2021 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, OCT. 25–28, 2021, GOLD COAST, AUSTRALIA

DYNAMIC GRAPH CONVOLUTIONAL NETWORK: A TOPOLOGY OPTIMIZATION PERSPECTIVE

Bowen Deng, Aimin Jiang

College of Internet of Things Engineering, Hohai University, ChangZhou, China

ABSTRACT

Recently, graph convolutional networks(GCNs) have drawn increasing attention in many domains, *e.g.*, social networks, recommendation systems. It's known that, in the task of graph node classification, inter-class edges connecting nodes from different categories often degrade the GCN model performance. On the other hand, a stronger intra-class connection in terms of the edge number and edge weights is always beneficial to node classification. Most existing GCN models assume that the topology and edge weights of the underlying graph are both fixed. However, real-world networks are often noisy and incomplete. To take into account such uncertainty in graph topology, we propose in this paper a dynamic graph convolution network (DyGCN), where edge weights are treated as learnable parameters. A novel adaptive edge dropping (AdaDrop) strategy is developed for DyGCN, such that even graph topology can be optimized. DyGCN is also a flexible architecture that can be readily combined with other deep GCN models to cope with the oversmoothness encountered when the network goes very deep. Experimental results demonstrate that the proposed DyGCN and its deep variants can achieve competitive classification accuracy in many datasets.

***Index Terms*—** Graph deep learning, supervised learning, graph node classification, graph topology optimization

1. INTRODUCTION

GCN [1] and many other graph neural networks [2, 3, 4] suffer from the notorious oversmoothing issue [5] arisen when the model depth goes deep, which drives the node features so indistinguishable that classifiers fail to work. Many efforts have been made to overcome it. JKNet proposed in [6] combines the representations learnt from all layers by max or concatenate operation for each node. Inspired by ResNet [7] and APPNP [8], the GCNII layer is proposed in [9], which employs *initial residual connection* and *identity mapping* to prevent feature collapse. This kind of works aim to design deep architectures and pay no attention to the underlying graph topology, though it's closely related to the oversmoothness issue.

The *information-to-noise ratio* (INR) [10] is defined to reflect the oversmoothness, that is, the proportion of intra-class node pairs out of all contactable node pairs that have interactions through the entire GCN model. It has been empirically shown that *topology optimization* can effectively increase the INR and alleviate the oversmoothing issue. But, only hard operations (*e.g.*, removing inter-class edges and adding intra-class edges) are considered in [10]. On the other hand, GCN-LPA [11] seeks to utilize label propagation module to modify the underlying graph with soft operations (*e.g.*, increasing the weights of intra-class edges and decreasing those of inter-class ones). In GCN-LPA, the updates of edge weights and GCN parameters are executed in turn and label propagation module serves as a regularizer of graph topology.

In this paper, we propose a dynamic GCN layer – DyGCN. In the presented architecture, edge weights are regarded as trainable parameters and used to adjust the adjacency matrix(and hence the propagation matrix) of graph. Based on this, we further propose a hard topology optimization strategy, that adaptively drops possible inter-class edges according to learnt edge scores. In contrast to GCN-LPA, no extra modules are required, such that DyGCN is much easier to be integrated with existing GCN models. Our contributions presented in this paper are summarized as follows: **i**. A novel DyGCN architecture is proposed for the task of node classification. Edge weights and linear transform parameters are trained simultaneously, such that the graph topology is no longer fixed; **ii**. A theoretical analysis of DyGCN is made in terms of both expressive ability and potentials to perform a soft topology optimization. Furthermore, the core mechanism of DyGCN is transferred into JKNet and GCNII, giving birth to JKNet(Dy) and GCNII(Dy), respectively; **iii**. A novel hard topology optimization strategy is also developed to adaptively remove noisy edges(*i.e.*, inter-class edges connecting nodes from different categories).

This work was supported in part by the National Key Research and Development Program 2018AAA0100800, the National Nature Science Foundation of China under grant 61801055, the Fundamental Research Funds for the Central Universities of China under grants 2018B23014 and 2018B47114, and the Key Development Program of Jiangsu Province of China under grants BE2017071, BE2017647, and BE2018004-04.

