CoANE: Modeling Context Co-Occurrence for Attributed Network Embedding

I-Chung Hsieh and Cheng-Te Li, *Member, IEEE*

Abstract—Attributed network embedding (ANE) is to learn low-dimensional vectors so that not only the network structure but also node attributes can be preserved in the embedding space. Existing ANE models do not consider the specific combination between graph structure and attributes. While each node has its structural characteristics, such as highly-interconnected neighbors along with their certain patterns of attribute distribution, each node's neighborhood should be not only depicted by multi-hop nodes, but consider certain clusters or social circles. To model such information, in this paper, we propose a novel ANE model, *Context Co-occurrence-aware Attributed Network Embedding* (CoANE). The basic idea of CoANE is to model the context attributes that each node's involved diverse patterns, and apply the convolutional mechanism to encode positional information by treating each attribute as a channel. The learning of context co-occurrence can capture the latent social circles of each node. To better encode structural and semantic knowledge of nodes, we devise a three-way objective function, consisting of positive graph likelihood, contextual negative sampling, and attribute reconstruction. We conduct experiments on five real datasets in the tasks of link prediction, node label classification, and node clustering. The results exhibit that CoANE can significantly outperform state-of-the-art ANE models.

Index Terms—Network embedding, attributed graphs, context co-occurrence, convolutional layers, graph representation learning

1 Introduction

Networks are important data structures to represent the relationships between entities. Modern techniques in Web, storage, and computation allow us to process, retrieve, and discover knowledge from a variety of network data. In the real world, for example, social networks depict the relationships and interactions between people, and academic citation networks encode how papers are referred to each other. In addition to the graph structure, there are usually attributes associated with nodes in the networks. In social networks, such as Facebook and Instagram, users can maintain their profiles. In academic citation networks, such as Google Scholar, researchers possess affiliation, expertise, and profiles. Jointly modeling the network structure and node attributes can benefit applications, such as recommender systems [6] and fake news detection [18].

Network embedding (NE) is an essential technique in various network mining and prediction tasks [2]. The basic idea is to learn low dimensional feature representation vectors of nodes so that the graph neighborhood of each node can be encoded in the feature space. The performance of typical tasks, such as link prediction, node label classification, and community detection, can get improved based on NE. Several typical NE models were proposed, such as DeepWalk [24], LINE [26], and node2vec [7]. The general idea is to generate contexts of nodes and apply the skip-gram model [20] to learn and produce the embedding vectors. However, typical models focus on utilizing network structure, but attributes associated with nodes are neglected.

The recent focus shifts to attributed network embedding (ANE), whose goal is to preserve not only the network structure but also node attributes when learning embeddings. GAT2VEC [25] transforms node-attribute relations into a bipartite graph and merges the embeddings of structure and attributes based on random walk and skip-gram model. LANE [9] incorporates all features by adding node labels and preserves their correlation. NEEC [10] concentrates on attributed network learning with expert cognition that requires queries and answers from the experts to improve embedding. The non-linear mapping with random walk process is also developed for ANE in UPP-SNE [34]. SANE [31] further imposes an attention mechanism to discriminate the correlation between nodes. The state-of-the-art methods are DANE [4] and ASNE [16]. DANE captures the high non-linearity and preserves various proximities in both topological structure and node attributes. ASNE preserves both structural proximity and attribute proximity so that the global structure and the homophily effect in attributes can be captured.

Although several attributed network embedding models were proposed, we find that they all focus on either learning representation of network structure or attributes individually, or find a proper approach to combine the feature representations of such two parts. In this way, the target node's diverse aspects between network structure and attributes cannot be modeled. For example, in real-world networks like a social network, a node links to a number of nodes, and has multi-hop neighboring nodes consisting of several egocentric communities (i.e., the so-called *social circles*), such as family and school, in which nodes are tightly interconnected and surely have attributes similar to each other. And those belonging to one or more distinct communities can be differed from not only topological connections in

• *The authors are with the Institute of Data Science, National Cheng Kung University, Tainan 701, Taiwan. E-mail: {q00117888, reliefli}@gmail.com.*

Manuscript received 16 October 2019; revised 24 March 2021; accepted 26 April 2021. Date of publication 14 May 2021; date of current version 7 December 2022. (Corresponding author: Cheng-Te Li.)
Recommended for acceptance by E. Chen.
Digital Object Identifier no. 10.1109/TKDE.2021.3079498

their neighborhood, but also their attributes. Only when we jointly model network structure and attributes together, we will be able to better exploit their underlying correlation. State-of-the-art methods GAE [13] and VGAE [13] can simultaneously model network structure and attributes. They scan the first- or second-order graph neighborhoods recursively so that the further and wider order region in the network can be indirectly reached and exploited. Then all neighbors in the same order would be considered to have the same importance for the target node. We think that without better use of connections between neighbors in a fine-grained manner, it is less possible to distinguish same-hop neighbors from each other, which could belong to different *latent social circles*. For instance, neighbors of a student may have multiple social circles, such as "CS dept", "family", and "labmates", whose sizes are different from one another. Friends from "CS dept" circle with dominated attributes would dilute the information on "family" circle. Besides, an effective embedding learning model needs to better exploit wider and deeper interconnected neighbors of multiple latent social circles.

This paper aims to capture specific contexts representing the latent social circles of the target node and to leverage *context co-occurrence in both network structure and node attributes* for better embedding learning even though some edges are missing or unobserved. In other words, the co-occurrence of node contexts in both network structure and node attributes provides information about which neighboring nodes and their relative attributes are correlated in the context. Comparing to existing studies [4], [16], [35], ASNE [16] simply treats attributes as the model input to predict the representation of a node. Though DANE [4] additionally considers fusing preservation, and it results in higher optimization costs and more complex architectures. ANRL [35] leverages common neighbors derived from random walk for topology preservation, but it does not model node attributes with the context co-occurrence structure. Therefore, we think that the better modeling of the interplay between network structure and node attributes can enhance the embedding power.

