Learning Graph Representations with Maximal Cliques

Soheila Molaei*, Nima Ghanbari Bousejin*, Hadi Zare, Mahdi Jalili, *Senior Member, IEEE*, Shirui Pan

***Abstract*—Non-Euclidean property of graph structures has faced interesting challenges when deep learning methods are applied. Graph convolutional networks can be regarded as one of the successful approaches to classification tasks on graph data, although the structure of this approach limits its performance. In this work, a novel representation learning approach is introduced based on spectral convolutions on graph-structured data in a semi-supervised learning setting. Our proposed method, COOL (COnvOlving cLiques), is constructed as a neighborhood aggregation approach for learning node representations using established graph convolutional network architectures. This approach relies on aggregating local information by finding maximal cliques. Unlike the existing graph neural networks which follow a traditional neighborhood averaging scheme, COOL allows for an aggregation of densely connected neighboring nodes of potentially differing locality. This leads to substantial improvements on multiple transductive node classification tasks.**

***Index Terms*—Deep learning, Graph representation learning, Graph convolutional networks, Graph neural networks, Network embedding**

I. Introduction

Many interesting phenomena such as social, biological, financial and brain connectomes involve data that can be readily represented in the graph-based structures with acquiring dependencies between the elements under the study [1]. One of the main challenges to graph-structured data is the extraction of structural graph information to embed in a machine learning models named as representation learning. The primary aim is to optimize the embedding process to preserve the original graph structure in the learned space [2]. Motivated by the success of Deep Learning (DL), several attempts have been made to apply DL as a mature learning technology on graph-structured data from social, biological, and financial domains. Graph Neural Networks (GNNs) have emerged as a promising approach for representation learning on relational data [3]–[6]. They were introduced as a generalization of the convolutional operator from regular grids to graph-structured data [7]. GNNs have shown remarkable success for relational reasoning over graph-structured representations [8], [9].

Representation learning on graphs typically requires aggregation of useful neighborhood information to succeed. Better representation learning provides more generalization power on graph-structured inputs. To learn meaningful representations, semi-supervised learning is one of the most promising methods [10]. Depending on the availability of the label information and the aim of the underlying task, there are supervised [11], [12], semi-supervised [13] and unsupervised [14] approaches [15]. Graph-level classification is comprised of high-level node representation in convolutional layers, graph pooling layers to perform downsampling and readout layers to provide the final graph representation for the aim of label prediction on a graph structure [16], [17]. Unsupervised learning of graph embedding is commonly performed by exploiting the edge-level structures via the autoencoder framework [18].

Semi-supervised Graph Neural Network (GNN) approaches involves constructing an end-to-end procedure by applying a bunch of graph convolutional layers aligned with a softmax layer. Due to the challenges of labeling large datasets, most real networks consist of only limited labeled nodes while others mostly being unlabeled. Therefore, semi-supervised learning is categorized among the most practical methods to deal with real networks [19]. Several approaches have been proposed for semi-supervised representation learning with graph-structured data. Examples include GCN [19], Planetoid [20], Chebyshev [21] and APPNP [22]. These approaches rely on message passing-based objectives that over-emphasize proximity information at the expense of limiting the representational capacity of the model [23]. Since the neighborhood is a local concept defined by some notion of proximity in the network, approaches that encode nodes using a generalized notion of their neighborhood, enforce an inductive bias that neighboring nodes have similar representations.

In this paper, we propose an alternative unsupervised graph learning approach that is based upon mutual information. Our approach relies on aggregating neighborhood representations from the potentially differing locality. The idea is to compute the hidden representations of each node in the graph by attending over its highly-connected neighbors. Our approach allows focusing on the most relevant parts of the input, and can thus be a strong predictor for node classification tasks. We demonstrate that the representation learned by COOL is competitive on six standard benchmark datasets and outperforms state of the art baselines in our experiments on both clean and noisy data. Contributions of our study are as follows:

- We propose COOL, a new neighborhood aggregation method for learning node representations.
- We propose a novel clique-based edge-weight computation that is general and architecturally simple.
- The proposed approach learns meaningful representations

*Equal contribution

Hadi Zare, Soheila Molaei, and Nima Ghanbari Bousejin are with the Faculty of New Sciences and Technologies, University of Tehran, Iran. Mahdi Jalili is with the School of Engineering, RMIT University, Melbourne, Australia, Shirui Pan is with Faculty of Information Technology, Monash University.

E-mail: soheila.molaei@ut.ac.ir, bousejin@gmail.com, h.zare@ut.ac.ir, mahdi.jalili@rmit.edu.au, shirui.pan@monash.edu

Manuscript received ; revised .

of graph structures by obtaining maximal cliques from neighbors at various distances.
- We establish two scenarios to showcase the strong denoising capabilities of our model in a range of node classification tasks.

Our paper shows significant leaps in performance compared to previous methods in a range of node classification tasks.

II. Related Works

Of particular interest to this work are methods that employ neighborhood aggregation, graph convolutional network and node representation learning.

Neighborhood Aggregation. Some works have been considered neighborhood aggregation for representation learning on graphs, including [24], [25]. A flexible approach in [24] was proposed by leveraging different neighborhood ranges for each node to enable better structure-aware representation. In [25], a new embedding for a node was considered by attending to all previous embeddings of its neighbors. GAT [1] leveraged masked self-attentional layers to address the shortcomings of prior methods using graph convolutions. COOL combines maximal cliques across different neighborhood branches. Our integrated approach learns node representations in an efficient way to contrast the features of immediate neighbors from those further away.

Graph Convolutional Network. Motivated by the success of Convolutional Neural Network (CNN) [26], Spectral Graph Convolutional Network [27] defined the convolution operation in the Fourier domain by computing the eigendecomposition of the graph Laplacian, resulting in potentially intense computations and non-spatially localized filters. Chebyshev [21] reduces computation complexity by approximating the filters by means of a Chebyshev expansion of the graph Laplacian. GCN [19] simplified the Chebyshev method by restricting the filters to operate in a one-hop neighborhood around each node. SGC [28] simplified GCN by erasing all the nonlinearities and collapsing the learning parameters into a single matrix. DGI [29] relies on maximizing mutual information between a high-level global representation and local parts of the input in an unsupervised manner.

Node Representation Learning. The goal of node representation learning is to learn distributed representations of nodes in graphs so that nodes with similar local connectivity tend to have similar representations [30]. Some representative methods include DeepWalk [31], LINE [32], node2vec [33] and GraphSAGE [34]. Despite their effectiveness in a variety of applications, these methods rely on random walk-based objectives. It is not that much clear whether they provide any useful signal, as these encoders already enforce an inductive bias that neighboring nodes have similar representations [29]. Here, we focus on graph convolutional networks which generate node representations by repeated aggregation over local node neighborhoods.

