Learning Graph Representations With Maximal Cliques

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

***Abstract*— Non-Euclidean property of graph structures has faced interesting challenges when deep learning methods are applied. Graph convolutional networks (GCNs) 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 semisupervised learning setting. Our proposed method, COnvOlving cLiques (COOL), is constructed as a neighborhood aggregation approach for learning node representations using established GCN 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 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 (DL), graph convolutional networks (GCNs), graph neural networks (GNNs), graph representation learning, 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 machine learning models named 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 the aggregation of useful neighborhood information to succeed. Better representation learning provides more generalization power on graph-structured inputs. To learn meaningful representations, semisupervised 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], semisupervised [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].

Semisupervised GNN approaches involve 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 are mostly being unlabeled. Therefore, semisupervised 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 graph convolutional network (GCN) [19], Planetoid [20], Chebyshev [21] and personalized propagation of neural predictions (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 brief, 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 COnvOlving cLiques (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.

1) We propose COOL, a new neighborhood aggregation method for learning node representations.
2) We propose a novel clique-based edge-weight computation that is general and architecturally simple.
3) The proposed approach learns meaningful representations of graph structures by obtaining maximal cliques from neighbors at various distances.
4) We establish two scenarios to showcase the strong denoising capabilities of our model in a range of node classification tasks.

Our brief 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, GCN and node representation learning.

A. Neighborhood Aggregation

Some works have been considered neighborhood aggregation for representation learning on graphs, including [24], [25]. A flexible

Manuscript received November 28, 2019; revised April 18, 2020, September 12, 2020, February 7, 2021, and June 21, 2021; accepted August 11, 2021. The work of Mahdi Jalili was supported by Australian Research Council under Project DP200101199 and Project LP180101309. *(Soheila Molaei and Nima Ghanbari Bousejin contributed equally to this work.) (Corresponding author: Hadi Zare.)*

Soheila Molaei, Nima Ghanbari Bousejin, and Hadi Zare are with the Faculty of New Sciences and Technologies, University of Tehran, Tehran 1417935840, Iran (e-mail: h.zare@ut.ac.ir).

Mahdi Jalili is with the School of Engineering, RMIT University, Melbourne, VIC 3000, Australia.

Shirui Pan is with the Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3104901.

Digital Object Identifier 10.1109/TNNLS.2021.3104901

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. Graph attention network (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.

B. Graph Convolutional Network

Motivated by the success of convolutional neural network (CNN) [26], Spectral GCN [27] defined the convolution operation in the Fourier domain by computing the eigendecomposition of the graph Laplacian, resulting in potentially intense computations and nonspatially 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. Deep graph infomax (DGI) [29] relies on maximizing mutual information between a high-level global representation and local parts of the input in an unsupervised manner.

C. 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], large-scale information network embedding (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 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 GCNs which generate node representations by repeated aggregation over local node neighborhoods.

D. Message Passing

Some works extended GCNs to send and aggregate information through higher-order paths [19]. APPNP [22] proposed sampling nodes in the lower layers conditioned on their top layer. Actional-structural GCN (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

A. 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.

B. Notation

We use standard graph representations in semisupervised node classification tasks. Graph G with N nodes is represented as a pair (X, A), where $X \in \mathbb{R}^{\mathbb{N}\times\mathbb{F}}$ denotes the node feature matrix with F features per node, and $A \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$ 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.

C. COOL

Our approach for learning meaningful representations is to obtain maximal cliques from neighbors at various distances. We employ an attention function, $\mathcal{T} : \mathbb{R}^{\mathbb{N}\times\mathbb{N}} \rightarrow \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$, 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 Fig. 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, $\mathcal{E} : \mathbb{R}^{\mathbb{N}\times\mathbb{N}} \times \mathbb{R}^{\mathbb{N}\times\mathbb{F}} \rightarrow \mathbb{R}^{\mathbb{N}\times\mathbb{F}'}$, such that $\mathcal{E}(X, \tilde{A}) = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}$ represents high-level representations $\vec{h}_i \in \mathbb{R}^{\mathbb{F}'}$ for each node i. Our $\mathcal{E}$ function is a two-layer GCN model [19] with the following layer-wise propagation rule:

