Going Deep: Graph Convolutional Ladder-Shape Networks

Ruiqi Hu,[1] Shirui Pan,[2] Guodong Long,[1] Qinghua Lu,[3] Liming Zhu,[3] Jing Jiang[1]
[1]Centre for Artificial Intelligence, University of Technology Sydney, Australia
[2]Faculty of IT, Monash University, Australia
[3]Data61, CSIRO
{ruiqi.hu, guodong.long, jing.jiang}@uts.edu.au, shirui.pan@monash.edu
{qinghua.lu, liming.zhu}@data61.csiro.au

Abstract

Neighborhood aggregation algorithms like spectral graph convolutional networks (GCNs) formulate graph convolutions as a symmetric Laplacian smoothing operation to aggregate the feature information of one node with that of its neighbors. While they have achieved great success in semi-supervised node classification on graphs, current approaches suffer from the over-smoothing problem when the depth of the neural networks increases, which always leads to a noticeable degradation of performance. To solve this problem, we present graph convolutional ladder-shape networks (GCLN), a novel graph neural network architecture that transmits messages from shallow layers to deeper layers to overcome the over-smoothing problem and dramatically extend the scale of the neural networks with improved performance. We have validated the effectiveness of proposed GCLN at a node-wise level with a semi-supervised task (node classification) and an unsupervised task (node clustering), and at a graph-wise level with graph classification by applying a differentiable pooling operation. The proposed GCLN outperforms original GCNs, deep GCNs and other state-of-the-art GCN-based models for all three tasks, which were designed from various perspectives on six real-world benchmark data sets.

Introduction

Graphs are of the essence for constructing non-Euclidean data and they are omnipresent in most areas. In the social media industry, for instance, users, through their personal profile information, are linked with and interact with other users (e.g., friends, colleagues), and the entire social media network is modeled as an attributed graph for product/friend/community recommendations (node clustering); In medicine and pharmacology, molecules and chemical bonds can be constructed as graphs to potentially discover new drugs by identifying their bio-activities (node classification (Defferrard, Bresson, and Vandergheynst 2016; Gilmer et al. 2017)); In academic citation networks, papers are connected by their citations, and the titles, authors, venues and keywords form graph characteristics for automatic categorization (semi-supervised node classification (Velickovic et al. 2017; Kipf and Welling 2016b; Zhang et al. 2016) and image classification(Liu et al. 2019b; 2019a)).

There has recently been a rise in interest in leveraging embedding methods or deep learning algorithms for graphs. Many probabilistic models (Grover and Leskovec 2016; Tang et al. 2015) or matrix factorization-based works (Cao, Lu, and Xu 2015; Wang et al. 2017) aim to capture the patterns (walks) with neighborhood connections and the first or second order proximities from a graph, and to encode the graph into a low-dimensional, compact vector space. The well-learned embedding can be directly analyzed by conventional machine learning approaches. These methods exploit and preserve the topological characteristics and lose sight of the contextual features of the nodes in the graph.

Other methods simultaneously consider structural characteristics and node features to learn a robust embedding of the graph. Examples in this category include content enhanced network embedding methods and neighborhood aggregation (or message passing) algorithms. Content enhanced methods associate text features with the representation architectures, examples like TADW (Yang et al. 2015) which incorporates the feature information into network representation under a matrix factorization framework and TriDNR (Pan et al. 2016) which trains a neural network architecture to capture both structural proximity and attribute proximity from the attributed graph. Neighborhood aggregation algorithms (Dai et al. 2018; Li, Han, and Wu 2018; Klicpera, Bojchevski, and Günnemann 2018), represented as spectral graph convolutional networks (spectral GCNs) (Kipf and Welling 2016b), introduce the Laplacian smoothing operation to propagate the attributes of a node over its neighborhood. Spectral-based algorithms have commanded more attention among the aforementioned approaches due to their significant improvements over semi-supervised tasks.

Spectral-based algorithms borrow the idea of filters from the graph signal processing domain to conduct the graph convolution operation, which can also be interpreted as eliminating noises from graph signal. The major difference between one type of spectral GCN and another lies in the design of the filter. Spectral CNNs (Bruna et al. 2013) formulate the filter with a collection of learnable parameters to implement the convolution operation based on the spectrum

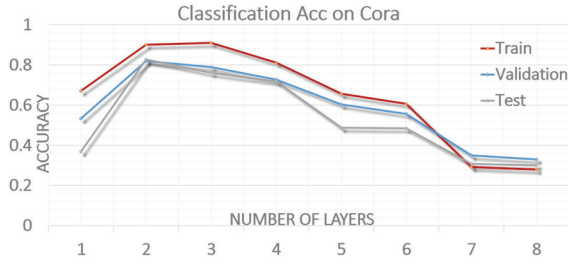

Figure 1: Influence of the depth of graph convolutional networks on semi-supervised node classification performance. When GCN goes deep (layer>3), its classification accuracy is decreased as the number of layers increases.

of the graph Laplacian. ChebNet (Defferrard, Bresson, and Vandergheynst 2016) considers the filter as Chebyshev polynomials of the diagonal matrix of eigenvalues, where the Chebyshev polynomial is defined with a recursive formulation. Spectral GCNs (Kipf and Welling 2016b) can be interpreted as the first order approximation of ChebNet, which assumes that the order of the polynomials of the Laplacian is restricted to 1 while the maximum of the eigenvalues is restricted to 2. Most recently, g-U-Nets (Gao and Ji 2019) attempts to generate the position information of selected nodes by using proposed gPooling and gUnpool operations, so the classic computer vision methods can be applied in graphs and increases the depth of the architecture. However, this method heavily relays on the graph preprocessing and can be difficult to train for good performance.