Message Passing. Some works extended graph convolutional networks to send and aggregate information through higher-order paths [19]. APPNP [22] proposed sampling nodes in the lower layers conditioned on their top layer. AS-GCN [35] developed a framework to accelerate the training of GCN through developing a sampling method by constructing the network layer by layer.

III. Preliminaries

Definition 1. Given a graph $G = (V, E)$ with a collection of two finite sets of V nodes and E edges, a clique is a subset of nodes such that each node in the set is adjacent to all other nodes in the set. In other words, a subset $C \in V$ is a clique if and only if $(v, v') \in E$ for all nodes v and $v' \in C$.

Cliques often unveil the tangible properties of the graph. Assuming that nodes in the graph represent people and edges friendship relationships, then a clique is a set of more than two people if they are all mutual friends of one another [36]. Given the notion of cliques, one can define maximal and maximum cliques as follows:

Definition 2. A maximal clique in a graph is a clique that cannot be extended by including one more adjacent node without compromising the connectivity property of the clique. In other words, a maximal clique is not a subset of a larger clique. Similarly, the maximum clique is the maximal clique with the highest number of nodes.

While cliques capture local representations of the graph, the maximum clique represents the global information content of the entire graph.

IV. Proposed Method

Motivation. Our aim is to perform node classification of graph-structured data by introducing a clique-based architecture. The idea is to specify different weights to different nodes in a neighborhood in which nodes are able to attend over their neighborhood's features through edge-weight strategy. The proposed architecture is efficient since it is parallelizable across node-neighbor pairs and it can be applied to graph nodes having different degrees by specifying arbitrary weights to the neighbors [1]. We validate the proposed approach by achieving state-of-the-art results that highlight the potential of clique-based models when dealing with arbitrarily structured graphs.

Notation. We use standard graph representations in semi-supervised node classification tasks. Graph G with N nodes is represented as a pair (X, A), where $X \in \mathbb{R}^{N \times F}$ denotes the node feature matrix with F features per node, and $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix. The objective is to classify each node into one of the target classes. Furthermore, our method assumes unweighted and directed graphs, i.e. $A_{ij} = 1$ if there is an edge from i to j and $A_{ij} = 0$ otherwise.

COOL. Our approach for learning meaningful representations is to obtain maximal cliques from neighbors at various distances. We employ an attention function, $\mathcal{T} : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}^{N \times N}$, to find all maximal cliques from the original graph using Bron-Kerbosch algorithm [37] with modifications to improve performance on large real-world graphs [38]. The algorithm has three disjoint sets of nodes R, P, and X, where R is a clique being constructed, P is a set of candidates to add to clique R, and nodes in X must be excluded from the clique. The algorithm chooses a candidate node v in P to

add to the clique R and v is moved to X when the recursive call returns. When P and X are empty, R is returned as a maximal clique. After finding maximal cliques, we construct a weighted adjacency matrix by sampling one-hop neighboring nodes such that $A'_{ij} = c$ represents nodes i and j are in a $(c+1)$-clique as shown in Figure 1. In extremely sparse settings, we find the two-hop neighboring nodes in addition to the one-hop neighborhood.

Our adjacency matrix, $\tilde{A} = \alpha \times A + \beta \times A'$, is a combination of unweighted and weighted adjacency matrices balanced by hyperparameters $\{\alpha, \beta\} \in \{0, 1\}$. To incorporate high-order proximities, we normalize this adjacency matrix using Algorithm 1 which takes as an input the adjacency matrix, $\tilde{A}$, the order, $T \geqslant 1$, which determines the order of proximities, and the threshold, $\nu > 0$, that helps to remove links with small probabilities to enhance sparsity. Our proposed method with this normalization procedure denoted as **COOL-norm**.

To generate high-level node representations, $\vec{h_i}$, we pass node features, X, and the normalized adjacency matrix, $\tilde{A}$, through the encoder function, $\mathcal{E} : \mathbb{R}^{N \times N} \times \mathbb{R}^{N \times F} \rightarrow \mathbb{R}^{N \times \mathbb{F}'}$, such that $\mathcal{E}(X, \tilde{A}) = \{\vec{h_1}, \vec{h_2}, \ldots, \vec{h_N}\}$ represents high-level representations $\vec{h_i} \in \mathbb{R}^{F'}$ for each node i. Our $\mathcal{E}$ function is a two-layer Graph Convolutional Network (GCN) model [19] with the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma\left(\hat{D}^{\frac{-1}{2}} \hat{A} \hat{D}^{\frac{-1}{2}} H^{(l)} W^{(l)}\right) \tag{1}$$

where $\hat{A} = \tilde{A} + I_N$ is the normalized adjacency matrix with inserted self-loops by an $N \times N$ identity matrix, I_N, and $\hat{D}$ is its corresponding degree matrix; i.e. $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$. For the nonlinearity, σ, we have applied ReLU function. $H^{(l)} \in \mathbb{R}^{N \times F}$ is the matrix of activations in the l-th layer with $H^{(0)} = X$ and $W^{(l)} \in \mathbb{R}^{F \times F'}$ is a layer-wise trainable linear transformation applied to every node, with $F' = 16$ features being computed.

The standard following cross-entropy loss function is used as our objective criterion over all labeled examples:

$$\mathcal{L} = -\sum_{l \in Y_L} \sum_{f=1}^{F} y_{lf} \log \widehat{y}_{lf} \tag{2}$$

where Y_L is the set of node indices that have labels, F is node features and y and $\widehat{y}$ are true labels and our predictions, respectively. A summary of the main steps of COOL is as follows:

1) Find all maximal cliques from the original graph by passing it through the attention function: $A' = \mathcal{T}(A)$.
2) Normalize the adjacency matrix, $\tilde{A} = \alpha \times A + \beta \times A'$, by combining the unweighted and weighted adjacency matrices balanced by hyperparameters, $\{\alpha, \beta\} \in \{0, 1\}$.
3) Obtain high-level representations, $\vec{h_i}$, for the maximal cliques of the input graph by passing it through the encoder function: $\mathcal{E}(X, \tilde{A}) = \{\vec{h_1}, \vec{h_2}, \ldots, \vec{h_N}\}$.
4) Update parameters of $\mathcal{E}$ by applying gradient descent to minimize Equation (2).