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

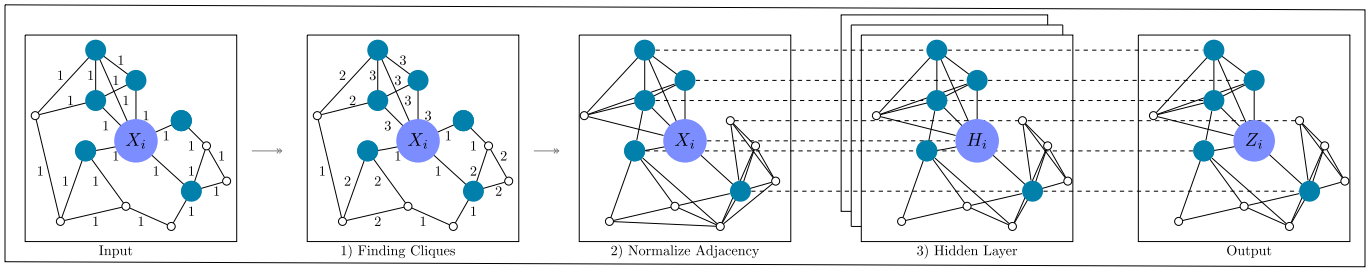


Fig. 1. High-level overview of COOL-norm method with semisupervised loss function: 1) find all maximal cliques; 2) normalize the adjacency matrix; 3) obtain high-level representations for normalized graph; and 4) update parameters.

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 rectified linear unit (ReLU) function. $H^{(l)} \in \mathbb{R}^{\mathbb{N}\times\mathbb{F}}$ is the matrix of activations in the lth layer with $H^{(0)} = X$ and $W^{(l)} \in \mathbb{R}^{\mathbb{F}\times\mathbb{F}'}$ 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 (2).

Algorithm 1 Normalize Adjacency Matrix to Capture High-Order Proximities. One 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 $\upsilon > 0$ Is a Threshold

Input : $\tilde{A}$, T, υ
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
$\tilde{A} \leftarrow \frac{1}{T} S \circ (1_{n*n} - I)$
$\tilde{A} \leftarrow \tilde{A} \circ (\tilde{A} > \upsilon)$
$\tilde{A} \leftarrow \tilde{A} + \tilde{A}^T + 2 \times I$
$\tilde{A} \leftarrow diag^{-1/2}\left(\tilde{A}\vec{1}\right)\ \tilde{A}\ diag^{-1/2}\left(\tilde{A}\vec{1}\right)$
return $\tilde{A}$

D. COOL-DGI

The existing methods often use semisupervised learning with GCNs [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. 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 the mutual information maximization principle and show that the learned embeddings can encode valuable information for node classification tasks.

The unsupervised training setup is equivalent to the one in DGI. After finding all maximal cliques using the attention function, $\mathcal{T} : \mathbb{R}^{\mathbb{N}\times\mathbb{N}} \to \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$, an encoder, $\mathcal{E} : \mathbb{R}^{\mathbb{N}\times\mathbb{N}} \times \mathbb{R}^{\mathbb{N}\times\mathbb{F}} \to \mathbb{R}^{\mathbb{N}\times\mathbb{F}'}$, computes high-level representation $\vec{h}_i$ for node i, using node features, X, and the normalized adjacency matrix, $\tilde{A}$. The discriminator, $\mathcal{D} : \mathbb{R}^{\mathbb{F}\times\mathbb{F}} \to \mathbb{R}$, 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 $\mathcal{R} : \mathbb{R}^{\mathbb{N}\times\mathbb{F}} \to \mathbb{R}^{\mathbb{F}}$ 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 $\mathcal{C} : \mathbb{R}^{\mathbb{N}\times\mathbb{N}} \times \mathbb{R}^{\mathbb{N}\times\mathbb{F}} \to \mathbb{R}^{\mathbb{M}\times\mathbb{M}} \times \mathbb{R}^{\mathbb{M}\times\mathbb{F}}$. Both the encoder and discriminator are jointly trained to distinguish between positive and negative samples by maximizing the following standard binary cross-entropy loss:

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

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 is $O(N^2)$; thus, the overall time complexity is $O(N^2)$ in the worst case. The space complexity of the proposed approach is $O(LNF + LF^2)$ where L, N, and F are the number of layers, nodes, and features, respectively.