The major problem with all the aforementioned algorithms is that the quality of the neighborhood aggregation procedure inevitably declines in a deep neural network architecture that has many graph convolutional layers (Klicpera, Bojchevski, and Günnemann 2018). The line chart (Fig.1) illustrates how the performance is degraded as the depth of the GCN model is increased. The main reason for this problem is that Laplacian smoothing in these aggregation (or propagation) schemes over-smooths the node features and renders the nodes from different clusters indistinguishable (Xu et al. 2018). Explicitly, adding multiple GCN layers is equivalent to repeatedly conducting a Laplacian smoothing operation, and the features of neighboring connected nodes in the graph will converge to the same values. In the case of spectral GCN, the features will converge in proportion to the square root of the node degree (Li, Han, and Wu 2018), which is the over-smoothing problem addressed in this paper.

In this paper, we propose a novel graph convolutional architecture, graph convolutional ladder-shape networks (GCLN), which is built upon one contracting path and one expanding path with contextual feature channels. The ladder-shape architecture allows the network to directly transmit and fuse the contextual information from the shallow layers to the deeper layers, which challenges the assumption that neighborhood aggregation-based architectures can only be shallow networks (e.g., the GCN performance will necessarily be degraded when there are more than three convolutional layers) because of the over-smoothing of features during the propagation procedure.

To further extend GCLN to exploit the hierarchical embedding information of graphs, we also fuse a differentiable graph pooling module (Ying et al. 2018) with our framework, so that GCLN can infer and aggregate the local topological structure as well as the coarse-grained topological structure over graphs. We have not only evaluated the effectiveness of GCLN with a semi-supervised task and unsupervised task at the node-wise level, but have also conducted graph classification to validate its predictive performance at the graph-wise level.

Our main contributions are summarized as follows:

- We propose a novel graph convolutional network symmetrically constructed on one contracting and one expanding path with contextual feature channels, which dramatically extends the depth of spectral graph convolutional networks.
- The proposed graph convolutional ladder-shape network (GCLN) solves the problem of over-smoothed features with many convolutional layers in neighborhood aggregation-based neural networks.
- Extensive experiments on semi-supervised and unsupervised node-wise node level tasks, and a graph-wise task compared with sufficient classical and state-of-the-art baselines demonstrate the effectiveness of GCLN.

Problem Definition

A graph is defined as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\{\mathsf{v}_i\}_{i=1,\cdots,n} \in \mathcal{V}$ represents all the nodes in a graph. $\mathsf{e}_{i,j} = (\mathsf{v}_i, \mathsf{v}_j)_{i,j=1,\cdots,n} \in \mathcal{E}$ indicates the set of binary linkage relationships between two nodes, where $\mathsf{e}_{i,j} = 1$ if there is an edge between nodes v_i and v_j, otherwise, $\mathsf{e}_{i,j} = 0$. An adjacency matrix $\mathbf{A}$ is used to mathematically present the topological information of the graph $\mathcal{G}$, where each cell of the matrix $\mathbf{A}$ maps the corresponding edge information encoded in $\mathcal{E}$. A feature matrix $\mathbf{X}$ preserves the features $\mathsf{x}_i \in \mathbf{X}$ pertinent to the node v_i.

Node-wise Embedding Definition

Given $\mathcal{G}$, the objective of graph neural networks is to embed all the nodes $\mathcal{V}$ into a compact space $\mathbf{Z} \in \mathbb{R}^{\mathsf{n}\times\mathsf{d}}$ through $f : \mathbf{(A, X)} \longmapsto \mathbf{Z}$, where d, normally a smaller number, is the dimension of the learned embedding. $\mathsf{z}_i \in \mathbf{Z}$ indicates the encoded node v_i with associated topological information from $\mathbf{A}$ and feature information from $\mathbf{X}$. Ideally, nodes with similar characteristics are expected to remain close in the embedding space. With well-learned embedding $\mathbf{Z}$ of the graph $\mathcal{G}$, the classical machine learning methods can be directly applied for semi-supervised tasks like node classification and unsupervised tasks like node clustering.