COOL-DGI. The existing methods often use semi-supervised learning with graph convolutional networks [19], which is often not possible as most graph data in the world is unlabeled. Thus, it is better to use an unsupervised learning approach for learning node representations within graph-structured data. Deep Graph Infomax (DGI) [29], is an unsupervised learning approach for learning node representations within graph-structured data. DGI uses graph convolutions [19] to build upon the deep mutual information maximization principle [39]. Here we use mutual information maximization principle and show that the learned embeddings can encode valuable information for node classification tasks.

Algorithm 1: Normalize adjacency matrix to capture high-order proximities. 1 is the vector of all ones, diag() means a diagonal matrix with respect to the given diagonal entries, $T \geq 2$ is the order and $v > 0$ is a threshold.

Input : $\tilde{A}$, T, ν
Output: Normalized $\tilde{A}$

$\tilde{A} \leftarrow diag^{-1}(\tilde{A}\vec{1})\tilde{A}$
$S, B \leftarrow \tilde{A}$
for $t \leftarrow 2$ **to** T **do**
  $B \leftarrow B\tilde{A}$
  $S \leftarrow S + B$
end for
$\tilde{A} \leftarrow \frac{1}{T} S \circ (1_{n*n} - I)$
$\tilde{A} \leftarrow \tilde{A} \circ \left(\tilde{A} > \nu\right)$
$\tilde{A} \leftarrow \tilde{A} + \tilde{A}^T + 2 \times I$
$\tilde{A} \leftarrow diag^{-1/2}\left(\tilde{A}\vec{1}\right)\ \tilde{A}\ diag^{-1/2}\left(\tilde{A}\vec{1}\right)$
return $\tilde{A}$

The unsupervised training setup is equivalent to the one in DGI. After finding all maximal cliques using the attention function, $\mathcal{T} : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}^{N \times N}$, an encoder, $\mathcal{E} : \mathbb{R}^{N \times N} \times \mathbb{R}^{N \times F} \rightarrow \mathbb{R}^{N \times \mathbb{F}'}$, computes high-level representation $\vec{h_i}$ for node i, using node features, X, and the normalized adjacency matrix, $\tilde{A}$. The discriminator, $\mathcal{D} : \mathbb{R}^{F \times F} \rightarrow \mathbb{R}$, then receives pairs of $(\vec{h_i}, \vec{s})$ containing the node representation and graph-level summary vector and outputs a score corresponding to whether a given pair represents a positive or negative sample by applying the following bilinear scoring function:

$$\mathcal{D}\left(\vec{h_i}, \vec{s}\right) = \sigma\left(\vec{h_i}^T W \vec{s}\right) \tag{3}$$

Here, W is a learnable scoring matrix and σ is the logistic sigmoid nonlinearity. Graph-level summary vectors, $\vec{s}$, is obtained using a readout function $\mathcal{R} : \mathbb{R}^{N \times F} \rightarrow \mathbb{R}^F$ which is a simple averaging of all the node's features:

$$\mathcal{R}(H) = \sigma\left(\frac{1}{N} \sum_{i=1}^{N} \vec{h_i}\right) \tag{4}$$

We refer to such a pair of node representation and graph-level summary as a positive sample if both are drawn from the same graph. A negative sample will consist of a node representation and graph-level summary obtained from a corrupted version of the graph, derived by randomly permuting the node features of the graph using a corruption function $\mathcal{C} : \mathbb{R}^{N \times N} \times \mathbb{R}^{N \times F} \rightarrow \mathbb{R}^{M \times M} \times \mathbb{R}^{M \times F}$. Both the encoder and discriminator are jointly trained to distinguish between

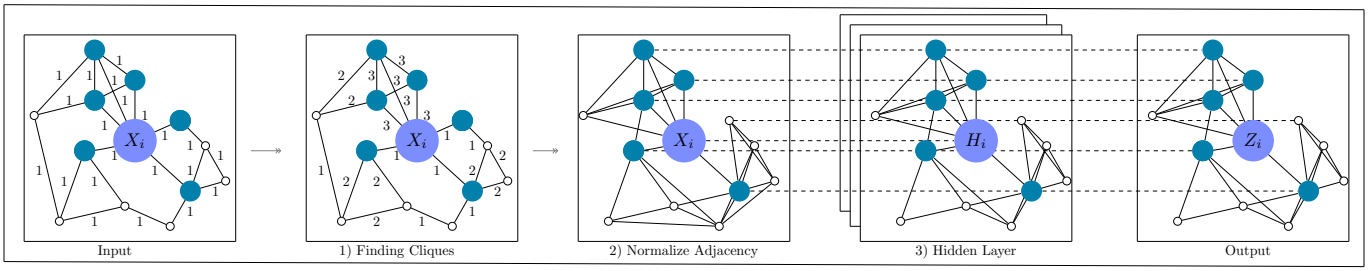


Fig. 1: A high-level overview of COOL-norm method with semi-supervised loss function: (1) find all maximal cliques (2) normalize the adjacency matrix (3) obtain high-level representations for normalized graph (4) update parameters.

positive and negative samples by maximizing the following standard binary cross-entropy loss:

$$\mathcal{L} = \frac{1}{N+M} \left(\sum_{i=1}^{N} \mathbb{E}_{(X,\tilde{A})} \left[\log \mathcal{D}\left(\vec{h_i}, \vec{s}\right) \right] + \sum_{i=1}^{M} \mathbb{E}_{(\tilde{X},\tilde{A})} \left[\log \left(1 - \mathcal{D}\left(\vec{\tilde{h}_j}, \vec{s}\right) \right) \right] \right) \tag{5}$$

The time complexity of finding cliques in sparse graphs is much less than dense graphs and could be done in $O(M+N)$, where M and N denote the number of edges and nodes respectively [40]. The time complexity of our encoders in COOL and COOL-DGI are of $O(N^2)$; thus, the overall time complexity is of $O(N^2)$ in the worst case. The space complexity of the proposed approach is of $O(LNF + LF^2)$ where L, N, and F are the number of layers, nodes, and features respectively.