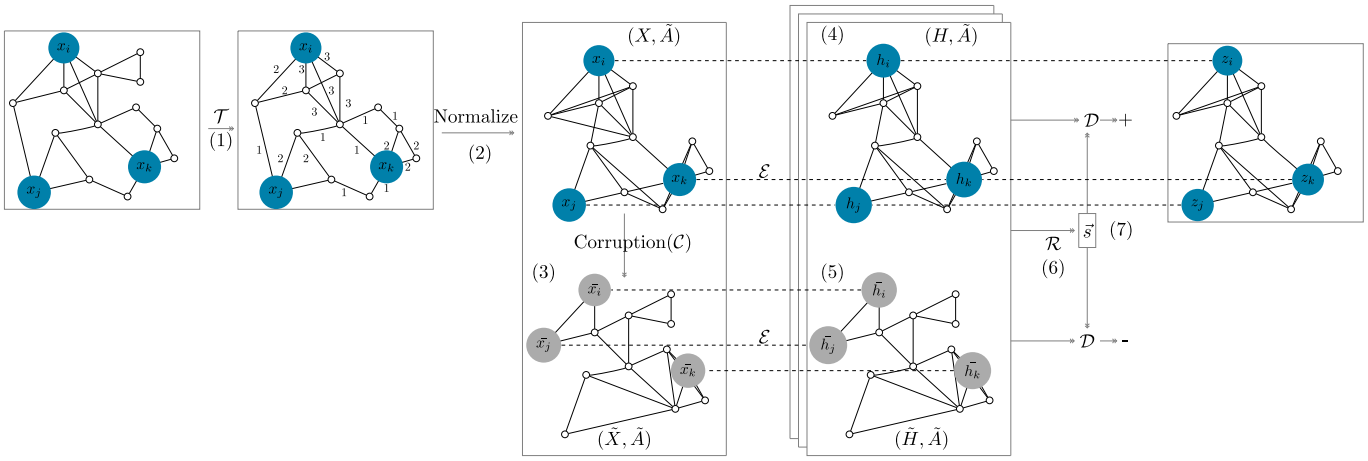


Fig. 2. High-level overview of COOL-DGI method with unsupervised loss function: 1) find all maximal cliques; 2) normalize the adjacency matrix; 3) sample a negative example; 4) obtain high-level representations for the positive graph; 5) obtain high-level representations for the negative graph; 6) summarize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

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: $(\widetilde{X}, \widetilde{A}) \sim \mathcal{C}(X, \widetilde{A})$.
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, $\vec{\widetilde{h}}_i$, for the maximal cliques of the negative graph by passing it through the encoder: $\widetilde{H} = \mathcal{E}(\widetilde{X}, \widetilde{A}) = \{\vec{\widetilde{h}}_1, \vec{\widetilde{h}}_2, \ldots, \vec{\widetilde{h}}_N\}$.
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 (5).

This algorithm is fully summarized in Fig. 2.

V. Experiments

A. 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 briefs 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 are 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 briefs 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 five 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

B. Baselines

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

C. Settings

Our model is initialized using Glorot initialization [44] and trained for a maximum of 200 epochs using Adam stochastic gradient descent (SGD) optimizer [45], with an initial learning rate of 0.01. Training is terminated if validation accuracy does not improve for ten 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 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 $T = 5$, $\nu = 10^{-4}$.

