Decoupled Representation Learning for Attributed Networks

Hao Wang, Defu Lian, Hanghang Tong, Qi Liu, Zhenya Huang, and Enhong Chen, *Senior Member, IEEE*

Abstract—Network representation learning or network embedding, which targets at learning the low-dimension representation of graph-based data, has attracted wide attention due to its effectiveness on various network-oriented applications in recent years. Though large efforts have been made on the joint analysis combining node attributes with the network structure, they usually model the interactions between nodes reflected by network structure and attributes in a coupled way and fail to address the common sparse attribute issues. To this end, in this article, we comprehensively study the problem of learning attributed network embedding, which focuses on characterizing different types of interactions among nodes and alleviating the sparse attribute problem as well. Specifically, we propose a novel **DeC**oupled **N**etwork **E**mbedding (DCNE) model to learn node representations in a unified framework. We first respectively project both nodes and attributes into a low-dimensional vectorial space. Then, we introduce a novel "decoupled-fusion" learning process into each graph layer to iteratively generate node embeddings. In particular, we propose two adapted graph convolution modules to decouple the learning of network structure and attributes respectively, and a fusion module to adaptively aggregate the information. Next, we adopt a modified mini-batch algorithm to iteratively aggregate the higher-order information of both nodes and attributes within a multi-task learning framework. Extensive experiments on five public datasets demonstrate that DCNE could outperform state-of-the-art methods on multiple benchmark tasks. Moreover, several qualitative analyses further indicate DCNE can learn more robust and representative node embeddings than other comparison methods for attributed networks.

Index Terms—Network embedding, attributed network, sparsity, decoupled and fusion, graph neural network

1 Introduction

NETWORKS are ubiquitous in our daily lives, such as social networks, citation networks, road maps, and protein networks [1], [2]. As one of the most common data structures in the real world, learning to process network data becomes the cornerstone in the research community. Among these studies, one of the fundamental issues is how to effectively generate the network data representation, which attracts much attention for a long time [3]–[6].

To achieve this goal, as shown in Fig. 1, network embedding, as a kind of promising techniques, targets at representing a network in a low-dimensional hidden space, where each node (i.e., $\{v_1, \cdots, v_{12}\}$) can be assigned with a vectorial embedding. Generally, the learned node embeddings are essential for capturing their relationships which are originally reflected by their topological structure with respect to the edge links [7]. In fact, by effectively encoding the topological characteristics, network embedding can be directly applied in various downstream applications, such as node classification [3], [4], link prediction [2], [8], network clustering [9] and social influence analysis [10]–[13].

In the literature, there are many efforts in designing network embedding methods. Traditionally, researchers mainly explore the effects of local or global network structure and propose a series of structure-preserved network embedding algorithms to capture the essential properties [3], [4], [14]. Generally, the basic idea is that nodes with similar structures should be closer in the embedding space. Along this line, earlier efforts try to exploit the neighborhood contexts between nodes for network embedding with random walk strategy, such as DeepWalk [3] and Node2vec [4]. Later, LINE [15] and GraRep [16] capture the high order proximity of node, which preserve more global network structures. Recently, aiming to keep higher-order nonlinear network structure, several deep learning based methods have been proposed, whose representative ones are SDNE [14], DNGR [17], etc. Besides the topological structure, nodes in the network are usually along with informative side attributes (e.g., category, content), which help identify some individual properties. Therefore, researchers incorporate such attributes, following the assumption that nodes with similar attributes should also be similarly reflected by their representations [18]–[21]. For instance, TADW [18] and LANE [19] respectively take the text information and label distribution in matrix factorization. Then, recent state-of-the-art studies iteratively generate the node embedding by aggregating attribute information from its local k-order neighbors following the message-passing-receiving mechanism, such as GraphSAGE [22] and GAT [23].

Although many previous work has demonstrated the advanced performance of attributed network embedding methods, they still have certain limitations for characterizing the complex interactions between nodes in the sparse situation. Specifically, 1). *sparse attributes*: most of the at-

- *H. Wang, D. Lian (corresponding author), Q. Liu, Z. Huang, and E. Chen (corresponding author) are with the Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Techonology, University of Science and Technology of China, Hefei, Anhui, 230026, China. Email: wanghao3@mail.ustc.edu.cn, {liandefu,qiliuql,huangzhy,cheneh}@ustc.edu.cn*
- *H. Tong is with the Department of Computer Science, University of Illinois at Urbana-Champaign. Email: htong@illinois.edu.*

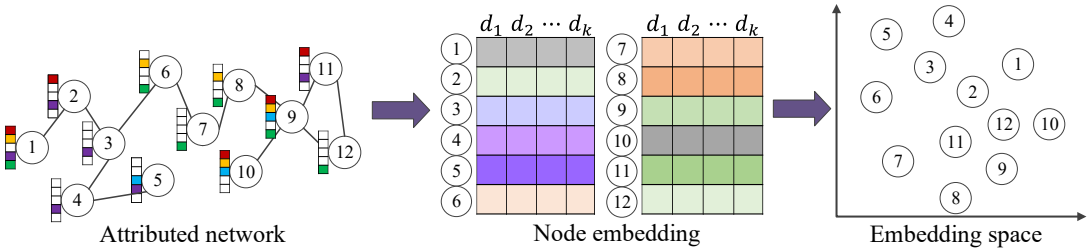


Fig. 1. A toy example of network embedding.

tributed networks face the sparsity problem of node attributes that the number of valued attributes is extremely small. As we know, the number of node attributes has an important impact on network embeddings, which could enhance the inference of the structural connections between nodes [18]. Therefore, it's necessary to generalize node attributes for alleviating this sparsity issue and improving the representative ability of embeddings, which has been ignored by many previous works; 2). *complex interactions*: most recent work typically models the interactions between nodes in a coupled way, where the informative attributes are utilized by initialization or side information for learning node embeddings. In other words, they generally assume that the similarities between nodes reflected by topological structure and attribute correlation in the network are consistent. Therefore, they cannot explicitly distinguish the different effects of topology structure and node attributes with regard to node connections. To this end, in this article, we aim to explore the complex interactions of such two different properties between nodes via a decoupled manner and alleviate the sparse attribute issues.

However, there are several technical challenges along this line. First, in a general network, node attributes are extremely sparse and endowed with incomplete values in practical scenarios. It is a challenge how to enhance the generalization ability of these limited valued attributes for alleviating the sparse attribute problem. Second, learning embeddings for attributed networks should consider both structure similarity and attribute similarity, where they often show partial consistency in practical networks stated in [24], [25]. There are two fundamental theories that can explain such phenomenon, topological similarity, and homogeneity. Specifically, topological similarity indicates that nodes sharing similar neighbor contexts tend to establish link connections [3], [4]. For example in Fig. 1, node v_{12} connects v_9 since they both have similar graph contexts (v_{11}). Meanwhile, homogeneity indicates that nodes with similar attributes would be prone to form the edges [26], e.g., node v_{10} is linked with node v_9 because of owning similar attributes. In fact, such properties (of node v_{10} and v_{12}) are not completely consistent in practical scenarios, and require us to quantify their different effects. Third, nodes in the network differently prefer to the properties of their structure proximity and attribute correlation. For instance, in Fig. 1, the representation of node v_{10} will depend more on its attributes, while node v_{12} is more inclined to the structural information around it. Consequently, it's a nontrivial problem to adaptively quantify different importances in the integration of structure and attribute similarity for each node. Last but not least, it's also critical to learn attributed network embeddings in a unified framework by explicitly distinguishing the differences between interactions of structure and attributes among nodes.

In our preliminary work [27], we have proposed a Sparse Attributed Network Embedding (SANE) model to learn the node embeddings by explicitly modeling the structural and attribute relationships among nodes. Specifically, we first projected each node and attribute into a low-dimensional latent space. Then we introduced a pairwise method to effectively capture the interactions between nodes and sparse attributes, and adopted the CBOW model [28] with attention mechanism to integrate the attributes into the network structure. In SANE, we have demonstrated the effectiveness by distinguishing the interactions of network structure and attributes in a shallow manner, which still has limitations in capturing the differences between their similarities.

In this article, we further conduct a comprehensive study of attributed network embedding with addressing all the above challenges from a new perspective. Specifically, we propose a novel **De**Coupled **N**etwork **E**mbedding (DCNE) model to learn node embeddings in a unified framework. First, we initialize the network by respectively projecting both nodes and attributes into a latent space with vectorial representations to enhance their generalization. Then, we introduce a novel *decoupled* process to learn different node embeddings by two modified graph convolution modules named *N-GAT* and *A-GAT* to capture network structure and attributes respectively. Especially, *N-GAT* and *A-GAT* hold partial independent parameters but share some as well to maintain consistency and distinguish the differences. Furthermore, a subsequent *fusion* process is performed to adaptively integrate the structural proximity and attribute property into a comprehensive embedding for each node. By iteratively stacking the whole "decoupled-fusion" process with multiple layers, the high-order information of node neighbors and attributes can be aggregated together for further alleviating the sparsity issue. Finally, a noise-contrastive estimation method is adopted to jointly learn DCNE in an end-to-end unified manner. In summary, the main contributions of this article are listed as follows:

- We study the problem of attributed network embedding from a new perspective, which focuses on distinguishing the complex interactions among nodes and alleviating the sparse attribute issues.
- In the technical part, we propose a network embedding model DCNE. We introduce the "decoupled-fusion" learning process in each graph layer to separately capture similarities of network structure and attributes and further generate the node embedding with regards to the inherent node characteristics.
- Furthermore, we adopt a modified mini-batch algorithm to iteratively aggregate immediate nodes and attributes to incorporate the higher-order information in a joint framework with multi-task learning.