Graph-wise Classification Definition

Given a set of aforementioned graphs $\mathsf{G} = \{(\mathcal{G}_1, \mathsf{y}_1), (\mathcal{G}_2, \mathsf{y}_2), ..., (\mathcal{G}_k, \mathsf{y}_k)\}$ where $\mathcal{G}_k \in \mathsf{G}$ with the corresponding labels $\mathsf{y}_k \in \mathsf{Y}$, the purpose of graph-wise tasks such as graph classification is to learn a mapping $f : \mathsf{G} \longmapsto \mathsf{Y}$ which assigns the given graphs to the labels.

Each given graph is embedded into a low-dimensional vector based on which the class label of the whole graph can be inferred.

Preliminaries

The proposed GCLN is most related to two models: spectral graph convolutional networks and u-shape networks. We also employ the differentiable graph pooling module to graph-wisely validate the effectiveness of GCLN. We elaborate the details of these three models in this section.

Graph Convolutional Networks

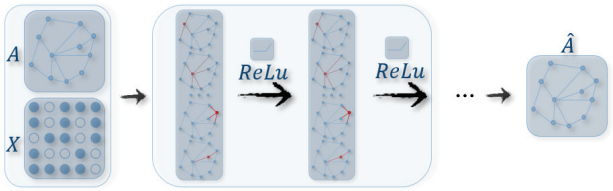

Figure 2: The architecture of a typical spectral graph convolutional network. The performance of GCNs with more than two or three layers will dramatically drop as a result of the over-smoothing problem.

Standard graph convolutional networks (GCNs) (Kipf and Welling 2016b), as shown in Figure 2, comprise a two-layer semi-supervised framework which propagates the features of a node over its neighboring nodes. The objective of GCNs is to learn a spectral function $f(\mathbf{X}, \mathbf{A})$ where the adjacency matrix $\mathbf{A}$ represents the topological characteristics of the graph and matrix $\mathbf{X}$ preserves the interdependence of the nodes and features. The propagation rule is more likely to represent each node in the graph by aggregating its neighborhood with self-loops, and the output of the networks is a normalized node-level representation (or graph embedding), which represents each node with a low-dimensional vector. Because the spectral function $f(\bullet)$ conducts the feature smoothing on the topological adjacent matrix $\mathbf{A}$, the networks can distribute the gradient through the supervised error and learn the representation of both annotated and unlabeled nodes.

U-Net

The U-Net (Ronneberger, Fischer, and Brox 2015) is an elegant U-shape fully convolutional network architecture developed for bio-medical image segmentation. The architecture consists of two parts: the down-sampling path and the up-sampling path.

The down-sampling path is comprised of four blocks with two convolutional layers and one max pooling (stride=2) layer for each block. At each step following the down-sampling path, the number of feature channels is doubled. The contracting path captures the contextual features from the input image for segmentation and passes the features to the up-sampling path with skip connections.

The up-sampling path is also comprised of four similar de-convolutional blocks and enables the model to simultaneously obtain accurate localization information and sufficient contextual information from the down-sampling path.

The U-Net adequately and simultaneously exploits the precise localization from the up-sampling path and the contextual information from the down-sampling path. The two sources of information are correspondingly concatenated through the feature channels during the training for bio-medical image segmentation.

Differentiable Graph Pooling

The differentiable graph pooling module (Ying et al. 2018) is designed to enable graph neural networks to conduct predictive tasks over graphs rather than over nodes of one graph.

Graph pooling is necessary for graph-wise tasks such as graph classification, and the challenge of the pooling operation in the graph setting, compared to the computer vision setting, is that unlike an image, a graph has no natural spatial location and unified dimension.

The differentiable graph pooling module solves the aforementioned challenges by learning a differentiable soft assignment at each layer of a GNN model, assigning nodes to a set of clusters according to the learned representations. Thus, the pooling operation of each GNN layer coarsens the given graph iteratively and generates a hierarchical representation of every input graph on completion of the entire training procedure.

Graph Convolutional Ladder-shape Networks

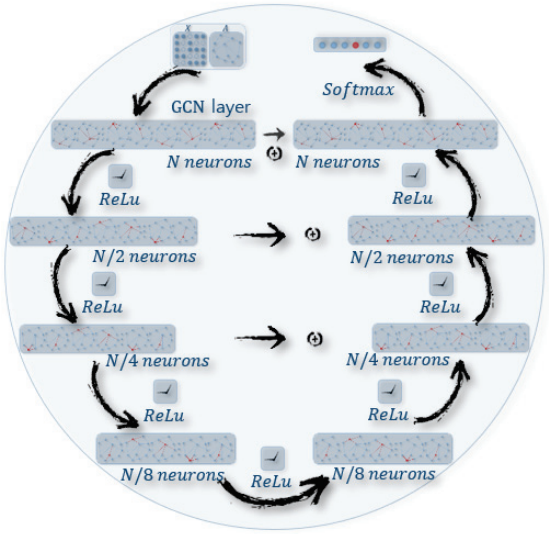

Figure 3: Illustration of a graph convolutional ladder-shape networks (GCLNs). The network consists of two symmetric paths: a contracting path (left) and an expanding path (right). The contextual feature channels between the two paths are used to pass and fuse contextual information from the contracting path to the location information from the expanding path in a simple but elegant operation.