978-1-7281-6338-3/21/$31.00 ©2021 IEEE

2. DYNAMIC GCN

2.1. DyGCN Layer

Given an undirected and connected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ consisting of N nodes and M edges, we denote its adjacency and degree matrices by $\mathbf{A}$ and $\mathbf{D}$. Its augmented adjacency and degree matrices are given, respectively, by $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$. The famous vanilla GCN [1] is then formulated as:

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\tilde{\mathbf{P}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}\right), \tag{1}$$

where $\tilde{\mathbf{P}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ is a fixed propagation matrix recomputed before training and inference, $\mathbf{H}^{(\ell)}$ denotes the input features to the ℓ-th GCN layer, $\mathbf{W}^{(\ell)}$ is the weight matrix of the ℓ-th layer, and $\sigma(\cdot)$ represents an activation function.

Naive DyGCN is more a framework than a novel layer since only the propagation matrix is different from traditional GCNs, as shown by the l-th layer formula below.

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\mathbf{P}^{(\ell)}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}\right), \tag{2}$$

$$\mathbf{P}^{(\ell)} = \text{SparseSoftMax}\left(\text{LeakyReLU}\left(\tilde{\mathbf{A}}^{(\ell)}\right)\right). \tag{3}$$

It can be readily verified that $\mathbf{P}^{(\ell)}$ is a right stochastic matrix, whose rows are obtained by the sparse softmax function applied on the corresponding rows of the augmented adjacency matrix $\tilde{\mathbf{A}}^{(\ell)}$. It's sparse since only nonzero elements of $\tilde{\mathbf{A}}^{(\ell)}$ are involved in this operation. Another noticeable feature of (2) is that $\tilde{\mathbf{A}}^{(\ell)}$ is learnable and its entries is no longer required to be nonnegative values as softmax will normalize the propagation weights.

2.2. Deep Variants

Though many experiments(see Section3.3) demonstrate the better robustness to oversmoothness than other shallow GCN models, *e.g.*, GCN and GCN-LPA [11], DyGCN still incurs this issue when models go very deep. As the GCN w/o ReLU activation is proven both empirically [12] and theoretically [5] more resistant to oversmoothness than GCN w/ ReLU, and DyGCN has the same aggregation scheme(2) with GCN(1), we investigate the expressive ability of linear DyGCN, *i.e.*, DyGCN w/o ReLU, to show the limitation of naive deep DyGCN. Formally, a L layer linear DyGCN

$$\mathbf{H}^{(L+1)} = \mathbf{P}^{(L)} \cdots \mathbf{P}^{(0)}\mathbf{H}^{(0)}\mathbf{W}^{(0)} \cdots \mathbf{W}^{(L)}. \tag{4}$$

is constrained by the following theorem.

Theorem 1. *(Linear DyGCN's expressive ability theorem) If a L-layer DyGCN model is linear and the Markov chain corresponding to $\mathbf{P}^{(\ell)}$ is ergodic, the final output $\mathbf{H}^{(L)}$ converges to a rank-1 matrix as $L \to \infty$.*

Proof. Since $\mathbf{P}^{(\ell)}$ is a right stochastic matrix, that has eigenvalue 1 with the corresponding eigen vector $\mathbf{1}$. If the Markov chain corresponding to $\mathbf{P}^{(\ell)}$ is ergodic, we have a unique eigen vector with eigenvalue 1 and the modulus of the rest of eigenvalues is less than 1. To proceed, we further define

$$\mathbf{Q}^{(\ell)} = \mathbf{P}^{(\ell)} - \frac{1}{N}\mathbf{1}\cdot\mathbf{1}^\top = \mathbf{P}^{(\ell)}\left(\mathbf{I} - \frac{1}{N}\mathbf{1}\cdot\mathbf{1}^\top\right). \tag{5}$$

From (5), we readily obtain that $\mathbf{Q}^{(\ell)}$ has a zero eigenvalue, which corresponds to eigen vector $\mathbf{1}$. Furthermore, $\mathbf{Q}^{(\ell)}$ share the rest of eigenvalues of $\mathbf{P}^{(\ell)}$ except 1. According to the Gershgorin circle theorem, the spectra radius $\rho\left(\mathbf{Q}^{(\ell)}\right)$ of $\mathbf{Q}^{(\ell)}$ is thus less than 1. This further implies

$$\lim_{L\to\infty}\rho\left(\prod_{\ell=0}^{L}\mathbf{Q}^{(\ell)}\right) \le \lim_{L\to\infty}\prod_{\ell=0}^{L}\rho\left(\mathbf{Q}^{(\ell)}\right) = 0. \tag{6}$$

