implement the prime ChebNet, and it mainly aggregates messages from the direct dependencies.

$$\mathbf{Z}^{(0)} = \sigma\left(\widetilde{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}\right), \tag{7}$$

where $\mathbf{W}^{(0)} \in \mathbb{R}^{d\times d}$ and $\widetilde{\mathbf{A}}$ is the normalized symmetric adjacency matrix.

3.2 High-order Dynamic (HD) ChebNet

As the multi-hop neighbors can be interacted via the 1-hop neighbors, we take the $\mathbf{Z}^{(0)}$ output from the prime ChebNet as input of the HD-ChebNet. The multi-vote based cross-attention (MVCAttn) mechanism first adaptively learns the direction of node probability transition $\mathbf{A}_d^{(k)}$, its schematic is showed in Figure 2 (b). MVCAttn has two phases - graph information aggregation and diffusion.

Graph Information Aggregation coarsens the node embeddings $\mathbf{Z}^{(k-1)}$ to a small supernode set $\mathbf{S}^{(k)} \in \mathbb{R}^{M\times d}$, $M \ll |\mathcal{V}|$.

The first step is multi-vote projection (MVProj). In which node embeddings $\mathbf{Z}^{(k-1)}$ are projected to multiple votes $\mathbf{V}^{(k)} \in \mathbb{R}^{|\mathcal{V}|\times M\times d}$, and these votes are aggregated to supernode set $\mathbf{S}^{(k)} = \{\boldsymbol{s}_m^{(k)}\}_{m=1}^{M}$.

$$\begin{aligned}
\boldsymbol{s}_m^{(k)} &= \text{MVProj}\left(\mathbf{Z}^{(k-1)}\right) \\
&= \text{norm}\left(\sum_{v=1}^{|\mathcal{V}|} \boldsymbol{z}_v^{(k-1)}\mathbf{W}_m^V\right)
\end{aligned} \tag{8}$$

where $|\mathcal{V}| \geq v \geq 1$, $M \geq m \geq 1$, $\mathbf{W}_m^V \in \mathbb{R}^{d_k\times d_k}$ is the projection weight and norm $()$ represents the **LayerNorm** operation.

Next, the forward cross-attention (FCAttn) up-

dates the supernode values as:

$$\begin{aligned}\widehat{\mathbf{S}}^{(k)} &= \text{FCAttn}\left(\mathbf{Z}^{(k)}, \mathbf{S}^{(k)}\right) \\ &= \mathbf{A}_f^{(k)}\mathbf{Z}^{(k-1)}\mathbf{W}_{fv} \\ \mathbf{A}_f^{(k)} &= \text{Softmax}\left(\frac{\mathbf{Z}^{(k-1)}\mathbf{W}_{fk}\mathbf{W}_{fq}\mathbf{S}^{(k)}}{\sqrt{d}}\right)\end{aligned} \tag{9}$$

where $\mathbf{W}_{fk} \in \mathbb{R}^{d_k \times d_c}$, $\mathbf{W}_{fq} \in \mathbb{R}^{d_c \times d_k}$ and $\mathbf{W}_{fv} \in \mathbb{R}^{d_k \times d_k}$.

Graph Information Diffusion feeds the supernodes $\widehat{\mathbf{S}}^{(k)}$ back to update node set $\mathbf{Z}^{(k)}$. With the node embeddings $\mathbf{Z}^{(k-1)}$ and supernode embeddings $\widehat{\mathbf{S}}^{(k)}$, the backward cross-attention (BCAttn) is defined as,

$$\begin{aligned}\mathbf{Z}^{(k)} &= \text{BCAttn}\left(\widetilde{\mathbf{S}}^{(k)}, \mathbf{Z}^{(k-1)}\right) \\ &= \mathbf{A}_b^{(k)}\mathbf{Z}^{(k-1)}\mathbf{W}_{bv} \\ \mathbf{A}_b^{(k)} &= \text{Softmax}\left(\frac{\widehat{\mathbf{S}}^{(k)}\mathbf{W}_{bq}\mathbf{W}_{bk}\mathbf{Z}^{(k-1)}}{\sqrt{d}}\right)\end{aligned} \tag{10}$$

where $\mathbf{W}_{bq} \in \mathbb{R}^{d_k \times d_a}$, $\mathbf{W}_{bk} \in \mathbb{R}^{d_a \times d_k}$ and $\mathbf{W}_{bv} \in \mathbb{R}^{d_k \times d_k}$.