The proposed GCLN is a symmetric architecture constructed of one contracting path and one expanding path with GCN layers. Three contextual feature channels allow the context features captured from the contracting path to fuse with the localization information learned through the expanding path. Each layer is built with the spectral graph convolution and there are eight GCN layers in total in the proposed framework.

Graph Convolutional Operation

The layers of GCLN are constructed on the first order approximation of Chebyshev spectral CNNs. The graph convolution here can be defined by the normalized graph Laplacian matrix :

$$g_\theta \otimes \mathsf{x} = \theta_0 \mathbf{I}\mathsf{x} - \theta_1 \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathsf{x}, \tag{1}$$

where g_θ is the spectral filter, $\mathsf{x} \in \mathbb{R}^n$ indicates the signal of the graph and $\otimes$ represents the convolution operator. $\mathbf{I}\mathsf{x}$ is the identity matrix. $\mathbf{D}$ denotes the diagonal degree matrix of topological adjacency matrix $\mathbf{A}$ and θ_k is the Chebyshev coefficients.

Due to the condition of the first order approximation (Kipf and Welling 2016b), the Chebyshev coefficients are further simplified with $\theta = \theta_1 = -\theta_0$, and the graph convolution is updated as:

$$g_\theta \otimes \mathsf{x} = \theta \left(\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)\mathsf{x}, \tag{2}$$

The convolution matrix is normalized through:

$$\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \longmapsto \mathbf{I} + \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}} \tag{3}$$

where $\widetilde{\mathbf{D}} = \sum_j \widetilde{\mathbf{A}}_{ij}$ and $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$.

Adopting the definition of graph convolution, given a graph signal with m feature channels (e.g., $\mathbf{X} \in \mathbb{R}^{n\times m}$, m features associated with each node), the layer-wise propagation rule of the spectral graph convolutional networks is defined as:

$$\mathbf{H}^{(l+1)} = \varphi\left(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^l\mathbf{W}^l\right). \tag{4}$$

where $\mathbf{H}^0 = \mathbf{X}$ and $\mathbf{H}^l$ is the activation from the l_{th} GCN layer. $\mathbf{W}^l$ is the trainable weight matrix and φ is the activation function such as $ReLU(\bullet)$ and $Linear(\bullet)$.

For semi-supervised multi-class classification, the softmax activation function is applied, row-wisely, to the final embedding of the graph. The form of the classifier is defined as follows:

$$\mathcal{C} = \mathsf{softmax}\left(\hat{\mathbf{A}}ReLU\left(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(l-1)}\right)\mathbf{W}^l\right). \tag{5}$$

where $\hat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}$ and $\mathbf{W}^l$ is the weight matrix between the last GCN-layer and the output layer.

The node classification loss function is written with cross-entropy error as follows:

$$\mathcal{L} := \sum_{i\in V_l}\sum_{f=1}^{F} \mathcal{Y}_{if}\ln \mathcal{C}_{if}. \tag{6}$$

where $\mathbf{V}_l$ denotes the set of annotated nodes and $\mathbf{F}$ is the number of classes. $\mathcal{Y}_{if}$ is the label indicator matrix mapping the predicted nodes.

GCLN Contracting Path

The contracting path is a four-layer graph convolutional embedding architecture in which each layer halves the size (the number of neurons) of the previous layer. Each GCN layer conducts the graph convolution with Eq.(4), followed by the $ReLU$ activation and dropout.

The contracting path can be considered as the encoder part if GCLN is interpreted as an end-to-end encoder-decoder architecture which encodes the topological characteristics and features associated with each node into feature representations at multiple echelons. The contextual information of the graph is captured through the contracting path and preserved at each layer, and will be correspondingly transmitted to the expanding path via the contextual feature channels.

GCLN Expanding Path

The expanding path mirrors the architecture of the contracting path with the arrangement of the layers reversed. Each layer doubles the number of neurons in the previous layer. An element-wise summation operation is conducted to receive and fuse the contextual information skipped from the contracting path as follows:

$$\mathbf{H}^l_{expanding} = summation\left(\mathbf{H}^l, \mathbf{H}^{(\mathbf{L}-l)}\right). \tag{7}$$

where $\mathbf{H}^l$ is the latent representation from the l^{th} layer and $\mathbf{L}$ is the number of graph convolutional layers. For the experiments described in this paper, $\mathbf{L} = 8$.

The contextual feature channels allow the network to go deeper and to fuse the contextual features from the contracting side with the up-convolutional features from the expansive side. This summation enables representations to be better localized and obtained following many convolutions. The indistinguishable features of nodes caused by repetitively conducting a symmetrically normalized Laplacian smoothing operation are directly characterized and enhanced, enabling them to merge with the contextual features from the contracting path. As a result, the over-smoothing problem of neighborhood aggregation-based algorithms is overcome.

A softmax activation takes the output from the last graph convolutional layer to conduct the semi-supervised node classification with Eq.(5) and Eq.(6).