- Extensive experiments on publicly available datasets demonstrate that DCNE could outperform state-of-the-art methods on the benchmark tasks. Moreover, several qualitative analyses further indicate our DCNE can generate more robust and representative node embeddings for the attributed network.

2 Related Work

2.1 Structure-preserved Network Embedding

One of the most popular techniques is structure-preserved network embedding, which considers the topological relationship among nodes in the network. To the best of our knowledge, DeepWalk [3] was the first work to learn network embedding via the skip-gram model [28] by truncated random walks. It assumed that nodes with similar network structures would have similar representations. On this basis, node2vec [4] improved DeepWalk with the weighted random walk, which can capture the homogeneity and structure equivalence for different networks. Then, some work capture the k-order distance relationship between nodes, such as LINE [15] and GraRep [16]. In particular, LINE defined both first-order proximity and second-order proximity of nodes, and preserved both relationships in joint learning. GraRep further considered the structure information of surrounding k-order nodes to enhance node representations. In the next generation, aiming to keep higher-order nonlinear network structures, several deep learning-based methods have been proposed. For example, Want et al. [14] proposed a semi-supervised auto-encoder model SDNE to generate node embeddings by preserving both global and local network structural information. Similarly, DNGR [17] adopted a random surfing model to capture the graph structural information to obtain the PPMI matrix [29], and utilized the stacked denoising auto-encoder for network embedding. In recent years, to make the model more robust, some state-of-the-art work introduces generative adversarial network, such as GraphGAN [30] and ANE [31]. For instance, GraphGAN [30] designed a generator to learn the underlying connectivity distribution and the discriminator to predict the probability of edge existence between node pairs. Meanwhile, motivated by unsupervised learning in many fields [32], [33], DGI [34] proposed a general unsupervised approach for graph-structured data representations, which maximized the mutual information between patch node embeddings and high-level graph representations.

2.2 Attributed Network Embedding

Besides network topology, nodes in the network are usually endowed with informative side attributes (e.g., category, content, and label), which help identify individual properties. Therefore, researchers attempt to incorporate attribute information into network embedding. Generally, they assume that nodes with similar attributes would be closer in the embedding space. From a technical perspective, related work makes sufficient efforts to fuse different types of attributes to improve the performance of original structure-preserved methods. Along this line, some work tried to incorporate attributes into the factorization models [18], [35], which plays an important role in network embedding in the earlier time. For example, TADW [18] first designed an inductive matrix complement framework to incorporate the text features of nodes into network representation, and HSCA [35] further added the regularization with network homogeneity. Then, as the deep walk based models have shown superiority, researchers have developed several enhanced models by leveraging different attributes, such as HNE [36], CENE [37] and UPP-SNE [20]. Specifically, CENE and UPP-SNE respectively exploited the benefits of text features and social profiles along with nodes to learn informative node embeddings. In addition, many auto-encoder models are explored to ensure the robustness [38], [39]. Among them, DANE [38] made the fundamental attempt, which learned both high-order embeddings of network structure and attributes, and designed a constraint objective to ensure their consistency. In recent years, considering the promising achievement of graph neural networks, several variants play the dominant role. For example, Hamilton et al. [22] proposed GraphSAGE, which iteratively generated node embeddings by sampling and aggregating features from their local neighborhood. One step further, Velivckovic et al. [23] considered the different importance of nodes by introducing a self-attention mechanism. To address the model complexity, [40] proposed a lightweight model Graph-MLP with a pure multi-layer neural network. Meanwhile, graph U-Nets [41] proposed pooling and unpooling with the graph convolution network under the auto-encoder framework for connection augmentation, to adapt the pooling and upsampling operations naturally to the graph data.

2.3 Label-enhanced Network Embedding

In addition to the unsupervised approaches above, several works differently formulate the network embedding as a supervised problem, where the node labels are incorporated to improve the correlation between embedding results and specific tasks [9], [19], [42], [43]. For example, TriDNR [42] learned node embeddings by jointly learning the three different interactions including inter-node relationship, node-word correlation as well as label-word correspondence. LANE [19] performed the spectral techniques on matrices of node&node, node&attribute, and node&label for learning comprehensive node embeddings in a common space. Though such methods have performed satisfactory results, many real-world networks suffer from the problem of limited annotations (small labeled data) since getting the high-quality labels is labor costing. Thus, many works try to leverage both labeled and unlabeled data in a semi-supervised learning framework [6], [44], [45]. For instance, Kipf et al. [6] proposed a novel variant of convolutional neural network, namely graph convolutional Network (GCN), for learning graph-structured data representations, which could be optimized in a semi-supervised learning manner. Alone the semi-supervised setting, GMNN [46] combined a conditional random field with graph neural network to jointly model the dependency of node labels and embeddings, and AM-GCN [25] proposed an adaptive multi-channel GCN to learn deep correlation information between topology and attributes. From another perspective, some work explored how to improve the consistency and robustness of GCN models for enhancing the performance in semi-supervised node classification. For instance, NodeAug [47]

studied the problem of adapting Data Augmentation techniques to strengthen the GCN by consistency training and [48] introduced Mixup methods into graph convolution learning for typical node and graph classification problems. Wang et al. [49] investigated how to design robust GCNS that are resistant to the adversarial, and further proposed ProSup [50] training strategy to supervise embeddings of all graph layers progressively toward the desired characteristics. For more detailed network embedding discussions, please refer to the comprehensive surveys [51]–[53].

In this article, we focus on the unsupervised attributed network without the supervised label information in the learning embedding stage, since it's often difficult for us to obtain the label information for most attributed networks in real-world scenarios. Besides, different from existing methods, we aim to explicitly explore the different complex interactions of both network structure and attributes among nodes via a decoupled manner, and jointly alleviate the sparse attribute issues in a unified framework.

3 Problem Definition

Let $G = (V, E, F)$ denotes a general attributed network. Specifically, the set $V = \{v_1, v_2, ..., v_{|V|}\}$ denotes all nodes in the network, and $E = \{e_{i,j}\}_{i,j=1}^{|V|}$ represents the set of edges between nodes. Moreover, each edge $e_{i,j}$ is associated with a corresponding weight $w_{ij} \geq 0$ indicating the link strength between node v_i and node v_j. Generally, the weight w_{ij} on edge $e_{i,j}$ is often defined as a binary value, and $w_{ij} = 1$ indicates that node v_i and v_j are linked by an edge e_{ij}, and vice versa. Besides, matrix $F \in \mathbb{R}^{|V| \times m}$ collects the node attributes, where each row f_i corresponds to the attribute vector of node v_i with the dimension m.

In the real world, such networks can be specified with many different types. For example, in a social network, nodes V can represent users, edges E denote their relationships (e.g., friendship or follow-up), and attributes F can collect their profiles, such as gender, location, and profession. In a citation network, nodes V can represent the papers/articles, edges E denote reference relationships with each other, and attributes F can be the titles or words appearing in the paper. Moreover, as we illustrated in the previous section, real-world networks usually exist the sparsity issue for both node structures and attributes, which means nodes in the network just establish a small number of links to each other and have extremely incomplete attributes.

Based on the terminologies described above, the goal of attributed network embedding is to learn an effective way to represent the network in a low-dimensional hidden space, where each node is assigned with a d_k dimensional vector, namely *node embedding* [3]. Specifically, the learned node embedding needs to preserve the original network property which can be evaluated by many network-based application tasks, such as node classification [54], link prediction [8], and recommendation [55], [56]. Therefore, the key to this goal is how to effectively integrate both network topological structure and node attribute information into the target node embeddings. However, most of the previous work learns the attributed network embedding in a coupled manner, which ignores the role of different interactions between nodes. To the end, we argue that it is worthwhile to distinguish the similarity of two different properties in a decoupled manner for fusing node embeddings. Along this line, we first elaborate the formal problem definition as follows:

Problem 1 (Attributed Network Embedding). *Given a general attributed network $G = (V, E, F)$, we aim to learn a low-dimensional embedding representation $U \in \mathbb{R}^{|V| \times d} (d << |V|)$ for all nodes in the network, and target these learned node embeddings should satisfy the following properties: 1) the representations of nodes with the similar network structure should be more similar; 2) the representations of nodes with similar attributes should be closer in the embedding space.*

4 DeCoupled Network Embedding

4.1 Framework

In this paper, we propose a novel **De**Coupled **N**etwork **E**mbedding (DCNE) model to learn attributed node embeddings by jointly exploiting the effects of network topological structure and attribute correlation in a decoupled manner. We illustrate the general architecture in Fig. 2, which consists of three main processes including initialization process, decoupled process, and fusion process. Specifically, we first initialize the network by respectively projecting both nodes and attributes into a latent space with vectorial representations. Then, we decouple the process of learning network embeddings by designing two modified graph convolution modules, to learn different node representations that represent the similarity of network structure and attributes respectively. Furthermore, we perform an attention network to adaptively quantify and fuse these two representations. Besides, the whole "decoupled-fusion" process is conducted with multiple layers iteratively to aggregate high-order node and attribute information. At last, we utilize a noise-contrastive estimation method to jointly learn node embeddings in an end-to-end framework. In the following, we will introduce the technical details of each component.