The last step is adding the probability transition with $\widetilde{\mathbf{A}}$. The output of k-th order HD-ChebNet (Equation A) is,

$$\widehat{\mathbf{Z}}^{(k)} = \sigma\left(\widetilde{\mathbf{A}}\mathbf{Z}^{(k)}\mathbf{W}^{(k)}\right) \tag{11}$$

Finally, the outputs from the prime ChebNet and HD-ChebNets are integrated as the node embeddings,

$$\mathbf{H} = \text{norm}\left(\mathbf{Z}^{(0)} + \sum_{k=1}^{K/2}\widehat{\mathbf{Z}}^{(k)}\right). \tag{12}$$

3.3 Classifier Layer

Node Classification The node representations $\mathbf{H}$ output from the last graph convolutional layer are straightforward fed into a *Softmax* classifier for node classification.

$$\widehat{y}_v = \text{Softmax}\left(\text{MLP}\left(\boldsymbol{h}_v\right)\right) \tag{13}$$

Graph Classification The representation on the whole graph is constructed via a readout layer on the outputs $\mathbf{H}$,

$$\begin{aligned}\boldsymbol{h}_v &= \sigma\left(f_1\left(\boldsymbol{h}_v\right)\right) \odot \tanh\left(f_2\left(\boldsymbol{h}_v\right)\right) \\ \boldsymbol{h}_g &= \frac{1}{|\mathcal{V}|}\sum_{v=1}^{|\mathcal{V}|}\boldsymbol{h}_v + \text{Maxpool}\left(\boldsymbol{h}_1 \cdots \boldsymbol{h}_{|\mathcal{V}|}\right)\end{aligned} \tag{14}$$

where $\odot$ denotes the Hadamard product and $f_1()$, $f_2()$ are two non-linear functions.

The graph representation $\boldsymbol{h}_g \in \mathbb{R}^d$ is fed into the *Softmax* classifier to predict the graph label.

$$\widehat{y}_g = \text{Softmax}\left(\boldsymbol{h}_g\right) \tag{15}$$

All parameters are optimized by minimizing the cross-entropy function:

$$\mathcal{L} = -\frac{1}{N}\sum_{n=1}^{N} y_{n/g}\log(\widehat{y}_{n/g}) \tag{16}$$

4 Experiments

In this section, we evaluate HDGCN on *transductive* and *inductive* NLP tasks of text classification, aspect-based sentiment classification, natural language inference, and node classification. In experiment, each layer of HDGCN is fixed with $K = 6$ order Chebyshev approximation and the model stacks $L = 1$ layer. The dimension of input node embeddings is $d = 300$ of GlVe or $d = 768$ of pre-trained BERT, and the hyper-parameter $d_k = 64$, $d_a = 64$. So the weights $\mathbf{W}^{(0)} \in \mathbb{R}^{d \times 64}$, $\mathbf{W}_d^l, \mathbf{W}^{(k)} \in \mathbb{R}^{64 \times 64}$ and $\mathbf{W}_{fk}, \mathbf{W}_{fq}, \mathbf{W}_{bq}, \mathbf{W}_{bk} \in \mathbf{R}^{64 \times 64}$. The number of super-nodes is set as $M = 10$. Our model is optimized with adaBelief (Zhuang et al., 2020) with a learning rate $1e-5$. The schematics about the HDGCN is shown in Figure 2.

To analyze the effectiveness of MVCAttn in avoiding over-smoothing, we report the results of ablation study - HDGCN-*static* in Table 1, 2 5. The ablation model - HDGCN-*static* is an implementation of Equation 5, in which the node state transition is determined by the static adjacency matrix $\widetilde{\mathbf{A}}^{2k}$.

4.1 Text Classification

The first experiment is designed to evaluate the performance of HDGCN on the text graph classification. Four small-scale text datasets[2] - MR, R8, R52, Ohsumed, and four large-scale text datasets - AG's News[3], SST-1, SST-2[4], Yelp-F[5] are used in this task. The graph structures are built on word-word co-occurrences in a sliding window

[2]https://github.com/yao8839836/text_gcn
[3]http://groups.di.unipi.it/ gulli/AG_corpus_of_news_articles.html
[4]https://nlp.stanford.edu/sentiment/treebank.html
[5]https://www.yelp.com/dataset

(width=3 and unweighted) on individual documents. HDGCN is initialized with word embeddings pre-trained by 300-d GloVe and 768-d BERT-base on small and large scale datasets respectively. The baselines include TextCNN, TextRNN, fastText, SWEM, TextGCN, GraphCNN, TextING, minCUT, BERT-base, DRNN, CNN-NSU, CapNets, LK-MTL, TinyBERT, Star-Transformer.