Algorithm for GCLN

The pseudo-code of GCLN is described in Algorithm 1. Step 2 walks through the contracting path and collects the contextual information of the graph layer by layer, while Steps 3 and 4 lead the network to go deep along the expansive path and fuse the corresponding contextual information via the feature channels. Steps 5 and 6 conduct the semi-supervised node classification and update the network.

In particular, the input for each graph convolution layer in the expanding path is the corresponding result of the summation, rather than the direct latent matrix $\mathbf{H}_{exp}$.

Fig. 3 demonstrates the construction of the proposed GCLN.

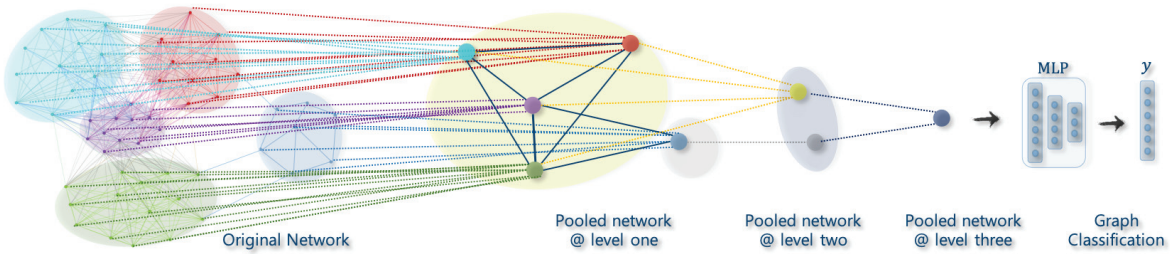

Figure 4: Illustration of differentiable graph pooling procedure. At each hierarchical layer, the embeddings of nodes are learned and obtained through a GCN layer, after which the nodes are clustered according to the learned embeddings into the coarsened graph for another GCN layer. The process is iterated for l layers and the final output is used for the graph classification task.

Algorithm 1 Graph Convolutional Ladder-Shape Networks for Node Classification

Require:
$\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$: a graph with nodes and edges;
$\mathbf{X}$: the feature matrix;
T: the number of iterations;
o: the number of the neurons in the first convolution layer;

Ensure: o is divisible by 8.

1: **for** iterator = 1,2,3, $\cdots\cdots$, T **do**
2: Generate latent matrix $\mathbf{H}$ via Eq.(4) passing contracting path;
3: Generate direct latent matrix $\mathbf{H}_{exp}$ via Eq.(4) passing expanding path;
4: Conduct corresponding summation via Eq.(7) with $\mathbf{H}$ and $\mathbf{H}_{exp}$;
5: Get the prediction results via Eq.(5)
6: Update the model with the loss computed via Eq.(6);
7: **end for**

Extension - GCLN with Differentiable Pooling

To extend GCLN for the graph-wise task, we have added a differentiable pooling layer behind each GCN layer and taken the output from the last layer for graph classification. The differentiable graph pooling formulates a general recipe for hierarchically pooling nodes across a set of graphs by generating a cluster assignment matrix $\mathcal{S}$ over the nodes leveraging the output of each GCN layer. The cluster matrix $\mathcal{S}$ on layer l is computed as follows:

$$\mathcal{S}^{(l)} = softmax\left(\mathsf{GCN}\left(\mathbf{A}^{(l)}, \mathbf{X}^{(l)}\right)\right). \tag{8}$$

where $\mathsf{GCN}(\mathbf{A}, \mathbf{X})$ is the graph convolutional operation elaborated in the last subsection and the softmax function is row-wisely conducted.

With cluster assignment matrix $\mathcal{S}$, the differentiable pooling layer coarsens the given graph, generating a new adjacency matrix $\mathbf{A}^{(l+1)}$ and a new matrix $\mathbf{X}^{(l+1)}$ by applying the following equations:

$$\mathbf{X}^{(l+1)} = {\mathcal{S}^{(l)}}^T \mathbf{H}^l \in \mathbb{R}^{\mathsf{n}_{l+1} \times \mathsf{d}}. \tag{9}$$

$$\mathbf{A}^{(l+1)} = {\mathcal{S}^{(l)}}^T \mathbf{A}^l \mathcal{S}^{(l)} \in \mathbb{R}^{\mathsf{n}_{l+1} \times \mathsf{n}_{l+1}}. \tag{10}$$

The generated $\mathbf{A}^{(l+1)}$ and $\mathbf{X}^{(l+1)}$ will be processed into the next GCN layer.

Fig. 4 illustrates the general process of the differentiable graph pooling.

Experiments

In this section, we set up the experiments compared to 15 classical and state-of-the-art methods to demonstrate the solid performance of GCLN on graph node classification. The experiments with 8-layer GCN and GAT validate the existence of the over-smoothing problem and demonstrate that GCLN is the solution to the problem. The contrast experiments compared residual GCN and GAT to illustrate that residual connection cannot effectively ameliorate the problem. We also reveal the effectiveness of GCLN on unsupervised learning tasks with clustering experiments compared to 16 peers. In addition, GCLN proves its predictive capability with the differentiable graph pooling module on graph-wise tasks such as graph classification.

