Dual Self-Paced Graph Convolutional Network: Towards Reducing Attribute Distortions Induced by Topology

Liang Yang[1,2], **Zhiyang Chen**[1,2], **Junhua Gu**[1,2,*] and **Yuanfang Guo**[3]
[1]School of Artificial Intelligence, Hebei University of Technology, China
[2]Hebei Province Key Laboratory of Big Data Calculation, Hebei University of Technology, China
[3]School of Computer Science and Engineering, Beihang University, China
yangliang@nextseason.cc, valarzychen@gmail.com, jhgu@hebut.edu.cn, andyguo@buaa.edu.cn

Abstract

The success of graph convolutional neural networks (GCNNs) based semi-supervised node classification is credited to the attribute smoothing (propagating) over the topology. However, the attributes may be interfered by the utilization of the topology information. This distortion will induce a certain amount of misclassifications of the nodes, which can be correctly predicted with only the attributes. By analyzing the impact of the edges in attribute propagations, the simple edges, which connect two nodes with similar attributes, should be given priority during the training process compared to the complex ones according to curriculum learning. To reduce the distortions induced by the topology while exploit more potentials of the attribute information, Dual Self-Paced Graph Convolutional Network (DSP-GCN) is proposed in this paper. Specifically, the unlabelled nodes with confidently predicted labels are gradually added into the training set in the node-level self-paced learning, while edges are gradually, from the simple edges to the complex ones, added into the graph during the training process in the edge-level self-paced learning. These two learning strategies are designed to mutually reinforce each other by coupling the selections of the edges and unlabelled nodes. Experimental results of transductive semi-supervised node classification on many real networks indicate that the proposed DSP-GCN has successfully reduced the attribute distortions induced by the topology while it gives superior performances with only one graph convolutional layer.

1 Introduction

Graph partitioning has contributed tremendously in both the traditional unsupervised node classification techniques and the latest semi-supervised node classification approaches [Kipf and Welling, 2017; Veličković *et al.*, 2018]. From either the spectral or spatial perspectives, many graph convolutional neural networks (GCNNs) have been proposed recently [Defferrard *et al.*, 2016; Hamilton *et al.*, 2017]. By incorporating supervised information, GCNN based semi-supervised node classification methods have significantly improved the performances of the traditional unsupervised techniques.

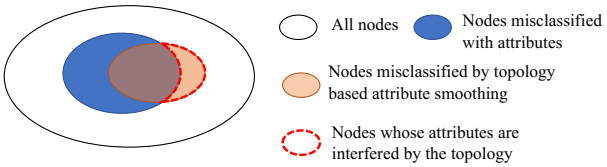

Figure 1: Different types of misclassified nodes. The nodes may be misclassified with respect to the original attributes only. It may also be misclassified by topology based attribute smoothing. The topology will correct certain incorrect classifications, while interfere some correct predictions. This distortion will induce a certain amount of misclassifications (the area surrounded by the red dashed lines) of the nodes.

Similar to other semi-supervised learning tasks [Chapelle *et al.*, 2009; Zhu, 2006], semi-supervised node classification requires efficient incorporations of the unsupervised information. When classifying the nodes in the attributed graphs, the attributes of the unlabelled nodes serve as the unsupervised information. Due to the sparsity of the node attributes, prediction only based on the original node attributes cannot fully exploit their relationships and usually results in overfittings. Therefore, many GCNNs [Kipf and Welling, 2017; Veličković *et al.*, 2018] perform operations which are equivalent to the attribute smoothing (propagating) over the graph [Li *et al.*, 2018]. Therefore, GCNNs actually utilizes the unsupervised information by augmenting the node attributes in local neighbourhoods.

Unforunately, this propagation strategy possesses certain weaknesses. Typically, over propagation may degrade the prediction performance [Li *et al.*, 2018]. Since the over propagation tends to amend all the nodes to have the identical augmented attributes which certainly reduces the discriminabilities of the nodes. Therefore, existing GCNNs only employ two stacked graph convolutional layers, i.e., two-hop propagations to prevent the over propagation. Besides, the attributes may be interfered by the utilization of the topology information. This distortion will induce a certain amount of misclassification of the nodes, which can be correctly predicted with only the attributes, as illustrated in Fig. 1.

*Corresponding author.

In this paper, we intend to reduce the negative effects, i.e., the distortions, caused by the utilization of the topology. Thus, the impact of the edges in attribute propagations is analyzed. As elaborated in Fig. 2, there exists two types of edges. If the two nodes v_i and v_j, whose attributes are similar, are connected by an edge (green line), the attributes of each node will vary slightly after propagations. Since this kind of edges usually contain less information and the network can quickly learn from them, we denote them as the simple edges. If the two nodes v_i and v_k, whose attributes are quite different, are connected by an edge (red line), the attributes of each node will be significantly changed after propagations. This kind of edges contain rich information and the network usually requires more trainings to learn from them. Therefore, these edges are denoted as the complex edges.