Since $\rho(\cdot)$ measures the largest absolute value of eigenvalues of a square matrix, (6) also indicates that all the eigenvalues of $\prod_{\ell=0}^{L}\mathbf{Q}^{(\ell)}$ approach zeros as $L \to \infty$ and, accordingly, $\prod_{\ell=0}^{L}\mathbf{Q}^{(\ell)}$ finally becomes a zero matrix.

On the other hand, expanding $\prod_{\ell=0}^{L}\mathbf{Q}^{(\ell)}$ yields

$$\begin{aligned}\prod_{\ell=0}^{L}\mathbf{Q}^{(\ell)} &= \prod_{\ell=0}^{L}\left(\mathbf{P}^{(\ell)} - \frac{1}{N}\mathbf{1}\cdot\mathbf{1}^\top\right)\\ &= \prod_{\ell=0}^{L}\mathbf{P}^{(\ell)} - \frac{1}{N}\mathbf{1}\cdot\mathbf{1}^\top\prod_{\ell=0}^{L-1}\mathbf{P}^{(\ell)},\end{aligned} \tag{7}$$

where we recursively apply $\mathbf{P}^{(\ell)}\cdot\mathbf{1} = \mathbf{1}$. Let $\mathbf{G}^{(L)}$ be $\prod_{\ell=0}^{L}\mathbf{P}^{(\ell)}$. Then, we obtain from the above equation

$$\begin{aligned}\lim_{L\to\infty}\prod_{\ell=0}^{L}\mathbf{Q}^{(\ell)} &= \left(\mathbf{I} - \frac{1}{N}\mathbf{1}\cdot\mathbf{1}^\top\right)\cdot\mathbf{G}^{(\infty)}\\ &= \mathbf{0}.\end{aligned} \tag{8}$$

Obviously, $\mathbf{G}^{(\infty)}$ is a rank-1 matrix. □

Theorem 1 motivates us to combine DyGCN with deep architectures for deep variants. Specifically, the fixed propagation matrix in JKNet and GCNII is replaced by (3). In the subsequent discussion, we refer to the resulting deep models as JKNet(Dy) and GCNII(Dy), separately.

2.3. DyGCN & Soft Topology Optimization

The proposed DyGCN is also trained by the back-propagation (BP) method. It can be demonstrated that, based on our architecture, the BP method can effectively enhance the intra-class connections and weaken the inter-class ones. We focus on the gradient of propagation matrix $\mathbf{P}^{(\ell)}$ instead of augmented adjacency matrix because it's more straightly related to feature

aggregation. To simplify the analysis, only the last propagation matrix employed to classify nodes is under consideration. With adopting the cross-entropy loss $\mathcal{L}_{gcn}$, it is readily to obtain

$$\begin{aligned}\frac{\partial \mathcal{L}_{gcn}}{\partial \mathbf{P}^{(L)}} &= \frac{\partial \mathcal{L}_{gcn}}{\partial \mathbf{H}^{(L+1)}} \left(\mathbf{H}^{(L)}\mathbf{W}^{(L)}\right)^\top \\ &= (\mathbf{Y}-\mathbf{T})\left(\mathbf{X}^{(L)}\right)^\top, \end{aligned} \tag{9}$$

where $\mathbf{X}^{(L)} = \mathbf{H}^{(L)}\mathbf{W}^{(L)}$ can be regarded as classification scores not propagated by $\mathbf{P}^{(L)}$ and its j-th row is denoted by $\mathbf{x}_j^{(L)}$, and $\mathbf{Y} \in [0,1]^{N\times C}$ is the predication matrix showing the possibilities of C classes for each node. Because we focus on the last layer, in the subsequent discussion, the superscript $^{(L)}$ will be omitted for the ease of notations. For the (i,j)-th element of $\frac{\partial \mathcal{L}_{gcn}}{\partial \mathbf{P}^{(L)}}$, (9) is reduced to

$$\left[\frac{\partial \mathcal{L}_{gcn}}{\partial \mathbf{P}^{(L)}}\right]_{ij} = (\mathbf{y}_i - \mathbf{t}_i)\cdot \mathbf{x}_j^\top. \tag{10}$$

Let c_i be the gold label corresponding to node i for $i = 1,\ldots,N$. Suppose that a neighbor j is correctly classified with high confidence, *i.e.*, $x_{j,c_j} \gg x_{j,k}$ for $k \neq c_j$, where $x_{j,k}$ denotes the k-th element of $\mathbf{x}_j$. In this context, we consider two scenarios.