To deal with these issues, in this paper, we develop a novel ANE model, *Context Co-occurrence-aware Attributed Network Embedding* (CoANE),[1] to learn node embeddings in attributed networks. Our CoANE is devised to make the embeddings of nodes preserve three-fold information: (a) the graph neighborhood of nodes from sampling contexts: making nodes tightly interconnected have similar embeddings, (b) the context co-occurrences of nodes: driving the embeddings to become closer if nodes whose direct and indirect neighbors (i.e., contexts) share similar attributes, and (c) the attributes of nodes: shaping the embeddings of nodes to preserve their own attribute information. To realize such ideas, CoANE is developed to have three novel components. First, we generate the crucially structural contexts of nodes via random walk and treat them from the perspective of attributes. Second, by considering each attribute as a *channel*, we adopt a 1-D convolutional layer with different filters to model contexts with their corresponding channels and summarize the features from the target's contexts into the derived embedding. Third, for more effective learning, we extend two contextual likelihood approaches in the design of our objective function. One is the *positive graph likelihood* whose goal is to make one- and high-order structural context co-occurrences be better preserved in the embeddings. The other is a *contextually negative sampling*, which considers the co-occurrence frequency of the target node and the negative samples to have a more effective negative sampling. Besides, we also perform the reconstruction of attribute values based on the derived node embeddings so that the semantics of nodes can be better preserved.

We summarize the contributions of this paper as follows.

- We propose a novel ANE model, CoANE, which is able to better generate and model the representational contexts of the target node. The main idea of CoANE is to capture the latent social circles of the target node.
- Technically, a convolutional mechanism is applied to distill positional information and latent social-circle features from the attributed contexts, which cannot be captured by existing solutions that use fixed-hop neighborhood. Besides, the extended graph likelihood, the contextually negative sampling method, and the reconstruction of attributes are also proposed for preserving higher-order structural relationships and nodes' semantic knowledge.
- Experiments conducted on five real datasets in tasks node classification, node clustering, and link prediction show that CoANE can significantly and consistently outperform state-of-the-art ANE methods.

This paper is organized as below. We discuss relevant studies in Section 2, and present the technical details of CoANE in Section 3. The experimental settings and results are described in Section 4. Section 5 concludes this work.

2 Related Work

We first introduce random walk-based methods that are popularly adopted for network embedding to attributed network embedding (ANE) in Section 2.1. Then, we discuss well-known deep graph reconstruction approaches to ANE in Section 2.2. We point out that in the literatures of network embedding, the techniques of subgraph aggregation can better incorporate network structure with node attributes in Section 2.3. Last, in Section 2.4, we review the common optimization designs for the ANE methods discussed from Sections 2.1, 2.2, and 2.3, and point out their insufficiencies.

2.1 Random Walk-Based Approaches

To review the random walk-based methods for network embedding, we first introduce DeepWalk [24], which generates fixed-length paths to have neighboring correlated nodes, and then computes the embedding similarity between the center node and a random-selected node inside/outside the window. Then, node2vec [7] devises the biased random walk to balance the wide and deep neighbors. Both approaches can transform the network structure to shortly and tightly interconnected sentence-like paths. Recent methods incorporate deep learning with random-walk node sequences. NetRA [33]

1. The code of CoANE can be accessed via the following Github link: https://github.com/ICHproject/CoANE/

presents a generative adversarial training and produces node embeddings by reconstructing random walk sequences and discriminating positive and negative samples. However, DeepWalk, node2vec, and NetRA are not designed for ANE. Thus, STNE [17] utilizes seq2seq in machine translation. It considers the random walk sequence as a sentence and learns the higher-order information. GraphRNA [11] conducts a random walk on the bipartite network between nodes and attributes, and then learns the features from the mixed context via sequence modeling for node classification. Besides, ANRL [35] combines the skip-gram model as an additional reconstruction for node attributes' autoencoder model with a jointly-learning process. Although obtaining the promising performance, both STNE and GraphRNA ignore the distribution of sequence sampling and node context diversity. ANRL only applies the distribution to the information preservation, and similar attributes in the context are not discussed. For this issue, we find that metapath2vec [3] can learn the node embeddings considering various relational patterns of nodes and edges, but it requires a heterogeneous network as the input graph. The strength of metapath2vec is on modeling diverse types of relationships between nodes, rather than incorporating node attributes. Hence, the fine-grained and diverse semantics distributed over contexts cannot be encoded into node embeddings.

2.2 Graph Reconstruction-Based Approaches

For deep learning applications, the autoencoder-based methods get attention, in which the input is squeezed into the embeddings and maintains the important features by optimizing the error between the original input and the decoder's output. To extend to the ANE framework, ASNE [16] learns the embeddings from different information sources and feeds them into the multi-layer neural network to distill high-level features. Then, DANE [4] adopts both the multi-layer mapping and the complementary loss to enforce two embeddings being as consistent as possible. SCAN [19] further considers combining diverse knowledge, including graph structure, node features, and node labels, for ANE in a semi-supervised setting. Although autoencoder-based methods can produce promising better performance, neither diverse contexts nor context importance is encoded in the embedding learning. Besides, there is a trade-off between performance and computation cost, especially for sparse features (e.g., the adjacency matrix), and the multi-source information fusion. Recent studies argue the incompatibility of the network embedding on euclidean space for the structure of real-world networks. To this end, DRNE [28] further incorporates the regular equivalence into node embedding learning, which is proven to preserve some typical node centrality measures.