Spontaneously, a question can be raised that whether these two kinds of edges should be learned simultaneously by the network in the training stage? Curriculum Learning (CL [Bengio *et al.*, 2009]) and Self-Paced Learning (SPL [Kumar *et al.*, 2010]), which mimic the learning process of human beings, conclude that by learning from simple concepts to complex ones, the learning of the network can be significantly improved. Based on this philosophy, we believe that the topology information should be evaluated and feed into the network gradually from the simple edges to the complex ones during the training process of GCNNs.

To reduce the distortions generated by utilizing the topology information while exploit more potentials of the attribute information, Dual Self-Paced Graph Convolutional Network (DSP-GCN) is proposed in this paper. Specifically, the unlabelled nodes with confidently predicted labels are gradually included into the training set in the node-level self-paced learning to better exploit the attribute information. To reduce the distortions when utilizing the topology information, edges are gradually, from the simple edges to the complex ones, added into the graph during the training process in the edge-level self-paced learning. Since the nodes and edges are highly correlated, the node-level and edge-level self-paced learnings are also correlated. Therefore, these two learning strategies are designed to mutually reinforce each other by coupling the selections of the edges and unlabelled nodes. If an unlabelled node is included during the training process, its edges will likely to be included. On the contrary, if most of the edges of an unlabelled node is added, this node as well as its predicted label tend to be put into the training with a high probability.

Our contributions are summarized as below:

- We observe that the utilization of the topology in GCNNs tends to interfere the correct prediction results which can be obtained with respect to the attributes only, according to the experiments.
- We propose an edge-level self-paced learning strategy by gradually train the network from the simple edges to the complex ones to reduce the negative effects caused by the utilization of the topology, instead of feeding the entire topology directly into the network.
- We propose a Dual Self-Paced Graph Convolutional Network (DSP-GCN) which jointly exploits the node-level and edge-level self-paced learning strategies.
- Experimental results on eight real networks indicate that DSP-GCN can successfully reduce the distortion induced by the topology and yield superiority performances with only one graph convolutional layer.

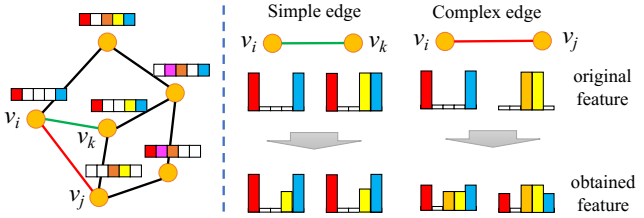

Figure 2: Simple and complex edges. A simple edge (green line) connects two nodes which contain similar attributes. After smoothing over the simple edge, their node attributes do not vary significantly. A complex edge (red line) connects two nodes which contain much different attributes. After smoothing over the complex edge, the discriminabilities of the node attributes are reduced.

2 Preliminaries

2.1 Notations

A network can be represented by an attributed graph $G = (V, E, X)$. $V = \{v_i | i = 1, ..., N\}$ is a set of $|V| = N$ vertices, where v_i is associated with a feature $x_i \in \mathbb{R}^K$. $X \in \mathbb{R}^{N \times K}$ represents the collection of the features. Each row of X corresponds to a node. E stands for a set of edges. Each of the edges connects two vertices in V. The adjacency matrix $A = [a_{ij}] \in \mathbb{R}^{N \times N}$ is obtained according to the network topology. If an edge connects the vertices v_i and v_j, $a_{ij} = 1$, and vice versa. If self-edges are allowed in the network, then $a_{nn} = 1$. Otherwise $a_{nn} = 0$. a_n, which denotes the n^{th} column of A, can be utilized to represent the local neighbourhood of the vertex v_n. $d_n = \sum_j a_{nj}$ stands for the degree of v_n while $D = \mathrm{diag}(d_1, d_2, ..., d_N)$ is the degree matrix of A. The graph Laplacian and its normalized form are defined as $L = D - A$ and $L = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$, respectively. For a network, given the labels $Y \in \mathbb{R}^{|V_l| \times F}$ (where F represents the number of classes), which belong to a set of vertices $V_l \subset V$, a typical semi-supervised node classification algorithm classifies other nodes in $V - V_l$ according to the attributed graph. For simplicity, the first l nodes $\{v_i\}_{i=1}^{l}$ are assumed to be labelled.