1. If node i is of the same class, that is, $c_i = c_j$, from (10) we then have

$$\begin{aligned}\left[\frac{\partial \mathcal{L}_{gcn}}{\partial \mathbf{P}^{(L)}}\right]_{ij} &= x_{j,c_i}(y_{i,c_i}-1) + \sum_{k\neq c_i} x_{j,k}y_{i,k} \\ &= \sum_{k\neq c_i} y_{i,k}(x_{j,k}-x_{j,c_i}) < 0, \end{aligned} \tag{11}$$

   which tells that the intra-class propagation weight of edge from node j to i will be augmented in the next update, no matter whether node i is recognized correctly or not.

2. If nodes i and j are from different classes (*i.e.*, $c_i \neq c_j$) and node i is misclassified as c_j with a very high probability, meaning $y_{i,c_j} \gg y_{i,k}$ for $k \neq c_j$, (10) is turned to

$$\begin{aligned}\left[\frac{\partial \mathcal{L}_{gcn}}{\partial \mathbf{P}^{(L)}}\right]_{ij} &= x_{j,c_j}y_{i,c_j} + x_{j,c_i}(y_{i,c_i}-1) \\ &\quad + \sum_{k\neq c_i,c_j} x_{j,k}y_{i,k} \\ &= \left(x_{j,c_j}-x_{j,c_i}\right)y_{i,c_j} \\ &\quad + \sum_{k\neq c_i,c_j} y_{i,k}\left(x_{j,k}-x_{j,c_i}\right) \\ &\approx x_{j,c_j} - x_{j,c_i} > 0. \end{aligned} \tag{12}$$

   Under this scenario, the propagation weight $\left[\mathbf{P}^{(L)}\right]_{ij}$ will be whittled.

Although a clear conclusion cannot be reached except in the above situations, many experiments (see Section3.2) demonstrate that (**Hypothesis 1**)intra-(inter-) connection can be enhanced(weakened) to some extent by soft topology optimization via propagation matrix $\mathbf{P}^{(\ell)}$ in DyGCN, JKNet(Dy), and GCNII(Dy).

2.4. AdaDrop

Soft topology optimization does not abandon any element of $\mathbf{P}^{(\ell)}$. In practice, if some edges are identified as inter-class edges, they can be removed earlier. To this end, we further develop the adaptive dropping (AdaDrop) for DyGCN. Denoting the set of predecessors(or neighbors for undirected graph) of node i by $\mathcal{N}_1(i)$, the propagation weights of messages from its predecessors(neighbors) are initialized equal to $\frac{1}{|\mathcal{N}_1(i)|+1}$. After a number of epochs, some propagation weights become smaller than $\frac{1}{|\mathcal{N}_1(i)|+1}$. According to **Hypothesis 1**, the edges corresponding to these weaken edges are more likely to be inter-class ones and hence removed by us randomly. The resulting graph topology is then taken into the subsequent training. Experimental results reveal that removing all possible inter-class edges does not always lead to the improvement. Hence, a probability p is specified for AdaDrop to control the chance of a potential inter-class edge being removed. The details of AdaDrop are provided in Algorithm. 1.

3. EXPERIMENTS

3.1. Implementation, Datasets & Experimental Setting

DyGCN is built on pytorch_sparse[1], a third-party PyTorch extension library for sparse tensors. To incorporate dynamic propagation matrices, GCNII and JKNet are reimplemented using pytorch_sparse. The official implementation of vanilla GCN and GCN-LPA along with PyG [13] are adopted in our experiments. As mentioned before, assigning a differentiable propagation matrix for each layer is computationally expensive and the resulting model is vulnerable to the overfitting. Thus, we assume that L_{block} consecutive layers share one common propagation matrix. Furthermore, the L_1 regularization and weight decay are applied to $\tilde{\mathbf{A}}^{(\ell)}$s and $\mathbf{W}^{(\ell)}$s, separately.