Model	MR	R8	R52	Ohsumed
TextCNN*	77.75	95.71	87.59	58.44
TextRNN*	77.68	96.31	90.54	49.27
fastText*	75.14	96.13	92.81	57.70
SWEM*	76.65	95.32	92.94	63.12
TextGCN*	76.74	97.07	93.56	68.36
GraphCNN*	-	97.80	94.60	69.40
minCUT (Bianchi et al., 2019)	76.52	97.42	93.53	66.37
TextING (Zhang et al., 2020b)	79.82	98.04	95.48	70.42
BERT-base (Jin et al., 2019)	85.80	97.92	96.37	71.04
HDGCN-*static*	79.70	98.05	95.49	70.75
HDGCN	**86.50**	**98.45**	**96.57**	**73.97**

Table 1: Test accuracy (%) on small-scale English datasets, where the results labeled with $\star$ are cited from (Zhang et al., 2020b).

Model	AG	SST-1	SST-2	Yelp-F
fastText (Joulin et al., 2017)	92.5	-	-	63.9
DRNN (Wang, 2018)	93.6	47.3	86.4	65.3
CNN-NSU (Li et al., 2017)	-	50.8	89.4	-
CapNets (Yang et al., 2018)	92.6	-	86.8	-
LK-MTL (Xiao et al., 2018)	-	49.7	88.5	-
BERT (Xiao et al., 2018)	94.5	50.1	89.3	65.8
TinyBERT (Jiao et al., 2020)	94.7	51.6	**92.6**	66.1
Star-Transformer (Guo et al., 2019)	-	52.9	-	-
HDGCN-*static*	94.0	52.1	90.8	65.2
HDGCN	**95.5**	**53.9**	92.3	**69.6**

Table 2: Test accuracies (%) on large-scale English datasets.

Table 1 shows the test accuracies on four small-scale English datasets, in which HDGCN ranks top with accuracies 86.50%, 98.45%, 96.57%, 73.97% respectively. HDGCN beats the best baselines achieved by TextING (the newest GNN model) and the fine-tuned BERT-base model. Our ablation model HDGCN-*static* also achieves higher accuracies than the newest GNN models - TextING and minCUT. Therefore, the outperformance of HDGCN verifies that (1) the node probability transition in high-order Chebyshev approximation improves the spectral graph convolution; (2) the MVCAttn mechanism in high-order ChebNet further raises the effectiveness by avoiding the over-smoothing problem.

Table 2 shows the test accuracies of HDGCN and other SOTA models on large-scale English datasets. HDGCN achieves the best results 95.5%, 53.9%, 69.6% on AG, SST-1, Yelp-F respectively, and performs a slight gap 0.3% with the top-1 baseline (TinyBERT) on SST-2. These results support that HDGCN outperforms the fully-connected graph module in Transformers and corresponding pre-trained models. Additionally, these comparisons also demonstrates that the combination of prior graph structures and self-adaptive graph structures in graph convolution is able to improve the multi-hop graph reasoning.

4.2 Multi-hop Graph Reasoning in Text Graph

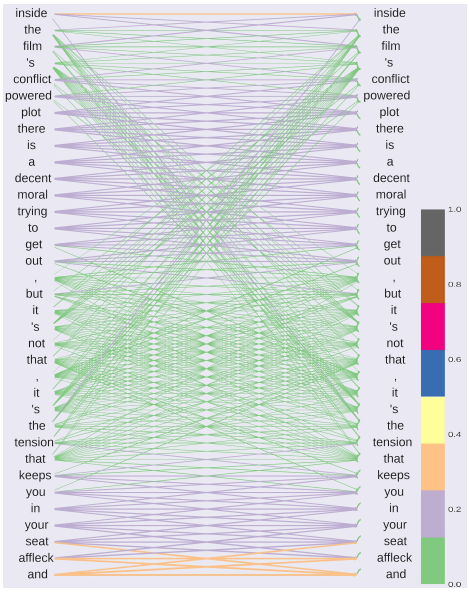

Figure 3: The message aggregation on adjacency matrix $\widetilde{\mathbf{A}}$ with word-word co-occurrence in document.

In the second experiment, we make a case study on the MR dataset to visualize how the HDGCN improve multi-hop graph reasoning. Here, we take the positive comment "*inside the film's conflict powered plot there is a decent moral trying to get out, but it's not that , it's the tension that keeps you in your seat Affleck and Jackson are good sparring partners*" as an example.

First, the word interactions on prior graph structure $\widetilde{\mathbf{A}}$ (word-word co-occurrence in a sliding window with width=3) is showed in Figure 3. We can see that the word mainly interacts with its consecutive neighbors. It is hard for the vanilla GCN to encode multi-hop and non-consecutive word-word interactions as the example shown in Figure 1.

