Topology and Content Co-Alignment Graph Convolutional Learning

Min Shi, *Student Member, IEEE*, Yufei Tang, *Member, IEEE*, and Xingquan Zhu, *Senior Member, IEEE*

***Abstract*— In traditional graph neural networks (GNNs), graph convolutional learning is carried out through topology-driven recursive node content aggregation for network representation learning. In reality, network topology and node content each provide unique and important information, and they are not always consistent because of noise, irrelevance, or missing links between nodes. A pure topology-driven feature aggregation approach between unaligned neighborhoods may deteriorate learning from nodes with poor structure-content consistency, due to the propagation of incorrect messages over the whole network. Alternatively, in this brief, we advocate a co-alignment graph convolutional learning (CoGL) paradigm, by aligning topology and content networks to maximize consistency. Our theme is to enforce the learning from the topology network to be consistent with the content network while simultaneously optimizing the content network to comply with the topology for optimized representation learning. Given a network, CoGL first reconstructs a content network from node features then co-aligns the content network and the original network through a unified optimization goal with: 1) minimized content loss; 2) minimized classification loss; and 3) minimized adversarial loss. Experiments on six benchmarks demonstrate that CoGL achieves comparable and even better performance compared with existing state-of-the-art GNN models.**

***Index Terms*— Graph convolutional learning, graph mining, network embedding, network representation learning, neural networks.**

I. Introduction

Recent years have witnessed a significant growth of graph neural networks (GNN) in domains involving data with dependence relationships, such as social network mining [1] and image recognition [2]. This is mainly attributed to GNN's efficient message-passing mechanism to encode network topology and node content/features in a unified latent space, through iterative feature aggregation of neighborhoods for each node [3], [4]. Because node content (or node features) is propagated through network topology, both node features and topological structures are naturally preserved throughout the interactive learning process between features and structures. Under this learning paradigm, tremendous effort has been focused on developing effective feature aggregators for improved node representation learning, such as GAT [5] that learns different importance weights for various neighborhoods and GraphSAGE [4] that learns a set of aggregator functions for each node.

Despite promising results, existing graph convolutional learning approaches employ a topology-driven principle, requiring node content to be aggregated by following edge connections. By doing so, they consider that node features and edge relationships are largely consistent and are mutually enhanced during the learning [6]. In reality, network node content (e.g., text and image semantics) and graph topology (e.g., link relations) may be highly inconsistent due to unconscious or deliberate human behaviors [7], [8]. For example, irrelevant citations between scholarly publications result in noisy citation networks. Similarly, an attacker may create fake followers or manipulate friendships, resulting in inconsistent social networks [9]. In addition, incomplete or missing links between nodes are also common, i.e., an image graph based on tag sharing rules often has sparse tag information, therefore, results in missing edges. Fig. 1 shows two examples of inconsistent and incomplete graphs or networks. In summary, graph inconsistency combined with sparse node relations severely challenges existing GNN learning models for the following two reasons.

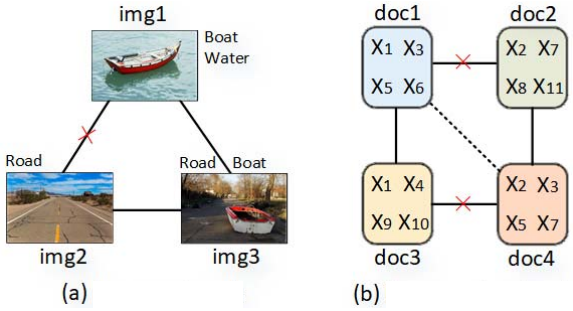

Fig. 1. Illustration of inconsistent/incomplete relationships in networks. (a) Image network. (b) Document network. The topology of both networks is given but is not always consistent with the node content. For example, when using image semantics as node content, the edge between img1 and img2 is inconsistent as they do not share any common semantics (they might be connected because of other unobserved correlations, such as pictures taken during the same trip). Analogically, the inconsistencies exist between (doc1, doc2) and (doc3, doc4), and there is a potential missing edge between doc1 and doc4 due to common features X_3 and X_5.

1) *Noisy Message Passing:* When neighborhood relationships are misaligned to node affinities reflected by node content, nodes will aggregate irrelevant information from neighbors, resulting in noisy content and inferior representation learning. Such noisy messages will continue to pass through graph structures and finally deteriorate the learning of all nodes.
2) *Node Relationship Impairing:* When two nodes have similar content but no link between them, there is no explicit constraint to force similar nodes to be similar in the embedding space. In addition, these absent relations will further impair the relation modeling between other pairs of nodes over the entire network.

Note that network topology is often not optimal, a recent study [10] proposes to optimize network topology structure to improve graph convolutional network (GCN) learning. However, such an approach is potentially risky because revising network topology to satisfy optimization tends to overfit the training data. Alternatively, in this brief, we take a different route to co-align network topology and node content for graph convolutional learning.

Specifically, we focus on information networks where nodes have rich features (content) such as texts and images. Instead of employing the traditional topology-driven principle, we treat network topology

Manuscript received March 27, 2020; revised November 10, 2020 and February 13, 2021; accepted May 18, 2021. This work was supported in part by the US National Science Foundation (NSF) under Grant OAC-2017597, Grant IIS-1763452, and Grant CNS-1828181. *(Corresponding author: Yufei Tang.)*

The authors are with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: mshi2018@fau.edu; tangy@fau.edu; xzhu3@fau.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3084125.

Digital Object Identifier 10.1109/TNNLS.2021.3084125

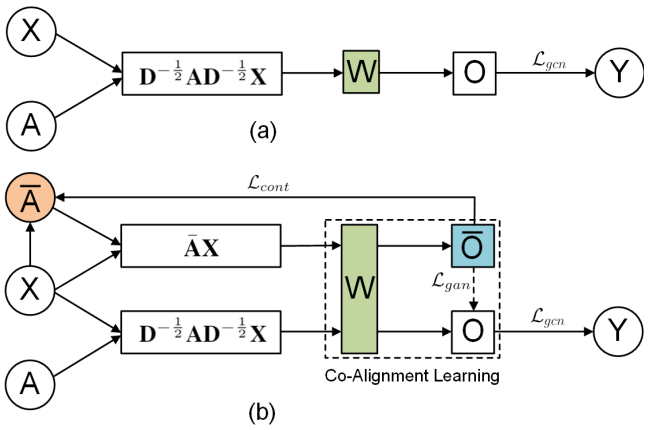

Fig. 2. Difference between GCN and our proposed CoGL. (a) GCN learns graph embeddings based on network topology $\mathbf{A}$ and node content $\mathbf{X}$ by optimizing classification loss $\mathcal{L}_{gcn}$. (b) CoGL first learns to construct a content network $\bar{\mathbf{A}}$ and then performs content and topology co-alignment embedding learning by optimizing a unified loss from three parts: content network construction loss $\mathcal{L}_{cont}$, node classification loss $\mathcal{L}_{gcn}$ and adversarial training loss $\mathcal{L}_{gan}$.

and node content as two distinct, but highly correlated, data sources, and explicitly characterize their differences for embedding learning. We advocate a new "co-alignment" learning principle where the topology network should respect underlying node content, and the content should comply with the topology network for optimized representation learning. A GNN model namely *co-alignment graph convolutional learning* (CoGL) is purposed for this purpose.