Data Sets

We conducted the experiments using three real-world bibliographic data sets: Cora, Citeseer and Pubmed (Sen et al. 2008) and the details of the data set statistics are summarized in Table 1. The data sets are used for both node-wise classification and clustering.

Node Classification

We conducted node classification to validate the effectiveness of GCLN on node-wise semi-supervised tasks.

Baseline Algorithms GCLN is compared with **Classic methods** and **GCN-based algorithms** such as Chebyshev (Defferrard, Bresson, and Vandergheynst 2016), GCN (Kipf and Welling 2016b), GraphInfoMax (Veličković et al. 2018), LGCN (Gao, Wang, and Ji 2018), StoGCN (Chen, Zhu, and Song 2018), DualGCN (Zhuang and Ma 2018), GAT (Velickovic et al. 2017), and g-U-Nets (Gao and Ji 2019).

Node Classification Setup We used an eight GCN-layer GCLN (except for the input layer and the layer with softmax) to conduct all the experiments. The first layer consists of 64 neurons and each following layer in the contracting

Table 1: Datasets for Node Classification and Clustering

	# Nodes	# Edges	# Features	# Classes	# Training Nodes	# Validation Nodes	# Test Nodes	Label rate
Citeseer	3,327	4,732	3,703	6	120	500	1,000	0.036
Cora	2,708	5,429	1,433	7	140	500	1,000	0.052
Pubmed	19,717	443,388	500	3	60	500	1,000	0.003

path halves the number of neurons in the previous layer. Because of the symmetric architecture of GCLN, the first layer in the expanding path starts with 8 neurons and each subsequent layer doubles the number of neurons, until 64 neurons are found in the last layer. 0.9 dropout and the ReLU activation function were applied after every graph convolutional operation. The learning rate was retained at 0.01 for all the experiments.

To verify the existence of the over-smoothing issue in the neighbor aggregation algorithms, we also set up the experiments for 8-layer GCN and GAT (the same number of layers as GCLN) to compare them with GCLN. For even fairer comparison, we constructed the residual 8-layer GCN and GAT, which adds a residual connection behind every layer, and compared it with GCLN. It is noteworthy that the residual 8-layer GCN and GAT contain many more parameters than GCLN.

Each experiment was conducted ten times and the average scores are reported as detailed below.

Node Classification Results The experimental results are summarized in Table 2. The GCN-based algorithms, such as GCN, GAT, LGCN, StoGCN and DualGCN, take advantage of the neighbor aggregation operation to propagate the feature of a node over its neighboring nodes, and outperformed classical methods such as TADW and DeepWalk.

Compared to two-layer GCN and two-layer GAT, the classification performance of the 8-layer variations were significantly degraded (more than 60%), which verified the over-smoothing issue in the graph convolution-based methods.

GCLN outperformed or matched all fifteen state-of-the-art/classical peers, 8-layer GCN and 8-layer GAT (with and without residual connections) across all three data sets. The large margin (26.3% to 173.1%) of difference in the classification results between GCLN, 8-layer GCN and 8-layer GAT directly proves that GCLN successfully addresses the over-smoothing problem of neighborhood aggregation-based algorithms caused by the repetitious application of the Laplacian smoothing operation.

To validate whether applying a residual mechanism to spectral graph convolutional algorithms could alleviate the over-smoothing problem, we set the experiments by adding residual connections to GCN and GAT with eight layers to vie with the normal eight-layer GCN and GAT as well as GCLN. The last five rows of Table 2 illustrate that adding a residual connection after each GCN layer dramatically enhanced performance when the network deepened (8 layers). However, there is still a large margin between GCLN and residual GCN. For GAT, the residual connections resulted in a performance reduction compared to 8-layer GAT in all three data sets. The last five sets of contrast experiments validate that the naïve residual connection is not an effective solution to the over-smoothing problem.

Table 2: Comparison of classification accuracy

Approaches	Cora (%)	Citeseer (%)	PubMed (%)
MLP	55.1	46.5	71.4
ManiReg	59.5	60.1	70.7
SemiEmb	59.0	59.6	71.7
LP	68.0	45.3	63.0
DeepWalk	67.2	43.2	65.3
TADW	76.7	64.8	54.7
Chebyshev	81.2	69.8	74.4
GraphInfoMax	82.3	71.8	76.8
LGCN	83.3	73.0	79.3
StoGCN	82.0	70.9	79.0
DualGCN	83.5	72.6	**80.0**
GCN	81.5	70.3	79.0
g-U-Nets	79.8	65.8	76.7
GAT	83.0	72.5	79.0
GAT 8 layers	31.9	21.8	40.7
GCN 8 layers	30.9	36.2	63.4
GCN 8 Residual	81.9	64.9	76.0
GAT 8 Residual	29.8	19.8	40.6
GCLN	**84.2**	**73.3**	**80.1**

Node Clustering

We evaluated the node-wise unsupervised predictive performance of GCLN by conducting a clustering task on the same real-world data sets used for the node classification. Following (Kipf and Welling 2016a), we remove the Softmax layer in the model and reconstruct the topological information at the last layer as the embedding of the given graph to train a k-means for node clustering task.