Three standard citation network datasets, namely, Cora, Citeseer, and Pubmed [14] are used for full-supervised node classification. Each dataset is randomly split into training, validation, and test sets according to $6:2:2$. For every run, one model is trained for 300 epochs by Adam [15] on training set and the test accuracy when validation accuracy gets maximized is reported finally.

[1] https://github.com/rusty1s/pytorch_sparse

Algorithm 1: AdaDrop

Input : A L-layer dynamic GCN model gcn, the feature matrix $\mathbf{X}$; the gold label matrix $\mathbf{T}$; the adjacency matrix $\mathbf{A}$; the max times of hard topology optimization K; the drop rate $p \in (0, 1]$; the training epoches E.

Output : The best model gcn_{best} and the corresponding set S_{best} of differentiable adjacency matrices.

Initialize: Denote by S the set of all $\tilde{\mathbf{A}}^{(\ell)}$ s in Eq. A2P . Initialize them with the same weights and let $S_0 = \{\tilde{\mathbf{A}}_0^{(\ell)}\}_{\ell=1:L}$; $accv_{max} = 0$, $S_{best} = S_0, gcn_{best} = null$

Function `Main() /* the main function */`
  for $k = 1$ **to** K **do**
    $acc_{val}, S_k, gcn_k =$ `OneRun(`$gcn, S_{k-1}, \mathbf{T}, E$`)`;
    if $acc_{val} > accv_{max}$ **then**
      $accv_{max} = acc_{val}$;
      $S_{best} = S_k$; $gcn_{best} = gcn_k$;
    end
    $\mathbf{B}$=`HardTopoOptim(`S_k, p`)`;
    for $\ell = 1$ **to** L *parallel* **do**
      $\tilde{\mathbf{A}}_k^{(\ell)} = \tilde{\mathbf{B}}$;
    end
    $S_k = \{\tilde{\mathbf{A}}_k^{(\ell)}\}_{\ell=1:L}$;
  end
  return S_{best}, gcn_{best}
end

Function `OneRun(`$gcn, S, \mathbf{X}, \mathbf{T}, E$`) /* regular training */`
  Initialize gcn;
  Fit the model gcn to $\mathbf{X}, \mathbf{T}$ for E epoches;
  Load the checkpoint with the best validation accuracy acc_{val} to get optimal gcn^*, S^*;
  return acc_{val}, S^*, gcn^*
end

Function `HardTopoOptim(`S, p`) /* drop less important edges */`
  for $\ell = 1$ **to** L **do**
    $\mathbf{P}^{(\ell)} =$ SparseSoftMax $\left(\text{LeakyReLU}\left(\tilde{\mathbf{A}}^{(\ell)}\right)\right)$;
  end
  $\mathbf{B} = \frac{1}{L}\sum_\ell \mathbf{P}^{(\ell)}$;
  for $i = 1$ **to** N *parallel* **do**
    for $j \in \mathcal{N}_1(i)$ *parallel* **do**
      if $\mathbf{B}_{i,j} < 1/|\mathcal{N}_1(i)|$ **then**
        Set $\mathbf{B}_{i,j} = 0$ with a probability of p;
      end
    end
  end
  $\mathbf{B} = \mathbf{B} + \mathbf{B}^T$;
  return $\mathbf{B}$
end

3.2. Proposed Models & Soft Topology Optimization

To quantify the topology optimization, we choose an index to reflect the proportion of inter-class edges, which tends to be low when the topology is optimized for node classification tasks. Formally, the inter-ratio(IR) of a propagation matrix is defined:

$$IR = \frac{w_2}{w_1 + w_2}, \tag{13a}$$

$$w_1 = \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_1(i)} \frac{P_{ij} \cdot |\mathcal{N}_1(i)|}{\sum_{k \in \mathcal{N}_1(i)} P_{ik}} \mathcal{I}(c_i, c_j), \tag{13b}$$