To enable co-alignment learning, CoGL reconstructs a content network from node features. The content network together with the original topology network is then set to perform co-alignment learning in an adversarial fashion: 1) the content network aims to learn good embeddings complying with graph topology while 2) the topology network trains to learn optimal embeddings with shared learning parameters for semisupervised node classification. In addition, the content network enforces adversarial training on the topology network in order to balance node content and topology information for optimal node representation learning. The difference between the proposed CoGL and existing GCNs [3] is explained in Fig. 2.

In summary, our main contribution is twofold: 1) we propose modeling inconsistency and discrepancy between network node content and topology for optimal and robust graph embedding and 2) we propose CoGL, a novel GNN model that enables co-alignment leaning between content-based network and the original topology network.

II. Related Work

Given a network with edge connections and content (features) associated with each node, graph embedding learns a low-dimensional vector for each node to preserve node content and network topology [11]. Many works have been proposed, ranging from unsupervised learning methods such as DeepWalk [12] to supervised learning methods such as SemiGraph [13]. The common idea is that nodes with similar topology or similar content are represented using embedding vectors close to each other in the latent space [14].

GNNs [6] are a family of neural network models specifically developed for learning from networked data. GNN models usually have efficient information aggregators [15] that apply directly to graphs to easily incorporate graph structures and features for unified node representation learning. GCNs [3] adopt a spectral-based convolution filter by which nodes can aggregate features from their respective local graph neighborhoods for representation learning. This convolution learning mechanism has been proven successful in many real-world analytic tasks such as link prediction [16], image recognition [2], and new drug discovery [17]. Following the similar convolution graph embedding scheme, many GNN models with more efficient information aggregators have been proposed. For example, graph attention network (GAT) [5] learns to assign different importance weights for various nodes so that nodes can highlight important neighborhoods while aggregating features. GraphSAGE [4] learns a set of aggregation functions for each node to flexibly aggregate information from neighborhoods within different hops. GraphAIR [18] explicitly models the neighborhood interaction in addition to neighborhood aggregation to better capture the complex and non-linear node features. Deep adaptive graph neural network (DAGNN) [19] aims to design deeper convolution layers to capture information from large and adaptive receptive fields.

For all existing GNN approaches, graph convolutional learning is carried out by using topology to drive feature aggregation. A widely accepted assumption is that node content and graph structures are consistent and complementary for measuring node closeness in the embedding space [20], [21]. In reality, the network topology is often noisy and inconsistent to node content [22]. A recent study [23] shows that network topology is not only inconsistent with node content at the individual node level but is also different from the affinity network (built from node content) at the network level (e.g. degree distributions). A recent topology optimization based GCN [10] proposes to revise network topology to improve GCN learning but requires revising/changing network topology which is risky to most users.

Different from the above models that consider graph structures and node content as consistent by default or revise them to ensure consistency, we propose a new co-alignment paradigm to explicitly model their inconsistency for optimal and robust graph embedding in an adversarial training fashion. It is worth mentioning that a couple of existing works [24], [25], including a recently proposed model ARGA [25] built on GCN, also use adversarial network embedding. However, our work is different from ARGA mainly in twofold. First, ARGA aims to design a graph autoencoder to encode network topology and node content in low-dimensional vectors, where topology and content are assumed to be consistent for aligned node relationships modeling based on GCN. In comparison, our work considers that network topology and node content could be inconsistent and aims to balance the two aspects for adversarial node relationships modeling. Second, ARGA aims to force latent node representations to match a prior distribution via an adversarial training scheme, whereas our work aims to balance the network topology and node content for unified and optimal embedding learning through adversarial training between them.

III. Problem Definition and Framework

A. Problem Definition

An information graph can be represented as $\mathrm{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$, where $\mathbf{V} = \{v_i\}_{i=1,\ldots,|\mathbf{V}|}$ is a set of unique nodes and $\mathbf{E} = \{e_{i,j}\}_{i,j=1,\ldots,|\mathbf{V}|;\ i\neq j}$ is a set of edges which can be equal to a $|\mathbf{V}| \times |\mathbf{V}|$ adjacency matrix $\mathbf{A}$ with $\mathbf{A}_{i,j} = w_{i,j} > 0$ if $e_{i,j} \in \mathbf{E}$ and $\mathbf{A}_{i,j} = 0$ if $e_{i,j} \notin \mathbf{E}$. $\mathbf{X} \in \mathbb{R}^{|\mathbf{V}|\times m}$ is a matrix containing all $|\mathbf{V}|$ nodes with their features, i.e., $\mathbf{X}_i \in \mathbb{R}^m$ represents the feature vector of node v_i, where m is the feature vector's dimension. It is easy to conclude that G can be any type of networked data where nodes have feature contents, such as citation networks with texts as node features and image networks with image semantics as node features. Given an information graph G,

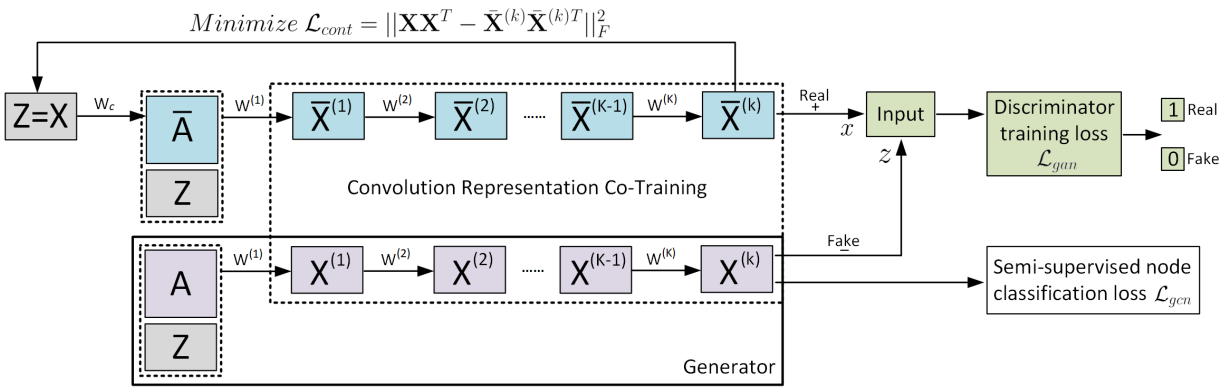

Fig. 3. Proposed CoGL model which performs two convolution representation learning on content network ($\bar{\mathbf{A}}$) and topology network ($\mathbf{A}$), respectively. The content network tries to learn embeddings that best reconstruct $\bar{\mathbf{A}}$ by minimizing $\mathcal{L}_{cont}$ while the topology network tries to learn optimal embeddigs with co-trained parameters by optimizing node classification loss $\mathcal{L}_{gcn}$. Meanwhile, The content network enforces an adversarial training on the topology network by optimizing binary classification loss $\mathcal{L}_{gan}$.

we aim to learn graph embeddings $\mathrm{O} \in \mathbb{R}^{|\mathbf{V}|\times c}$ for node classification in a semisupervised fashion, where c is the embedding size. In this brief, the expressions of graph embedding and representation learning are used interchangeably.