Baseline Algorithms GCLN is compared with two groups of peers: **1) Single source information-leveraged algo-**

Table 3: Clustering results

Algorithms	Cora		Citeseer		Pubmed	
	Acc(%)	F1(%)	Acc(%)	F1(%)	Acc(%)	F1(%)
GraphCoder	32.5	29.8	22.5	30.1	53.1	50.6
DNGR	41.9	34.0	32.6	30.0	45.8	46.7
GAE	59.6	59.5	40.8	37.2	67.2	66.0
VGAE	60.9	60.9	34.4	30.8	63.0	63.4
ARGA	64.0	61.9	57.3	54.6	**66.8**	65.6
GCLN	**68.8**	**66.1**	**59.3**	**59.8**	66.2	**66.2**

Table 4: Datasets for Graph Classification

	# Nodes(max)	# Nodes(avg.)	# Graphs	# Edges(avg.)	# Nodes Labels	# Classes	Sources
D&D	5,748	284.32	1,178	715.65	82	2	Bio
ENZYMES	126	32.60	600	63.14	6	6	Bio
PROTEINS	620	39.06	1,113	72.81	4	2	Bio

rithms such as DNGR (Cao, Lu, and Xu 2016), Graph Encoder (Tian et al. 2014), GAE* (Kipf and Welling 2016a), VGAE*; and **2) Both content and structure-leveraged algorithms** such as GAE, VGAE, ARGA. Due to the page limitation, some of baselines are not listed in the table.

Clustering Results Table 3 lists the experimental results on Cora, Citeseer and Pubmed respectively. We conducted experiments with other ten baselines and with more metrics including normalise mutual information (NMI), Precision, and Adjusted Rand index (ARI). We only demonstrate six representative peers due to the limited space.

GCLN outperforms all baselines on Cora and Citeseer under the five metrics, and achieved competitive performance compared with the latest algorithm on Pubmed. For example, on Cora, GCLN improves the accuracy from 7.5% (compared with ARGA) to 69.1% (compared with RMSC); raises the F1 score from 6.8% (compared with ARGA) to 79.6% (compared with K-means); and improves NMI from 21.8% (compared with TADW) to 72.3% (compared with DNGR).

The results from methods such as DeepWalk and BigClam, which only consider a single source of information from the given graph, are inferior to the performance of those that simultaneously leverage topological structure and node characteristics. The graph convolutional models with adversarial regularization also show their superiority over their peers on the clustering task. GCLN, with 8 layers, is not affected by the over-smoothing issue and maintains the optimal performance over all sixteen baselines.

Graph Classifcation

We validated the graph-wise performance of GCLN with differentiable pooling module by conducting graph classification on three biological graph data sets summarized in the Table 4.

Baseline Algorithms We compared GCLN with both **1) graph kernel-based methods** such as GK (Shervashidze et al. 2009), DGK (Yanardag and Vishwanathan 2015), WL-OA (Kriege, Giscard, and Wilson 2016); and **2) deep learning models** such as DCNN (Atwood and Towsley 2016), ECC (Simonovsky and Komodakis 2017), DGCNN (Zhang et al. 2018), DIFFPOOL; and **3) CapsuleNet-based** such as GCAPS-CNN (Verma and Zhang 2018) and GCAPS (Xinyi and Chen 2019).

Experimental Setup To objectively evaluate the effectiveness of GCLN on graph-wise level prediction, we applied ten-fold cross validation to conduct the graph classification experiments. Specifically, Eight-fold training was used to train the model, one training fold was used as the validation set for hyper-parameter adjustment, and the remaining fold was used for testing. Each experiment was conducted ten times and the average accuracy is reported.

Graph Classification Results Table 5 compares the performance of GCLN at a graph-wise level with other baselines. The results demonstrate that GCLN obtains the optimal average performance over all graph classification algorithms. The 8-layer GCLN with DIFFPOOL increases the accuracy compared to the 2-layer GraphSAGE(Hamilton, Ying, and Leskovec 2017) with DIFFPOOL. GCLN has also outperformed the CasuleNet-based algorithms by around 4.3% accuracy as well as graph kernel-based methods by more than 10%. The accuracy of the graph classification shows that the architecture of GCLN not only tackles the over-smoothing problem at a node-wise level, but also achieves stable performance at a graph-wise level.