Figure 4 shows the node interactions from node embeddings $\mathbf{Z}^{(0)}$ to supernodes $\widehat{\mathbf{S}}^{(1)}$ and the graph diffusion from $\widehat{\mathbf{S}}^{(1)}$ to node embeddings $\mathbf{Z}^{(1)}$. In which, the supernode $S4$ puts greater attention on the segment - *it's the tension that keeps you in your seat*. This segment determines its positive

Initialized embeddings	Model	TWITTER		LAP14		REST14		REST15		REST16	
		Acc.	F1.	Acc.	F1.	Acc.	F1.	Acc.	F1.	Acc.	F1
GloVe	AOA⋆	72.30	70.20	72.62	67.52	79.97	70.42	78.17	57.02	87.50	66.21
	TNet-LF⋆	72.98	71.43	74.61	70.14	80.42	71.03	78.47	59.47	89.07	70.43
	ASCNN⋆	71.05	69.45	72.62	66.72	81.73	73.10	78.47	58.90	87.39	64.56
	ASGCN-DT⋆	71.53	69.68	74.14	69.24	80.86	72.19	79.34	60.78	88.69	66.64
	ASGCN-DG⋆	72.15	70.40	75.55	71.05	80.77	72.02	79.89	61.89	88.99	67.48
BERT-base	AEN-BERT (Song et al., 2019)	-	-	79.93	76.31	83.12	73.76	-	-	-	-
	BERT-PT (Xu et al., 2019)	-	-	78.07	75.08	84.95	76.96	-	-	-	-
	SDGCN-BERT (Zhao et al., 2020)	-	-	**81.35**	**78.34**	83.57	76.47	-	-	-	-
	HDGCN (GloVe)	**73.41**	**71.52**	76.80	73.18	80.43	70.74	**81.18**	**67.40**	**89.12**	70.37
	HDGCN (BERT-base)	72.69	71.23	79.15	75.48	**85.89**	**79.33**	**81.18**	62.21	87.99	**71.28**

Table 3: Test accuracy (%) and macro-F1 score on aspect-based sentiment classification. The results labeled with ⋆ are cited from (Zhao et al., 2020).

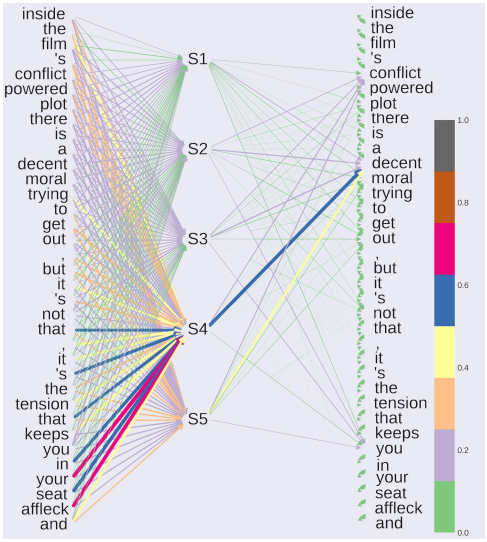

Figure 4: The word interactions in MVCAttn ($\mathbf{A}_f^{(1)} \times \mathbf{A}_b^{(1)}$) of the 1st HD-ChebNet, where $S1 \sim S5$ represent the supernodes.

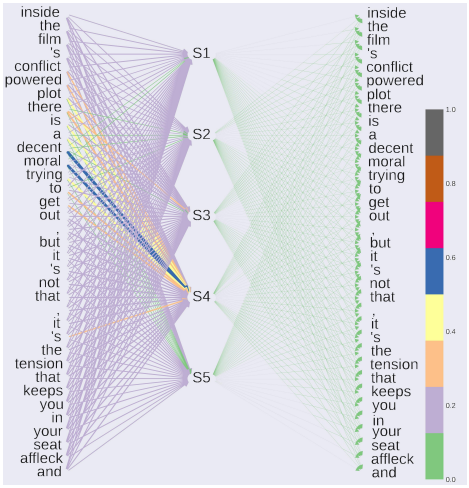

Figure 5: The word interactions in MVCAttn $\mathbf{A}_f^{(2)} \times \mathbf{A}_b^{(2)}$ of the 2nd HD-ChebNet, where $S1 \sim S5$ represent the supernodes.

polarity significantly. The other supernodes $S1$, $S2$, $S3$, $S5$ just aggregate messages from the global context evenly. Next, the messages aggregated in supernodes $S1 \sim S5$ are mainly diffused to four tokens - *conflict*, *decent*, *moral*, *you*. That verifies the self-adaptively learned graph structure $\mathbf{A}_f^{(1)} \times \mathbf{A}_b^{(1)}$ by the MVCAttn improves the multi-hop graph reasoning on nodes - *conflict, decent, moral, you*. From the perspective of semantics, these four words determine the positive sentiment in this comment significantly.