Assuming the single-graph setup (i.e., (X, A) provided as input), we now summarize the steps of the COOL-DGI procedure:

1) Find all maximal cliques from the original graph by passing it through the attention function: $A' = \mathcal{T}(A)$.
2) Normalize the adjacency matrix, $\tilde{A} = \alpha \times A + \beta \times A'$, by combining the unweighted and weighted adjacency matrices balanced by hyperparameters, $\{\alpha, \beta\} \in \{0, 1\}$.
3) Sample a negative example by using the corruption function: $\left(\widetilde{X}, \widetilde{A}\right) \sim \mathcal{C}\left(X, \widetilde{A}\right)$.
4) Obtain high-level representations, $\vec{h_i}$, for the maximal cliques of the input graph by passing it through the encoder: $H = \mathcal{E}(X, \widetilde{A}) = \{\vec{h_1}, \vec{h_2}, \ldots, \vec{h_N}\}$.
5) Obtain high-level representations, $\vec{\tilde{h}_i}$, for the maximal cliques of the negative graph by passing it through the encoder: $\widetilde{H} = \mathcal{E}(\widetilde{X}, \widetilde{A}) = \{\vec{\tilde{h}_1}, \vec{\tilde{h}_2}, \ldots, \vec{\tilde{h}_N}\}$.
6) Summarize high-level representations of the input graph by passing it through the readout function: $\vec{s} = \mathcal{R}(H)$.
7) Update parameters of $\mathcal{E}$, $\mathcal{R}$, and $\mathcal{D}$ by applying gradient descent to maximize Equation (5).

This algorithm is fully summarized in Figure 2.

V. Experiments

Datasets. A number of transductive node classification tasks are applied to examine COOL. The experiments are conducted on six standard network datasets including Cora, Citeseer and Pubmed [41] as well as ogbn-arxiv [42], Coauthor CS, and Amazon Photo [43]. Dataset statistics are summarized in Table I. We follow the experimental setup described in [19] to classify research papers into topics where nodes correspond to documents and edges represent citations. Each node has a label and a feature vector that corresponds to elements of a bag-of-words representation of a document.

Since the existing evaluation strategies for GNN models have some limitations [43], we used multiple data splits. For the Cora, CiteSeer, and PubMed datasetsthe data is first split into a train and a test set. For the train set, 1500 nodes were sampled and the test set contains all the remaining nodes. We used three different label sets in each experiment: A training set of 80 nodes per class, a validation set of 500 nodes, and a test set. For the ogbn-arxiv dataset, we trained on papers published until 2017, validated on those published in 2018, and tested on those published since 2019. For the Coauthor CS and Amazon Photo datasets, we used 30 labeled nodes per class as the training set, 30 nodes per class as the validation set, and the rest as the test set. Each experiment is run with 5 random initializations on each data split, leading to a total of 100 runs per experiment.

TABLE I: Summary of the datasets used in our experiments.

Dataset	Nodes	Edges	Features	Classes	Label Rate
Cora	2,708	5,429	1,433	7	0.052
Citeseer	3,327	4,732	3,703	6	0.036
Pubmed	19,717	44,338	500	3	0.003
ogbn-arxiv	169,343	1,166,243	128	40	0.625
Coauthor CS	18,333	81,894	6,805	15	0.0164
Amazon Photo	7,487	119,043	745	8	0.0214

Baselines. We compare COOL against GCN [19], GAT [1], DGI [29], SGC [28], AS-GCN [35], GraphSAGE [34] and APPNP [22].

Settings. Our model is initialized using Glorot initialization [44] and trained for a maximum of 200 epochs using Adam SGD optimizer [45], with an initial learning rate of 0.01. Training is terminated if validation accuracy does not improve for 10 consecutive steps; as a result, most runs finish in less than 200 steps. It is applied a fixed dropout [46] rate of 0.5 to input and hidden layer and an $\mathcal{L}_2$ regularization of 0.0005

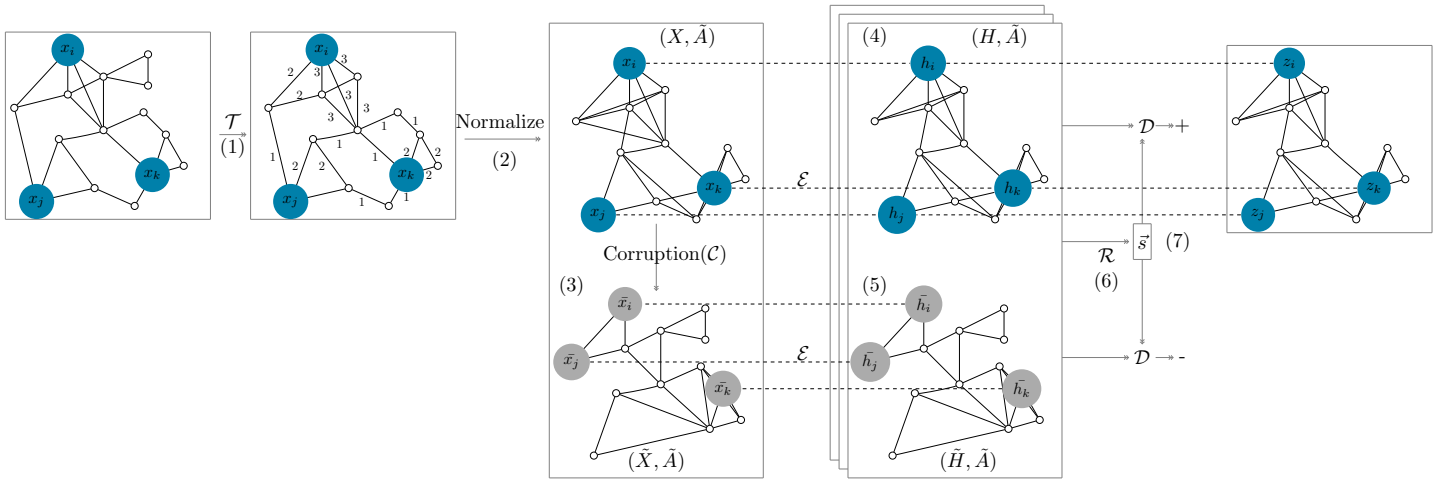


Fig. 2: A high-level overview of COOL-DGI method with unsupervised loss function: (1) find all maximal cliques (2) normalize the adjacency matrix (3) sample a negative example (4) obtain high-level representations for the positive graph (5) obtain high-level representations for the negative graph (6) summarize high-level representations and (7) update parameters.