D. 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

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

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%**

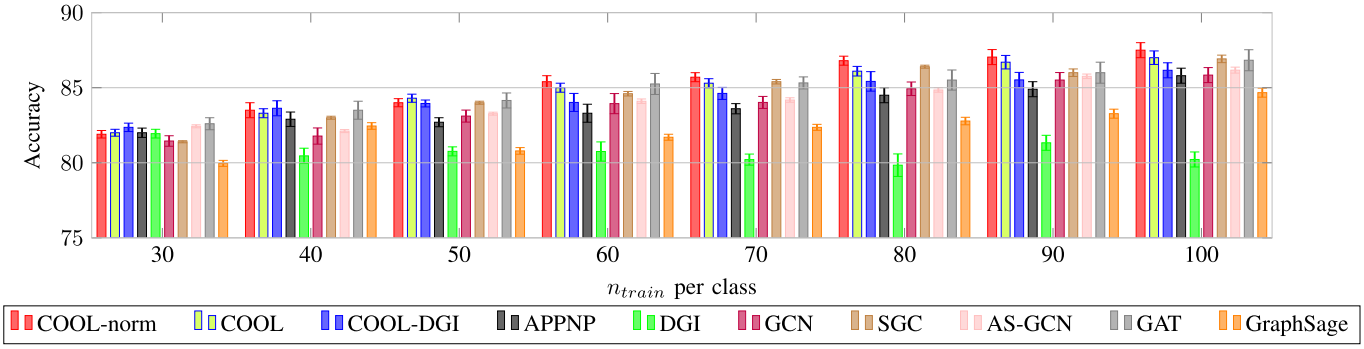


Fig. 3. Accuracy for different training set sizes (number of labeled nodes per class) on Cora.

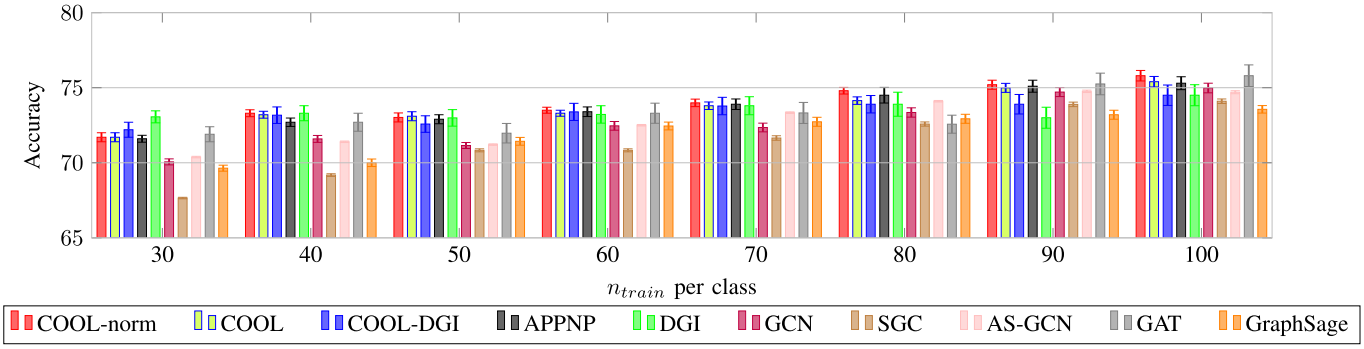


Fig. 4. Accuracy for different training set sizes (the number of labeled nodes per class) on CiteSeer.

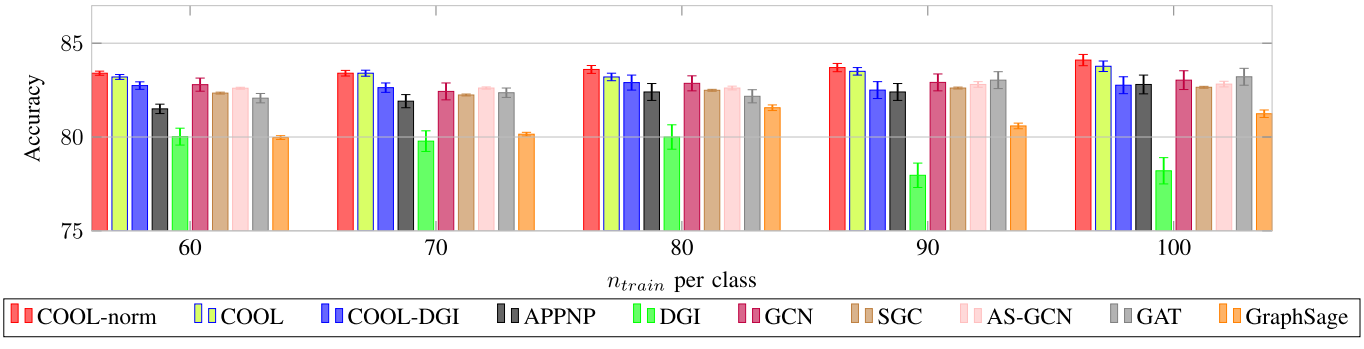


Fig. 5. Accuracy for different training set sizes (number of labeled nodes per class) on PubMed.