Figure 5 shows the message aggregation from node embeddings $\mathbf{Z}^{(1)}$ to supernodes $\widehat{\mathbf{S}}^{(2)}$ and the message diffusion from $\widehat{\mathbf{S}}^{(2)}$ to node embeddings $\mathbf{Z}^{(2)}$. We can see that the supernode $S4$ puts greater attention on another segment - *there is a decent moral young to get out*, which also contributes to the sentiment polarity. Then messages aggregated to supernodes $S1 \sim S5$ are diffused to all words evenly. The backward interactions from supernodes $S1 \sim S5$ to all graph nodes do not have visible differences. These results demonstrate that the multi-hop graph reasoning in HDGCN just needs one graph convolutional layer to attain the stationary state.

4.3 Aspect-based Sentiment Classification

The third experiment evaluates HDGCN's performance on the task of aspect-based sentiment classification. This task aims to identify whether the sentiment polarities of aspect are explicitly given in sentences (Zhao et al., 2020). The datasets used in this task include TWITTER, LAP14, REST14, REST15, REST16 (Zhao et al., 2020). The details about the statistics on these datasets are shown in Figure 6. The SOTA comparison models include AOA, TNet-LF, ASCNN, ASGCN-DT, ASGCN-DG, AEN-BERT, BERT-PT, SDGCN-BERT.

Each sample in this task includes a sentence pair, an aspect, and a label. The sentence pair and the aspect are concatenated into one long sentence, and the text graph is preprocessed with the dependency

tree on this sentence. HDGCN is tested twice with word embeddings initialized by pre-trained 300-d GloVe and 768-d BERT-base respectively.

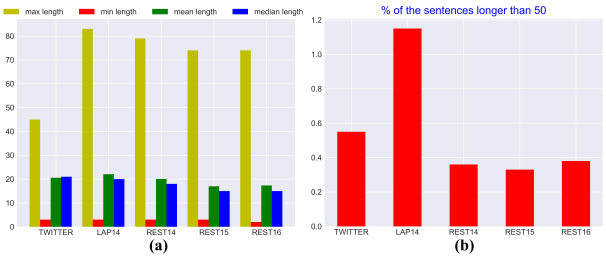

Figure 6: **(a):** The statistics of aspect-based sentiment classification datasets. **(b):** The percentages of sentences with length ≥ 50 in 5 datasets.

Table 3 shows the test accuracies and micro-F1 scores on 5 datasets, where HDGCN achieves new state-of-the-art results on TWITTER, REST14, REST15, REST16, and a top-3 result on the LAP14. As shown in Figure 6 that the LAP14 has the maximum percentage of long sentences among all datasets. A shallow network in HDGCN does not outperform the SOTA result on the LAP14. Additionally, compared with the newest ASGCN and attention-based AOA, HDGCN achieves the best results on TWITTER, LAP14, REST15, REST16 (Acc) and performs very close with the highest accuracy on REST14 and macro-F1 score on REST16. Above comparison supports that the matching between aspect and sentence pair in HDGCN is more accurate than the newest GNN and attention-based models, which verifies that the multi-hop graph reasoning is improved in HDGCN.

4.4 Natural Language Inference

The fourth experiment evaluates HDGCN's performance on the Stanford natural language inference (SNLI) task (Bowman et al., 2015). This task aims to predict the semantic relationship is *entailment* or *contradiction* or *neutral* between a premise sentence and a hypothesis sentence. All the comparison methods include fine-tuned BERT-base, MT-DNN (Liu et al., 2020), SMART (Jiang et al., 2020), and CA-MTL (Pilault et al., 2021).

In this task, the premise and hypothesis sentences are concatenated and constructed into a long sentence. Which is preprocessed to a text graph with the dependency tree. The word embeddings in HDGCN were initialized from the pre-trained 768-d BERT-base.

All test accuracies are shown in Table 4, where HDGCN achieves the new state-of-the-art results

Model	Total parameters	% data used		
		0.1%	1.0%	10%
BERT-base (Devlin et al., 2019)	1.0×	52.5	78.1	86.7
MT-DNN (Liu et al., 2020)	-	81.9	**88.3**	91.1
SMART (Jiang et al., 2020)	-	82.7	86.0	88.7
CA-MTL (Pilault et al., 2021)	1.12×	**82.8**	86.2	88.0
HDGCN	1.02×	80.3	85.6	**92.3**

Table 4: Test accuracy (%) on SNLI, where the total parameters take the BERT-base as base.

on the 10% data. As the MT-DNN, SMART and CA-MTL are all fine-tuned on multi-task learning, they perform better than HDGCN in low resource regimes (0.1% and 1.0% of the data). HDGCN just uses 0.02× more parameters than the BERT-base, and it outperforms the later model on all scales of data. These results verify that the combination of prior graph structure and self-adaptive graph structure in HDGCN performs comparably with the fully-adaptive graph structures in Transformers and BERT-based multi-task learning models.