4.2 Initialization Process

4.2.1 Representation Initialization

Given the attributed network $G = (V, E, F)$, we first initialize both nodes and attributes of the network in a principled way. As we mentioned in Section 1, nodes in the real-world networks not only generally establish a few links to each other, but also are endowed with incomplete attributes, which leads to extreme sparsity issues of both node structures and attributes in the modeling. To alleviate this problem, we respectively project all nodes and attributes into the common vectorial space and establish two lookup embedding matrices U^0 and Z^0 for the initialization representations of both respectively:

$$U^0 \in \mathbb{R}^{|V| \times m}, \quad Z^0 \in \mathbb{R}^{m \times m}, \tag{1}$$

where $|V|$ and m are the numbers of nodes and attributes, respectively. We initialize node embedding u_i^0 of node v_i with its attributes, and the attribute embedding z_j^0 of attribute f_j with the one-hot encoding. Although we initialize the network by just specifying both node embedding and attribute embedding on their own, we can get several

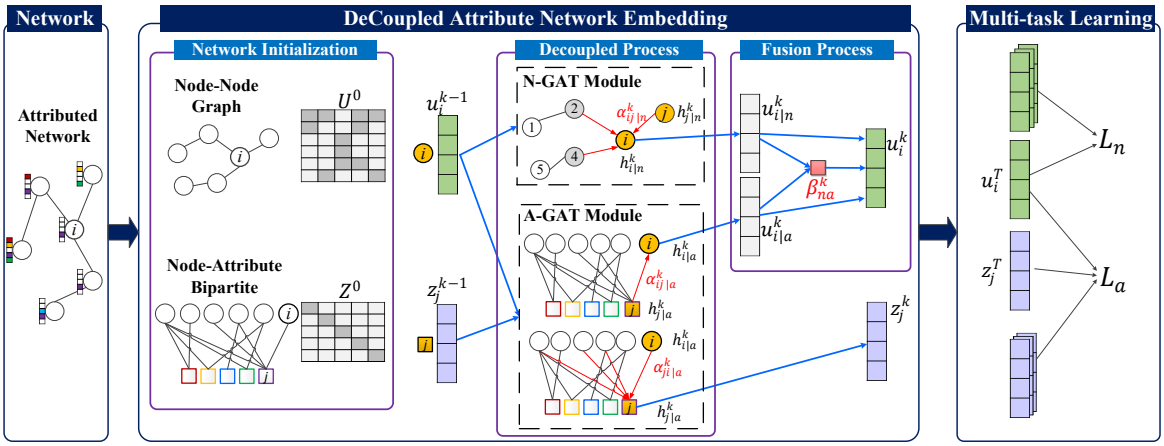


Fig. 2. DCNE: DeCoupled Network Embedding Framework.

advantages with addressing the sparsity issue. First, different from most existing work [18], [22] considering mixture node initialization, DCNE can facilitate the following decoupled network learning process. Second, we can capture and generalize the similarities of nodes and attributes, even with no interactions between them before [28].

4.2.2 Network Initialization

As shown in Fig. 1, nodes in the real-world networks would show inconsistent similarities from their structural proximity and attribute correlation perspectives. To explicitly distinguish these two properties, in this paper, we aim to learn general node embeddings by decoupling the unified network embedding process into two sub-processes including structure-level embedding and attribute-level embedding. To start up our modeling, we need first define two subgraphs from the general attributed network including *node-node graph* G_n and *node-attribute bipartite* G_a.

Specifically, given the attributed network $G = (V, E, F)$, the *node-node graph* $G_n = (V, E)$ illustrates the connections among nodes, where we just identify the nodes $V = \{v_1, v_2, ..., v_{|V|}\}$ as the vertexes, and set edges $E = \{e_{i,j}\}_{i,j=1}^{n}$ with strength weight w_{ij} as the same definition in original attributed network in Section 3. On the other hand, we also construct the *node-attribute bipartite* $G_a = (V, F)$ showing the connections between nodes and attributes. In particular, we identify both nodes V and attributes F as the vertexes, and only set the edges to indicate that node v_i endows with the attribute f_j. Therefore, in the bipartite $G_a = (V, F)$, we have two types of vertexes including nodes and attributes, and the edge link would be just established between different type nodes, and would not in the same. In the following, we present how our proposed model DCNE generates the node embedding through these two subgraphs via *decoupled process* and *fusion process* iteratively.

4.3 Decoupled Process

In the decoupled process, DCNE aims to generate two different node representations including *node2structure embedding* and *node2attribute embedding*, over the *node-node graph* and *node-attribute bipartite*, which can capture both structure-level and attribute-level similarities of nodes in the attributed network. Concretely, we propose two modified graph convolution modules, namely *N-GAT* and *A-GAT*, in the decoupled processes, following the general graph neural network with message-passing-receiving mechanisms [22].

4.3.1 Capturing Network Structure Similarity

For *N-GAT* module, our goal is to generate the *node2structure embedding* of each node through the *node-node graph* to capture the network topology. Our basic assumption is that nodes with similar structure should be closer with respect to their learned embeddings in the latent space, please note that in *N-GAT*, we just exploit the property of node-node connections while ignoring the effect of node-attribute correlation temporarily. Mathematically, given a certain node v_i in the *node-node graph* $G_n = (V, E)$, *N-GAT* generates its *node2structure embedding* $u^k_{i|n}$ at layer k with the following message-passing-receiving mechanism as:

$$(v \to e)_{N\text{-}GAT} : h^k_{i|n} = \left[W^{k-1}_c, W^{k-1}_n\right] u^{k-1}_i + b^{k-1}_n, \quad (2)$$

$$(e \to v)_{N\text{-}GAT} : u^k_{i|n} = \sigma\Big(\sum\nolimits_{j \in \mathcal{N}(i)} \alpha^k_{ij|n} h^k_{j|n}\Big). \quad (3)$$

Specifically, in Eq. (2), the message passing operation $(v \to e)_{N\text{-}GAT}$ allows node v_i sends its structural message $h^k_{i|n}$ to its adjacent nodes from its node embedding u^{k-1}_i at layer $k-1$. Here, $\left[W^{k-1}_c, W^{k-1}_n\right]$ and b^{k-1}_n are the trainable weight and bias parameters to transform the node messages, and notation $[,]$ denotes the matrix concatenation operation. It's worth noting that the weight parameters in *N-GAT* consist of two parts, i.e., the exclusive one W^{k-1}_n for itself and the shared one W^{k-1}_c that also exists in the next graph module *A-GAT*. Based on these two types of parameters, the passing message of each node $h^k_{i|n}$ can naturally preserve the consistency between structure property and attribute correlation, and meanwhile distinguish the special structural effect without attributes in the network.

Then, in Eq. (3), the message receiving operation $(v \to e)_{|n}$ aggregates all messages from neighbors $\mathcal{N}(i)$ of node v_i to update the *node2structure embeddings* $u^k_{i|n}$ at layer k. Here, $\sigma(x)$ is the activation function $\mathrm{ReLU}(x) = \max(0, x)$, and $\mathcal{N}(i)$ is the neighbor set of node i (including itself) with fixed-size samples, which is uniformly sampled to improve the computational efficiency for model training.

In addition, considering the common phenomenon that different neighbors would produce inconsistent influences on node v_i. Inspired by [23], we propose an attention network to model the effect from structural passing message $h^k_{j|n}$ of neighbor $v_j \in \mathcal{N}(i)$ by the weight function $\alpha^k_{ij|n}$:

$$\alpha_{ij|n}^{k} = \frac{\exp\Big(LeakyReLU\big(p_n^{k^T}[h_{j|n}^{k}, h_{i|n}^{k}]\big)\Big)}{\sum_{t\in\mathcal{N}(i)}\exp\Big(LeakyReLU\big(p_n^{k^T}[h_{t|n}^{k}, h_{i|n}^{k}]\big)\Big)}, \quad (4)$$

where $p_n^{k^T}$ is the transposition weight parameters corresponding to the attention network at layer k. All the calculated attention scores with respect to node v_i are normalized by the softmax function ($\exp(\cdot)$). By Eq. (4), we can assign greater weights to the neighbors that are more similar to v_i, in order to make them closer in the embedding space.

Moreover, at layer 0, we initialize the node embedding u_i^0 by looking up the initialized node representation matrix U^0 in Eq. (1). Through Eq. (2) and Eq. (3), *N-GAT* generates the *node2structure embedding* $u_{i|n}^{k}$ of each node at layer k with capturing the similarity of network structure.

4.3.2 Capturing Node Attribute Similarity

Then, in the *A-GAT* module, our goal turns to generate the *node2attribute embedding* of each node through the *node-attribute bipartite* to capture the attribute correlation between nodes in the network. Concretely, we assume that nodes with similar attributes should be closer with respect to their learned embeddings in the latent space, Since the *node-attribute bipartite* establishes the links by connecting two type vertexes including nodes and attributes, which is different from *node-node graph*, the module *A-GAT* consists of two consecutive steps with different directions, i.e., one step to learn the *node2attribute embedding* following the direction from attributes to nodes, and the other step to update the *attribute embedding* from nodes to attributes direction.