2.3 Subgraph Aggregation-Based Approaches

To better incorporate structure and attribute, the technique of subgraph aggregation is used in graph representation learning. The main idea is to utilize the neighboring subgraph to represent each node, and to learn an aggregation function that can fuse features of neighboring nodes with the target node. To fulfill the subgraph aggregation technique, since the convolutional mechanism has been a powerful detector of local features, especially for relationship pattern recognition. Graph Convolutional Network (GCN) [14] is proposed to adopt the spectral graph convolution for semi-supervised node classification. Its extended method Graph Attention Network (GAT) [30] can further distinguish the importance of neighbors. Nevertheless, we concentrate on node embedding learning in attributed networks without known labels. For the unsupervised setting, Graph Auto-Encoder (GAE) [13] and Variational Graph Auto-Encoder (VGAE) [13] revise the GCN and learn ANE by adding the reconstructed objective of the adjacency matrix, along with variational parameters for the probability function. Furthermore, GraphSAGE [8] is devised for learning both *transductive* and *inductive* node embeddings in a graph. In GraphSAGE, node features, along with the graph topology, are used to learn an embedding function that can be applied to both existing and new-coming nodes in the graph. GraphSage can be trained in either semi-supervised or unsupervised manners. To enhance node embeddings' quality and robustness, ARGA/ARVGA [23] extends the GAE/VAGE by the adversarial learning to generate more real negative samples for model learning.

However, these unsupervised subgraph aggregation models cannot distinguish the contributions of different hops of neighbors, and cannot capture the structural and attributed patterns, such as social circles. Moreover, their objective functions make embeddings fit the input adjacency matrix that only considers first-order neighbors. The preservation of higher-order information tends to be missing.

2.4 Optimization Design

In optimizing node embedding learning, the general design of loss function consists of two parts: (1) enhancing the similarity of nodes that are close and correlated with each other, and (2) using negative sampling to better separate irrelevant nodes from one another. For the first part, encoding the first-order relationship between nodes, such as GAE [13], VGAE [13] and STNE [17], is the common-used way. However, it would result in being less capable of learning the distributions of importance among different neighbors. Preserving higher-order relationships between nodes by DANE [4] can lead to better performance. Second, the negative sampling is to replace the high-cost computation of softmax mapping for large-scale networks. However, producing negative samples based on their appearance frequency cannot well separate irrelevant nodes from each other in the embedding space because nodes tend to have similar negative samples. In this work, we pay more attention to the preservation of topological distribution of the network by incorporating the weighted sampling probability of context into negative sampling.

3 The Proposed CoANE Model

We first describe the notations. Given an attributed network $\mathbf{G} = (V, \mathbf{E}, \mathbf{X})$, where V is the node set with n nodes ($|V| = n$), $\mathbf{E}$ is the adjacency matrix, in which $\mathbf{E}_{ij}$ represents the weight between $v_i \in V$ and $v_j \in V$, and $\mathbf{X} \in I\!R^{n \times d}$ is the node-attribute matrix, where d is the number of attributes. In the task of NE learning, we aim at generating a low dimensional vector to represent each node in a network. As for ANE, the goal is to preserve two properties from an attributed network, *structural proximity* and *semantic proximity*. The former

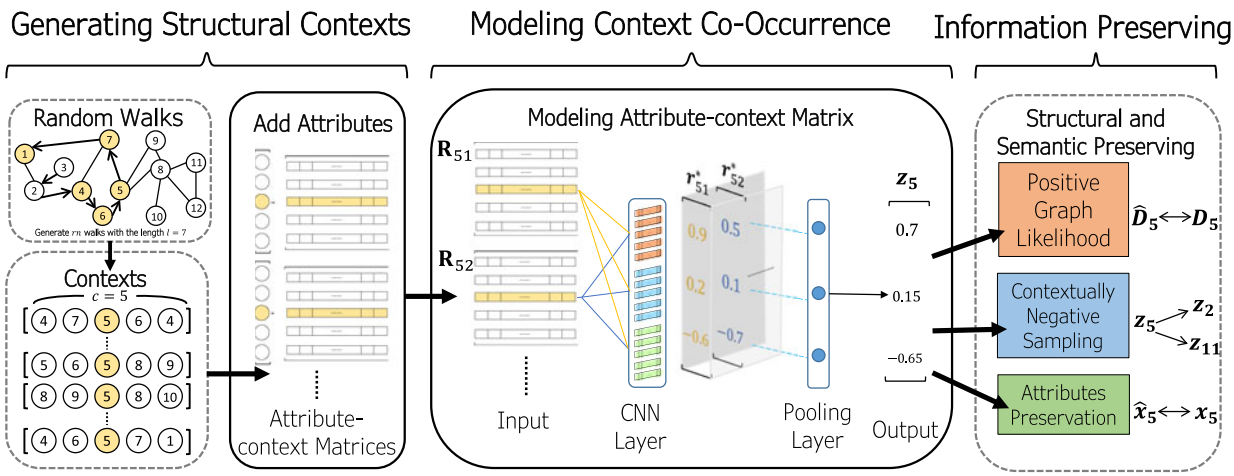


Fig. 1. The overview of CoANE. In this illustration, we generate r random walks with length $l = 7$, as indicated by bold directed edges, for $n = 12$ nodes. The contexts of the midst node 5 are extracted from random walks with size $c = 5$. The attribute-context matrix $\mathbf{R}_{5*}$ is fed into 1-D convolutional layer and then 1-D pooling layer. The low-dimensional embedding is the output. For structural and semantic preserving, we update the model parameters by an extended graph likelihood, the proposed contextually negative sampling, and the attribute preservation.

indicates that nodes with similar structural neighborhood tend to have similar embedding vectors. The latter aims at making nodes sharing similar attributes possess similar embedding vectors. We think such structural and semantic proximities are correlated with one another in the form of *social circles*, and should be simultaneously modeled in learning node embeddings. Here the *social circles* [15] refers to that there are multiple groups of friends that share similar attributes and have tight connections to each other in a node's ego network. The preservation of structural and semantic proximity in CoANE is to capture the latent social circles surrounded by each node in the network. The framework of CoANE contains three parts including *generating structural contexts*, *modeling context co-occurrence*, and *proximity preserving*, as shown in Fig. 1. We first find the contexts via random walk and combine them with attributes for modeling their context co-occurrence features. In the end, we compute the likelihood of preserving co-occurrence proximities in updating our model and deriving the resulting embeddings. We elaborate details of our CoANE in the following subsections.