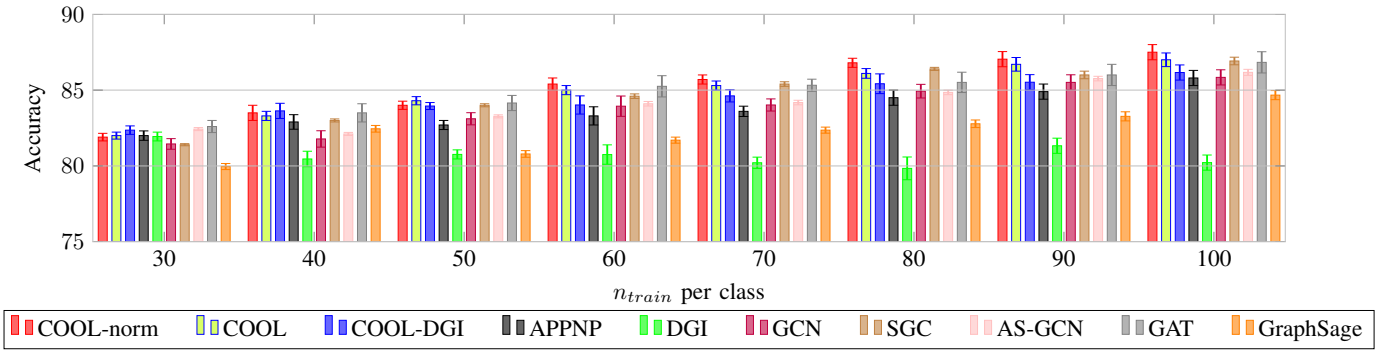


Fig. 3: Accuracy for different training set sizes (number of labeled nodes per class) on Cora.

on the weights are added. COOL-DGI model uses a one-layer GCN model as an encoder with the effective hidden size of 512 (especially 256 on Pubmed due to memory limitation) and the parametric ReLU (PReLU) [47] nonlinearity[1]. For COOL-norm, we follow intuitions from [48] to normalize the adjacency matrix. Following experimental settings proposed in [48], we use $T = 5$, $\nu = 10^{-4}$.

Results. Tables II and III demonstrate how our model performs on multiple data splits and initializations. The best results are highlighted in bold. We report the mean classification accuracy (with standard deviation) on the test nodes after 100 runs of training, and reuse the metrics already reported in [19] for the performance of GCN, as well as GAT [1], DGI [29], SGC [28], AS-GCN [35], GraphSAGE [34], and APPNP [22].

We achieve a test accuracy of 86.1% (85.4%), 74.1% (73.9%) and 79% (82.9%) with COOL (COOL-DGI) on Cora, Citeseer, and Pubmed datasets, respectively. The results obtained by COOL outperform all the competing baselines. We particularly note that the COOL-DGI approach with unsupervised loss exceeding the results reported for the GCN with the supervised loss on all datasets. We assume that these benefits stem from the fact that the COOL-DGI approach aggregates high-level localized information from the most relevant parts of the graph and allows for every node to have access to structural properties of the entire graph while the supervised GCN is limited to only two-layer neighborhoods. Although COOL and COOL-DGI demonstrate strong performance being achieved across all three datasets, COOL-norm considerably outperforms all the competing approaches with a test accuracy of 86.8%, 74.8%, and 86.3% on Cora, Citeseer, and Pubmed as well as 73.8%, 93.3%, and 89.3% on ogbn-arxiv, Coauthor CS, and Amazon Photo, verifying the potential of normalizing the adjacency matrix in improving generalization of the model.

Since the labeling rate is often very small for real-world datasets, we analyzed the performance of the models with a small number of training samples. Figures 3 - 5 illustrate how the number of training nodes per class impacts the accuracy on Cora, Citeseer, and Pubmed. Our results demonstrate strong performance achieved in this sparsely labeled setting across all six datasets, verifying the potential of the proposed clique-based method in the transductive node classification domain. Figure 6 shows the learning curves on the Cora dataset. We can

[1]A reference COOL implementation can be found at https://github.com/SoheilaMolaei/COOLnorm.

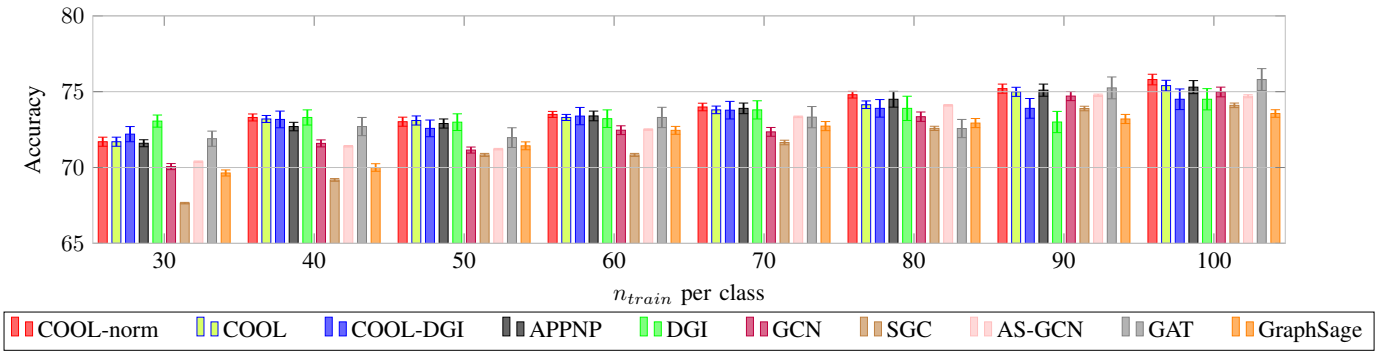


Fig. 4: Accuracy for different training set sizes (the number of labeled nodes per class) on CiteSeer.

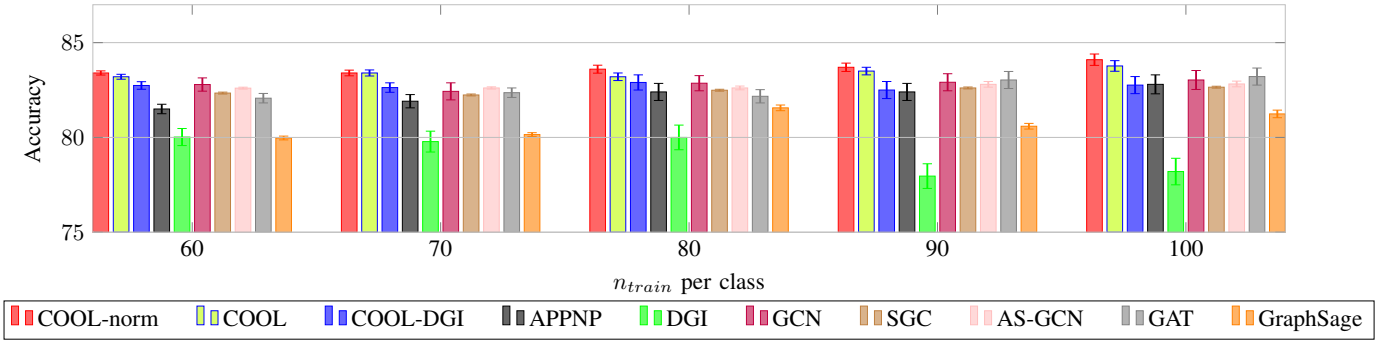


Fig. 5: Accuracy for different training set sizes (number of labeled nodes per class) on PubMed.