Mathematically, in the first step, given a certain node vertex v_i in the *node-attribute bipartite* $G_a = (V, F)$, *A-GAT* generates its *node2structure embedding* $u_{i|a}^{k}$ at layer k with the similar message-passing-receiving architecture as Eq. (2) and Eq. (3), which is defined as follows:

$$(v \to e)_{\textit{A-GAT}} : h_{j|a}^{k} = [W_c^{k-1}, W_a^{k-1}] z_j^{k-1} + b_a^{k-1}, \quad (5)$$

$$(e \to v)_{\textit{A-GAT}} : u_{i|a}^{k} = \sigma\Big(\sum\nolimits_{j\in\mathcal{N}(i)} \alpha_{ij|a}^{k} h_{j|a}^{k}\Big). \quad (6)$$

where $[W_c^{k-1}, W_a^{k-1}]$ and b_a^{k-1} are the trainable network parameters. However, note that both operations in *A-GAT* exist some differences as follows. First, it can only generate the attribute message $h_{j|a}^{k}$ to its adjacent node vertexes from the neighbor attribute embedding z_j^{k-1} at layer $k-1$ in the message passing operation (Eq. (5)). Second, it can only aggregate all messages from neighbor attributes $\mathcal{N}(i)$ of node v_i ($\mathcal{N}(i)$ only contains the connected attribute vertexes with respect to node v_i) to update the *node2attribute embeddings* $u_{i|a}^{k}$ at layer k in the message receiving operation (Eq. (6)). Similarly, the weight parameter $\big[W_c^{k-1}, W_a^{k-1}\big]$ in *A-GAT* also consist of two parts, i.e., the exclusive one W_a^{k-1} distinguishing the special attribute effects, and the same one W_c^{k-1} shared with *N-GAT* preserving common consistency. Moreover, we still consider the different effects on node v_i from its attribute neighbor set $\mathcal{N}(i)$ with fixed-size samples according to the weight score $\alpha_{ij|a}^{k}$ by a similar attention network as follows:

$$\alpha_{ij|a}^{k} = \frac{\exp\Big(LeakyReLU\big(p_a^{k^T}[h_{j|a}^{k}, h_{i|a}^{k}]\big)\Big)}{\sum_{t\in\mathcal{N}(i)}\exp\Big(LeakyReLU\big(p_a^{k^T}[h_{t|a}^{k}, h_{i|a}^{k}]\big)\Big)}. \quad (7)$$

In the second step, given a certain attribute vertex a_j in the *node-attribute bipartite* G_a, *A-GAT* produces its updated *attribute embedding* z_j^{k} at layer k by an additional similar architecture, which is defined as follows:

$$(v \to e)_{\textit{A-GAT}} : h_{i|a}^{k} = \big[W_c^{k-1}; W_a^{k-1}\big] u_i^{k-1} + b_a^{k-1}, \quad (8)$$

$$(e \to v)_{\textit{A-GAT}} : z_j^{k} = \sigma\Big(\sum\nolimits_{i\in\mathcal{N}(j)} \alpha_{ji|a}^{k} h_{i|a}^{k}\Big), \quad (9)$$

where u_i^{k-1} is the *node embedding* of neighbor node v_i at last layer $k-1$, and $\mathcal{N}(j)$ consists of the sampled fixed-size neighbor nodes holding with attribute a_j.

4.4 Fusion Process

After the decoupled process in DCNE, we decompose the general network embedding to obtain two separate presentations of a certain node including *node2structure embedding* and *node2attribute embedding*. In the fusion process, we address the problem of how to generate the comprehensive node embedding by aggregating them together to make it satisfy both properties of network structure and attribute correlation simultaneously at the current layer. In Fig. 1, we notice that nodes in the attributed network differently prefer to such both properties. For example, some nodes like v_{12} and v_9 are more dependent on their topological position since they are linked directly, while others like v_{10} and v_9 have the same attributes to show the similar attribute correlation. Therefore, it is necessary to distinguish and quantify the dominance of *node2structure embedding* and *node2attribute embedding* in the fusion process. Specifically, we combine them at layer k to update *node embedding* u_i^{k} for node v_i in general as follows:

$$u_i^{k} = \beta_i^{k} u_{i|n}^{k} + (1 - \beta_i^{k}) u_{i|a}^{k}, \quad (10)$$

where β_i^{k} is the weight score that balances the *node2structure embedding* $u_{i|n}^{k}$ and *node2attribute embedding* $u_{i|a}^{k}$ for node v_i, which is implemented by an attention network as:

$$\beta_i^{k} = h^{k^T} ReLU\big(W_\beta^{k}[u_{i|n}^{k}, u_{i|a}^{k}]\big), \quad (11)$$

where h^{k^T} and W_β^{k} are the trainable parameters at layer k.

Therefore, each node can adaptively make a trade-off between its structural and attribute preferences, so as to facilitate the embedding process in the next layer $k+1$.

Through our decoupled and fusion process in one graph layer, we can update the *node embedding* u_i^{k} of each node and the *attribute embedding* z_j^{k} of each attribute in an attributed network, where the adjacent structure and attribute information of immediate neighborhood around nodes can be aggregated. Then, we continuously stack K these graph layers with multiple such "decoupled-fusion" processes to capture the propagation of surrounding K-order structure and attribute information. Specifically, different from the intuitive method that simply fusion once at last with the output node embeddings by respectively stacking both *N-GAT* and *A-GAT* with multi-round (message-passing-receiving) operations, our DCNE with iterative "decoupled-fusion" graph layers could gain two main advantages for network embedding as follows. First, it could characterize the complex high-order interactions between nodes and attribute during the propagation iteratively. Second, it could enhance

the representation ability of nodes and attributes by alleviating the sparsity problem gradually, since our modeling of nodes and attributes are not totally independent but related with each other to some extend. Moreover, we denote the final output of *node embedding* and *attribute embedding* at last graph layer K as $U \in R^{|V|\times d}$ and $Z \in R^{|V|\times d}$ for the concise notations in the subsequent technical parts.

4.5 Model Learning

4.5.1 Joint Learning in a unified framework

DCNE introduces the specific modeling for both nodes and attributes in an attributed network. Therefore, we propose a unified framework to jointly learn two tasks including structure similarity and attribute correlation. Specifically, for the former task, we utilize the widely used graph-based loss criterion [22] with negative sampling [27] as:

$$\mathcal{L}_n = \log \sigma(u_i^T u_j) + \sum\nolimits_{t=1}^{neg} E_{v_t \sim P_n(v)} \log \sigma(-u_i^T u_t), \quad (12)$$

where u_j is the *node embedding* of the positive node v_j that co-occurs near target node v_i on a fixed-length random walk, u_t denotes the *node embedding* of sampled node v_t for target v_i from the noisy distribution $P_n(v)$ [57] with total number neg. $\sigma(x)$ is the sigmoid activation function. Specifically, we calculate the similarity score of both nodes by inner product with their embeddings. By optimizing Eq.(12), we can make nodes with similar structure closer in the embedding space, and vice versa, so as to preserve the structural property between nodes in the network.

For the latter task, we also define the similar objective function to learn the attribute correlation as:

$$\mathcal{L}_a = \log \sigma(u_i^T z_j) + \sum\nolimits_{t=1}^{neg} E_{f_t \sim P_a(f)} \log \sigma(-u_i^T z_t). \quad (13)$$

where z_j denotes the embedding of positive attribute f_j that node v_i owns, and z_t is the negative one, sampled from attribute noisy distribution $P_a(f)$. By Eq. (13), we could encourage nodes with similar attributes to be closer, while forcing the others with disparate attributes to be distinct.

Moreover, considering that both learning tasks above are often not completely independent but exist partially relevance in many real-world attributed network, the joint learning objective function of DCNE could be defined as the following multi-task learning framework combined with Eq. (12) and Eq. (13):

$$\mathcal{L} = \mathcal{L}_n + \mathcal{L}_a + \lambda ||\Theta||^2, \quad (14)$$

where Θ denotes all trainable parameters in DCNE, which are regularized with L_2 norm to prevent overfitting. λ is a balancing hyper-parameter. Please recall that DCNE shares part of weight parameters (W_c) in *N-GAT* and *A-GAT* (Eq. (2), Eq. (5) and Eq. (8)), which can improve the model robustness in the learning process. Moreover, since we optimize the node embeddings in both tasks, which can capture the correlation between network structure and attributes, and meanwhile strengthen the ability of model learning by distinguishing their inconsistency.

4.5.2 Model Optimization with Mini-Batch Training

Considering each part of our DCNE is differentiable, in most cases, we could utilize the Stochastic Gradient Descent (SGD) and Adam algorithms [58] to optimize the joint objective function Eq.(14) in an end-to-end trainable manner.

TABLE 1
Statistics of the datasets

Datasets	Nodes#	Links#	Attributes#	Labels#
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
DBLP	60,744	52,890	3,799	4
Facebook	4,039	88,234	1,406	–
BlogCatalog	5,196	171,743	8,189	–

Moreover, since there usually exist large-scale attributed networks in the real world, the computational complexity of generating *node embeddings* for all nodes at the same time is very high if we directly use the conventional algorithm (recall Section 4.3 and 4.4), which leads to unsatisfactory storage and time cost. Therefore, we introduce an adapted mini-batch training strategy to ensure the efficiency of our DCNE. Specifically, for nodes in a batch of data, we first sample the required adjacent nodes and attributes in the last layer K, where all samples can be treated as nodes to be computed in the previous layer $K-1$. Then we repeat this process recursively back to the initial layer 0, and in the end, we construct the subgraph with required nodes and attributes in the batch. Therefore, we can only generate the node embeddings from samples in each graph layer instead of computing the whole graph, which effectively reduces the storage requirement and speeds up DCNE for the large-scale attributed networks. It worth mentioning that different from the typical solutions that only sample multi-hops neighborhood nodes in most GNN models [12], [22], our sampling procedure would consider both high-order nodes and attributes, which characterizes their complex interactions more effectively. Consequently, more surrounding information can be utilized to alleviate the sparsity problem.

4.5.3 Time Complexity

In our mini-batch training setting, the time complexity of DCNE can be fixed at $\mathcal{O}(CB\Pi_{k=1}^{K}(|\mathcal{N}_n^k| + |\mathcal{N}_a^k|))$, where C is the number of negative samples, B is the number of nodes in each batch, and $|\mathcal{N}_n^k|$ and $|\mathcal{N}_a^k|$ are the number of sampling neighbors of *N-GAT* and *A-GAT* in each graph layer respectively. Compared with the complexity of ordinary graph neural network $\mathcal{O}(CB\Pi_{k=1}^{K}(|\mathcal{N}_n^k|)$, the complexity of DCNE is several times that of it, mainly in the aggregation of attribute information $|\mathcal{N}_a^k|$ in each layer. In general, we often set the layer number K=2 to make the trade-off between computational cost and satisfactory performance based on previous work [6] and experimental results. Such time complexity is acceptable for most large-scale attributed networks, which is proportional to the number of sampling neighbors, negative samples, and batch size.