B. Overall Framework

The proposed CoGL model for semisupervised graph embedding and node classification is shown in Fig. 3. It performs co-alignment learning between the constructed content network and the original topology network through three collaborative components.

1) *Content-aligned Graph Topology Learning:* This section intends to learn the content-aligned graph topology network $\bar{\mathbf{A}}$ (content network) from input node features $\mathbf{X}$ by minimizing a graph reconstruction loss.
2) *Semisupervised Graph Embedding:* This section aims to learn convolution graph embeddings from the original graph topology network (topology network) $\mathbf{A}$ for node classification in a semi-supervised manner.
3) *Adversarial Graph Embedding Training:* In this section, the content network embedding enforces an adversarial training on the topology network embedding.

IV. Proposed Method

A. Content-Aligned Graph Topology Learning

We aim to learn a nonnegative matrix $\bar{\mathbf{A}} \in \mathbb{R}^{|\mathbf{V}|\times|\mathbf{V}|}$ (also called the content network) that could reveal the underlying pairwise node relationships from node features $\mathbf{X}$. An *ad hoc* solution is to build a k-nearest neighbor graph [26] or simply calculate the Euclidean distance for each pair of nodes by $\bar{\mathbf{A}}_{i,j} = \|\mathbf{X}_i - \mathbf{X}_j\|$. This, however, does not generate optimal graph embeddings for specific problems. Instead, we adopt a single-layer feed-forward neural network parametrized by a weight vector $\mathbf{W}_c \in \mathbb{R}^{m\times 1}$ and followed by a nonlinear transformation (e.g., $\mathrm{ReLU}(u) = \max(0, u)$). It takes the feature difference $|\mathbf{X}_i - \mathbf{X}_j|$ between ith and jth nodes as the input and outputs the corresponding relevance weight by

$$w_{i,j} = \mathrm{ReLU}(\mathbf{W}_c^T|\mathbf{X}_i - \mathbf{X}_j|). \tag{1}$$

Then, we apply softmax normalization on each node and finally obtain the weight matrix $\bar{\mathbf{A}}$ as

$$\bar{\mathbf{A}}_{i,j} = \mathrm{softmax}(w_{i,j}) = \frac{\exp(\mathrm{ReLU}(\mathbf{W}_c^T|\mathbf{X}_i - \mathbf{X}_j|))}{\sum_{j=1}^{|\mathbf{V}|}\exp(\mathrm{ReLU}(\mathbf{W}_c^T|\mathbf{X}_i - \mathbf{X}_j|))}. \tag{2}$$

After the normalization, $\bar{\mathbf{A}}$ is no longer a symmetric matrix (i.e., $\bar{\mathbf{A}}_{i,j} \neq \bar{\mathbf{A}}_{j,i}$), which is rational as the target is to learn an aligned approximation of nodes to their respective neighborhoods for optimal graph embedding. The calculation of (2) is expensive especially when input node features are with very high dimensions. For efficient calculation, we can first map input node features ($\mathbf{X}$) to a dimension-reduced space [5], [27] and (2) can be rewritten as

$$\bar{\mathbf{A}}_{i,j} = \frac{\exp(\mathrm{ReLU}(\mathbf{W}_c^T|\mathbf{X}_i\mathbf{W}_p - \mathbf{X}_j\mathbf{W}_p|))}{\sum_{j=1}^{|\mathbf{V}|}\exp(\mathrm{ReLU}(\mathbf{W}_c^T|\mathbf{X}_i\mathbf{W}_p - \mathbf{X}_j\mathbf{W}_p|))} \tag{3}$$

where $\mathbf{W}_p \in \mathbb{R}^{m\times d}$ is a learned matrix and $d < m$.

To derive a consistent content network that best aligns node features, we first perform a two-layer convolutional representation learning [3] based on $\bar{\mathbf{A}}$ by

$$\bar{\mathbf{X}}^{(2)} = \bar{\mathbf{A}}\mathrm{ReLU}(\bar{\mathbf{A}}\mathbf{X}\mathbf{W}^{(1)})\mathbf{W}^{(2)} \tag{4}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{m\times h}$ and $\mathbf{W}^{(2)} \in \mathbb{R}^{h\times c}$ are the first and second convolution-layer embedding parameters, respectively. Then, we minimize the reconstruction error between learned node embeddings and input node features as

$$\mathcal{L}_{cont} = \| \mathbf{X}\mathbf{X}^T - \bar{\mathbf{X}}^{(2)}\bar{\mathbf{X}}^{(2)^T} \|_F^2 \tag{5}$$

where both $\mathbf{X}$ and $\bar{\mathbf{X}}^{(2)}$ have been normalized to ensure stable parameter learning, i.e., through $\mathbf{X} = \mathrm{softmax}(\mathbf{X})$ and $\bar{\mathbf{X}}^{(2)} = \mathrm{softmax}(\bar{\mathbf{X}}^{(2)})$. The idea of above topology learning process is analogical to an autoencoder which aims to learn the suitable content-aligned network topology $\bar{\mathbf{A}}$ by minimizing [i.e., through (5)] the input node features $\mathbf{X}$ and output node features $\bar{\mathbf{X}}^{(2)}$ after the convolution learning. Because the input and output node features may have different dimensions, we transform them to minimize their respective gram matrices which have the same shape.