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

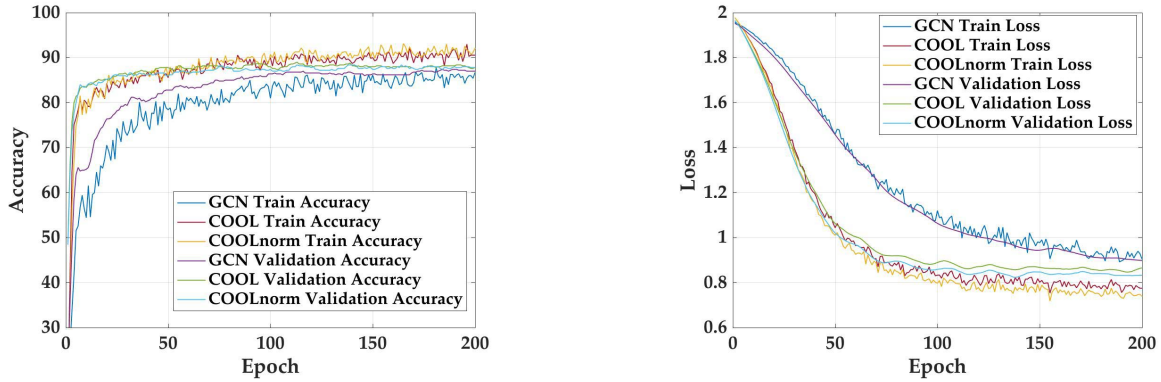
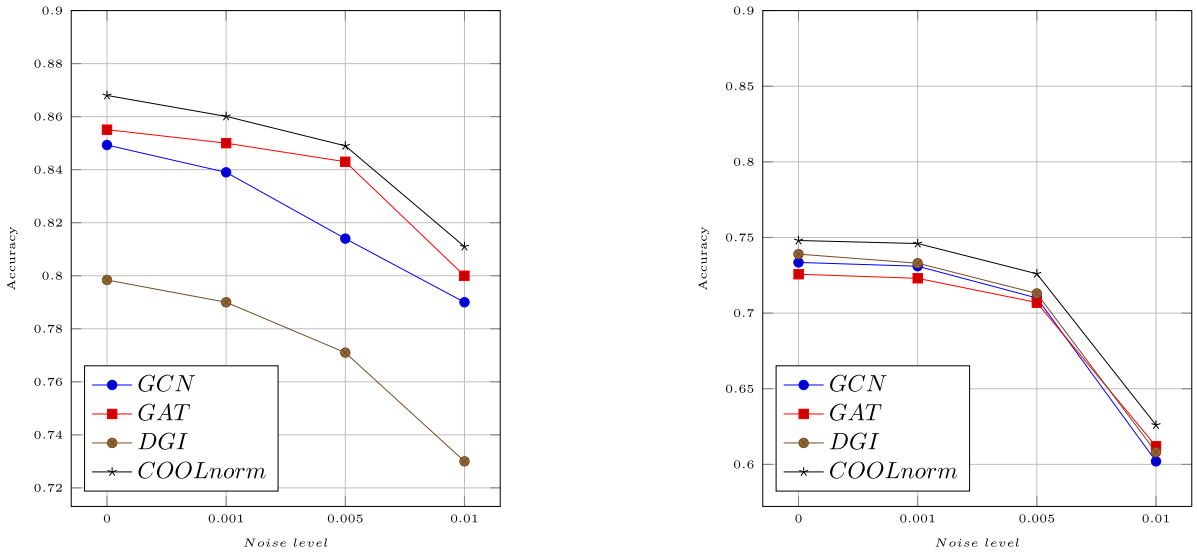


Fig. 6. Accuracy and loss during training of GCN, COOL, and COOL-norm averaged over 100 runs on the Cora dataset.