4.5 Graph Node Classification

The fifth experiment evaluates the effectiveness of HDGCN on the node classification task. We use three standard citation network benchmark datasets - Cora, Citeseer, and Pubmed, to compare the test accuracies on *transductive* node classification. In the three datasets, the nodes represent the documents and edges (undirected) represent citations. The node features correspond to elements of a bag-of-words representation of a document (Veličković et al., 2018). We also use the PPI dataset to compare the results on *inductive* node classification, which consists of graphs corresponding to different human tissues. The baselines for comparison include GCN, GAT, Graph-Bert, GraphNAS, LoopyNet, HGCN, GRACE, GCNII. The results of our evaluation are recorded in Table 5.

	Transductive (ACC, %)			***Inductive*** (micro-F1)
Model	Cora	Citeseer	Pubmed	PPI
GCN (Kipf and Welling, 2017)	85.8	73.7	88.1	69.7
GAT (Veličković et al., 2018)	86.4	74.3	87.6	97.3
Graph-Bert (Zhang et al., 2020a)	84.3	71.2	79.3	-
GraphNAS (Gao et al., 2019)	84.2	73.1	79.6	98.6
LoopyNet (Zhang and Meng, 2019)	83.9	73.7	83.0	-
HGCN (Chami et al., 2019)	79.9	-	80.3	74.6
GRACE (Zhu et al., 2020)	83.3	72.1	86.7	96.9
GCNII (Chen et al., 2020)	86.4	76.5	85.6	**99.5**
HDGCN-*static*	84.2	73.2	90.3	50.4
HDGCN	**88.6**	**77.0**	**91.0**	**99.5**

Table 5: Test accuracy (%) on Cora, Citeseer, Pubmed and micro-F1 score (%) on PPI.

HDGCN achieves the new state-of-the-art results on Cora, Citeseer and Pubmed, and performs equally best with the newest GCNII on PPI. Our

ablation model, HDGCN-*static*, also achieves close results with the newest GNNs on Cora, Citeseer, Pubmed, but it performs poorly on PPI. Which verifies that the high-order Chebyshev approximation of spectral graph convolution has more serious over-smoothing problem in *inductive* node classification than *transductive* node classification. All comparisons in this experiment demonstrate the effectiveness of MVCAttn to avoid the over-smoothing problem.

5 Conclusions

This study proposes a multi-hop graph convolutional network on high-order dynamic Chebyshev approximation (HDGCN) for text reasoning. To improve the multi-hop graph reasoning, each convolutional layer in HDGCN fuses low-pass signals (direct dependencies saved in fixed graph structures) and high-pass signals (multi-hop dependencies adaptively learned by MVCAttn) simultaneously. We also firstly propose the multi-votes based cross-attention (MVCAttn) mechanism to alleviate the over-smoothing in high-order Chebyshev approximation, and it just costs the linear computation complexity. Our experimental results demonstrate that HDGCN outperforms compared SOTA models on multiple transductive and inductive NLP tasks.

Acknowledgments

This work is supported by Natural Science Foundation of China (Grant No.61872113, 62006061), Strategic Emerging Industry Development Special Funds of Shenzhen (Grant No.XMHT20190108009), the Tencent Group Science and Technology Planning Project of Shenzhen (Grant No.JCYJ20190806112210067) and Shenzhen Foundational Research Funding (Grant No.JCYJ20200109113403826).

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. Etc: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284.

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. 2019. Mincut pooling in graph neural networks. *arXiv preprint arXiv:1907.00481*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Ines Chami, Rex Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. *In NeurIPS*, 32:4869.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *In ICML*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *In NAACL*.

Vijay Prakash Dwivedi and Xavier Bresson. 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*.

Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2019. Graphnas: Graph neural architecture search with reinforcement learning. *arXiv preprint arXiv:1904.09981*.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer. In *In NAACL*, pages 1315–1325.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *In ACL*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *In EMNLP*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is bert really robust? natural language attack on text classification and entailment. *arXiv preprint arXiv:1907.11932*, 2.

Armand Joulin, Édouard Grave, Piotr Bojanowski, and Tomáš Mikolov. 2017. Bag of tricks for efficient text classification. In *In EACL*, pages 427–431.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *In ICLR*.