3.1 Generating Structural Contexts

To capture the latent social circles, we start from generating sequences of correlated nodes, i.e., structural contexts, via random walk based on word2vec [21]. At each walking step, the random walker starts at a given node $v_i \in V$ and decides which adjacent node to visit next by sampling with probability $p(v_i) = \frac{\mathbf{E}_{ij}}{\sum_j \mathbf{E}_{ij}}$. For each node as the starting one, we repeat the sampling process until the length of walks is up to a predefined value l. We can repeat this process r times for each node, and then have rn sequences with length l. Then, we compile each node's context by scanning the sequences and copying a specific fragment as a context. We set a fixed window (context) size c and align the midst of the window with the starting node of sequences. A context contains the target node's previous and latter neighbors. Since the beginning position of the scanning contains empty slots, i.e., the first half of the window has no previous neighbors, we perform padding to fill in the empty slots like the image padding for the convolutional neural network (CNN). Then, we move the window towards the next positions in the sequence, and adopt subsampling [21] to alleviate over-emphasizing the nodes that high-frequently appear in the context.

The subsampling is to deal with the imbalance of occurrence frequency between rare and frequent nodes. To alleviate the over-frequent occurrence for some nodes, those nodes with higher frequency should be ruled out with higher probabilities. That said, in subsampling, we increase the possibility of being sampled for rare nodes, i.e., those rarely appear in the generated node sequences so that the quality of embeddings of rare nodes can be improved. The subsampling probability is given by $p_{sub}(v) = 1 - \sqrt{t/f(v)}$, where t is a constant, and $f(v)$ is the frequency of node v's appearance in the generated node sequences. The context of the midst with the frequency $f(v)$ higher than t would tend to be discarded. In addition, we set $p_{sub}(v) = 1$ if v is the starting node for each sequence to ensure that each node has at least one context neighbor.

The original skip-gram model [21] considers only the center node and its neighbors in the sampled node sequence for embedding learning. Since nodes outside the context window are discarded, the original skip-gram needs to sample more node sequences, which produces additional computational cost. We think the entire context (i.e., the sampled node sequence) can reflect the latent social circles that the center node involves. Besides, the distribution of positional information located from near-by to far-away neighbors with respect to the center node can also be used to unfold the sizes of different latent social circles. Hence, we fully exploit all of the generated contexts, and accordingly model the context co-occurrence to learn latent social circles of every node.

An illustration architecture of our random walk mechanism is shown in the left dotted blocks of Fig. 1. In the random walks block and contexts block, we demonstrate the generated contexts for node 5, which are sampled from the random walk node sequences. We can find the nodes adjacent to the node 5 must be the 1-hop, 2-hop, and higher-order neighbors. In the following, we denote the collection of contexts with the same central node v as a set $context(v)$. Because the generated context comes from the sampling process, the size of each node v's context set $context(v)$

tends to be different. This indicates that a node with more diverse neighbors is surrounded by richer social circles. Such a setting depicts that a neighbor that frequently occurs in the context can possess similar traits (sharing common neighbors or similar attributes) as the midst node. Hence, we need to learn the specific patterns from the context to distinguish features between nodes.

Note that although the undirected networks do not have positional information, the distance between nodes in the network can indicate the relationship strength between nodes. The higher-order relationship (i.e., long-distance) can help find the latent features. It could also introduce noise if the random walk goes in the wrong way. Therefore, we adopt the convolutional mechanism, in which the filters with context-like length are used, to learn which parts in the high-order relationships contribute more in depicting the context of nodes. The positional information is captured by weights learned from convolution filters that are applied to generated contexts. Different nodes can have various distributions of useful positional information. The positional information can help better represent nodes because high-order relationships in nodes' context can be distinguished by convolutional weights.

Modeling context co-occurrence can capture positional information. We create the *co-occurrence matrix* $\mathbf{D}$ to represent the co-occurrences of nodes by counting the node occurring the contexts of the node v_i (i.e., $\mathbf{D}_{ij} =$ the counts of v_j in all $context(v_i)$). Since one-hop context neighbors of a node can best represent that node, to enhance the preservation of local structure, we also define 1-hop co-occurrence $\mathbf{D^1}$, where $\mathbf{D^1}_{ij} = \mathbf{D}_{ij}$ if $\mathbf{E}_{ij} > 0$. These two co-occurrence matrices help us preserve the graph information in the optimization step.

3.2 Modeling Context Co-Occurrence

We learn the pattern from each node's context and combine the context features. We adopt the convolutional mechanism, along with multiple filters, to extract the similar attitudes and positional information of nodes in a context, and then pool all context features to generate node embeddings. Similar to the CNN model for the feature extraction of the image: a cat image can be assembled by several small matrices depicting a long tail, a furry body, and round pupils. We can imagine such the context set of the target node is a *picture* consisting of specific features from their nodes' attributes and topological neighbors. Since the diversity of contexts for the target needs to be encoded, we need a multi-view model to recognize and extract a variety of patterns of "pixels" in different contexts.

The idea of modeling context co-occurrence is to capture the *latent social circles* surrounded by each node in the network. The social circles mean that in a social network, the neighborhood (e.g., friends and friends of friends) of a user tends to contain multiple neighboring subsets, in which nodes in each subset are tightly connected with one another and share similar attributes. Each neighboring subset is considered as a social circle. For example, in reality, each social circle (i.e., neighboring subset) can be "basketball club", "family", and "labmates." The convolutional mechanism that learns the combination between graph structure and context attributes can model the latent social circles through a variety of convolutional filters. Since the attributes of nodes further away from the midst in the context are less correlated with the target node, we learn different weights by wider convolutional filters to adjust the contribution of different positional information in the context.