$$w_2 = \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_1(i)} \frac{P_{ij} \cdot |\mathcal{N}_1(i)|}{\sum_{k \in \mathcal{N}_1(i)} P_{ik}} [1 - \mathcal{I}(c_i, c_j)], \tag{13c}$$

where w_1 and w_2 are the global intensities of intra- and inter-class edges, and $\mathcal{I}(c_i, c_j)$ is an indicator function, whose value is equal to 1 when $c_i = c_j$ and 0 otherwise. This metric consider both edge weights and presence and hence suits for both soft and hard topology. Another interesting observation is that a node with a larger neighborhood is more likely to be misclassified by deep vanilla GCNs [9]. Based on this, for each target node we scale the associated propagation weights by the number of predecessors in (13).

The first set of experiments are to support the **Hypothesis 1** mentioned earlier. All three models are composed by 4 layers divided to 2 blocks, each maintaining one common propagation matrix, *i.e.*, $\mathbf{P}^{(0)} = \mathbf{P}^{(1)} = \mathbf{P}_0$, $\mathbf{P}^{(2)} = \mathbf{P}^{(3)} = \mathbf{P}_1$. They are trained with a fixed learning rate 0.01. Variations of IR (computed for each propagation matrix) vs epochs are depicted in Figure 1. Curves of DyGCN, JKNet(Dy) and GCNII(Dy) are evaluated on both the subgraph induced by training nodes and the whole graph. It can be observed that IR monotonically decreases with the increase of epochs. Furthermore, the variation trend of $\mathbf{P}_1$ appears more dramatic, because it is closer to the output layer of each model.

3.3. Performance Comparison

Following the common setting, classification performance of various GCN models with different network depth are evaluated and compared in this subsection. As there is no public separation of these datasets, we randomly divide each dataset into 10 splits. For a fair comparison, all the models are evaluated on each split and the averaging accuracy of each model is reported in Table 1. When equipped with AdaDrop, a dynamic model shares the same hyperparameters except the drop rate p and the max times of hard topology optimization K. To reveal the influence of network depth, all parameters except L are shared within one group of experiments (corresponding to one row in Table 1) for a shallow model (*i.e.*, GCN, GCN-LPA, or DyGCN). In Table 1, DyGCN(Lin) denotes the variant of DyGCN using the identity activation function and "AD" indicates the usage of AdaDrop.

Table 1: Classification accuracies (in percentage) of various models with various depths in full-supervised setting

Dataset	Model	# of layers employed				
		2	4	8	16	32
Cora	GCN [1]	87.95	87.55	50.85	29.98	29.98
	JKNet(Add)[6]	-	86.03	85.87	86.53	86.05
	GCN-LPA [11]	87.20	86.62	60.46	31.88	31.22
	GCNII [9]	-	88.15	**88.82**	88.78	88.89
	DyGCN	89.06	87.93	74.91	28.52	30.06
	DyGCN(Lin)	88.80	87.88	86.66	81.66	33.47
	DyGCN(AD)	**89.28**	88.19	85.87	33.75	30.00
	JKNet(Dy)	-	88.41	87.93	88.06	85.92
	JKNet(AD,Dy)	-	88.76	88.49	88.14	88.08
	GCNII(Dy)	-	88.36	88.36	**88.89**	**89.26**
	GCNII(AD, Dy)	-	**89.00**	88.71	88.71	89.21
Citeseer	GCN	76.08	75.18	53.97	26.07	21.25
	JKNet(Max)	-	75.17	75.62	75.81	75.42
	GCN-LPA	**76.37**	72.79	27.00	25.35	23.77
	GCNII	-	76.13	76.67	76.49	76.79
	DyGCN	76.20	75.11	73.29	31.98	20.98
	DyGCN(Lin)	76.13	74.67	73.86	71.25	41.40
	DyGCN(AD)	76.26	75.14	73.42	50.47	27.07
	JKNet(Dy)	-	75.23	75.93	75.71	75.33
	JKNet(AD,Dy)	-	75.23	76.08	75.92	75.47
	GCNII(Dy)	-	76.07	76.89	**76.85**	**77.03**
	GCNII(AD, Dy)	-	**76.27**	**77.21**	76.79	76.94
Pubmed	GCN	87.69	85.58	73.67	49.60	39.69
	JKNet(Add)	-	86.21	86.10	85.99	85.79
	GCN-LPA	86.97	85.35	72.24	57.92	55.30
	GCNII	-	89.68	89.33	89.01	88.85
	DyGCN	87.83	85.10	83.66	65.60	39.50
	DyGCN(Lin)	87.39	84.99	83.96	82.03	70.27
	DyGCN(AD)	**87.87**	85.15	83.75	80.21	46.81
	JKNet(Dy)	-	85.74	85.79	85.61	85.41
	JKNet(AD,Dy)	-	85.92	85.75	85.66	85.63
	GCNII(Dy)	-	89.90	89.69	89.24	88.97
	GCNII(AD, Dy)	-	**89.94**	**89.80**	**89.41**	**89.27**