2.2 Graph Convolutional Network

By simplifying the complex existing models, Graph Convolutional Network (GCN) [Kipf and Welling, 2017] defines the graph convolution operation as

$$H_{GCN} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X, \tag{1}$$

where $\tilde{A} = A + I_N$ and $\tilde{D}_{nn} = \sum_j \tilde{A}_{nj} = d_n + 1$. Then, the to-be-predicted labels can obtained by feeding H_{GCN} into a fully-connected layer as

$$T_{GCN} = H_{GCN} W. \tag{2}$$

By minimizing the cross-entropy, which is defined in Eq. (3), between the predictions and given labels, the parameter W can be obtained.

$$\mathcal{L} = -\sum_{n \in V_l} \sum_{f=1}^{F} y_{nf} \log(t_{nf}) \tag{3}$$

[Li et al., 2018] concludes the mechanism and success of GCN that it is equivalent to perform a symmetric Laplacian smoothing ($H_{GCN} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$) operation before the actual predictions. It aims at augmenting the attributes with topology to alleviate the problem of attribute sparsity. The performance of the smoothing based classification obviously outperforms the classification based on the original attributes. Unfortunately, the employment of the topology information also induces certain negative effects to the classification results. As illustrated in Fig. 1, some nodes (i.e. the non-overlapped orange set), which can be correctly classified according to the original attributes, are actually misclassified with respect to the smoothed attributes after the attribute propagations.

2.3 Self-paced Learning

Inspired by the learning process of human beings, i.e., learning from easy concepts to complex ones, [Bengio et al., 2009] proposes a curriculum learning (CL) strategy which gradually includes the training samples from the easy samples to the complex ones during the training process. Self-paced Learning (SPL [Kumar *et al.*, 2010]) adopts this strategy by incorporating the curriculum design as a regularization term into the learning objective function as

$$\sum_{i=1}^{N} q_i \ell(y_i, f(x_i, w)) - \lambda \sum_{i=1}^{N} q_i, \tag{4}$$

where $f(x_i, w)$ represents the prediction function with learnable model parameter w, $\ell(y_i, f(x_i, w))$ stands for the loss function between the prediction $f(x_i, w)$ and ground truth y_i, and $q_i \in \{0, 1\}$ is the sample weight to indicate whether the data point x_i has been included in the training process. Note that the second term $\lambda \sum_{i=1}^{N} q_i$ is the self-paced regularization term, with λ being the pace parameter which controls the learning pace. The model is trained by jointly minimizing the objective function in Eq. (4) with respect to the model parameter w and the sample weights q_i's by gradually increasing the pace parameter λ. For a fixed w, the optimal q_i^* is

$$q_i^* = \begin{cases} 1 & \text{if } \ell_i < \lambda \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

where $\ell_i = \ell(y_i, f(x_i, w))$ represents the loss of predicting the data sample x_i as $f(x_i, w)$. With a small λ, only the data samples with small losses (i.e. easy to be predicted) are included in the training data. When λ gradually increases, more complex data samples are added.

Self-paced learning has already been extended to semi-supervised learning and unsupervised matrix factorization. Self-paced semi-supervised learning augments the training set with the unlabelled data samples based on the confidence predictions [Ma *et al.*, 2017]. The unlabelled data samples are gradually included by combining the self-paced strategy and co-training of two views. Meanwhile, self-paced matrix factorization extends the previous self-paced learning strategy which is designed for the data samples to a self-paced learning for the pairs of data samples and their corresponding attributes [Zhao *et al.*, 2015].

Self-paced Network Embedding

Recently, self-paced learning is applied to network embedding. [Zhou *et al.*, 2018] follows the co-training strategy in [Ma *et al.*, 2017] and extends the co-training of two views to two tasks, i.e., embedding and prediction. However, this node-level self-paced strategy cannot reduce the negative impact to the attributes induced by the topology. [Gao and Huang, 2018] adopts the self-paced strategy to select the negative samples and performs a self-paced training with the pair of data samples similar to [Zhao *et al.*, 2015]. Unfortunately, self-paced negative sampling requires to compute the similarities between almost all the pairs of nodes, which is computationally inefficient. Besides, similar to other semi-supervised node classification methods introduced above, these two methods also ignore the impact of topology in the self-paced node selections.

3 Proposed Work

In this section, Dual Self-Paced Graph Convolutional Network (DSP-GCN), which reduces the topology distortions on the attributes, is proposed. DSP-GCN consists of two interactive self-paced learning strategies, node-level and edge-level strategies. The node-level self-paced strategy gradually includes the unlabelled nodes, whose labels are predicted with high confidence scores, into the training set. The edge-level self-paced strategy gradually incorporates the edges, from the simple edges to the complex ones, into the graph convolutional operation of GCN. Note that both the strategies will mutually reinforce each other, because the nodes and edges in the graph is usually highly correlated. If an unlabelled node is added into the training set, its edges tend to be included. Meanwhile, if most of the edges of an unlabelled node has already been included , this node as well as its predicted label will likely to be added into training set.