John Boaz Lee, Ryan A Rossi, Sungchul Kim, Nesreen K Ahmed, and Eunyee Koh. 2019. Attention models in graphs: A survey. *In TKDD*, 13(6):1–25.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Shen Li, Zhe Zhao, Tao Liu, Renfen Hu, and Xiaoyong Du. 2017. Initializing convolutional filters with semantic features for text classification. In *In EMNLP*.

Xiaodong Liu, Yu Wang, Jianshu Ji, Hao Cheng, Xueyun Zhu, Emmanuel Awa, Pengcheng He, Weizhu Chen, Hoifung Poon, Guihong Cao, et al. 2020. The microsoft toolkit of multi-task deep neural networks for natural language understanding. In *In ACL*.

Hoang Nt and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*.

Jonathan Pilault, Amine Elhattami, and Christopher Pal. 2021. Conditionally adaptive multi-task learning: Improving transfer learning in nlp using fewer parameters & less data. *In ICLR*.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. Dropedge: Towards deep graph convolutional networks on node classification. In *In ICLR*.

Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. 2019. Attentional encoder network for targeted sentiment classification. *arXiv preprint arXiv:1902.09314*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *In NIPS*, pages 6000–6010.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

Baoxin Wang. 2018. Disconnected recurrent neural networks for text categorization. In *In ACL*.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuan-Jing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. In *In ACL*.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *In EMNLP*.

Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. Learning what to share: Leaky multi-task network for text classification. In *In Coling*.

Hu Xu, Bing Liu, Lei Shu, and Philip Yu. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. In *In NACL*.

Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. 2018. Investigating capsule networks with dynamic routing for text classification. In *In EMNLP*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *In AAAI*, volume 33, pages 7370–7377.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. *Advances in Neural Information Processing Systems*.

Chen Zhang, Qiuchi Li, and Dawei Song. 2019a. Aspect-based sentiment classification with aspect-specific graph convolutional networks. In *In EMNLP-IJCNLP*.

Haopeng Zhang and Jiawei Zhang. 2020. Text graph transformer for document classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8322–8327, Online. Association for Computational Linguistics.

Jiawei Zhang. 2020. Get rid of suspended animation problem: Deep diffusive neural network on graph semi-supervised classification. *arXiv preprint arXiv:2001.07922*.

Jiawei Zhang and Lin Meng. 2019. Gresnet: Graph residual network for reviving deep gnns from suspended animation. *CoRR*, abs/1909.05729.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020a. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*.

Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Ustebay. 2019b. Bayesian graph convolutional neural networks for semi-supervised classification. In *In AAAI*, volume 33, pages 5829–5836.

Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. 2020b. Every document owns its structure: Inductive text classification via graph neural networks. In *In ACL*.

Zhengchao Zhang, Meng Li, Xi Lin, and Yinhai Wang. 2020c. Network-wide traffic flow estimation with insufficient volume detection and crowdsourcing data. *Transportation Research Part C: Emerging Technologies*, 121:102870.

Pinlong Zhao, Linlin Hou, and Ou Wu. 2020. Modeling sentiment dependencies with graph convolutional networks for aspect-level sentiment classification. *Knowledge-Based Systems*.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. 2020. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *NeurIPS*, 33.

A Appendices

Here, we give the completed proof about our high-order Chebyshev approximation on the spectral graph convolution. We exhibit how to deduce the spectral graph convolution to 4th-order Chebyshev polynomials as follows.

$$\boldsymbol{g_\theta} \star \boldsymbol{x} = \mathbf{U}\boldsymbol{g_\theta}\mathbf{U}^T\boldsymbol{x}$$

where $\boldsymbol{g_\theta} = \boldsymbol{g_\theta}(\widetilde{\mathbf{\Lambda}})$ is the graph filter defined in spectral domain.

$$\begin{aligned}\boldsymbol{g_\theta} &\approx \boldsymbol{\theta}_0 + \boldsymbol{\theta}_1\widetilde{\mathbf{\Lambda}} + \boldsymbol{\theta}_2\left(2\widetilde{\mathbf{\Lambda}}^2 - 1\right) + \boldsymbol{\theta}_3\left(4\widetilde{\boldsymbol{\Lambda}}^3 - 3\widetilde{\mathbf{\Lambda}}\right)\\ &= \boldsymbol{\theta}_0 + \boldsymbol{\theta}_1\widetilde{\mathbf{\Lambda}} + 2\boldsymbol{\theta}_2\widetilde{\mathbf{\Lambda}}^2 - \boldsymbol{\theta}_2 + 4\boldsymbol{\theta}_3\widetilde{\mathbf{\Lambda}}^3 - 3\boldsymbol{\theta}_3\widetilde{\mathbf{\Lambda}}\end{aligned}$$