TABLE II: Results of node classification on the Cora, Citeseer, and Pubmed datasets. In the first column, we highlight the kind of data available to each method during training (**X**: features, **A**: adjacency matrix, **Y**: labels).

Available data	**Method**	**Cora**	**Citeseer**	**Pubmed**
X, A, Y	GCN [19]	84.93 ± 0.45 %	73.35 ± 0.31 %	82.86 ± 0.4%
X, A, Y	GAT [1]	85.51 ± 0.67%	72.57 ± 0.6 %	82.17 ± 0.35 %
X, A	DGI [29]	79.84 ± 0.75 %	73.9 ± 0.8 %	80 ± 0.65%
X, A, Y	SGC [28]	86.4 ± 0.1%	72.58 ± 0.14%	82.49 ± 0.05%
X, A, Y	AS-GCN [35]	84.85 ± 0.15 %	74.10 ± 0.05%	82.61 ± 0.1 %
X, A, Y	GraphSAGE [34]	82.78 ± 0.25%	72.93 ± 0.3 %	81.56 ± 0.15 %
X, A, Y	APPNP [22]	84.5 ± 0.5%	74.5 ± 0.52%	82.4 ± 0.45%
X, A	**COOL-DGI**	85.42 ± 0.65%	73.9 ± 0.58 %	82.9 ± 0.4 %
X, A, Y	**COOL**	86.1 ± 0.32%	74.14 ± 0.25%	83.2 ± 0.2%
X, A, Y	**COOL-norm**	**86.8 $\pm$ 0.3%**	**74.8 $\pm$ 0.22%**	**83.6 $\pm$ 0.21%**

TABLE III: Results of node classification accuracies on the ogbn-arxiv, Coauthor CS, and Amazon Photo datasets.

Model	**ogbn-arxiv**	**Coauthor CS**	**Amazon Photo**
GCN	0.7174 ± 0.0029%	91.8 ± 0.1%	87.3 ± 1.0%
APPNP	72.28%	91.69 ± 0.4%	**89.38 $\pm$ 1.2%**
GraphSage	71.49 ± 0.25%	0.9045%	87.4 ± 1.1%
COOL-norm	**73.8%**	**93.3%**	89.3 ± 1.0%

observe that the proposed approach presents higher training and testing scores during learning.

In order to demonstrate the effectiveness of our approach with respect to graph signal denoising, we designed two types of noise cases in terms of structural and attribute noises. Attribute noise adds noise to node features while structural noise randomly removes or adds a small portion of the edges in the original graph. In the attribute noise case, we added Gaussian noise with noise levels of 0.001, 0.005, and 0.01 to the node features. As Figure 7 shows, COOL-norm outperforms other baselines on Cora and Citeseer datasets, demonstrating robust performance across node classification benchmarks.

In a structural noise setting, we randomly add or remove a small portion of the edges with noise ratios of 0.05, 0.1, and 0.2. Figure 8 illustrates robust COOL-norm performance against our strong baselines, demonstrating our approach extracts highly useful features that reduce the negative impact of the noise in structural information.

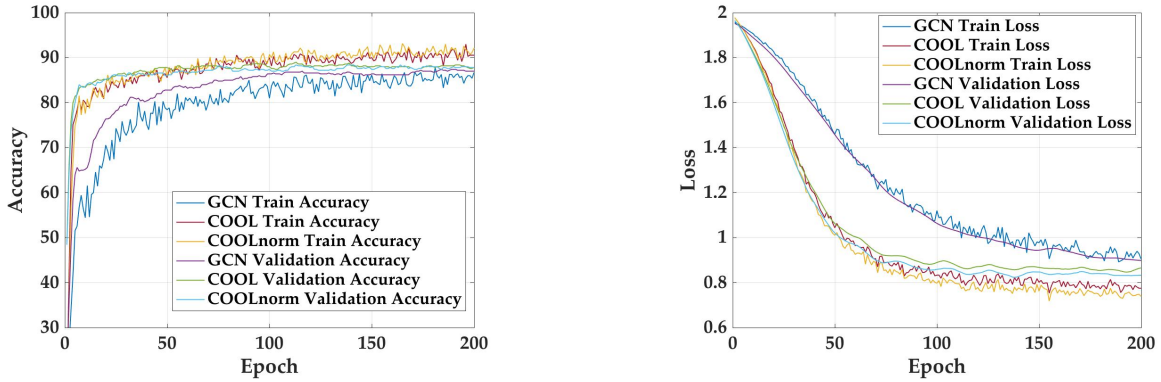
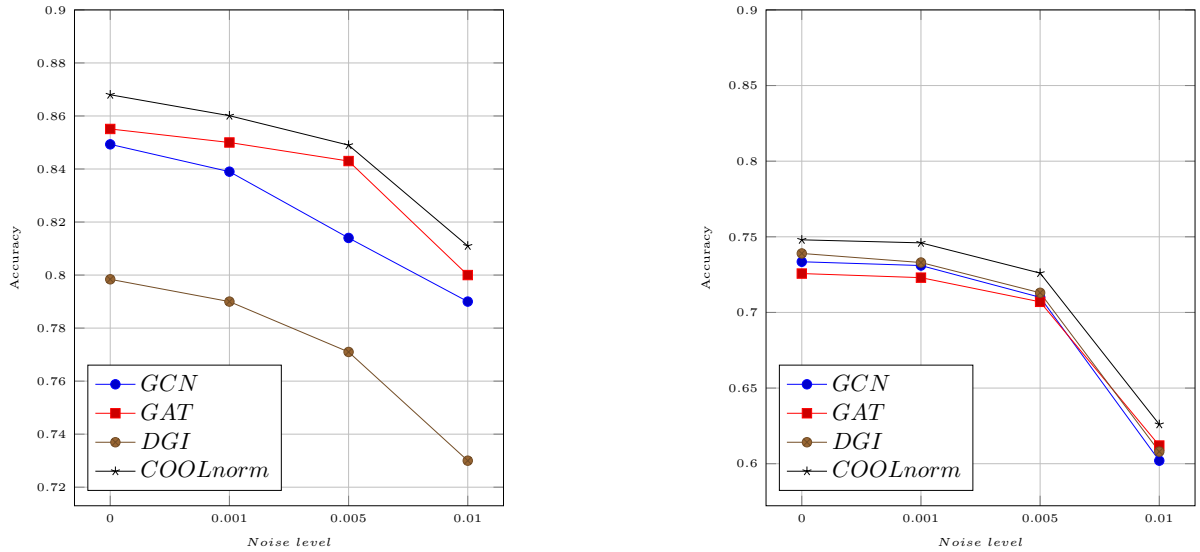


Fig. 6: Accuracy and loss during training of GCN, COOL, and COOL-norm averaged over 100 runs on the Cora dataset.