3.1 Node-level Self-Paced Learning

Our node-level self-paced learning augments the training set to fully exploit the unsupervised information (i.e., the attributes of the unlabelled nodes). Since we gradually add the unlabelled nodes, which possess the predicted labels with high confidence scores, into the training set, the objective function can be formulated as

$$\mathcal{L}_{node} = \sum_{i=1}^{l} \ell(y_i, g(x_i, A, W)) + \sum_{k=l+1}^{N} q_k \ell(y_k, g(x_k, A, W)) - \lambda_{node} \sum_{k=l+1}^{N} q_k, \tag{6}$$

where the three terms at the right-hand side of the equation are a supervised loss, an unsupervised loss and a self-paced

regularizer, respectively. y_i and y_k denote the given labels of the labelled nodes and the predicted labels of the unlabelled nodes, respectively. $q_k \in \{0, 1\}$ is the weight assigned to the unlabelled node v_k to indicate whether it as well as its predicted label y_k has been added into the training set. λ_{node} is the pace parameter which controls the learning pace. If λ_{node} increases, more unsupervised nodes will be added into the training set, i.e., more unsupervised nodes satisfy the condition of $q_k^* = 1$, and vice versa. The prediction model $g(x_k, A, W)$ is identical to GCN as shown in Eqs. (1) and (2), and the loss function $\ell(.,.)$ is the cross-entropy as shown in Eq. (3). With a fixed W, minimizing Eq. (6) with respect to q_k gives the same optimal solution as Eq. (5) presents.

3.2 Edge-level Self-Paced Learning

Unfortunately, optimizing Eq. (6) with a fixed topology A is equivalent to directly add all the edges (i.e., the entire topology) into the training of GCN, which may induce certain negative effects to the attributes as elaborated in Fig. 1. To alleviate this issue, the negative impact of the topology to the discriminabilities of the node attributes is considered and formulated. Since there exists different kinds of edges, which possess different degrees of training difficulties and have been illustrated in Sec. 1 and Fig. 2, we propose to exploit the self-paced learning strategy [Kumar *et al.*, 2010] to gradually include edges with different degrees of training difficulties (from the simple edges to the complex ones) into the training process. To measure the difficulty of learning from the edge e_{ij}, we exploit the attribute variations on the nodes v_i and v_j after the attribute propagations as the measurement, which also indicates the amount of information possessed by e_{ij}. This quantity can be described as the inner product of the attributes of nodes v_i and v_j as

$$s_{ij} = 1 - \frac{\exp(x_i B x_j^T)}{\sum_{k \in N(i)} \exp(x_i B x_k^T)}, \tag{7}$$

where B is the learnable parameter to measure the consistency between the corresponding attributes of the two nodes. For example, "learning" and "inference" are two similar attributes in machine learning community. If the attribute 'learning" is propagated to a node v_i which possesses the attribute "inference" yet without the attribute 'learning", the attribute amendment occurred on node v_i tends to be minor and only slightly increases the entropy of node v_i.

Instead of adding a self-paced regularizer as the node-level strategy, our edge-level self-paced learning strategy directly selects the edges which are included in the training process via

$$\hat{a}_{ij} = \begin{cases} 1 & \text{if } s_{ij} < \lambda_{edge} \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where $\hat{a}_{ij}$ is edge weight and λ_{edge} is the pace parameter to control the learning pace of the edges. When λ_{edge} is small, only the edges e_{ij} with high s_{ij}, which will not significantly increase the attribute entropy of the connected nodes, is added. When λ_{edge} increases, more edges will be included.

3.3 Dual Self-Paced Learning

The node-level self-paced learning in Eq. (6) and edge-level self-paced learning in Eq. (8) can be respectively tuned with two pace parameters λ_{node} and λ_{edge}. Meanwhile, both strategies will mutually reinforce each other.

Intuitively, if the prediction confidence of an unsupervised node is high, this node will be added into the training set according to the self-paced learning strategy. The high prediction confidence of a node is usually induced by the consistency between it and its neighbourhoods. Besides, this consistency indicates that the edges between them are the simple ones, as shown in Fig. 2. Then, these edges are also added into the graph. On the other hand, the addition of certain edges into the training set may also induce the addition of certain nodes in a similar manner to the above situation.

To model the interactions between the two strategies, the weight q_k of unlabelled node and weight $\hat{a}_{ij}$ of edge are jointly considered to form the loss of the interactions as

$$\mathcal{L}_{inter} = -\gamma \sum_{k=l+1}^{N} \left(q_k \sum_{j \in N(k)} \frac{\hat{a}_{kj}}{\hat{d}_k + 1} \right), \tag{9}$$

where $\hat{d}_k = \sum_j \hat{a}_{kj}$ represents the current degree of the node v_k in graph $\hat{A}$. $\hat{d}_k + 1$, which is equivalent to adding a self-loop to each node as GCN, is employed to avoid the zero divisor. Since the edge-level self-paced learning in Eq. (8) is performed as a constraint instead of a loss term as the node-level self-paced learning. A simple combination of the two losses in Eqs. (6) and (9) cannot achieve the reinforcement from the node-level learning to the edge-level learning. To alleviate this issue, Eq. (8) is modified as

$$\hat{a}_{ij} = \begin{cases} 1 & \text{if } s_{ij} < \lambda_{edge} + \gamma \frac{q_i}{\hat{d}_i + 1}. \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

If the unlabelled node v_i has been added to the training set, i.e., $q_i = 1$, the probability of edge e_{ij} being included will increase from λ_{edge} to $\lambda_{edge} + \gamma \frac{q_i}{\hat{d}_i + 1}$.