The mathematical details of modeling these contexts are depicted as follows. First, we represent each context for the same central node in the form of the matrix. For every node in the context, we put and align their attributes of these context nodes together according to their order in the context. An *attribute-context matrix* $\mathbf{R}_{vi} \in I\!R^{c \times d}$ for the ith context with midst node v can be derived. We use all the attribute-context matrices involved by the same midst v to distill its features, where each matrix can be viewed as the source of a feature. Specifically, the attribute-context matrices corresponding to the same midst v can be concatenated vertically as a large attribute-context matrix $\mathbf{R}_v \in I\!R^{c' \times d}$, where $c' = c \cdot |context(v)|$. Its attributes in the second dimension of the matrix are independent. Hence, we can view each attribute as a channel, like the RGB color values of an image, to depict the specific pattern between network structure and attributes. Hence, the matrix can be squeezed along the second dimension as a sequence with d channels with its features. The summarized matrix from these contexts has become a fixed form, which is exactly compatible with the input form of the convolutional neural network in euclidean space. Since a sequence of attribute-context matrices is corresponding to contexts per midst node, then we can directly adapt the *1-dimension convolutional neural network* (1-D CNN) to learn the similar attributes of nodes at different positions for each context. Note that we do not consider the overlapping of the receptive region because each context individually represents the target node. We let the filter of CNN scan the region in the length of context instead of the overlapping scan. In other words, the setting of our 1-D CNN model includes: the number of input channels (attribute dimension) d, the number of output channel (embedding dimension) d', and the volume (receptive field size) $= c$ (context size) with stride $= c$ so that each movement of the filter is equal to the length of context as an unit. That is, the model filters the attribute-context matrix $\{x_{v_{-c'}}, \ldots, x_v, \ldots, x_{v_{c'}}\}$ for the context $\{v_{-c'}, \ldots, v, \ldots, v_{c'}\}$, where x_v is the corresponding attribute vector of node v and $c' = (c-1)/2$, and they are convolutionally summed by d' filters to be a feature vector $\mathbf{z} \in I\!R^{d'}$. After having scanned all of the sequences, we can derive a number of feature vectors for each node. The results of feature vectors are fed into a pooling layer, where we choose to average the different number of diverse vectors (i.e., 1-D average pooling) from the same midst as the final embedding vector. Such a process can be illustrated as Fig. 1 (Modeling Context Co-Occurrence) and Fig. 2 (bottom). Finally, we formulate the process of the 1-D convolutional layer as well as the average pooling layer for node v, which can be respectively represented as below. The convolutional part is given by: $\mathbf{r}^*_{vij} = \sum \mathbf{R}_{vi} \odot \mathbf{\Theta}_j$, for $i = 1, 2, \ldots, |context(v)|$ and $j = 1, 2, \ldots, d'$, where $\mathbf{r}^*_{vij} \in I\!R$ is the ith context's convolutional value from the jth filter, $\mathbf{R}_{vi} \in I\!R^{c \times d}$ is the ith attribute-context matrix with midst node v, $\odot$ is Hadamard product, and $\mathbf{\Theta}_j \in I\!R^{c \times d}$ is the parameter matrix of the jth filter. The pooling part is given by: $\mathbf{z}_v = \mathbf{\theta}_{pool}\mathbf{R}^*_v$, where $\mathbf{z}_v$ is the embedding vector of node v, $\mathbf{\theta}_{pool} \in I\!R^{1 \times |context(v)|}$ is 1-D average pooling operator,

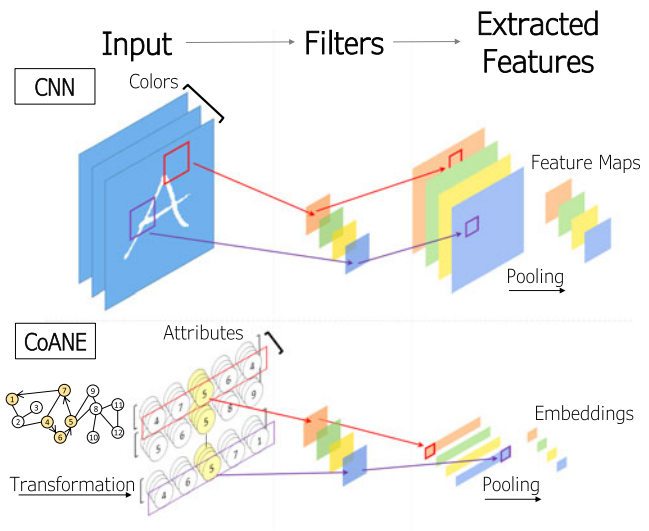


Fig. 2. The differences of convolution between CNN and CoANE. After converting the attributed network into a set of attribute-context matrices, each matrix can be viewed as a segment of photo with certain features that we perform convolution through multiple filters.

and $\mathbf{R}_v^* \in I\!R^{|context(v)| \times d'}$ is the collection of convolution results from $\mathbf{r}_{vij}^*$, i.e., $\mathbf{R}_v^* = \{\mathbf{r}_{vij}^*\}$.

Discussion. The proposed method can be further discussed in comparisons of related studies. First, in Fig. 2, we present an outline of CNN (top) and how CoANE utilizes the convolutional mechanism (bottom). Though euclidean's input of CNN is not generally compatible with network data, we can use attribute-context matrices in a similar configuration as images so that CNN can be applied. CNN can recognize a cat picture via some filters to score and mine its tail and pupils. Similarly, CoANE can distinguish neighboring nodes via learning latent social circles, like "baseball team" and "colleague", using the various filters that learn higher weights on the correlated attributes like "favorite sport" and "job."