5 Experiments

5.1 Experimental Dataset and Setup

5.1.1 Datasets

We select five public datasets in our experiments, which consist of two types of attributed networks including citation networks (Cora, Citeseer, DBLP) and social networks (Facebook, BlogCatalog). For the preprocessing of datasets, we regard them as undirected networks without loss of generality (Note that our model can adapt both directed and

undirected networks). Moreover, we remove the nodes that are not connected in the network to ensure the reliability of experimental results. Table 1 shows the statistics of all datasets. We list the details of them as follows:

Cora [18] contains 2,708 machine learning papers from 7 categories and 5,429 citation links. Each paper is represented as a binary vector of 1,433 dimensions indicating the presence of words, which can be regarded as attributes.

Citeseer [18] contains 3,312 publications of 6 classes and 4,732 citation links. Each document is represented as a binary vector of 3,703 dimensions, which are attributes.

DBLP [42] contains 60,744 papers and 52,890 citation links. Papers come from 4 research areas including database, data mining, artificial intelligent and computer vision, which can be treated as the ground-truth labels. We use the title of each article as node attributes, represented by a $3,799$ dimensional TF-IDF vector.

Facebook [39] is an online social network, which contains $4,039$ nodes and $88,234$ friendship links. The nodes' attributes are social users' anonymous profiles, where are encoded as a 1406 dimensional binary feature vector.

BlogCatalog [39] is constructed with $171,743$ social links between $5,196$ bloggers on BlogCatalog website, where node attributes are generated by the description keywords.

5.1.2 Baselines

We compare our DCNE model with several representative state-of-the-art network embedding algorithms to demonstrate its' effectiveness on different tasks. First, we choose the models that only explore the effect of network structure including Node2vec, LINE, and SDNE. Second, we introduce the ones that utilize both structures and attribute information including TADW, UPP-SNE, GAT, DANE, and CAN. We also introduce an intuitive model Attri and take our preliminary SANE model [27] as baselines. Please note that we do not introduce the (semi-)supervised methods since DCNE follows the unsupervised manner without incorporating node label information. The details of them are illustrated as follows:

- **Attri**: Considering the dimensions of node attributes can be very large, we directly leverage Singular Value Decomposition (SVD) [18] to reduce the original dimensions to 200, as node embeddings.
- **Node2vec** [4]: Node2vec utilizes the skip-gram model to learn node embeddings with a biased truncated random walks to explore diverse neighbors.
- **LINE** [15]: It learns the network embedding by preserving the *first-order proximity* or *second-order proximity* of the network structure separately. We utilize the best of them as the final baseline.
- **SDNE** [14]: It proposes an auto-encoder to obtain node embeddings by preserving the global and local network structure information.
- **Node2vec+Attri**: It is an improved version of original Node2vec model by incorporating the attribute feature vectors in Attri.
- **LINE+Attri**: It is an improved version of LINE with incorporating Attri's attribute feature vectors.
- **TADW** [18]: TADW integrates the network structure and rich text information for learning the fused node embeddings based on inductive matrix completion.
- **UPP-SNE** [59]: It constructs node embedding via a non-linear mapping from attributes and adopt skip-gram model to jointly learn node embeddings, which achieves improvement in sparse attribute situation.
- **DANE** [38]: It utilizes the auto-encoder to learn both high-order embeddings of network structure and attributes, and combine them to ensure the consistency.
- **CAN** [39]: It proposes a variational auto-encoder that embeds nodes and attributes in the same semantic space with Gaussian distributions.
- **GAT** [23]: On the basis of GraphSage [22], GAT utilizes the attention mechanism to capture the different effects between nodes, in order to adaptively aggregate the features from nodes' neighbors.
- **SANE** [27]: We previously propose SANE, which adopts a pairwise method to capture the interactions between nodes and attributes. The model is learned by CBOW model with attention mechanism.

5.1.3 Parameter Setting

in the experiments, there are several hyper-parameters in our DCNE model that should be specified. First, we set the number of graph layer in DCNE as K=2 (recall Section 4.4), where the dimensions of each layer (i.e., *node embedding*) are defined as [125, 100]. We make the grid search for the numbers of sampled neighbors in *N-GAT* and *A-GAT* from the set [50, 40,..., 10]. Second, in the training stage, we initialize DCNE parameters with Gaussian Distribution (with a mean of 0 and a standard deviation of 0.01). Then, we also specify the weight parameters in *N-GAT* and *A-GAT* as the same at the beginning (recall Section 4.3), in order to obtain a better initialization point. Then, we set the hyper-parameter λ and negative sample sizes in the objective function (Eq. (14)) as 0.0001 and 5, respectively. In addition, we set the initial learning rate of Adam as 0.003 and the mini-batch size as 64. Third, as for baseline settings, we set their parameters as the same according to their original papers and tune them to be optimal. Moreover, to make a fair comparison, we set all the node embedding dimension size of compared baselines as 100 with the same as ours. Specifically, we implement all models by Pytorch platform and run all the experiments independently on a Linux server with four 2.0GHZ Intel Xeon E5-2620 CPUs and a Tesla K80 GPU.

5.2 Performance Comparison

As mentioned above, network embedding generally follows an unsupervised fashion. The results can benefit many downstream tasks, where the most representative ones include multi-label node classification and link prediction. We will evaluate the performance of DCNE by comparing with baselines on such two benchmark tasks.

5.2.1 Multi-label Node Classification Task

Multi-label node classification is a typical task to evaluate network embedding performance. Specifically, we would regard the output node representations by network embedding algorithms as the node features, and then such features can be applied into the subsequent trainable classifiers to inference the node labels. We select Core, Citeseer, and

TABLE 2
Results of multi-label node classification on **Cora** dataset using Micro-F1 metric

Training ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Node2vec	58.48	65.55	67.95	70.12	71.43	72.21	73.06	74.41	75.07	75.25
LINE	41.11	45.96	50.06	52.32	54.46	56.74	57.31	59.62	59.81	60.47
SDNE	47.98	59.38	61.56	65.70	66.30	67.63	68.31	69.02	70.26	70.37
Node2vec + Attri	60.31	67.96	72.26	74.16	74.88	75.44	75.73	76.55	76.98	77.43
LINE + Attri	43.98	53.50	59.07	61.99	64.32	66.04	67.32	68.90	69.48	71.20
TADW	55.72	64.95	70.64	73.53	75.09	76.72	78.07	78.67	79.49	80.40
UPP-SNE	62.78	72.59	75.05	77.66	78.23	78.63	79.38	79.40	80.04	80.27
DANE	61.48	69.13	73.50	75.75	76.88	77.96	78.08	78.67	79.45	79.61
CAN	61.94	68.54	73.41	75.64	76.13	76.90	77.34	77.86	78.19	79.45
GAT	62.39	68.78	73.83	74.87	75.82	76.97	77.68	77.79	78.46	79.59
SANE	70.55	78.78	80.23	82.31	82.50	82.80	83.60	83.64	84.01	84.47
DCNE	**73.59**	**79.09**	**81.27**	**82.88**	**82.98**	**83.27**	**83.95**	**84.15**	**84.38**	**84.86**

TABLE 3
Results of multi-label node classification on **Citeseer** dataset Micro-F1 metric.

Training ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Node2vec	40.51	44.05	47.71	49.22	50.48	51.43	52.00	52.54	52.94	53.15
LINE	29.53	34.16	35.74	37.32	38.92	40.33	40.41	41.45	42.50	43.02
SDNE	32.54	36.74	38.62	39.94	41.26	43.89	44.63	44.78	45.74	46.32
Node2vec + Attri	48.02	57.62	60.52	62.27	63.09	64.22	65.16	65.30	66.15	66.36
LINE + Attri	43.28	51.28	56.04	59.26	59.84	61.77	62.46	63.08	63.25	64.07
TADW	46.65	56.38	61.38	64.68	66.87	67.28	67.39	68.65	69.78	70.06
UPP-SNE	55.62	64.40	66.24	66.91	66.98	68.07	68.42	68.44	68.59	68.85
DANE	54.89	60.51	63.52	64.27	65.24	66.09	66.14	66.97	68.15	68.69
CAN	57.64	62.41	64.85	65.90	66.46	67.30	67.70	68.34	68.62	68.74
GAT	60.34	65.87	66.55	68.51	68.93	69.46	69.83	70.15	70.33	70.41
SANE	66.56	70.02	70.96	71.68	71.70	72.20	72.35	72.97	72.98	73.27
DCNE	**67.31**	**70.55**	**71.27**	**71.96**	**72.07**	**72.32**	**72.67**	**73.26**	**73.49**	**73.62**

TABLE 4
Results of multi-label node classification on **DBLP** dataset Micro-F1 metric.