(a) Cora (b) Citeseer

Fig. 7: Node classification accuracy of models in attribute noise case.

VI. Qualitative Analysis

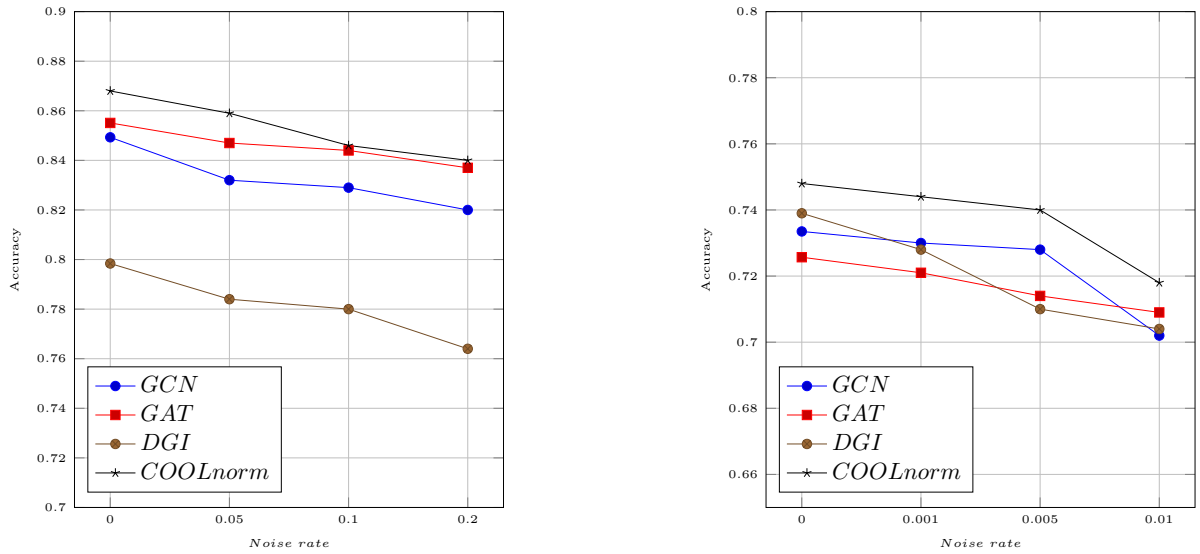
To better understand thoroughly the effectiveness of COOL, we provide a standard set of t-SNE plots [49] of the representations learnt by the COOL algorithm on the Cora dataset. Our analysis is performed by using GraphTSNE [50] with the weight of the graph clustering loss $\alpha = 0.4$ as shown in Figure 9. Colors denote document class. As expected given the quantitative results, the learnt representations exhibit discernible clustering in the projected 2D space (especially compared to the raw features), which respects the seven topic classes of Cora.

VII. Conclusions

We introduced a new approach for learning semi-supervised representations on graph-structured data, leveraging maximal cliques. Our model allows for (implicitly) assigning various importance weights to different nodes within a neighborhood, enabling a leap in model capacity. This enables state-of-the-art results across six well-established node classification benchmarks.

References

[1] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017).
[2] W. L. Hamilton, R. Ying, J. Leskovec, Representation learning on graphs: Methods and applications, arXiv preprint arXiv:1709.05584 (2017).
[3] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, Neural message passing for quantum chemistry, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1263–1272.
[4] R. Liao, Z. Zhao, R. Urtasun, R. S. Zemel, Lanczosnet: Multi-scale deep graph convolutional networks, arXiv preprint arXiv:1901.01484 (2019).
[5] K. Sun, Z. Lin, H. Guo, Z. Zhu, Virtual adversarial training on graph convolutional networks in node classification, in: Chinese Conference on Pattern Recognition and Computer Vision (PRCV), Springer, 2019, pp. 431–443.
[6] K. Sun, Z. Lin, Z. Zhu, Multi-stage self-supervised learning for graph convolutional networks on graphs with few labels, arXiv preprint arXiv:1902.11038 (2019).



(a) Cora (b) Citeseer

Fig. 8: Node classification accuracy of models in structural noise case.

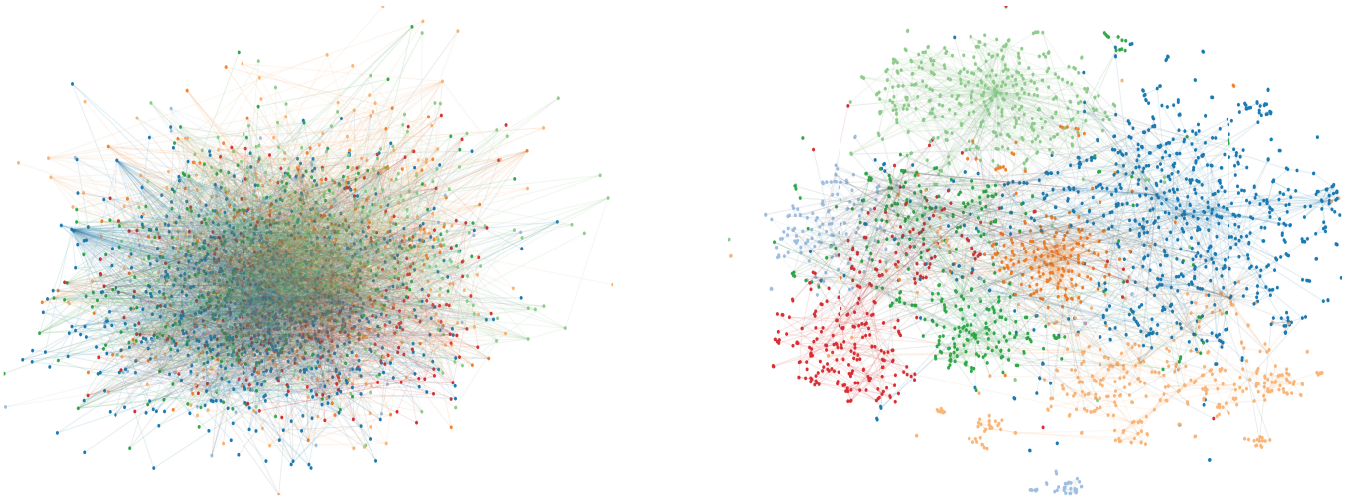


Fig. 9: t-SNE embeddings of the nodes in the Cora dataset from the raw features (**left**) and features from a learned COOL model (**right**). The clusters of the learned COOL model's representations are clearly defined, with the weight of the graph clustering loss $\alpha = 0.4$.