3.4 Objective Function and Optimization

By combining Eqs. (6), (9) and (10), the overall objection function is constructed as

$$\begin{aligned} \mathcal{L} = & \sum_{i=1}^{l} \ell(y_i, g(x_i, \hat{A}, W)) + \sum_{k=l+1}^{N} q_k \ell(y_k, g(x_k, \hat{A}, W)) \\ & - \lambda_{node} \sum_{k=l+1}^{N} q_k - \gamma \sum_{k=l+1}^{N} \left(q_k \sum_{j \in N(k)} \frac{\hat{a}_{kj}}{\hat{d}_k + 1} \right) \\ & \text{s.t. } \hat{a}_{ij} = \begin{cases} 1 & \text{if } s_{ij} < \lambda_{edge} + \gamma \frac{q_i}{\hat{d}_i + 1} \\ 0 & \text{otherwise} \end{cases}, \end{aligned} \tag{11}$$

with the hyper-parameters γ, λ_{node} and λ_{edge}. The parameters to be learned in Eq. (11) are W, B, y_k, q_k and $\hat{a}_{ij}$. This objective function can be minimized according to an alternative optimization strategy. The optimization process are

Algorithm 1 Dual Self-Paced GCN

Input: Attributed graph $G = (V, E, X)$, labels $\{y_i\}_{i=1}^{l}$
Parameters: γ, pace parameters λ_{node} and λ_{edge}
Output: Labels $\{y_i\}_{i=l+1}^{N}$ of the target nodes $\{v_i\}_{i=l+1}^{N}$

1: **while** not converged **do**
2: Update q_k via Eq. (12).
3: Update y_k via Eq. (13).
4: Update $\hat{a}_{ij}$ via Eq. (10).
5: Update W and B on augmented training set with gradient back-propagation.
6: Augment λ_{node} and λ_{edge}.
7: **end while**
8: **return** $\{y_i\}_{i=l+1}^{N}$

shown in Algorithm 1 with the main steps elaborated as follows:

Update q_k: This step is equivalent to minimizing $\mathcal{L}$ in Eq. (11) without the supervised term and the constraint. Since $q_k \in \{0, 1\}$, its optimal solution is

$$q_k^* = \begin{cases} 1 & \text{if } \ell_k < \lambda_{node} + \gamma \sum_j \left(\frac{\hat{a}_{kj}}{\hat{d}_k + 1} \right) \\ 0 & \text{otherwise} \end{cases}. \quad (12)$$

Here, $\ell_k = \ell(y_k, g(x_k, \hat{A}, W))$ represents the difference between the predictions of two consecutive iterations, where y_k stands for the predicted label in the previous iteration, and $g(x_k, \hat{A}, W)$ represents the predicted label in the current iteration according to the new $\hat{a}_{ij}$, W and B. The difference threshold is $\lambda_{node} + \gamma \sum_j \left(\frac{\hat{a}_{kj}}{\hat{d}_k + 1} \right)$, where the first term λ_{node} is the pace parameter. The second term $\gamma \sum_j \left(\frac{\hat{a}_{kj}}{\hat{d}_k + 1} \right)$ indicates the probability of this node v_k being included in the training set, if more edges $\hat{a}_{kj}$ of the unlabelled node v_k is included. By updating q_k, we can improve the self-paced node selections with our self-paced edge selections.

Update y_k: Since only the second term of $\mathcal{L}$ in Eq. (11) contains the variable y_k, the optimal y_k^* can be obtained by minimizing $\ell(y_k, g(x_k, \hat{A}, W))$. Therefore,

$$y_k^* = g(x_k, \hat{A}, W), \quad (13)$$

where $g(x_k, \hat{A}, W)$ is identical to GCN as introduced in Eqs. (1) and (2), i.e., label is predicted by GCN with parameters W, B and the topology $\hat{a}_{ij}$ obtained in the previous iteration.

Update $\hat{a}_{ij}$: The optimal solution of a_{ij} has been shown in Eq. (10). By updating a_{ij}, the proposed self-paced edge selections can be further enhanced with respect to the self-paced node selections.