Training ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Node2vec	73.12	76.61	77.96	78.65	79.06	79.44	79.65	79.78	79.81	79.85
LINE	65.44	69.04	70.96	72.36	73.12	73.26	73.94	74.20	74.46	74.61
SDNE	63.04	63.31	64.88	65.36	66.82	65.03	68.04	68.71	71.26	71.86
Node2vec + Attri	76.14	78.01	78.56	78.92	79.51	79.79	80.04	80.15	80.39	80.50
LINE + Attri	66.10	71.35	73.69	74.89	76.07	76.78	77.14	77.91	78.02	78.33
TADW	72.42	77.07	79.18	80.35	80.44	80.96	81.23	81.57	81.69	81.79
UPP-SNE	77.13	78.17	78.42	79.30	79.59	79.98	80.05	80.18	80.52	80.62
DANE	76.71	77.72	78.25	78.76	79.07	79.37	79.51	79.61	79.66	79.75
CAN	76.58	77.47	78.16	78.23	78.68	79.23	79.54	79.56	79.69	80.13
GAT	77.85	78.08	79.68	80.75	81.21	81.56	81.87	82.02	82.21	82.27
SANE	80.87	81.94	82.46	82.84	82.95	83.01	83.19	83.21	83.22	83.39
DCNE	**81.46**	**82.25**	**82.95**	**83.10**	**83.13**	**83.22**	**83.31**	**83.45**	**83.59**	**83.81**

DBLP datasets for this experiment since they supply the labels for us. To start up the experiments, we first implement the linear SVM model as the classifier, which corresponds to many previous work [4], [27], in order to reduce the inconsistency of classifier performance. Then, for data partition, we randomly select the labeled nodes at different ratios as the training sets and regard the remaining as the test set. Specifically, the training ratios are varied from 1% to 10% by gradually increasing 1%. We adopt the Micro-F1 to evaluate the performance. (We also conduct the experiments with Macro-F1 and accuracy metrics, but omit it for brevity since we observe they have the same performance trend with Micro-F1). We independently repeat the experiments for all models 10 times, and report the average results using the Micro-F1 metric in Table 2, Table 3 and Table 4, where the best results are especially highlighted in **boldfaced**.

From the results, we can conclude several observations as follows. First, our DCNE model consistently achieves significant performance improvements on all datasets, especially facing the restricted situations that the training data are extremely fewer. It proves that DCNE can generate more effective and robust node embeddings than other models for node classification. Second, the fusion models considering both network structure and attributes(i.e., Node2vec+Attri, LINE+Attri, TADW, UPP-SNE, DANE, CAN, and GAT) outperform the methods with only structure information (i.e., Node2vec, LINE, and SDNE), which illustrates the node attributes are essential and effective for learning representative node embeddings in practice. Third, by comparing with the competitive methods GAT and SANE, DCNE still presents better performances, which demonstrates the effectiveness of our proposed joint learning framework with the novel "decoupled-fusion" architecture for network embedding. Consequently, DCNE can alleviate the similarity inconsistency issue between network structure and attribute to generate more representative node embeddings .

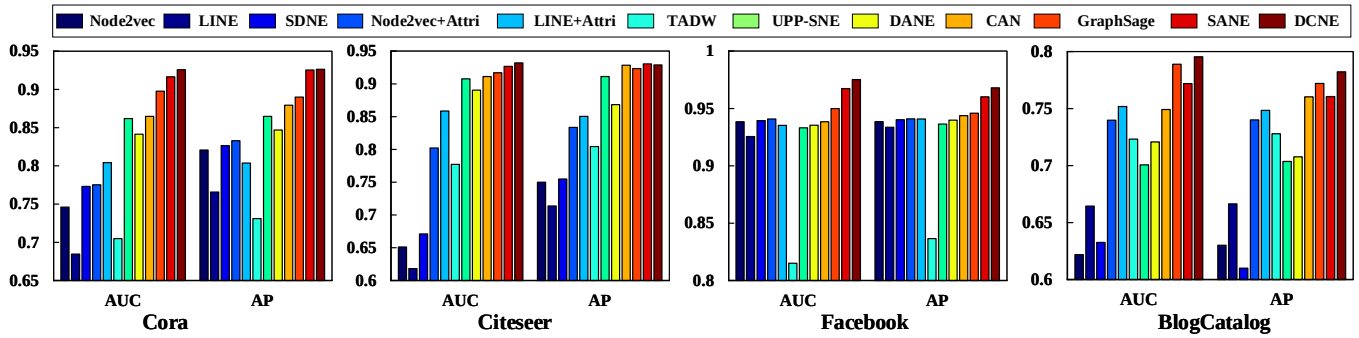
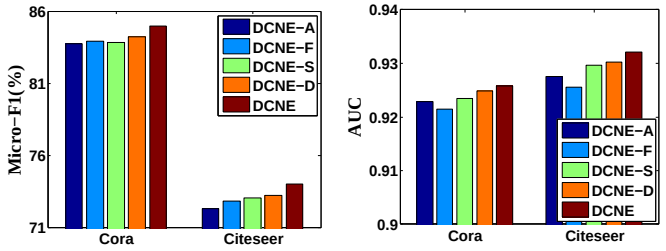


Fig. 3. Performance results of all models on link prediction task.



(a) Node Classification (b) Link prediction

Fig. 4. Ablation Study on Node Classification and Link Prediction Task.

5.2.2 Link Prediction Task

Link prediction is another typical task for evaluating network embedding methods, which aims to predict whether there exists a link between two nodes in the network. We select two citation networks (Cora, Citeseer) and two social networks (Facebook, BlogCatalog) for this experiment. To start up, in each dataset, we randomly remove 30% edges of the network as test sets, and preserve the connectivity between the rest of the nodes to learn node embeddings on the remaining sub-graph. Then, in the testing, we regard the removed edges as the positive links and randomly sample an equal number of non-existing links as negative instances. Our goal is to rank the positive links as the top before negative ones. In this task, we select the area under curve (AUC) and average precision (AP) as the metrics for evaluation. Similarly, we also run all experiments 10 times, and report the average performances to guarantee fairness.

Fig. 3 shows the overall results on this task. First, our DCNE achieves the satisfactory results of link prediction among all methods, which shows that node embeddings learned by DCNE can better maintain network structure and infer link relation between nodes. Second, a different phenomenon, which should be pointed out, is that some fusion models do not always perform better than methods only considering structure information. For example, Node2vec+Attri and LINE+Attri, incorporating attributes, achieve better results than basic ones Node2vec and LINE on Cora and Citeseer datasets, but fail on Facebook. This illustrates that just combing network structure and attributes in a simple way is not capable of characterizing the complex interactions between them for different kinds of networks since they may be coupled together as we mentioned before. Third, comparing with the competitive GAT and SANE, we find that DCNE consistently outperforms others on all datasets. The results demonstrate that DCNE, with the novel "decoupled-fusion" process, can improve the node embedding abilities for different types of networks by adaptively integrating structure and attribute information together.

5.3 Ablation Study

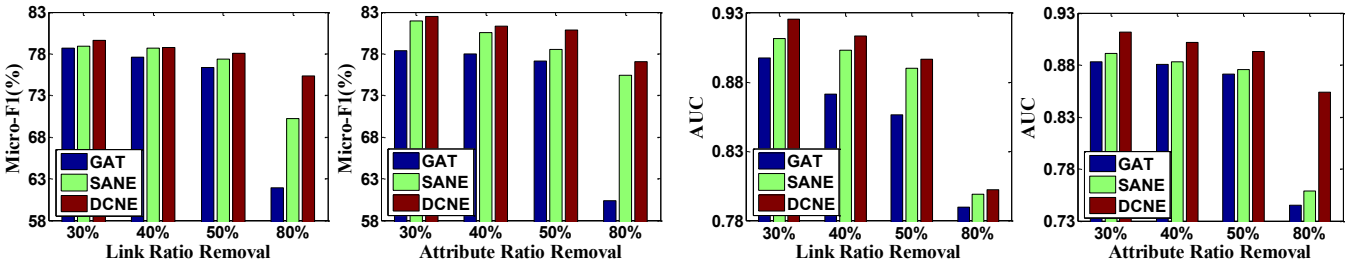
Furthermore, we verify the effectiveness of each part in DCNE by ablation study in this part. Specifically, we introduce four simplified versions of DCNE, where the details are: 1). DCNE-A: It's the reduction version that removes the attention mechanism of *N-GAT* and *A-GAT* defined in Eq.(4) and Eq.(7); 2). DCNE-F: It utilizes the average strategy to combine the *node2structure embedding* and *node2attribute embedding* instead of the adaptive attention function (Eq.(11)) in fusion process; 3). DCNE-S: It regards the weight parameters of *N-GAT* and *A-GAT* in each graph layer as the same, i.e., we remove the parameters W_n^{k-1} and W_a^{k-1} in DCNE. 4). DCNE-D: Contrary to DCNE-S variant, it considers the weight parameters completely different, i.e., we remove the W_c^{k-1} parameters in DCNE. Here, we adopt the same experimental settings as before for all methods on both node classification and link prediction tasks on Cora and Citeseer datasets. For better illustration, we report the results of the node classification task at 10% training ratio and link prediction task of 70% training edges.

Fig. 4 shows the ablation results. We can observe DCNE achieves the best performance of two tasks on both datasets, which indicates the effectiveness of each component in DCNE for network embedding. Specifically, first, comparing with DCNE-A, DCNE consistently performs better results, which demonstrates that it's essential to utilize the attention mechanism to consider the different importance of adjacent nodes and attributes. Second, DCNE generates more satisfactory results than the variant DCNE-F, which verifies the proposed attentional fusion function can adaptively incorporate network structure with node attributes by quantifying their properties. Third, Compared with variants DCNE-S and DCNE-D, DCNE, sharing parts of the same weight parameters in *N-GAT* and *A-GAT*, can generate better performance. It indicates the proposed decoupling process can not only preserves the consistency between structural property and attribute correlation, but also distinguishes the special exclusive effects of them in the embedding fusion process, so as to learn flexible and representative node embeddings.