Table 5: Graph Classification Results

Approaches	D&D (%)	ENZYMES (%)	PROTEINS (%)
GK	78.45	32.7	71.67
DGK	73.5	53.43	75.68
WL-OA	79.04	60.13	75.26
DCNN	58.09	42.44	61.29
ECC	72.54	45.67	72.65
DGCNN	79.37	51	75.54
DIFFPOOL	80.64	62.53	76.25
g-U-Nets	81.01	61.77	75.12
GCAPS-CNN	77.62	61.83	76.40
CapsGNN	75.38	54.67	76.28
GCLN+DP	**81.20**	**62.79**	**77.47**

Conclusion

Neighborhood aggregation algorithms inevitably suffer from the over-smoothing problem when the depth of the network is increased, because repeatedly conducting the Laplacian smoothing operation leads to the features of neighboring connected nodes in the graph converging to the same values. In this paper, we have proposed graph convolutional ladder-shape networks (GCLNs) which address the over-smoothing problem with a symmetric ladder-shape architecture. The network consists of a contracting path, expanding path and contextual feature channels which characterize and enhance indistinguishable features by fusing the corresponding contextual information from the contracting side to the deeper layers. A comparison of the results of our experiments on classical and state-of-the-art peers, 8-layer GCN,

8-layer GAT and their residual versions prove the superiority of our method.

We have experimentally evaluated the performance of GCLN from the perspective of a node-wise semi-supervised task and node-wise unsupervised task as well as a graph-wise task. All the experiments results validate the optimal effectiveness of GCLN from multiple perspectives.

References

Atwood, J., and Towsley, D. 2016. Diffusion-convolutional neural networks. In *NIPS*, 1993–2001.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Cao, S.; Lu, W.; and Xu, Q. 2015. Grarep: Learning graph representations with global structural information. In *CIKM*, 891–900. ACM.

Cao, S.; Lu, W.; and Xu, Q. 2016. Deep neural networks for learning graph representations. In *AAAI*, 1145–1152.

Chen, J.; Zhu, J.; and Song, L. 2018. Stochastic training of graph convolutional networks with variance reduction. In *ICML*, 941–949.

Dai, H.; Kozareva, Z.; Dai, B.; Smola, A.; and Song, L. 2018. Learning steady-states of iterative algorithms over graphs. In *ICML*, 1114–1122.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 3844–3852.

Gao, H., and Ji, S. 2019. Graph U-nets. In *Proceedings of The 36th International Conference on Machine Learning*.

Gao, H.; Wang, Z.; and Ji, S. 2018. Large-scale learnable graph convolutional networks. In *KDD*, 1416–1424. ACM.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*.

Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *KDD*, 855–864. ACM.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NIPS*.

Kipf, T. N., and Welling, M. 2016a. Variational graph auto-encoders. *NIPS*.

Kipf, T. N., and Welling, M. 2016b. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2018. Combining neural networks with personalized pagerank for classification on graphs.

Kriege, N. M.; Giscard, P.-L.; and Wilson, R. 2016. On valid optimal assignment kernels and applications to graph classification. In *NIPS*, 1623–1631.

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. *arXiv preprint arXiv:1801.07606*.

Liu, L.; Zhou, T.; Long, G.; Jiang, J.; Yao, L.; and Zhang, C. 2019a. Prototype propagation networks (ppn) for weakly-supervised few-shot learning on category graph. In *IJCAI*.

Liu, L.; Zhou, T.; Long, G.; Jiang, J.; and Zhang, C. 2019b. Learning to propagate for graph meta-learning. In *NeurIPS*.

Pan, S.; Wu, J.; Zhu, X.; Zhang, C.; and Wang, Y. 2016. Tri-party deep network representation. *Network* 11(9):12.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 234–241. Springer.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine* 29(3):93.

Shervashidze, N.; Vishwanathan, S.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. 2009. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, 488–495.

Simonovsky, M., and Komodakis, N. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, 3693–3702.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *WWW*, 1067–1077. International World Wide Web Conferences Steering Committee.

Tian, F.; Gao, B.; Cui, Q.; and et.al. 2014. Learning deep representations for graph clustering. In *AAAI*, 1293–1299.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* 1(2).

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341*.

Verma, S., and Zhang, Z.-L. 2018. Graph capsule convolutional neural networks. *arXiv preprint arXiv:1805.08090*.

Wang, X.; Cui, P.; Wang, J.; Pei, J.; Zhu, W.; and Yang, S. 2017. Community preserving network embedding. In *AAAI*, 203–209.

Xinyi, Z., and Chen, L. 2019. Capsule graph neural network. In *ICLR*.

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.-i.; and Jegelka, S. 2018. Representation learning on graphs with jumping knowledge networks. *arXiv preprint arXiv:1806.03536*.

Yanardag, P., and Vishwanathan, S. 2015. Deep graph kernels. In *SIGKDD*, 1365–1374. ACM.

Yang, C.; Liu, Z.; Zhao, D.; Sun, M.; and Chang, E. Y. 2015. Network representation learning with rich text information. In *IJCAI*, 2111–2117.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. In *NIPS*, 4800–4810.

Zhang, Q.; Wu, J.; Yang, H.; Lu, W.; Long, G.; and Zhang, C. 2016. Global and local influence-based social recommendation. In *CIKM*, 1917–1920. ACM.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhuang, C., and Ma, Q. 2018. Dual graph convolutional networks for graph-based semi-supervised classification. In *WWW*, 499–508.