Update W and B: In this step, we only have to optimize the first two terms of $\mathcal{L}$ in Eq. (11), i.e.,

$$\sum_{i=1}^{l} \ell(y_i, g(x_i, \hat{A}, W)) + \sum_{k=l+1}^{N} q_k \ell(y_k, g(x_k, \hat{A}, W)),$$

whose parameters are W and B. Here, only the unlabelled nodes, which are included in the training set, i.e., $q_k = 1$, will contribute to the second term. Since the cross-entropy in Eq. (3) is adopted as a loss, the gradient back-propagation can be utilized to obtain W and B similar to GCN [Kipf and Welling, 2017] and GAT[Veličković *et al.*, 2018].

Remark 1: Comparing with the existing self-paced network embedding approaches, DSP-GCN possesses two obvious advantages. 1) DSP-GCN considers self-paced learning on edges to reduce the negative effects caused by employing the topology. 2) DSP-GCN considers and models the correlations between the node-level and edge-level self-paced learning to improve the prediction efficiencies.

Remark 2: The graph construction is the updating of $\hat{a}_{ij}$ in Eq. (10) with s_{ij} defined in Eq. (7). The computational complexity of each edge s_{ij} is $O(K)$, where K is the number of features. Therefore, the overall computational complexity of the graph construction is $O(MK)$ where M is the number of edges in the graph, i.e., the computations of the graph construction will linearly increase when the scale of the network increases.

Dataset	#Nodes	#Edges	#Classes	#Features
Texas	187	328	5	1,703
Cornell	195	304	5	1,703
Washington	230	446	5	1,703
Wisconsin	265	530	5	1,703
Wiki	3,363	45,006	19	4,972
CiteSeer	3,327	4,732	6	3,703
Cora	2,708	5,429	7	1,433
PubMed	19,717	44,338	3	500
NELL	65,755	266,144	210	5,414

Table 1: Datasets.

4 Experimental Results

In this section, we validate the proposed DSP-GCN by empirically evaluating the performances in the transductive semi-supervised node classification task. In DSP-GCN, only one graph convolutional layer is employed instead of the two layers utilized in GCN and GAT. Here, we set $\gamma = 1$ and initialize $\lambda_{node} = 0.12$ and $\lambda_{edge} = 0.2$ on all the networks. Note that λ_{node} and λ_{edge} are augmented via two multipliers ranging from 1.05 to 1.11, respectively.

4.1 Datasets

In the experiments, three common citation networks [Sen *et al.*, 2008], Cora, CiteSeer and PubMed, and a knowledge graph NELL [Carlson *et al.*, 2010] are employed as shown in Table 1. In each citation network, papers and undirected citations are defined as the nodes and edges, respectively. The node content is represented by the bag-of-word representation of the documents. Papers are classified into various categories according to their disciplines. A bipartite graph is extracted from the knowledge graph as in [Yang *et al.*, 2016]. Besides of the entity nodes, each relation tuple (e_i, r, e_j) is decomposed into two connected relation nodes (e_i, r) and (e_j, r). The topology is constructed by adding edges between entity node e_i and relation node (e_i, r). Besides, five more networks, including Cornell, Texas, Washington, Wisconsin

Methods	Cora	Citeseer	Pubmed	NELL
MLP	55.1%	46.5%	71.4%	22.9%
ManiReg	59.5%	60.1%	70.7%	21.8%
SemiEmb	59.0%	59.6%	71.7%	26.7%
LP	68.0%	45.3%	63.0%	26.5%
DeepWalk	67.2%	43.2%	65.3%	58.1%
ICA	75.1%	69.1%	73.9%	23.2%
Planetoid	75.7%	64.7%	77.2%	61.9%
Chebyshev	81.2%	69.8%	74.4%	-
GCN	81.5%	70.3%	79.0%	66.0%
MoNet	81.7%	69.9%	78.8%	64.2%
GAT	83.0%	72.5%	79.0%	-
DSP-GCN	**85.0%**	**74.2%**	**81.2%**	**67.3%**

Table 2: Transductive node classification results.

and Wiki, are employed. Four of them, i.e., Texas, Cornell, Washington and Wisconsin, are the sub-networks of the WebKB network. Each of them is the collection of the webpages from an university in U.S.. Similarly, nodes in Wiki network are the webpages from Wikipedia.

4.2 Methods

For comparisons, we employ 11 state-of-the-art semi-supervised node classification algorithms, including multilayer perceptron (MLP), label propagation (LP) [Zhu *et al.*, 2003], semi-supervised embedding (SemiEmb) [Weston *et al.*, 2012], manifold regularization (ManiReg) [Belkin *et al.*, 2006], graph embedding (DeepWalk) [Perozzi *et al.*, 2014], iterative classification algorithm (ICA) [Lu and Getoor, 2003], graph-based semi-supervised learning framework (Planetoid) [Yang *et al.*, 2016], graph convolution with Chebyshev filters [Defferrard *et al.*, 2016], graph convolutional network (GCN) [Kipf and Welling, 2017], mixture model networks (MoNet) [Monti *et al.*, 2017], and graph attention networks (GAT) [Veličković *et al.*, 2018]. Besides, to give a comprehensive understanding, we also compare DSP-GCN with 5 community detection methods on the attributed networks. Degree-corrected Stochastic Block Model (DCSBM) [Karrer and Newman, 2011] and NetMRF [He *et al.*, 2018] only adopt the network topology, while PCLDC [Yang *et al.*, 2009], SCI [Wang *et al.*, 2016] and NEMBP [He *et al.*, 2017] utilize both the network topology and node attributes. All the results of the baselines are produced by running the codes from the authors with their default settings.