5.4 Qualitative Analysis

5.4.1 Sparsity Analysis

As we mentioned before, our DCNE has a superior ability of learning node embeddings by addressing the sparse node attribute issues. To this end, we conduct the qualitative experiments to discuss the sparsity sensitivity on both node classification and link prediction tasks on Cora dataset (as the representative one). Specifically, we first remove parts of



(a) Node Classification (b) Link Prediction

Fig. 5. Sparsity Results of DCNE compared with SANE and GAT, with removing different link ratio and attribute ratio on both tasks.

(a) Node2vec (b) LINE (c) GAT (d) SANE (e) DCNE

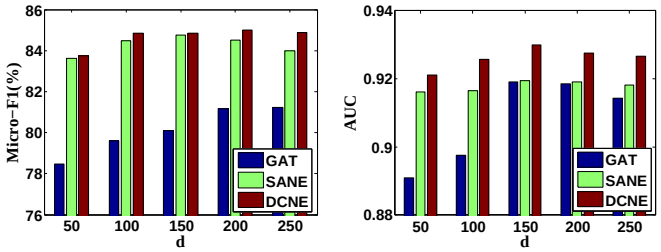
Fig. 6. Visualization of network representations learned by different models on the Cora dataset. Points with different colors indicate the nodes with the corresponding labels. The text marked in figures corresponds to the label name.

network links and node attributes at different ratios respectively (i.e., 30%, 40%, 50%, 80%), and utilize the remaining attributed network for the evaluation. In particular, for the link prediction task, we make a little difference compared with the settings above that we regard the links as the test edges and consider all the rest as for training. Moreover, we introduce the competitive GAT and SANE for comparisons, and report the results in Fig. 5.

From these figures, we can find the performance of GAT on both two evaluation tasks decreases more dramatically than SANE and DCNE as the ratios of removed links and attributes increase. This indicates that GAT is more sensitive to the sparsity of attributed networks, and cannot handle the extremely sparse situation (of removing 80% links or attributes). Meanwhile, DCNE and SANE achieve more stable results than GAT, with the help of respectively projecting both nodes and attributes into different vectors to improve the generalization ability. Furthermore, DCNE consistently performs the best on both tasks, especially in extreme sparsity issues (80% removal situation), because it considers the complex interactions between nodes and attributes by "decoupled-fusion" processes, and further utilize the high-order nodes and attributes in joint learning, which can effectively alleviate the sparsity issues.

5.4.2 Embedding Visualization

Here, we demonstrate the representation ability of DCNE by visualizing its embedding results, so that we can intuitively observe the relevance between nodes [60]. Specifically, we randomly sample 150 nodes of each different label from Cora dataset, and then project their node embeddings, learned from DCNE, into a two-dimensional space by the widely used visualization algorithm t-SNE [61]. We also introduce the results including Node2vec, LINE, GAT, and SANE for comparison. Besides, we mark the nodes with their labels using different colors, and present the corresponding name in the center of each label to make the visualization more intuitive to be observed. Fig. 5.2.2 shows



(a) Node Classification (b) Link Prediction

Fig. 7. Results of DCNE with different embedding size on both tasks.

all the embedding visualizations. First, the performance of Node2vec and LINE is not very satisfactory, since nodes are all mixed together. This is because both of them only consider network structure for learning embedding without attribute effects, and therefore, all the information is coupled. Second, GAT and SANE achieve better visualization results than Node2vec and LINE, where nodes with the same labels are easier to be grouped together. It proves that incorporating the attribute information into the network structure can effectively enhance the representative ability of learned node embeddings. Third, DCNE performs the best by comparing with others, especially with the most competitive SANE. The distances between nodes with the same labels are closer to form clusters, while nodes with different labels are even farther away from each other. We can conclude that DCNE can generate more effective and discriminative node embeddings, which explains the reason that DCNE achieves better performance on node classification and link prediction tasks from another perspective.

5.5 Parameter Analysis

5.5.1 The Impact of Embedding Size d

The node embedding size d plays an important role in the model, since it greatly affects the representation ability of learned node embeddings. In this experiment, we select DCNE, SANE, and GAT for model comparison with different dimension size settings in the set $\{50, 100, 150, 200, 250\}$.

TABLE 5
Results of DCNE with different graph layer number on two tasks.

Layer Number	Node Classification	Link Prediction
K=1	0.8417	0.8642
K=2	0.8486	0.9258
K=3	0.8343	0.8535
K >3	-	-

The comparison results are shown in Fig. 7. There are several observations. First, the performances of all models raise as the dimension d increases at the beginning, which means node embeddings with more dimensions can preserve more information of the attributed network. However, their performances will decrease when d surpasses about 200, since node embedding with too large size would introduce more sparsity and noises to reduce the performance. Moreover, DCNE performs more stable results compared with others in different settings, which demonstrates DCNE can generate more representative and robust node embeddings.

5.5.2 The Impact of Sampled Size $\mathcal{N}_i$

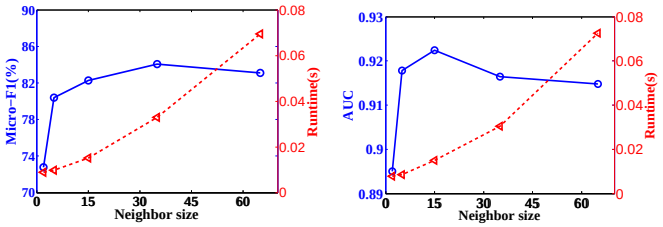
The sampled neighbor size $\mathcal{N}_i$ has different impacts on the effectiveness of training our DCNE. To further analyze the time efficiency, we fix both the number of sampling neighbors $\mathcal{N}_i$ in both *N-GAT* and *A-GAT* to the same sizes varying from 2 to 70 in the set $\{2, 5, 15, 35, 55\}$. Fig. 8 reports the performance and the corresponding runtime results of DCNE. From the figures, we observe that as the number of sampling neighbors $\mathcal{N}_i$ increases, the margin of model performance gradually decreases, and the runtime increases rapidly on both tasks. Thus, we set the number of sampling neighbors as 20, in our experiments above, in order to make the trade-off between model performance and running time.

5.5.3 The Impact of Layer Number K

For graph neural network-based methods, The number of graph layers K is an important factor for the final evaluation performances. Therefore, we conduct the comparative experiments with respect to different layer numbers to analyze its effect on DCNE, and report the results of $K \leq 3$ in Table. 5 due to the limitation of GPU memory. From the table, we can find the trend of results on both node classification and link prediction tasks are similar, where the performance is improved as the layer number increases to 2, but gradually decreases when layer number achieves 3. We argue that a larger number of graph layers, capturing more high-order information, is beneficial for model performances. However, an excessive layer number also could lead to worse results or even worse than the performance of K=1, since it contains more redundant and noise information from many overlapping or invalid high-order neighbors, especially on small-scale datasets like Cora.

5.5.4 The Impact of Initialization for Node Embedding

In the field of deep learning, parameter initialization always has an important impact on model convergence and performances. From our empirical experiments, we also find the initialization of node embeddings has more significant effects than other parameters in DCNE. Consequently, we set up the initial node embeddings U^0 with three different methods: node attributes, random values, and the matrix with all-1 values, and perform the compared results in Fig. 9. From the figure, we can observe the initialization method with attributes converges faster and achieves better performances than random and all-1 methods. To this result, we argue the attribute initialization method could start from a better initial point and have the preliminary discrimination for node embeddings.



(a) Node Classification (b) Link Prediction

Fig. 8. DCNE results with different sampled neighbor sizes on two tasks.

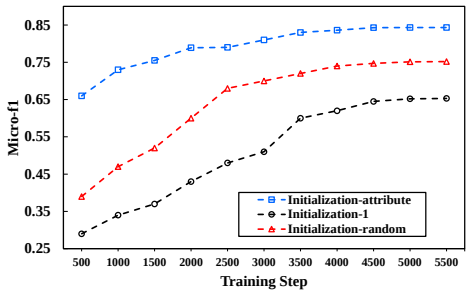


Fig. 9. Results of different initialization methods for node embedding.

6 Conclusions

In this paper, we studied attributed network embedding problem, which focused on distinguishing the complex interactions among nodes and alleviating the sparse attribute issues. Specifically, we first initialized all the nodes and attributes with low-dimensional vectors, and further proposed a novel "decoupled-fusion" process to adaptively capture and integrate both topology and attribute similarities in networks according to the node characteristics. Then, we continuously stacked multiple graph layers to iteratively aggregate the higher-order node and attribute information. At last, we jointly learned the node embeddings in a unified multi-task framework based on a modified mini-batch training strategy. Compared with state-of-the-art baselines on benchmark evaluation tasks, DCNE demonstrated the effectiveness and robustness from multiple aspects. In the future, we will continue to explore the intrinsic mechanism of attributed network and try to integrate more information such as node labels into embeddings with efficient manners.

Acknowledgements

This research was partially supported by grants from the National Key Research and Development Program of China (No. 2020AAA0103800), and the National Natural Science Foundation of China (No.s 61922073, 61976198 and U20A20229). Hanghang Tong is partially supported by NSF (1947135, 2003924 and 1939725). Hao Wang would like to thank the China Scholarship Council for their support (No. 201906340183).

REFERENCES

[1] Z. Hao, C. Lu, Z. Huang, H. Wang, Z. Hu, Q. Liu, E. Chen, and C. Lee, "Asgn: An active semi-supervised graph neural network for molecular property prediction," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 731–752.

[2] D. Du, H. Wang, T. Xu, Y. Lu, Q. Liu, and E. Chen, "Solving link-oriented tasks in signed network via an embedding approach," in *Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics*, 2017, pp. 75–80.

[3] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[4] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.