Second, compared to the state-of-the-arts GAE [13] and VGAE [13] that only uses fixed neighborhood in embedding learning, CoANE more emphasizes the learning in the specific patterns of similar attributes and positional information in the context. Hence, CoANE tends to incorporate more precise information from network structure and attributes, which is more capable of detecting various situations like one or multiple latent social circles (e.g., some friends likes baseball while another set of neighbors are colleagues who like jazz music).

3.3 Information Preservation and Objective Function

While the network structure and node attributes have been encoded by the proposed attribute-context convolutional mechanism, now we present how to design the learning objective so that the derived embedding vectors can have structural and semantic preserving in the view of the context. The design of objective function can be divided into three parts: Positive Graph Likelihood, Contextually Negative Sampling, and Attribute Preservation. The first part is to preserve co-occurrence matrices $\mathbf{D}$ and $\mathbf{D^1}$ while the second is to make nodes tightly interconnected and overlapped with one another in terms of attributes that have similar embedding vectors. The third is to preserve node semantics by reconstructing the original attributes.

3.3.1 Positive Graph Likelihood

We take advantage of the idea of autoencoder that reconstructs features between input and output to impose the network structure into the embedding vectors. We extend the *graph likelihood* [1] to reconstruct a new co-occurrence matrix $\mathbf{D}'$ by embedding vectors based on matrix factorization, and the aim is to minimize a reconstruction loss between $\mathbf{D}'$ and $\mathbf{D}$ (and $\mathbf{D^1}$). Let the embedding matrix be $\mathbf{Z}$, and $\mathbf{Z} = [\mathbf{L}|\mathbf{R}]$, where $\mathbf{L}, \mathbf{R} \in I\!R^{n \times \frac{d'}{2}}$ are the left embedding and right embedding. The graph likelihood can be defined as follows:

$$\prod_{v_i, v_j \in V} \sigma(\mathbf{L}_i^T \mathbf{R}_j)^{\mathbf{D}_{ij}} \sigma(1 - \mathbf{L}_i^T \mathbf{R}_j)^{I(\mathbf{E}_{ij}=0)}, \tag{1}$$

where $\sigma(x) = (1 + \exp(-x))^{-1}$, I is the indicator function, L_i is the ith row of L and R_j is the jth row of R. The idea is to use embedding vectors to generate the co-occurrence matrix. An embedding learning can preserve more about the network if it can lead to high graph likelihood in the reconstruction of co-occurrence matrix $\mathbf{D}$.

However, the original graph likelihood needs to enumerate pairs of nodes (leading to high computation cost) and is not able to precisely emphasize on real edges of the network. We leverage only the positively relational term and avoid sparser and higher cost negative term, and also strengthen the graph likelihood of one-hop nodes by adding $\mathbf{D^1}$.

The adjusted graph likelihood L_{pos} can be rewritten in the form of negative log-likelihood as below:

$$L_{pos} = -\sum_{v_i \in V} \sum_{v_j \in V, i \neq j} \tilde{\mathbf{D}}_{ij} log(\sigma(\mathbf{L}_i^T \mathbf{R}_j)), \tag{2}$$

where $\tilde{\mathbf{D}} = \mathbf{D^N} + \mathbf{D^1}$, and $\mathbf{D^N}$ is the normalized $\mathbf{D}$.

Note that the first-order (one-hop) neighbors should be more emphasized in learning node embeddings. This idea comes from random walk with restart (RWR) [27], i.e., personalized PageRank [22]. Although one-hop neighbors already have higher arriving probability values, the restarting probability in RWR is used to give one-hop neighbors much higher arriving probability values. The underlying intuition is that one-hop neighbors can directly represent the target node. Hence, we need to pay more attention to them by adding $\mathbf{D^1}$ to strengthen the graph likelihood. In addition, to further emphasize the contribution of one-hop neighbors in graph likelihood, we choose to use $\mathbf{D^N} + \mathbf{D^1}$, rather than the normalization of $\mathbf{D} + \mathbf{D^1}$. Adding $\mathbf{D^1}$ to the normalized $\mathbf{D}$ will give one-hop neighbors higher importance in learning node embeddings.

Furthermore, we think that the extremely small values in $\tilde{D}_{ij}$ could be noise when the graph is sparse. We want to lower down the effect of noisy structural information, and aim at preserving the most significant positive neighbors for each node. We consider the top-k_p co-occurrence neighbors scored by $\tilde{D}_{ij}$. Specifically, for each node v_i, we compute $\mathbf{L}_i^T \mathbf{R}_j)$ for only $j \in$ top-$k_p(\tilde{\mathbf{D}}_{ij} | j = 1, \ldots, n)$. We determine the number of significant positive neighbors k_p by $k_p = \max_{i=1,\ldots,n}(|context(v_i)|)$, where the $|context(v_i)|$ is the number of the sampled context nodes for v_i. We can treat k_p as a kind of latent neighborhood size.

With the preservation of top-k_p significant neighbors, our loss function is able to not only pay more attention to preserve strong connections, but also better alleviate the impact of noisy neighborhood, which especially appears in sparse graphs that result in more less-frequent neighbors sampled by random walks.

3.3.2 Contextually Negative Sampling

In learning node embeddings, the basic idea is to make correlated nodes close in the embedding space. To better separate correlated and irrelevant nodes considering network structure and node attributes, we develop an efficient and effective *contextually negative sampling* extended from negative sampling methods in word2vec [21] and AllVec [32]. In word2vec, the method is to select a small number of dissimilar samples and compute the loss using their logistic similarity by inner product. AllVec [32] considers all nodes out of the neighborhoods of the target word as the negative samples, and compute loss by square of inner product similarity. In addition to combining word2vec and AllVec, we further consider the contextual frequency of the target node and the negative samples so that the importance of negative samples can be modeled.