So,

$$\begin{aligned}\mathbf{U}\boldsymbol{g_\theta}\mathbf{U}^{\mathrm{T}}\boldsymbol{x} &\approx \boldsymbol{\theta}_0\boldsymbol{x} + \boldsymbol{\theta}_1\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\boldsymbol{x} + 2\boldsymbol{\theta}_2\mathbf{U}\widetilde{\mathbf{\Lambda}}^2\mathbf{U}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{\theta}_2\boldsymbol{x} + 4\boldsymbol{\theta}_3\mathbf{U}\widetilde{\mathbf{\Lambda}}^3\mathbf{U}^{\mathrm{T}}\boldsymbol{x} - 3\boldsymbol{\theta}_3\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\boldsymbol{x}\\ &= \boldsymbol{\theta}_0\boldsymbol{x} + \boldsymbol{\theta}_1\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\boldsymbol{x} + 2\boldsymbol{\theta}_2\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{\theta}_2\boldsymbol{x} + 4\boldsymbol{\theta}_3\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\boldsymbol{x} - 3\boldsymbol{\theta}_3\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\boldsymbol{x}\\ &= \mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}(\frac{\boldsymbol{\theta}_0}{\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}} + \frac{\boldsymbol{\theta}_1}{\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}} + 2\boldsymbol{\theta}_2 - \frac{\boldsymbol{\theta}_2}{\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}} + 4\boldsymbol{\theta}_3\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}} - 3\frac{\boldsymbol{\theta}_3}{\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}})\boldsymbol{x}\\ &= \mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\left(\frac{\boldsymbol{\theta}_0 - \boldsymbol{\theta}_2}{\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}} + \frac{\boldsymbol{\theta}_1 - 3\boldsymbol{\theta}_3}{\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}}\right)\boldsymbol{x} + \mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\left(2\boldsymbol{\theta}_2 + 4\boldsymbol{\theta}_3\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\right)\boldsymbol{x}\\ &= \left((\boldsymbol{\theta}_0 - \boldsymbol{\theta}_2) + (\boldsymbol{\theta}_1 - 3\boldsymbol{\theta}_3)\,\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\right)\boldsymbol{x} + \mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\left(2\boldsymbol{\theta}_2 + 4\boldsymbol{\theta}_3\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\right)\boldsymbol{x}\end{aligned}$$

Let assume $\boldsymbol{\theta}^{(0)} = \boldsymbol{\theta}_0 - \boldsymbol{\theta}_2 = -\boldsymbol{\theta}_1 + 3\boldsymbol{\theta}_3$, $\boldsymbol{\theta}^{(1)} = 2\boldsymbol{\theta}_1 = -4\boldsymbol{\theta}_3$,

$$\begin{aligned}\mathbf{U}\boldsymbol{g_\theta}\mathbf{U}^{\mathrm{T}}\boldsymbol{x} &\approx \boldsymbol{\theta}^{(0)}\left(\mathbf{I} - \mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\right)\boldsymbol{x} + \mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\boldsymbol{\theta}^{(1)}\left(\mathbf{I} - \mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\mathrm{T}}\right)\boldsymbol{x}\\ &\approx \boldsymbol{\theta}^{(0)}\widetilde{\mathbf{A}}\boldsymbol{x} + \widetilde{\mathbf{A}}^2\boldsymbol{\theta}^{(1)}\widetilde{\mathbf{A}}\boldsymbol{x}\end{aligned}$$

To avoid the over-smoothing problem in the node state transition $\widetilde{\mathbf{A}}^2$, the graph structure $\widetilde{\mathbf{A}}$ is approximated by the dynamic adjacency matrix $\mathbf{A}_d$ self-adaptively learned with attention mechanism. This way have the hidden pairwise interactions to improve the multi-hop graph reasoning in high-order Chebyshev polynomials. Therefore, we define the layer-wise propagation of multi-hop graph convolutional network as follows.

$$\begin{aligned}\mathbf{H} &\approx \sum_{k=0}^{K/2}\mathbf{Z}^{(k)},\\ \mathbf{Z}^{(0)} &= \underbrace{\sigma\left(\widetilde{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}\right)}_{\text{Prime ChebNet}},\\ \mathbf{Z}^{(k)} &= \underbrace{\sigma\left(\widetilde{\mathbf{A}}\left(\mathbf{A}_d^{(k)}\mathbf{Z}^{(k-1)}\mathbf{W}_d^{(k)}\right)\mathbf{W}^{(k)}\right)}_{\text{HD-ChebNet}}\end{aligned}$$

where $\mathbf{X}$ is the input features, and we introduce two nonlinear filterings $\mathbf{W}^{(k)}$ and $\mathbf{W}_d^{(k)}$ on node signals.