4.3 Results and Analysis

In the experiments, DSP-GCN is compared to 11 baseline methods on three citation networks by following the experiment protocols in [Yang *et al.*, 2016], where 20 nodes per class, 500 nodes and 1000 nodes are employed for training, validation and performance evaluation, respectively. On the extracted bipartite graph from NELL, we follow the settings in [Yang *et al.*, 2016] where the label rate is 0.1%. The performance is measured with accuracy (AC). As shown in Table 2, our DSP-GCN outperforms all the baseline methods. The improvement of DSP-GCN compared to GAT, which achieves the best performance among the baseline methods, is moderately significant. Although the performance gain of accuracy is 1.8% in average, a large proportion of the error rates has been reduced. Specifically, the average error rates of GAT and our proposed DSP-GCN on citation networks are 21.84% and 19.87%, respectively, Then the error rate reduction, which is achieved by the proposed DSP-GCN, is (21.84% - 19.87%)/21.84% = 9.02%. It clearly demonstrates the effectiveness of our DSP-GCN on jointly exploiting the node content and network topology, which reduces the distortions to the attributes induced by the topology.

To comprehensively evaluate the proposed DSP-GCN, we compare it with five methods, which can also be regarded as node classification techniques. In addition to the three citation networks, five webpage networks, which is often adopted to evaluate the community detection methods, are employed. For the four medium webpage networks, including Cornell, Texas, Washington and Wsicsonsin, we adopt 20% labelled nodes for training, 10% labelled nodes for validation, and the other nodes for testing. For the large Wiki network, the percentages of nodes for training and validation are 3% and 3%, respectively. The results are shown in Table 3. The performance is measured with accuracy (AC) and normalized mutual information (NMI). As can be observed, our DSP-GCN gives relatively consistent performances and significant improvements on the five webpage networks.

The extensive results have demonstrated the superiority and effectiveness of the proposed DSP-GCN under various circumstances compared to 16 baseline methods. On the other hand, the original GCN gives unsatisfactory results (comparable/worse performances compared to the community detection methods) on the webpage networks. The unstable performances of GCN have revealed that the simple attribute smoothing based on the topology may cause severe degradations to the attributes, thus induce an unsatisfactory prediction result. Besides, the NMI scores of GCN, which are more sensitive to the prediction results of the classes containing very few nodes than AC, are obviously worse than AC on the webpage networks. This phenomenon indicates that the prediction results of GCN for the categories possessing very few nodes are unsatisfactory [Zhou *et al.*, 2018], which indicates a severe degradations of the attributes on these rare nodes after the attribute smoothing. Meanwhile, DSP-GCN can obtain satisfactory results on rare node classifications.

4.4 Case Study

To further demonstrate the negative effects caused by the topology in classification, a case study on the Citeseer network is carried out. In this experiment, 25% of the misclassified nodes (74 nodes) in GCN can be correctly classified with respect to the attributes only, where the classification is performed by a single fully-connected layer. This number is reduced to 10% (27 nodes) in our proposed DSP-GCN. The significant gain of our DSP-GCN verifies that our DSP-GCN can successfully reduce the attribute distortions caused by the topology.

Datasets	**DCSBM**		**NetMRF**		**PCLDC**		**SCI**		**NEMBP**		**GCN**		**DSP-GCN**	
Criterion	AC	NMI	AC	NMI	AC	NMI	AC	NMI	AC	NMI	AC	NMI	AC	NMI
Texas	48.1	16.6	30.6	5.5	38.8	10.4	49.7	12.5	53.6	35.1	57.1	5.0	**72.1**	**52.0**
Cornell	37.9	9.7	31.8	7.3	30.3	7.2	36.9	6.8	47.2	18.7	46.3	9.1	**64.1**	**39.0**
Washington	31.8	9.9	35.0	5.8	30.0	5.7	46.1	6.8	42.9	21.1	54.9	10.5	**69.9**	**40.2**
Wisconsin	32.8	3.1	28.6	3.2	30.2	5.0	46.4	13.3	63.4	38.0	55.6	18.8	**73.1**	**54.4**
Wiki	2.6	31.2	31.1	25.8	28.8	26.9	29.5	23.4	46.3	47.2	16.4	3.7	**51.2**	**49.2**
CiteSeer	26.6	4.1	22.2	1.2	24.9	3.0	34.4	9.2	49.5	24.3	70.3	45.4	**74.2**	**48.7**
Cora	38.5	17.1	58.1	37.2	34.1	17.5	41.7	17.8	57.6	44.1	81.5	62.5	**85.0**	**67.1**
PubMed	53.6	12.3	55.5	16.9	63.6	26.8	-	-	65.7	28.3	79.0	28.0	**81.2**	**46.2**

Table 3: Comparisons with different kinds of community detection methods.