We develop and expect our contextually negative sampling can push the embeddings of dissimilar nodes in both the network structure and node attributes to be away from each other according to the contextual frequency. Our negatively relational loss L_{neg} for target node v_i is given by:

$$L_{neg}(v_i) = \sum_{j=1}^{k} E_{v_j \sim P_{V^*(v_i)}} (a(\mathbf{z}_{v_i}^T \mathbf{z}_{v_j})^2), \tag{3}$$

where k is number of negative samples, the contextual noise distribution $P_V(v)$ is defined by $\frac{|context(v)|}{\sum_{v \in V} |context(v)|}$ for any node set V, $V^*(v) = \{v' \in V | v' \notin context(v)\}$ is the set of nodes occurring out of the context of node v, and a is a controlling constant deciding the strength of negative loss.

In detail, the negative loss L_{neg} consists of two parts: the similarity between embedding vectors and the target-negative *co-occurrence* probability for negative sampling. For the similarity, we utilize the square inner product similarity that is the same as existing NE methods and emphasizes the magnitude of similarity. Second, we aim at selecting the most significant negative samples because they need more power to be pushed away in the embedding space. We think the volume of a node's generated contexts is related to its main representation in the graph as well as a domination in a cluster. Therefore, we choose a node as an appropriate candidate for negative samples if its contexts cover the larger region (i.e., more contexts) in the network and have a lower correlation with the target node. That said, we sample nodes based on the contextual probability derived by its co-occurrence frequency of the contexts. The first k nodes sampled from $V^*(v_i)$ are the more informative negative ones and considered as the negative samples for the target node v_i. To deal with the high sampling cost when V^* for each node has a diverse composition, we devise *pre-sampling* and *batch-sampling* to obtain negative nodes. For the pre-sampling, first, we offline sample the nodes more than k, respecting to the contextual probability $P_V(v)$ as the negative sets. Then we select the first k samples out of the contexts of target node v_i as negative samples. For batch-sampling, we consider nodes in the batch for negative sampling during training. This can avoid too much probability computation and reduce the times of comparisons between nodes. The pre-sampling can reduce the training cost due to selecting negative targets before training. The batch-sampling only needs to handle nodes in each batch.

3.3.3 Attribute Preservation

In addition to encoding structural information, we also aim at preserving semantic information based on node features, i.e., attribute values, in the process of embedding learning. We utilize a multi-layer perceptron (MLP) decoder to reconstruct node features from the derived node embeddings. That said, we can obtain the reconstructed attributes $\hat{\mathbf{X}}_{v_i} = MLP(\mathbf{z}_{v_i})$, where MLP is constructed by stacking two hidden layers with the ReLU non-linear activation function. The preservation loss is given as follows:

$$L_{att} = \gamma MSE(\hat{\mathbf{X}}, \mathbf{X}), \tag{4}$$

where MSE is mean square error, $\hat{\mathbf{X}}$ denote the reconstructed attribute values, γ is the hyperparameter that controls the importance of the reconstruction effect.

3.3.4 Model Optimization

The final objective function L_{obj} of CoANE is given by Eq. (5), which combines the positive graph likelihood Eq. (2), the contextual negative sampling Eq. (3), and the attribute preservation Eq. (4). In the equation, the positive part ensures the preservation of structural and semantic proximities while the negative part pushes dissimilar ones away from each other and avoids overfitting

$$L_{obj} = L_{pos} + \sum_{v_i \in V} L_{neg}(v_i) + L_{att}. \tag{5}$$

We use batch gradient descent to optimize the parameters in L_{obj} for the parameters of filters in our model, which nodes are randomly partitioned into several batches and make their corresponding embeddings update each epoch.

The overall CoANE algorithm is outlined in Algorithm 1. After the pre-processing phase, we can obtain each node's contexts $context(v)$ and two matrices $\mathbf{D}$ and $\mathbf{D^1}$. Then we start the training phase and initialize both model parameters and embedding vectors of all nodes using Xavier uniformly initialization [5]. At each training iteration, we apply batch updating, which further consists of Embedding Updating step and Loss Updating step, for n_B nodes of V_{n_B} sampled from V without replacement. That is, we randomly split V into a set of batches V_B with size n_B by the function $RandomlySplitBatch$, and perform updating procedures for each batch V_{n_B} sampled from V_B.

For the sampled nodes, we first update their embeddings by following our convolutional process, and then the loss is calculated for updating model parameters. We expect that filters of the model are updated by contexts and are kept improved in generating embeddings. The whole updating process repeats until the convergence of L_{obj} or the maximum step is achieved. Last, we need to refresh all embeddings the same as Embedding Updating step.

Algorithm 1. CoANE Algorithm

Input: $\mathbf{G} = (V, \mathbf{E}, \mathbf{X})$, repeat r, walk length l, scan probability $p(v)$, context length c, maximum epoch N_{max}, number of training nodes n_B and number of negative samples k
Parameter: parameters of filters $\mathbf{\Theta}$
Output: node embedding $\mathbf{Z}$

```
#Pre-processing phase
for all node v in V do
    context(v) = RandomWalkProcess(V, E), p(v), c, r, l)
Construct D, D^1 and negative samples V_neg and initialize Z and Θ
#Training phase
for n = 1 to N_max do
    V_B = RandomlySplitBatch(V, n_B)
    for V_nB in V_B do
        {R_v}_{V_nB} = AttributeConcat({context(v)}_{V_nB}, X)
        Update Z by {z_v}_{V_nB} = CNN(Θ, {R_v}_{V_nB}, d')
        Compute L_pos(Z, D, D^1) # Section 3.3.1
        Compute L_neg(Z, V_neg, k) # Section 3.3.3
        Compute L_att(MLP(Z), X) # Section 3.3.4
        L_obj(V_nB) = L_pos + L_neg + L_att
        Update Θ by GradientDescent(L_obj(V_nB))
Renew z_v for all nodes V
return Z
```