B. Semisupervised Graph Embedding

We perform graph representation learning for node classification in a semisupervised manner. In a similar way, the low-dimensional embeddings are derived through a two-layer recursive convolution learning based on the original topology network $\mathbf{A}$ by

$$\mathrm{O} = \tilde{\mathbf{A}}\mathrm{ReLU}(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}^{(1)})\mathbf{W}^{(2)} \tag{6}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-(1/2)}(\mathbf{I} + \mathbf{A})\mathbf{D}^{-(1/2)}$ denotes the normalized form of $\mathbf{A}$, $\mathbf{I}$ is an identity matrix with the same shape and $\mathbf{D}$ is the degree matrix of $(\mathbf{I} + \mathbf{A})$. Here, the convolution learning parameters

$\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ are shared and co-trained between the content network and the topology network, which are beneficial to balance node content and topology for graph construction and embedding learning. Then, we apply semisupervised training to learn the parameters by minimizing the node classification loss $\mathcal{L}_{gcn}$ as follows:

$$\mathrm{H} = \mathrm{softmax}(\mathrm{O}) = \frac{\exp(\mathrm{O})}{\sum_t \exp(\mathrm{O}_t)} \tag{7}$$

$$\mathcal{L}_{gcn} = -\sum_{i \in \mathcal{Y}_L} \sum_{j=1}^{c} \mathrm{Y}_{i,j} \ln \mathrm{H}_{i,j} \tag{8}$$

where the node embedding size c equals to the total number of labels for graph nodes. $\mathrm{Y} \in \mathbb{R}^{|\mathbf{V}| \times c}$ denotes the one-hot label indicators matrix for all nodes and $\mathcal{Y}_L$ is a set of node indices with labels known for semisupervised training.

C. Adversarial Graph Embedding Training

The learning balance between content and topology is achieved through adversarial training [28], where the goal is to consider both content and topology information for optimal graph embedding and node classification. As shown in Fig. 3, the topology network learning component is considered a generator $\mathcal{G}(\mathbf{A}, \mathbf{X})$ that generates node embeddings based on the topology network $\mathbf{A}$. During adversarial training, the discriminator tries to classify the real node (positive) sample $x \in \bar{\mathbf{X}}^{(2)}$ learned from the content network $\bar{\mathbf{A}}$ as class 1 and meanwhile classify the fake node (negative) sample $z = \mathcal{G}(\mathbf{A}, \mathbf{X})$ as class 0. On the other side, the generator tries to fool the discriminator by classifying z as class 1. The adversarial embedding training objective is given as

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{L}_{gan}(\mathcal{D}, \mathcal{G}) = \mathbb{E}_{x \sim \bar{\mathbf{X}}^{(2)}} \log \mathcal{D}(x) + \mathbb{E}_z \log(1 - \mathcal{D}(\mathcal{G}(\mathbf{A}, \mathbf{X}))). \tag{9}$$

D. Model Optimization and Training

As described above, the content network construction and graph embedding conform to an adversarial co-alignment learning fashion for optimal node classification performance. As the three components in Fig. 3 share and co-train convolution graph embedding parameters $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$, we finally seek to optimize the following combined objective as

$$\mathcal{L} = \mathcal{L}_{gcn} + \alpha \mathcal{L}_{cont} + \beta \mathcal{L}_{gan}(\mathcal{D}, \mathcal{G}) \tag{10}$$

where α and β are set to balance the content network construction and the adversarial embedding training, respectively. Note that $\mathcal{L}_{gan}(\mathcal{D}, \mathcal{G})$ in (9) involves both discriminator and generator trainings, where each part needs to combine both $\mathcal{L}_{gcn}$ and $\mathcal{L}_{cont}$ for collective model parameter optimization. The training procedure of our CoGL model is summarized in Algorithm 1.

Algorithm 1 Training the CoGL Model

Input : Graph topology $\mathbf{A}$ and node features $\mathbf{X}$
Output: Node embeddings $\mathrm{O} \in \mathbb{R}^{|\mathbf{V}| \times c}$
Initialization: $i = 0$, training epochs M and N

```
while i ≤ M do
    Ā ← build content network through (3);
    X̄^(2) ← learn content network embeddings through (4);
    O ← learn topology network embeddings through (6);
    for j = 0, ..., N do
        Sample n instances from content network embeddings X̄^(2);
        Sample n instances from topology network embeddings O;
        Update model parameters based on (10) in two steps:
            1) Optimize L while training the discriminator;
            2) Optimize L while training the generator.
    end
    i = i + 1.
end
```

V. Experiments

In this section, we evaluate the proposed model for supervised node classification on six benchmark datasets. The dataset statistics are summarized in Tables I and II.

A. Datasets

We use four benchmark networks, Cora, Citeseer, Pubmed, and DBLP that have been widely used for node-level classification in previous work [3], [29]. **Cora** contains 2708 research articles grouped into seven machine learning classes such as *Reinforce Learning* and *Genetic Algorithms*. There are 5429 edges between them and each paper node is described with a feature vector of 1433 dimensions. **Citeseer** contains 3327 research articles in six classes with 4732 links between them, where each paper node has a feature vector of 3703 dimensions. **Pubmed** contains 19 717 literature nodes and 44 338 edges. Each node belongs to one of the three classes and has a feature vector of 500 dimensions. **DBLP** contains 17 725 publications from four classes. It has 52 890 edges and each node is associated with a feature vector of 6974 dimensions.

TABLE I
Document Network Characteristics

Items	Cora	Citeseer	PubMed	DBLP
# Nodes	2708	3327	19717	17725
# Edges	5429	4732	44338	52890
# Features	1433	3703	500	6974
# Classes	7	6	3	4

TABLE II
Image Network Characteristics

Items (#)	Nodes	Edges	Features	Classes
MIR	5892	380808	500×375	152
ImageCLEF	3461	221185	500×375	134

We also use two multilabel image networks[1] MIR and ImageCLEF. **MIR** has 5892 nodes from 152 classes. Each node represents a 500×375 RGB color image and there are 380 808 edges for this network. **ImageCLEF** contains 3461 nodes and 221 185 edges. Each node is also a 500×375 RGB image that corresponds to one or more of the 134 classes. For each image in these two datasets, we extract a CNN feature descriptor and the feature dimensions for MIR and ImageCLEF are transformed to 152 and 134, respectively.

B. Baseline Methods and Settings

1) Baseline Methods: We employ the following baselines for semisupervised node classification comparison. For all baselines, DeepWalk [12], SemiEmb [30], and so on represent classical shallow node embedding methods. GCN [3], GAT [5], DGCN [31], and so on represent GNNs and attention mechanisms. GMNN [32]

[1] https://snap.stanford.edu/data/web-flickr.html

and GMI [33] represent the most recent approaches that actively model discrepancy between node content and topology for embedding learning.