(a) (b)

Fig. 7. Node classification accuracy of models in attribute noise case. (a) Cora. (b) Citeseer.

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. Figs. 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. Fig. 6 shows the learning curves on the Cora dataset. We can observe that the proposed approach presents higher training and testing scores during learning.

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 ± 1.2%**
GraphSage	71.49 ± 0.25%	0.9045%	87.4 ± 1.1%
COOL-norm	**73.8%**	**93.3%**	89.3 ± 1.0%

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 Fig. 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.

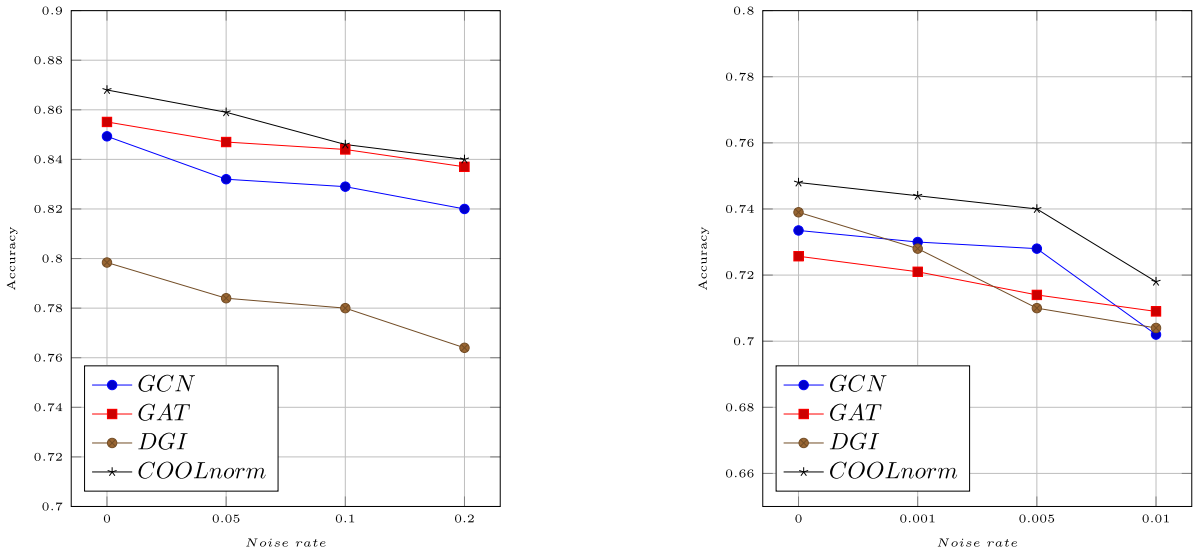


Fig. 8. Node classification accuracy of models in structural noise case. (a) Cora. (b) Citeseer.