The baseline JKNets using three aggregation strategies, *i.e.*, Concat, Max, and Add, are all tested. But only the result corresponding to the best strategy is reported in Table 1. For instance, Max is the best strategy for Citeseer. Hence, JKNet is marked as JKNet(Max) and JKNet(Dy) also utilizes this strategy. Regarding to baseline GCNII and GCN-LPA, the hyperparameters recommended by their authors are taken in our experiments.

On each dataset, the best result for each L is marked in bold font. The highest score among all the results is further underlined. It can be noticed that DyGCN and its deep variants generally outperform the baseline models. Impressively, 2-layer DyGCN with AdaDrop reaches the highest accuracy on Cora, even outperforming those 32-layer deep models, *e.g.*, JKNet and GCNII. Though shallow DyGCN and DyGCN(AD) demonstrate competitive performance, the degradation is significant when they become significantly deep, just as suggested by Theorem 1. Even so, a deep DyGCN outperforms a same-layer vanilla GCN or GCN-LPA noticeably, revealing much more robustness against the oversmoothing issue. Furthermore AdaDrop can enhance it such that a 16 or 32-layer DyGCN(AD) exceeds same-layer DyGCN 5.23% - 18.4% in accuracy. These observations demonstrate that the proposed models and AdaDrop are effective to alleviate oversmoothness via optimizing graph topology. Actually, AdaDrop can enhance all dynamic models listed here in most cases. Note that since JKNet itself has robustness to oversmoothness, the experiments of JKNet(Dy) and JKNet(Dy,AD) are all conducted in an ablation manner. That is, for one dataset, these models share the same hyperparameters except those extra ones required by JKNet(Dy) and JKNet(Dy,AD). The experiments of GCNII comply with this setting too.

4. CONCLUSION

It is known that the node classification performance of GCN highly relies on graph topology. In this paper, we present a novel dynamic GCN model, which enables effective topology optimization including AdaDrop. Although it is theoretically proven that the output of a deep DyGCN could collapse into a low-rank space, the resistance of DyGCN to the oversmoothness is much stronger than vanilla GCN and GCN-LPA and can be further enhanced via combining the proposed architecture with some other deep models, *e.g.*, JKNet and GCNII. To investigate the effectiveness of topology optimization, we define the inter-ratio(IR). It is empirically shown that DyGCN and its deep variants can effectively decrease the IR scores with the increase of epochs, and achieve superior performance in three citation datasets. The potential future work includes the design of more effective strategies to resist the oversmoothing issue and transferring DyGCN and AdaDrop into other tasks such as graph adversarial learning [16].