1) *DeepWalk [12]:* It performs random walks over the network to capture neighborhood relationships between nodes. The SkipGram model is then used to derive the low-dimensional node embeddings.
2) *SemiEmb [30]:* A semisupervised embedding method that applies "shallow" learning techniques such as kernel methods on deep network architectures by adding a regularizer at the output layer.
3) *Planetoid [34]:* A semisupervised method in which the embedding of each node is jointly trained to predict the class label of the node and the context on the network.
4) *Chebyshev [35]:* It extends the traditional convolutional neural networks to design fast localized convolutional filters on graphs.
5) *ARGA [25]:* It is a graph autoencoder where node embeddings are trained to reconstruct the graph structure. Meanwhile, node embeddings are enforced to match a prior distribution via an adversarial training scheme.
6) *GCN [3]:* It performs embedding learning with the spectral-based convolutional filter, where each node generates the representation by aggregating features from its immediate neighborhoods.
7) *GMNN [32]:* It performs semisupervised embedding learning with a graph Markov neural network. A conditional random field is used to model the joint distribution of node labels, and two GNNs are utilized to improve both the inference and learning procedures.
8) *DGCN [31]:* It performs semisupervised embedding learning by simultaneously preserving local neighborhood and global context information with two GCNs.
9) *GAT [5]:* It learns importance weights for different nodes and each node is able to highlight important neighborhoods for efficient embedding learning.
10) *GMI [33]:* It performs unsupervised embedding learning based on the concept of graph mutual information. The idea is to directly maximize the mutual information between the input and output of a graph neural encoder in terms of node features and topological structure.
11) *CoGL:* The proposed model, which first learns a content-aligned graph topology network, and then couples the learned network topology and the original network topology for adversarial supervised embedding learning.

2) Topology-Content Inconsistency Intervention: To further evaluate the proposed CoGL model in handling incomplete and inconsistent node relationships for improved embedding learning, we design experiments to learn inconsistent network topology created in two ways.

1) *Structure Removal:* We randomly remove some portion of structures (e.g., edges) from the original network to create incomplete node relationships in the network. We use r_{RS} to denote the portion of the total number of edges removed from the network. After removal the left number of edges will be $(1 - r_{RS})|\mathbf{E}|$.
2) *Structure Injection:* We randomly inject some portion of noisy structures in the original network to cause inconsistent node relationships in the network, i.e., manually add links for some pairs of nodes. We use r_{IS} to denote the ratio of the number of edges over the total added to original the network. After injection, the total number of edges will be $(1 + r_{IS})|\mathbf{E}|$.

TABLE III
NODE CLASSIFICATION ACCURACY ON CORA, CITESEER, PUBMED, AND DBLP DATASETS (THE TOP TWO BEST RESULTS ARE BOLDFACED AND UNDERSCORED, RESPECTIVELY)

Methods	Cora	Citeseer	Pubmed	DBLP
DeepWalk	67.2%	43.2%	65.3%	66.3%
SemiEmd	59.0%	59.6%	71.7%	72.1%
Planetoid	75.7%	64.7%	77.2%	74.7%
Chebyshev	81.2%	69.8%	74.4%	74.3%
ARGA	78.4%	61.3%	73.8%	65.4%
GCN	81.5%	70.9%	79.0%	75.1%
GMNN	<u>83.4%</u>	**72.6%**	**81.2%**	**80.2%**
DGCN	82.9 ± 0.4%	72.1 ± 0.4%	78.3 ± 0.3%	76.5 ± 0.3%
GAT	83.2 ± 0.7%	71.0 ± 0.7%	79.0 ± 0.3%	78.2 ± 0.7%
GMI	82.1 ± 0.1%	72.0 ± 0.7%	<u>79.6 ± 0.4%</u>	77.4 ± 0.1%
CoGL (**Ours**)	**84.1** ± 0.6%	<u>72.4</u> ± 0.5%	79.2 ± 0.3%	<u>79.8</u> ± 0.6%

Although we consider network content to be reliable information for node relationship reconstruction in this work, network content could be noisy which may degrade the embedding performance. Therefore, we design the following experiment to test the impact.

1) *Content Injection:* We randomly inject noisy content into the original node features, i.e., for the document networks the random irrelevant words can be added to the nodes. We denote r_{IC} as the injection ratio. After the content injection steps the total number of features in the network would be $(1 + r_{IC})|\mathbf{X}|$.

Since CoGL is developed from the GCN, we specially compare the performance between CoGL and GCN to validate the effectiveness of the proposed model for topology and content co-alignment embedding learning, where the comparison results on Cora, Citeseer, and DBLP are reported.

3) Parameter Settings: For experiments of baseline methods on Cora, Citeseer, Pubmed, and DBLP datasets, we follow the same settings as in previous works [3], [5]. Twenty labeled nodes for each class are used for semisupervised training. Five hundred validation nodes are used for fine-tuning the hyperparameters and the classification results are compared on 1000 test nodes. For the MIR and ImageCLEF datasets, we, respectively, select 300, 500, and 700 labeled image nodes for training the model. For other unlabeled nodes, we select 1000 and 2000 images for validation and test, respectively. For each experiment, data splits are performed randomly and the average performances are finally reported over ten runs.

In our approach, the hidden layer dimensions d for content network construction are set as 30 and the hidden convolution layer dimension h is set as 30. We train CoGL for a maximum of 1000 epochs based on the Adam algorithm with early stopping of 200 epochs. For the four document networks, the dropout probability and learning rate are set as 0.5 and 0.002, and they are 0.2 and 0.01 for MIR and ImageCLEF datasets. For comparison, we set the default balance parameters α and β as 0.4 and 0.8, respectively. Similar to previous work [3], we use a L_2 norm regularization where the weight decay is set as 5e-4. The implementation code of CoGL can be available at https://github.com/codeshareabc/CoGL.

C. Node Classification Results

Table III shows the node classification results on Cora, Citeseer, Pubmed, and DBLP datasets, and the results for MIR and ImageCLEF datasets are shown in Table IV. The top two best results in the two tables are boldfaced and underscored. From the

TABLE IV
NODE CLASSIFICATION ACCURACY (%) ON MIR AND IMAGECLEF DATASETS (THE TOP TWO BEST RESULTS ARE BOLDFACED AND UNDERSCORED, RESPECTIVELY)

Datasets	**MIR**			**ImageCLEF**		
# Labeled Nodes	300	500	700	300	500	700
DeepWalk	41.42 ± 0.22	42.98 ± 0.25	44.50 ± 0.33	43.34 ± 0.50	43.67 ± 0.39	44.98 ± 0.43
SemiEmd	46.75 ± 0.39	47.24 ± 0.48	48.61 ± 0.27	54.30 ± 0.67	55.22 ± 0.49	56.50 ± 0.42
Planetoid	50.74 ± 0.33	51.04 ± 0.40	51.91 ± 0.23	56.75 ± 0.36	57.77 ± 0.36	57.99 ± 0.49
Chebyshev	53.28 ± 0.32	54.28 ± 0.32	55.87 ± 0.45	58.12 ± 0.37	58.79 ± 0.51	59.58 ± 0.51
ARGA	52.14 ± 0.48	52.65 ± 0.53	53.51 ± 0.36	57.46 ± 0.41	58.39 ± 0.42	58.67 ± 0.45
GCN	55.43 ± 0.41	56.14 ± 0.38	57.65 ± 0.27	60.33 ± 0.34	61.04 ± 0.54	61.60 ± 0.38
GMNN	56.09 ± 0.21	57.42 ± 0.24	58.61 ± 0.29	61.12 ± 0.38	61.74 ± 0.32	62.83 ± 0.37
DGCN	55.76 ± 0.52	57.05 ± 0.47	58.37 ± 0.38	61.02 ± 0.54	61.54 ± 0.43	62.21 ± 0.46
GAT	55.59 ± 0.54	57.27 ± 0.76	58.38 ± 0.48	61.45 ± 0.52	61.47 ± 0.32	62.32 ± 0.41
GMI	54.62 ± 0.51	55.77 ± 0.49	57.08 ± 0.46	60.05 ± 0.61	61.68 ± 0.45	61.23 ± 0.48
CoGL (**Ours**)	**57.23 ± 0.46**	**58.78 ± 0.39**	**60.11 ± 0.45**	**61.77 ± 0.57**	**62.52 ± 0.40**	**63.86 ± 0.52**

comparison results, we have four major observations together with their possible explanations.