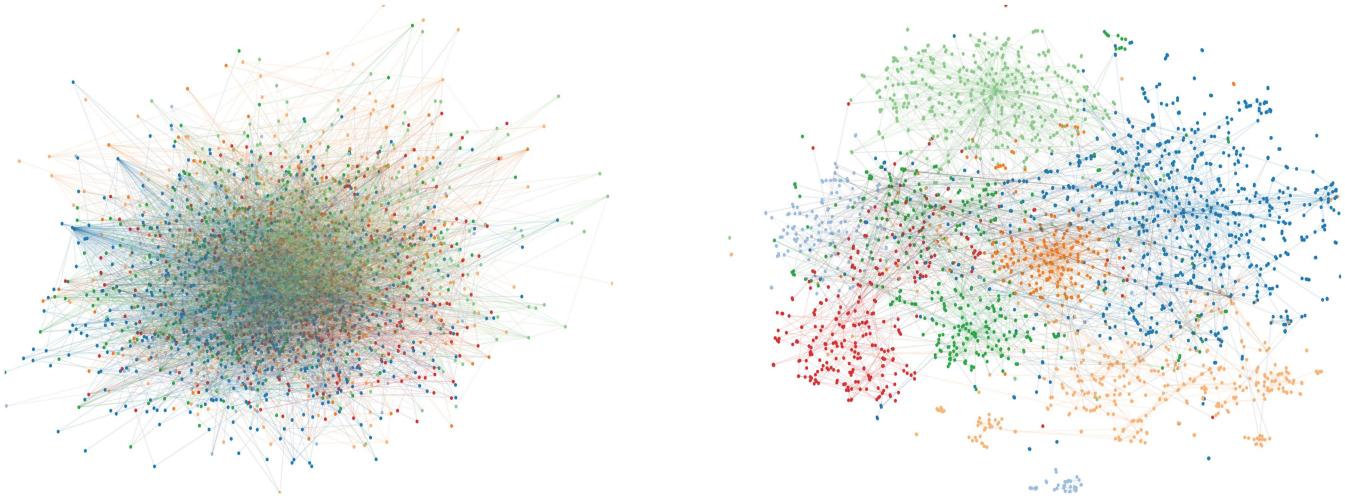


Fig. 9. (**Left**) t-SNE embeddings of the nodes in the Cora dataset from the raw features and (**Right**) features from a learned COOL model. The clusters of the learned COOL model's representations are clearly defined, with the weight of the graph clustering loss $\alpha = 0.4$.

Fig. 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.

VI. Qualitative Analysis

To better understand thoroughly the effectiveness of COOL, we provide a standard set of t-distributed stochastic neighbor embedding (t-SNE) plots [49] of the representations learned 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 Fig. 9. Colors denote document class. As expected given the quantitative results, the learned representations exhibit discernible clustering in the projected 2-D space (especially compared to the raw features), which respects the seven topic classes of Cora.

VII. Conclusion

We introduced a new approach for learning semisupervised 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. Liò, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*. [Online]. Available: http://arxiv.org/abs/1710.10903

[2] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," 2017, *arXiv:1709.05584*. [Online]. Available: http://arxiv.org/abs/1709.05584

[3] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1263–1272.

[4] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel, "LanczosNet: Multi-scale deep graph convolutional networks," 2019, *arXiv:1901.01484*. [Online]. Available: http://arxiv.org/abs/1901.01484

[5] K. Sun, Z. Lin, H. Guo, and Z. Zhu, "Virtual adversarial training on graph convolutional networks in node classification," in *Proc. 2nd Chin. Conf., PRCV*. Xi'an, China: Springer, Nov. 2019, pp. 431–443.

[6] K. Sun, Z. Lin, and Z. Zhu, "Multi-stage self-supervised learning for graph convolutional networks on graphs with few labels," 2019, *arXiv:1902.11038*. [Online]. Available: http://arxiv.org/abs/1902.11038

[7] S. Abu-El-Haija *et al.*, "MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," 2019, *arXiv:1905.00067*. [Online]. Available: http://arxiv.org/abs/1905.00067

[8] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[9] P. W. Battaglia *et al.*, "Relational inductive biases, deep learning, and graph networks," 2018, *arXiv:1806.01261*. [Online]. Available: http://arxiv.org/abs/1806.01261

[10] A. Wijesinghe and Q. Wang, "DFNets: Spectral CNNs for graphs with feedback-looped filters," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6007–6018.

[11] M. Huang *et al.*, "Supervised representation learning for multi-label classification," *Mach. Learn.*, vol. 108, no. 5, pp. 747–763, May 2019.

[12] S. Molaei, H. Zare, and H. Veisi, "Deep learning approach on information diffusion in heterogeneous networks," *Knowl.-Based Syst.*, vol. 189, Feb. 2020, Art. no. 105153.

[13] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," 2019, *arXiv:1903.11960*. [Online]. Available: http://arxiv.org/abs/1903.11960

[14] D. Rao, F. Visin, A. A. Rusu, R. Pascanu, Y. W. Teh, and R. Hadsell, "Continual unsupervised representation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7645–7655.