[5] H. Pei, B. Wei, K. Chang, C. Zhang, and B. Yang, "Curvature regularization to prevent distortion in graph embedding," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 20 779–20 790.

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

[7] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[8] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[9] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 203–209.

[10] Q. Liu, B. Xiang, N. J. Yuan, E. Chen, H. Xiong, Y. Zheng, and Y. Yang, "An influence propagation view of pagerank," *ACM Transactions on Knowledge Discovery from Data*, vol. 11, no. 3, p. 30, 2017.

[11] X. Wang, Y. Zhang, W. Zhang, X. Lin, and C. Chen, "Bring order into the samples: A novel scalable method for influence maximization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 2, pp. 243–256, 2016.

[12] H. Wang, T. Xu, Q. Liu, D. Lian, E. Chen, D. Du, H. Wu, and W. Su, "Mcne: An end-to-end framework for learning multiple conditional network representations of social network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1064–1072.

[13] H. Pei, B. Yang, J. Liu, and K. Chang, "Active surveillance via group sparse bayesian learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[14] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1225–1234.

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

[16] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp. 891–900.

[17] S. Cao, W. Lu, and Xu, "Deep neural networks for learning graph representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[18] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network representation learning with rich text information," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015, pp. 2111–2117.

[19] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, pp. 731–739.

[20] D. Zhanga, J. Yinb, X. Zhuc, and C. Zhanga, "User profile preserving social network embedding," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017, pp. 3378–3384.

[21] X. Huang, J. Li, and X. Hu, "Accelerated attributed network embedding," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, 2017, pp. 633–641.

[22] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1025–1035.

[23] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[24] T. L. Fond and J. Neville, "Randomization tests for distinguishing social influence and homophily effects," in *WWW '10*, 2010.

[25] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei, "Am-gcn: Adaptive multi-channel graph convolutional networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1243–1253.

[26] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[27] H. Wang, E. Chen, Q. Liu, T. Xu, D. Du, W. Su, and X. Zhang, "A united approach to learning sparse attributed network embedding," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 557–566.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[29] Y. Li, L. Xu, F. Tian, L. Jiang, X. Zhong, and E. Chen, "Word embedding revisited: A new representation learning and explicit matrix factorization perspective," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[30] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[31] Q. Dai, Q. Li, J. Tang, and D. Wang, "Adversarial network embedding," *arXiv preprint arXiv:1711.07838*, 2017.

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

[33] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, "Mutual information neural estimation," in *International Conference on Machine Learning*, 2018, pp. 531–540.

[34] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," 2019.

[35] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Homophily, structure, and content augmented network representation learning," in *Proceedings of IEEE 16th International Conference on Data Mining*, 2016, pp. 609–618.

[36] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 119–128.

[37] X. Sun, J. Guo, X. Ding, and T. Liu, "A general framework for content-enhanced network representation learning," *arXiv preprint arXiv:1610.02906*, 2016.

[38] H. Gao and H. Huang, "Deep attributed network embedding." in *IJCAI*, vol. 18. New York, NY, 2018, pp. 3364–3370.

[39] Z. Meng, S. Liang, H. Bao, and X. Zhang, "Co-embedding attributed networks," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 393–401.

[40] Y. Hu, H. You, Z. Wang, Z. Wang, E. Zhou, and Y. Gao, "Graph-mlp: Node classification without message passing in graph," *arXiv preprint arXiv:2106.04051*, 2021.

[41] H. Gao and S. Ji, "Graph u-nets," in *international conference on machine learning*. PMLR, 2019, pp. 2083–2092.

[42] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016, pp. 1895–1901.

[43] T. Huang, L. Zhou, L. Wang, G. Du, and K. Lü, "Attributed network embedding with community preservation," in *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2020, pp. 334–343.

[44] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," *arXiv preprint arXiv:1603.08861*, 2016.

[45] Z. Meng, S. Liang, J. Fang, and T. Xiao, "Semi-supervisedly co-embedding attributed networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 6507–6516.
[46] M. Qu, Y. Bengio, and J. Tang, "Gmnn: Graph markov neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 5241–5250.
[47] Y. Wang, W. Wang, Y. Liang, Y. Cai, J. Liu, and B. Hooi, "Nodeaug: Semi-supervised node classification with data augmentation," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 207–217.
[48] Y. Wang, W. Wang, Y. Liang, Y. Cai, and B. Hooi, "Mixup for node and graph classification," in *Proceedings of the Web Conference 2021*, 2021, pp. 3663–3674.
[49] Y. Wang, S. Liu, M. Yoon, H. Lamba, W. Wang, C. Faloutsos, and B. Hooi, "Provably robust node classification via low-pass message passing," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 621–630.
[50] Y. Wang, W. Wang, Y. Liang, Y. Cai, and B. Hooi, "Progressive supervision for node classification," in *Machine Learning and Knowledge Discovery in Databases*, F. Hutter, K. Kersting, J. Lijffijt, and I. Valera, Eds. Cham: Springer International Publishing, 2021, pp. 266–281.
[51] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, pp. 833–852, 2019.
[52] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.
[53] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, no. 1, p. 11, 2019.
[54] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, p. 93, 2008.
[55] Y. Cen, J. Zhang, G. Wang, Y. Qian, C. Meng, Z. Dai, H. Yang, and J. Tang, "Trust relationship prediction in alibaba e-commerce platform," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 5, pp. 1024–1035, 2019.
[56] D. Lian, X. Xie, E. Chen, and H. Xiong, "Product quantized collaborative filtering," *IEEE Transactions on Knowledge and Data Engineering*, 2020.
[57] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.
[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[59] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "User profile preserving social network embedding," in *IJCAI International Joint Conference on Artificial Intelligence*, 2017.
[60] Y. Luo, X. Qin, C. Chai, N. Tang, G. Li, and W. Li, "Steerable self-driving data visualization," *IEEE Transactions on Knowledge and Data Engineering*, 2020.
[61] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

Hao Wang is currently working toward the Ph.D. degree in the School of Computer Science and Technology at University of Science and Technology of China (USTC). His main research interests include data mining, representation learning, network embedding and recommender systems. He has published several papers in referred conference proceedings, such as TOIS, KDD, SIGIR, WSDM, ICDM, IJCAI, and AAAI.

Defu Lian is a professor in the School of Computer Science and Technology, University of Science and Technology of China (USTC). He received the B.E. and Ph.D. degrees in computer science from University of Science and Technology of China (USTC) in 2009 and 2014, respectively. His general research interests include spatial data mining, recommender systems and learning to hash. He has published prolifically in refereed journals and conference proceedings, e.g., TKDE, TOIS, KDD, IJCAI, AAAI, and WWW. He has served regularly in the program committees of a number of conferences, and is reviewers for the leading academic journals.

Hanghang Tong is currently an associate professor at Department of Computer Science at University of Illinois at Urbana-Champaign. Before that he was an associate professor at School of Computing, Informatics, and Decision Systems Engineering (CIDSE), Arizona State University. He received his M.Sc. and Ph.D. degrees from Carnegie Mellon University in 2008 and 2009, both in machine learning. His research interest is in large scale data mining for graphs and multimedia. He has received several awards, including SDM/IBM Early Career Data Mining Research award (2018), NSF CAREER award (2017), ICDM 10-Year Highest Impact Paper award (2015), and four best paper awards (TUP'14, CIKM'12, SDM'08, ICDM'06). He has published over 100 refereed articles. He is the Editor-in-Chief of SIGKDD Explorations (ACM), an action editor of Data Mining and Knowledge Discovery (Springer), and an associate editor of Knowledge and Information Systems (Springer); and has served as a program committee member in data mining, database and artificial intelligence venues (e.g., SIGKDD, SIGMOD, AAAI, WWW, CIKM).

Qi Liu is a professor at University of Science and Technology of China (USTC). He received the Ph.D. degree in Computer Science from USTC. His general research interest is data mining and knowledge discovery. He has published prolifically in refereed journals and conference proceedings, e.g., TKDE, TOIS, TKDD, TIST, KDD, IJCAI, AAAI, ICDM, SDM and CIKM. He has served regularly in the program committees of a number of conferences, and is a reviewer for the leading academic journals. Dr. Liu is the recipient of the KDD 2018 Best Student Paper Award (Research), the ICDM 2011 Best Research Paper Award, the Special Prize of President Scholarship for Postgraduate Students, Chinese Academy of Sciences (CAS), and the Distinguished Doctoral Dissertation Award of CAS.. He is supported by the Young Elite Scientist Sponsorship Program of CAST and the Youth Innovation Promotion Association of CAS.

Zhenya Huang recieved the B.E. degree from Shandong University, in 2014 and the Ph.D. degree from University of Science and Technology of China (USTC), in 2020. He is currently an associate researcher of the School of Computer Science and Technology, USTC. His main research interests include data mining, knowledge discovery, representation learning and intelligent tutoring systems. He has published more than 20 papers in refereed journals and conference proceedings including TKDE, TOIS, KDD, AAAI.

Enhong Chen (SM'07) is a professor and vice dean of the School of Computer Science at University of Science and Technology of China (USTC). He received the Ph.D. degree from USTC. His general area of research includes data mining and machine learning, social network analysis and recommender systems. He has published more than 100 papers in refereed conferences and journals, including IEEE Trans. KDE, IEEE Trans. MC, KDD, ICDM, NIPS, and CIKM. He was on program committees of numerous conferences including KDD, ICDM, SDM. He received the Best Application Paper Award on KDD-2008, the Best Student Paper Award on KDD-2018 (Research), the Best Research Paper Award on ICDM-2011 and Best of SDM-2015. His research is supported by the National Science Foundation for Distinguished Young Scholars of China. He is a senior member of the IEEE.