1) Graph convolutional kernel-based methods including CoGL, GAT, DHCN, GMNN, GCN, and Chebyshev perform generally better than other random walk and regularization based methods such as DeepWalk, SemiEmb, and Planetoid. The improvements are mainly attributed to the graph convolutional kernel which allows nodes to aggregate desired features from their neighborhoods for efficient end-to-end supervised classification training. From Tables III and IV, we can observe that CoGL consistently outperforms some strong baseline methods including GAT, DGCN, GCN, and ARGA, which demonstrate the effectiveness of the proposed co-alignment node structure and feature learning framework.
2) The results in Tables III and IV show that GMNN frequently outperforms other models, including GCN- and GAT-based approaches. This is mainly attributed to its unique design which models the joint distribution of node labels conditioned on node attributes, given the network topology structure. GMNN combines two GNN models through an EM (Expectation Maximization) learning process. This is similar to the co-training process employed in CoGL. Both CoGL and GMNN share similar design principles to consider/model discrepancy between node labels, topology, and node content for node embedding learning. However, the major difference is that CoGL explicitly models content as a content-aligned graph topology $\bar{\mathbf{A}}$, whereas GMNN still uses the original network and models node labels as a conditional random field.
3) CoGLachieves generally comparable results with the state-of-the-art method GMNN on the citation networks and performs slightly better than GMI on Cora and DBLP observed from Table III, while both GMNN and GMI outpace GCN. It means that the content-aligned graph topology learned in CoGL is enhancing the original graph topology learning (e.g., GCN) through the adversarial training process. In addition, the comparison results in Table IV show that CoGL outperforms all baseline methods. The reason is probably that the two image networks contain many irrelevant or missing links because of the coarse-grained way to construct image node connections, i.e., images sharing at least one tag build a link. However, images often correspond to multiple tags of different importance and the original network structure may not capture accurate node relationships. The results demonstrate that the introduced topology learning component in CoGL has contributed to the performance gain over the basic GCN model learning on the original image networks.
4) Both CoGL and ARGA are developed based on the adversarial training principle. We can observe from Tables IV and V that CoGL significantly outperforms ARGA to the node classification task. There are two main reasons to explain the difference. First, although both ARGA and CoGL use GCN to integrate network topology and content for unified embedding learning, CoGL is more specifically designed for the end-to-end node classification, while ARGA targets at implementing a graph autoencoder in an unsupervised manner. Second, the adversarial training in ARGA simply aims to force the latent node representations to follow a prior Gaussian distribution, whereas the adversarial training in CoGL aims to balance network topology and node content for consistent and improved embedding learning.
5) Among all baselines, GAT aims to assign varying importance weights to different links while performing the feature aggregation, which helps to encode important network topology aligned with node content for unified embedding learning. However, we can observe from the results that CoGL performs slightly better than GAT, where the potential reason is that the learned content-aligned network may have provided enriched information over the plain node content to reveal the relationships between nodes [27].

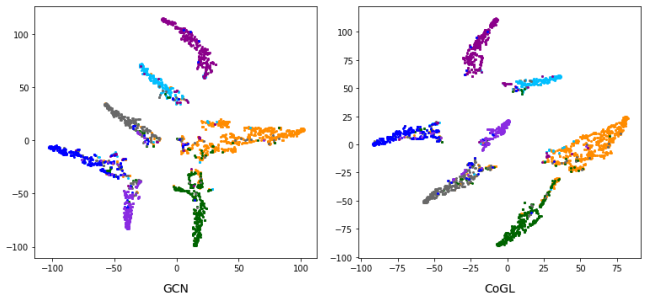

Fig. 4. Embedding 2-D visualization on Cora dataset.

The comparison results of CoGL against various strong baselines empirically verified the effectiveness of our approach for information network embedding learning. In addition, the tSNE-based

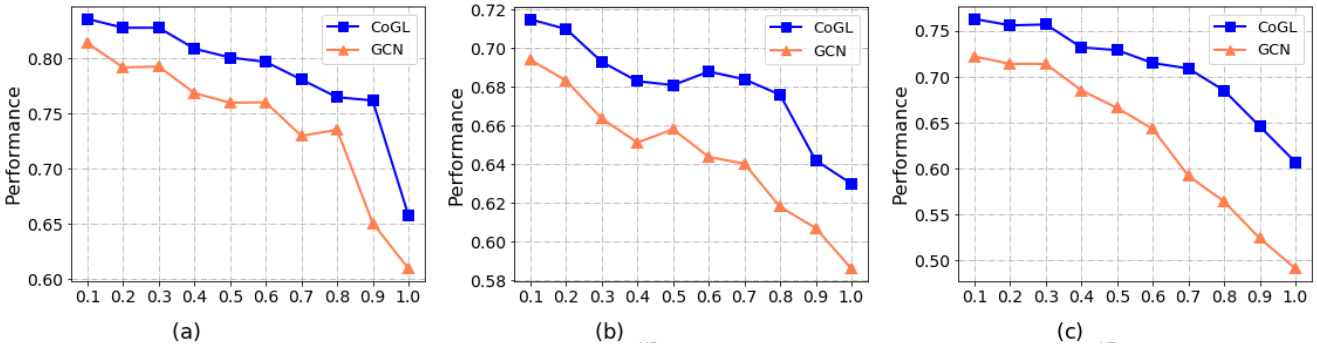

Fig. 5. Impact of removal ratio r_{RS} of original structure in Cora, Citeseer and DBLP networks. (a) r_{RS} on Cora. (b) r_{RS} on Citeseer. (c) r_{RS} on DBLP.