5 Conclusions

Topology information is a double-edged sword for GCNN based semi-supervised node classification. It can be exploited to alleviate the problem of attribute sparsity by attribute smoothing over the graph. On the other hand, its employment in the attribute smoothing also degrades the discriminabilities of the attributes. To reduce its distortions while exploit more potentials of the attribute information, we propose Dual Self-Paced Graph Convolutional Network (DSP-GCN) in this paper. It consists of two highly correlated strategies, node-level and edge-level self-paced learning strategies, which gradually add the unlabelled nodes with confidently predicted labels and the simple edges into training, respectively. Extensive experiments on real networks have demonstrated the superiority of the proposed DSP-GCN compared to the baseline methods and its ability to reduces the attribute distortions induced by topology with only one graph convolutional layer.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (No.2017YFC0820106), in part by the National Natural Science Foundation of China under Grant 61503281 and Grant 61802391, in part by the Foundation for Innovative Research Groups through the National Natural Science Foundation of China under Grant 61421003, and in part by the Fundamental Research Funds for the Central Universities.

References

[Belkin *et al.*, 2006] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 7:2399–2434, 2006.

[Bengio *et al.*, 2009] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pages 41–48, 2009.

[Carlson *et al.*, 2010] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, pages 1306–1313, 2010.

[Chapelle *et al.*, 2009] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE TNN*, 20(3):542–542, 2009.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3837–3845, 2016.

[Gao and Huang, 2018] Hongchang Gao and Heng Huang. Self-paced network embedding. In *ACM SIGKDD*, pages 1406–1415, 2018.

[Hamilton *et al.*, 2017] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1025–1035, 2017.

[He *et al.*, 2017] Dongxiao He, Zhiyong Feng, Di Jin, Xiaobao Wang, and Weixiong Zhang. Joint identification of network communities and semantics via integrative modeling of network topologies and node contents. In *AAAI*, pages 116–124, 2017.

[He *et al.*, 2018] Dongxiao He, Xinxin You, Zhiyong Feng, Di Jin, Xue Yang, and Weixiong Zhang. A network-specific markov random field approach to community detection. In *AAAI*, pages 306–313, 2018.

[Karrer and Newman, 2011] Brian Karrer and M. E. J. Newman. Stochastic blockmodels and community structure in networks. *PRE*, 83:016107, 2011.

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

[Kumar *et al.*, 2010] M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NIPS*, pages 1189–1197, 2010.

[Li *et al.*, 2018] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, pages 3538–3545, 2018.

[Lu and Getoor, 2003] Qing Lu and Lise Getoor. Link-based classification. In *ICML*, pages 496–503, 2003.

[Ma *et al.*, 2017] Fan Ma, Deyu Meng, Qi Xie, Zina Li, and Xuanyi Dong. Self-paced co-training. In *ICML*, pages 2275–2284, 2017.

[Monti *et al.*, 2017] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *IEEE CVPR*, pages 5425–5434, 2017.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *ACM SIGKDD*, pages 701–710, 2014.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

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

[Wang *et al.*, 2016] Xiao Wang, Di Jin, Xiaochun Cao, Liang Yang, and Weixiong Zhang. Semantic community identification in large attribute networks. In *AAAI*, pages 265–271, 2016.

[Weston *et al.*, 2012] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade - Second Edition*, pages 639–655. 2012.

[Yang *et al.*, 2009] Tianbao Yang, Rong Jin, Yun Chi, and Shenghuo Zhu. Combining link and content for community detection: a discriminative approach. In *ACM SIGKDD*, pages 927–936, 2009.

[Yang *et al.*, 2016] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, pages 40–48, 2016.

[Zhao *et al.*, 2015] Qian Zhao, Deyu Meng, Lu Jiang, Qi Xie, Zongben Xu, and Alexander G. Hauptmann. Self-paced learning for matrix factorization. In *AAAI*, pages 3196–3202, 2015.

[Zhou *et al.*, 2018] Dawei Zhou, Jingrui He, Hongxia Yang, and Wei Fan. SPARC: self-paced network representation for few-shot rare category characterization. In *ACM SIGKDD*, pages 2807–2816, 2018.

[Zhu *et al.*, 2003] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.

[Zhu, 2006] Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):4, 2006.