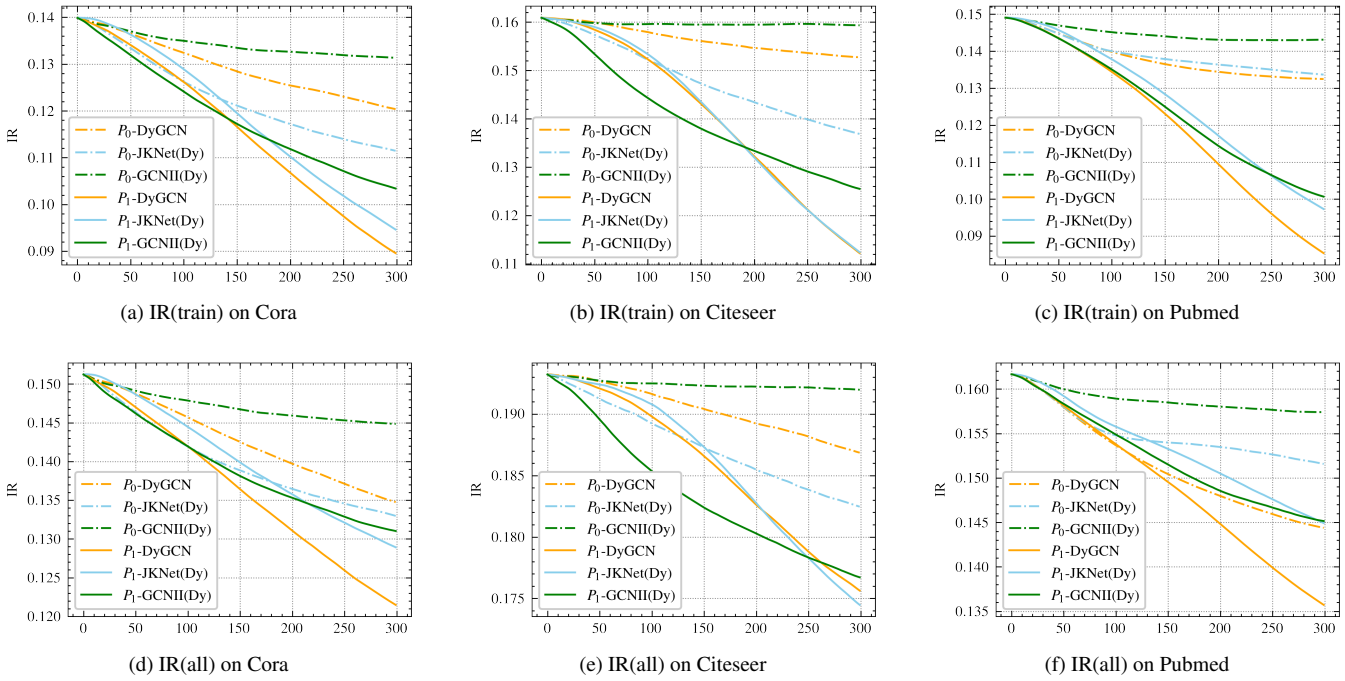
5. REFERENCES

[1] Thomas N. Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," *5th International Conference on Learning Representations*, 2017.

[2] Petar Veličković, Arantxa Casanova, Pietro Liò, Guillem Cucurull, Adriana Romero, and Yoshua Bengio, "Graph attention networks," *6th International Conference on Learning Representations*, 2018.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun, "Spectral networks and locally connected networks on graphs," in *International Conference on Learning Representations*, 2014.

[4] William L. Hamilton, Rex Ying, and Jure Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.



(a) IR(train) on Cora (b) IR(train) on Citeseer (c) IR(train) on Pubmed

(d) IR(all) on Cora (e) IR(all) on Citeseer (f) IR(all) on Pubmed

Fig. 1: IR-epoch curves of DyGCN, JKNet(Dy) and GCNII(Dy) obtained on three citation datasets. The first row presents experimental results evaluated on training nodes, while the second row consists of curves evaluated on the whole graph.

[5] Kenta Oono and Taiji Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *International Conference on Learning Representations*, 2020.

[6] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken Ichi Kawarabayashi, and Stefanie Jegelka, "Representation learning on graphs with jumping knowledge networks," *35th International Conference on Machine Learning*, 2018.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *7th International Conference on Learning Representations, ICLR*, 2019.

[9] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding Ding, and Yaliang Li, "Simple and deep graph convolutional networks," *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[10] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun, "Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[11] Hongwei Wang and Jure Leskovec, "Unifying graph convolutional neural networks and label propagation," 2020.

[12] Sitao Luan, Mingde Zhao, Xiao Wen Chang, and Doina Precup, "Break the ceiling: Stronger multi-scale deep graph convolutional networks," *Advances in Neural Information Processing Systems*, 2019.

[13] Matthias Fey and Jan Eric Lenssen, "Fast graph representation learning with pytorch geometric," 2019.

[14] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad, "Collective classification in network data," *AI Magazine*, 2008.

[15] Diederik P. Kingma and Jimmy Lei Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations*, 2015.

[16] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann, "Adversarial attacks on neural networks for graph data," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.