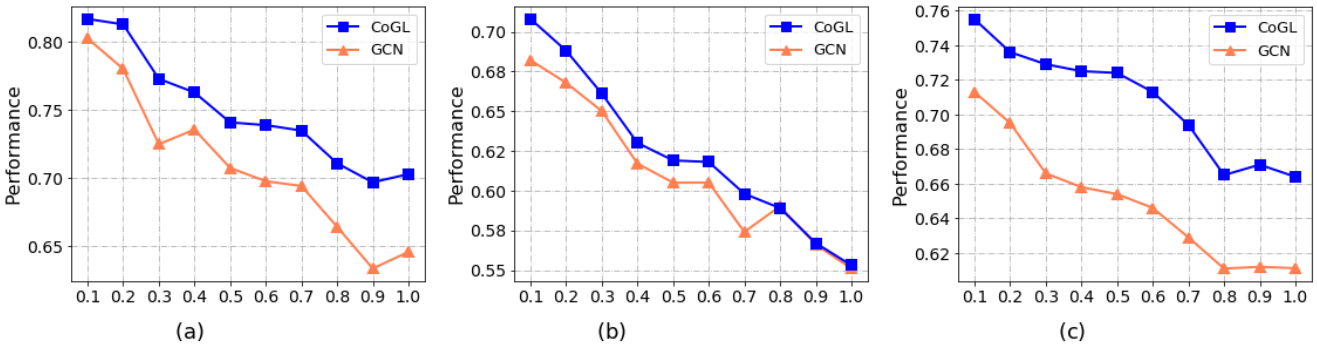

Fig. 6. Impact of injection ratio r_{IS} of noisy structure in Cora, Citeseer and DBLP networks. (a) r_{IS} on Cora. (b) r_{IS} on Citeseer. (c) r_{IS} on DBLP.

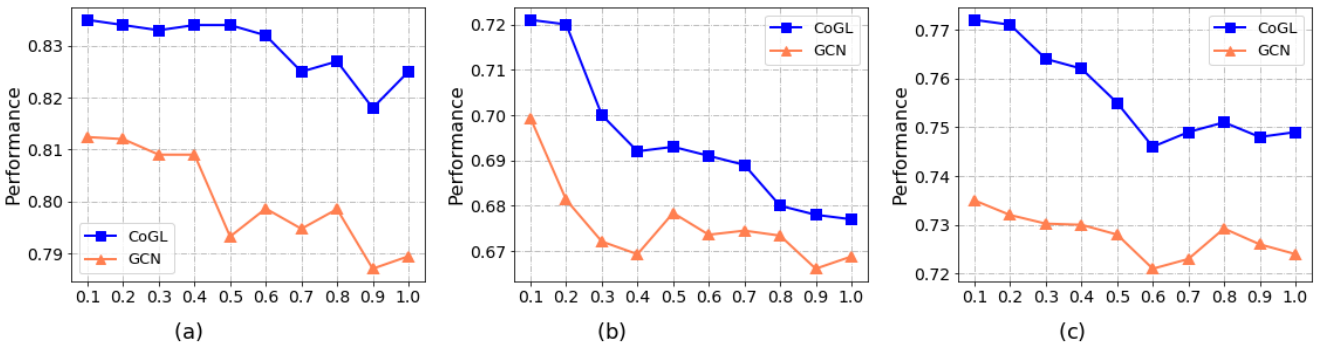

Fig. 7. Impact of injection ratio r_{IC} of noise content in Cora, Citeseer and DBLP networks. (a) r_{IC} on Cora. (b) r_{IC} on Citeseer. (c) r_{IC} on DBLP.

visualization results in the 2-D space shown in Fig. 4 demonstrate that CoGL can achieve slightly discriminative clusters compared with GCN, i.e., for CoGL, nodes in some clusters are slightly more compact and boundaries among clusters are slightly more clear as well, which is consistent with the node classification result that CoGL slightly outperforms GCN on the Cora dataset.

D. Topology-Content Inconsistency Intervention Results

Compared with GCN which relies on the original network topology for convolutional embedding learning, CoGL adds a content-aligned topology learning component to balance the content and topology parts for consistent learning. We manually disturb the inconsistency level between network topology and content by removing network links, injecting noisy network links, and injecting noisy node contents, which are controlled by parameters r_{RS}, r_{IS}, and r_{RC}, respectively.

Fig. 5 shows the impact of r_{RS} using different values. We can observe that performances of CoGL and GCN gradually decline with r_{RS} increasing from 0.1 to 1.0, where CoGL significantly outperforms GCN all the way. It is interesting to note that when r_{RS} is set to be larger, CoGL tends to achieve better performance gain in all three networks, i.e., $r_{RS} = 1$ means that nodes in the network are fully isolated and the embedding for each node is only derived from its content, which verifies that the content-aligned topology learning component in CoGL is beneficial to complement the missing linkage relationships between nodes. Fig. 6 shows the impact of r_{IS} where we can observe similar decreasing trends with more noisy structures injected in the networks. The noisy network structures would deteriorate the relationships among nodes and we can conclude the content-based network topology has benefited the performance when irrelevant links existed in the networks.

Similar decreasing trend and improvement of CoGL over GCN can be observed for r_{IC} shown in Fig. 7. As we have hypothesized, noisy

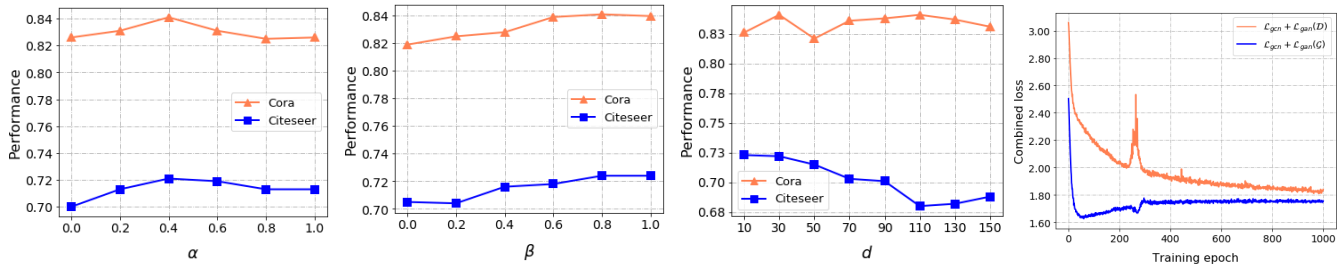

Fig. 8. Impact of parameters α, β, d, and the classification training loss.

features will be propagated over the network during the convolution learning, which may deteriorate the linkage relationships among nodes for GCN-based methods such as GCN and CoGL. However, we can conclude from Fig. 7 that the introduced content-aligned topology learning in CoGL has mitigated the above issue for two possible reasons. First, the learned network topology is able to span more enriched node relationships than the naive content that has enhanced the embedding learning. Second, although the input node content is noisy, the topology learning component is able to learn fair structural relationships among nodes to maximize the end-to-end classification accuracy, such that the unified embedding learning has benefited from the adversarial and balanced training between the original topology and the learned topology. It is interesting to note that the performance degradation is less dramatic for r_{IC} when compared with r_{RS} and r_{IS}, where the reason is that the GCN-based embedding learning methods rely more on the network structure than the content for modeling relationships between nodes, i.e., the original network topology stays intact for r_{IC} which has guaranteed the network embedding quality.