[15] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," Jan. 2019, *arXiv:1901.00596*. [Online]. Available: http://arxiv.org/abs/1901.00596

[16] S. Pan, J. Wu, X. Zhu, C. Zhang, and P. S. Yu, "Joint structure feature exploration and regularization for multi-task graph classification," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 3, pp. 715–728, Mar. 2016.

[17] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Red Hook, NY, USA: Curran Associates, 2018, pp. 4805–4815.

[18] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Proc. 27th Int. Joint Conf. Artif. Intell. (IJCAI)*, Stockholm, Sweden, Jul. 2018, pp. 2609–2615, doi: 10.24963/ijcai.2018/362.

[19] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*. [Online]. Available: http://arxiv.org/abs/1609.02907

[20] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," 2016, *arXiv:1603.08861*. [Online]. Available: http://arxiv.org/abs/1603.08861

[21] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[22] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," 2018, *arXiv:1810.05997*. [Online]. Available: http://arxiv.org/abs/1810.05997

[23] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 385–394.

[24] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-I. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," 2018, *arXiv:1806.03536*. [Online]. Available: http://arxiv.org/abs/1806.03536

[25] M. Fey, "Just jump: Dynamic neighborhood aggregation in graph neural networks," 2019, *arXiv:1904.04849*. [Online]. Available: http://arxiv.org/abs/1904.04849

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[27] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*. [Online]. Available: http://arxiv.org/abs/1312.6203

[28] F. Wu, T. Zhang, A. H. de Souza, Jr., C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," 2019, *arXiv:1902.07153*. [Online]. Available: http://arxiv.org/abs/1902.07153

[29] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," 2018, *arXiv:1809.10341*. [Online]. Available: http://arxiv.org/abs/1809.10341

[30] F.-Y. Sun, M. Qu, J. Hoffmann, C.-W. Huang, and J. Tang, "vGraph: A generative model for joint community detection and node representation learning," 2019, *arXiv:1906.07159*. [Online]. Available: http://arxiv.org/abs/1906.07159

[31] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 701–710.

[32] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee, May 2015, pp. 1067–1077.

[33] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 855–864.

[34] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[35] W. Huang, T. Zhang, Y. Rong, and J. Huang, "Adaptive sampling towards fast graph representation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4558–4567.

[36] A. Douik, H. Dahrouj, T. Y. Al-Naffouri, and M.-S. Alouini, "A tutorial on clique problems in communications and signal processing," 2018, *arXiv:1808.07102*. [Online]. Available: http://arxiv.org/abs/1808.07102

[37] C. Bron and J. Kerbosch, "Algorithm 457: Finding all cliques of an undirected graph," *Commun. ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[38] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in large sparse real-world graphs," *ACM J. Exp. Algorithmics*, vol. 18, pp. 1–3, Dec. 2013.

[39] R. D. Hjelm *et al.*, "Learning deep representations by mutual information estimation and maximization," 2018, *arXiv:1808.06670*. [Online]. Available: http://arxiv.org/abs/1808.06670

[40] D. Eppstein and D. Strash, "Listing all maximal cliques in large sparse real-world graphs," in *Proc. 10th Int. Symp., SEA*. Kolimpari, Greece: Springer, May 2011, pp. 364–375.

[41] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, p. 93, 2008.

[42] W. Hu *et al.*, "Open graph benchmark: Datasets for machine learning on graphs," 2020, *arXiv:2005.00687*. [Online]. Available: http://arxiv.org/abs/2005.00687

[43] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," 2018, *arXiv:1811.05868*. [Online]. Available: http://arxiv.org/abs/1811.05868

[44] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1026–1034.

[48] K. Sun, P. Koniusz, and Z. Wang, "Fisher–Bures adversary graph convolutional networks," 2019, *arXiv:1903.04154*. [Online]. Available: http://arxiv.org/abs/1903.04154

[49] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[50] Y. Y. Leow, T. Laurent, and X. Bresson, "GraphTSNE: A visualization technique for graph-structured data," in *Proc. ICLR Workshop Represent. Learn. Graphs Manifolds*, 2019, pp. 1–7.