[7] S. Abu-El-Haija, B. Perozzi, A. Kapoor, H. Harutyunyan, N. Alipourfard, K. Lerman, G. V. Steeg, A. Galstyan, Mixhop: Higher-order graph convolution architectures via sparsified neighborhood mixing, arXiv preprint arXiv:1905.00067 (2019).

[8] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: going beyond euclidean data, IEEE Signal Processing Magazine 34 (4) (2017) 18–42.

[9] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, R. Pascanu, Relational inductive biases, deep learning, and graph networks (2018). arXiv:1806.01261.

[10] W. A. S. Wijesinghe, Q. Wang, Dfnets: Spectral cnns for graphs with feedback-looped filters, in: Advances in Neural Information Processing Systems, 2019, pp. 6007–6018.

[11] M. Huang, F. Zhuang, X. Zhang, X. Ao, Z. Niu, M.-L. Zhang, Q. He, Supervised representation learning for multi-label classification, Machine Learning 108 (5) (2019) 747–763.

[12] S. Molaei, H. Zare, H. Veisi, Deep learning approach on information diffusion in heterogeneous networks, Knowledge-Based Systems 189 (2020) 105153.

[13] L. Franceschi, M. Niepert, M. Pontil, X. He, Learning discrete structures for graph neural networks, arXiv preprint arXiv:1903.11960 (2019).

[14] D. Rao, F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, R. Hadsell, Continual unsupervised representation learning, in: Advances in Neural Information Processing Systems, 2019, pp. 7645–7655.

[15] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P. S. Yu, A Comprehensive Survey on Graph Neural Networks, arXiv:1901.00596 [cs, stat] (Jan. 2019). arXiv:1901.00596.

[16] S. Pan, J. Wu, X. Zhu, C. Zhang, P. S. Yu, Joint Structure Feature Exploration and Regularization for Multi-Task Graph Classification, IEEE Transactions on Knowledge and Data Engineering 28 (3) (2016) 715–728. doi:10.1109/TKDE.2015.2492567.

[17] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, J. Leskovec, Hierarchical Graph Representation Learning with Differentiable Pooling, in: Proceedings of the 32Nd International Conference on Neural Information Processing Systems, NIPS'18, Curran Associates Inc., USA, 2018, pp. 4805–4815.

[18] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, C. Zhang, Adversarially regularized graph autoencoder for graph embedding, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden., 2018, pp. 2609–2615. doi:10.24963/ijcai.2018/362.
URL https://doi.org/10.24963/ijcai.2018/362

[19] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[20] Z. Yang, W. W. Cohen, R. Salakhutdinov, Revisiting semi-supervised learning with graph embeddings, arXiv preprint arXiv:1603.08861 (2016).

[21] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in neural information processing systems, 2016, pp. 3844–3852.

[22] J. Klicpera, A. Bojchevski, S. Günnemann, Predict then propagate: Graph neural networks meet personalized pagerank, arXiv preprint arXiv:1810.05997 (2018).

[23] L. F. Ribeiro, P. H. Saverese, D. R. Figueiredo, struc2vec: Learning node representations from structural identity, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 385–394.

[24] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, S. Jegelka, Representation learning on graphs with jumping knowledge networks, arXiv preprint arXiv:1806.03536 (2018).

[25] M. Fey, Just jump: Dynamic neighborhood aggregation in graph neural networks, arXiv preprint arXiv:1904.04849 (2019).

[26] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[27] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, arXiv preprint arXiv:1312.6203 (2013).

[28] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, K. Q. Weinberger, Simplifying graph convolutional networks, arXiv preprint arXiv:1902.07153 (2019).

[29] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, R. D. Hjelm, Deep graph infomax, arXiv preprint arXiv:1809.10341 (2018).

[30] F.-Y. Sun, M. Qu, J. Hoffmann, C.-W. Huang, J. Tang, vgraph: A generative model for joint community detection and node representation learning, arXiv preprint arXiv:1906.07159 (2019).

[31] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 701–710.

[32] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[33] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2016, pp. 855–864.

[34] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in neural information processing systems, 2017, pp. 1024–1034.

[35] W. Huang, T. Zhang, Y. Rong, J. Huang, Adaptive sampling towards fast graph representation learning, in: Advances in Neural Information Processing Systems, 2018, pp. 4558–4567.

[36] A. Douik, H. Dahrouj, T. Y. Al-Naffouri, M.-S. Alouini, A tutorial on clique problems in communications and signal processing, arXiv preprint arXiv:1808.07102 (2018).

[37] C. Bron, J. Kerbosch, Algorithm 457: finding all cliques of an undirected graph, Communications of the ACM 16 (9) (1973) 575–577.

[38] D. Eppstein, M. Löffler, D. Strash, Listing all maximal cliques in large sparse real-world graphs, Journal of Experimental Algorithmics (JEA) 18 (2013) 3–1.

[39] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, A. Trischler, Y. Bengio, Learning deep representations by mutual information estimation and maximization, arXiv preprint arXiv:1808.06670 (2018).

[40] D. Eppstein, D. Strash, Listing all maximal cliques in large sparse real-world graphs, in: International Symposium on Experimental Algorithms, Springer, 2011, pp. 364–375.

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

[42] M. Z. Y. D. H. R. B. L. M. C. J. L. Weihua Hu, Matthias Fey, Open graph benchmark: Datasets for machine learning on graphs, arXiv preprint arXiv:2005.00687 (2020).

[43] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, Pitfalls of graph neural network evaluation, arXiv preprint arXiv:1811.05868 (2018).

[44] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010, pp. 249–256.

[45] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

[47] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.

[48] K. Sun, P. Koniusz, J. Wang, Fisher-bures adversary graph convolutional networks, arXiv preprint arXiv:1903.04154 (2019).

[49] L. v. d. Maaten, G. Hinton, Visualizing data using t-sne, Journal of machine learning research 9 (Nov) (2008) 2579–2605.

[50] Y. Y. Leow, T. Laurent, X. Bresson, Graphtsne: A visualization technique for graph-structured data, in: ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.