E. Parameter Sensitivity and Classification Loss

The impacts of parameters α and β are reported in Fig. 8. The results show that ignoring the content network construction component ($\alpha = 0.0$) or the adversarial training component ($\beta = 0.0$) during training will cause the lowest classification performance, which verifies the benefit of combining content network and topology network for co-alignment training in this brief. We also studied the parameter d which is involved in (3). The impact of d is shown in Fig. 8 (third). We can observe that performances vary from the values of d, where performance on Cora tends to be lower with larger values of d, whereas d has a less classification impact on Citeseer.

To demonstrate the balanced learning between topology network and content network, we show the combined loss of node classification ($\mathcal{L}_{gcn}$) and adversarial binary classification (discriminator $\mathcal{L}_{gan}(\mathcal{D})$ or generator $\mathcal{L}_{gan}(\mathcal{G})$) in Fig. 8 (fourth). We observe that the two curves tend to be closer but there still a difference, which reflects an adversarial balance between topology network and content network for unified and optimal graph embedding and node classification.

VI. Conclusion

Network topology and node content (including labels) are two main information sources commonly used in many network embedding models. Although topology and node content are often inconsistent, existing methods, especially graph convolution networks, naturally ignore such inconsistency for embedding learning. In this brief, we took the inconsistency between graph structure and features into consideration for improved graph embedding learning, and proposed a novel CoGL model, CoGL. The merit of CoGL lies in that it co-aligns the original topology network and the constructed content network for optimal node embedding and classification. Experiments and validations on six benchmark datasets demonstrated the effectiveness of CoGL. The proposed model can be used to learn embeddings for a variety of information networks with rich node content, such as the widely seen document networks and image networks.

References

[1] W. Fan *et al.*, "Graph neural networks for social recommendation," in *Proc. World Wide Web Conf.*, 2019, pp. 417–426.
[2] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, "Multi-label image recognition with graph convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5177–5186.
[3] 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
[4] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Process. Syst.*, 2017, pp. 1024–1034.
[5] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*. [Online]. Available: http://arxiv.org/abs/1710.10903
[6] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," 2019, *arXiv:1901.00596*. [Online]. Available: http://arxiv.org/abs/1901.00596
[7] A. Hofmann, S. Perchani, J. Portisch, S. Hertling, and H. Paulheim, "DBkWik: Towards knowledge graph creation from thousands of Wikis," in *Proc. Int. Semantic Web Conf. Posters, Demos Ind. Tracks*, 2017, pp. 1–4.
[8] H. Liu, L. Bai, X. Ma, W. Yu, and C. Xu, "ProjFE: Prediction of fuzzy entity and relation for knowledge graph completion," *Appl. Soft Comput.*, vol. 81, Aug. 2019, Art. no. 105525.
[9] A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," 2018, *arXiv:1809.01093*. [Online]. Available: http://arxiv.org/abs/1809.01093
[10] L. Yang, Z. Kang, X. Cao, D. Jin, B. Yang, and Y. Guo, "Topology optimization based graph convolutional network," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019.
[11] M. Shi, Y. Tang, X. Zhu, J. Liu, and H. He, "Topical network embedding," *Data Mining Knowl. Discovery*, vol. 34, no. 1, pp. 75–100, 2020.
[12] 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.
[13] R. Hisano, "Semi-supervised graph embedding approach to dynamic link prediction," in *Proc. Int. Workshop Complex Netw.* Cham, Switzerland: Springer, 2018, pp. 109–121.
[14] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, Sep. 2018.
[15] S. S. Du, K. Hou, R. R. Salakhutdinov, B. Poczos, R. Wang, and K. Xu, "Graph neural tangent kernel: Fusing graph neural networks with graph kernels," in *Proc. Adv. Neural Process. Syst.*, 2019, pp. 5724–5734.
[16] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. Adv. Neural Process. Syst.*, 2018, pp. 5165–5175.
[17] M. Sun, S. Zhao, C. Gilvary, O. Elemento, J. Zhou, and F. Wang, "Graph convolutional networks for computational drug development and discovery," *Briefings Bioinf.*, vol. 21, no. 3, pp. 919–935, May 2020.

[18] F. Hu, Y. Zhu, S. Wu, W. Huang, L. Wang, and T. Tan, "GraphAIR: Graph representation learning with neighborhood aggregation and interaction," *Pattern Recognit.*, vol. 112, Apr. 2021, Art. no. 107745.

[19] M. Liu, H. Gao, and S. Ji, "Towards deeper graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 338–348.

[20] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," *Network*, vol. 11, no. 9, p. 12, 2016.

[21] A. Maheshwari, A. Goyal, A. Kumar, M. K. Hanawal, and G. Ramakrishnan, "Representation learning on graphs by integrating content and structure information," in *Proc. 11th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2019, pp. 88–94.

[22] J. Wu, S. Pan, X. Zhu, Z. Cai, and C. Zhang, "Multi-graph-view learning for complicated object classification," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2015, pp. 3953–3959.

[23] T. Guo, S. Pan, X. Zhu, and C. Zhang, "CFOND: Consensus factorization for co-clustering networked data," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 4, pp. 706–719, Apr. 2019.

[24] H. Wang *et al.*, "GraphGAN: Graph representation learning with generative adversarial nets," 2017, *arXiv:1711.08267*. [Online]. Available: http://arxiv.org/abs/1711.08267

[25] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," 2018, *arXiv:1802.04407*. [Online]. Available: http://arxiv.org/abs/1802.04407

[26] M. Maier, U. V. Luxburg, and M. Hein, "Influence of graph construction on graph-based clustering measures," in *Proc. Adv. Neural Process. Syst.*, 2009, pp. 1025–1032.

[27] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-supervised learning with graph learning-convolutional networks," in *Proc. IEEE CVPR*, Jun. 2019, pp. 11313–11320.

[28] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[29] J. Leskovec and A. Krevl, "SNAP datasets: Stanford large network dataset collection," Jun. 2014. [Online]. Available: http://snap.stanford.edu/data

[30] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural networks: Tricks of the trade*. Berlin, Germany: Springer, 2012, pp. 639–655.

[31] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *Proc. World Wide Web Conf. (WWW)*, 2018, pp. 499–508.

[32] M. Qu, Y. Bengio, and J. Tang, "GMNN: Graph Markov neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5241–5250.

[33] Z. Peng *et al.*, "Graph representation learning via graphical mutual information maximization," in *Proc. Web Conf.*, 2020, pp. 259–270.

[34] 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

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