Complexity Analysis. The time complexity relies on three parts: the convolutional mechanism with filters, the computation of co-occurrence matrices between contexts and attributes, and the attribute reconstruction for its preservation. The time complexity of the first part is $O(d * d' * c)$, where the context size c is extremely smaller than the attribute dimension d (i.e., $c \ll d$). This part is less than the state-of-the-art ANE model DANE ($O(d * d' * l)$) [4], where l is the number of parameters in the hidden layer and usually larger than embedding dimension d'. Second, for the co-occurrence matrices, since matrices are sparse, we use sparse data structure and operation. So, the complexity is $O(n * d')$, where $n = |V|$ is the number of nodes. Such complexity is also less than DANE that needs multi-proximity computation $O(n^2)$. We also leave the memory-efficient extensions via our batch updating. Then the complexity becomes $O(n_B * d')$ (i.e., $n_B \ll n$) for each sub-epoch. Third, our attribute reconstruction is similar to DANE; however, the actual computation of DANE requires encoding, decoding, and multiple loss functions to capture structure and attribute information. Our CoANE only utilizes an attribute decoder for attribute preservation, which is a shallow structure, comparing to the deeper architecture of DANE.

4 Experiments

4.1 Experiment Settings

Datasets. We employ five publicly available attributed network datasets, including Cora, Citeseer, WebKB, Pubmed,[2] and Flickr [9] , for the experiments. The statistics of such five datasets are presented in Table 1. Each node is associated with a list of attributes and one class label. The class label is considered as the ground truth in the task of node label classification and node clustering. Note that since WebKB contains four small networks, we run the experiments separately, and report the average score.

TABLE 1
Summary of the Adopted Datasets

Dataset	#nodes	#Attributes	#edges	density	#labels
Cora	2708	1433	5278	0.0014	7
Citeseer	3312	3703	4660	0.0008	6
Pubmed	19717	500	44327	0.0002	3
WebKB-Cornell	195	1703	286	0.0151	5
WebKB-Texas	187	1703	298	0.0171	5
WebKB-Washington	230	1703	417	0.0158	5
WebKB-Wisconsin	265	1703	479	0.0137	5
Flickr	7575	12047	239738	0.0084	9

Competing Methods. We consider well-known NE/ANE methods for performance comparison. The first two are commonly-used plain NE methods that consider no attributes: random-walk-based approaches node2vec [7] (with parameters $p = q = 1$) and LINE [26] that preserves second-order graph proximity. The next three are state-of-the-art subgraph-aggregation-based ANE methods, including GAE/VGAE [13] with 2 layers (256-128) and GraphSAGE [8] with the mean aggregation. The following two are state-of-the-art graph-reconstruction-based ANE methods, including DANE [4] [3] with 2 layers (128-64), and ASNE [16]. For the context-modeling-based approach, we consider STNE [17], in which the layer sizes of encoder and decoder are all 64.[4] Our CoANE is also a context-based approach like STNE. Last, CoANE is further compared with two recent state-of-the-art approaches ANRL [35] and ARGA/ARVGA [23] with layers (256-128) and discriminator (128-512) that follow the settings mentioned in the original paper. For node2vec, DANE, and ANRL, we set the parameters for the random walk: the window size $= 10$, the walk length $= 80$, and the number of walks $r = 10$. To have a fair comparison in generating low-dimensional node embeddings, the dimension of node embedding vector is set $d' = 128$ for all methods. For CoANE, we set the number of walks $r = 1$, the walk length $= 80$, the scan parameter $t = 10^{-5}$, and the number of negative samples $k = 20$. Besides, we consider a 2-layer multi-layer perceptron MLP with ReLU non-linear mapping for attribute preservation, and we choose Adam optimizer with learning rate $= 0.001$ [12].

On the tuning of CoANE hyperparameters, the negative loss controller a in Eq. (3), the context window size c, and the attribute preservation controller γ in Eq. (4) are tuned by the validation set with ranges: $a \in [1\text{e-}5, 1\text{e-}1]$, $c \in [3, 5, 7, 9, 11]$ and $\gamma \in [1\text{e}3, 1\text{e}7]$, respectively. Besides, we apply pre-sampling to obtain negative samples in Eqs. (2) and (3) for the denser graphs (i.e., WebKB and Flickr), and apply batch-sampling in Eqs. (2) and (3) for the sparser graphs (i.e., Cora, Citeseer, and Pubmed).

2. Datasets for Cora, Citeseer, WebKB, and Pubmed available via https://linqs.soe.ucsc.edu/data

3. To have fair comparison, we exclude the pre-training part in DANE source code because the pre-training part is never mentioned in the paper and all of the competing methods (including our CoANE) do not consider pre-training. That said, we faithfully utilize the end-to-end training of DANE as one of our competing methods.

4. All settings of encoder and decoder and the generation of contexts follow the original STNE paper.

TABLE 2
Macro and Micro F1 Scores for Node Label Classification of Cora, Citeseer, and Pubmed Datasets

Dataset	Cora						Citeseer						Pubmed					
Training ratio	5%	20%	50%	5%	20%	50%	5%	20%	50%	5%	20%	50%	5%	20%	50%	5%	20%	50%
Method	Macro F1			Micro F1			Macro F1			Micro F1			Macro F1			Micro F1		
node2vec	0.663	0.714	0.750	0.627	0.677	0.734	0.437	0.522	0.555	0.375	0.461	0.487	0.760	0.773	0.776	0.739	0.754	0.759
LINE	0.306	0.338	0.363	0.093	0.179	0.243	0.216	0.238	0.256	0.115	0.181	0.208	0.413	0.433	0.441	0.319	0.332	0.333
GAE	0.737	0.771	0.786	0.714	0.744	0.770	0.552	0.577	0.585	0.471	0.501	0.500	0.751	0.764	0.771	0.749	0.761	0.768
VGAE	0.669	0.782	0.817	0.649	0.762	0.807	0.506	0.645	0.684	0.441	0.585	0.620	0.819	0.826	0.829	0.812	0.820	0.824
GraphSAGE	0.622	0.652	0.657	0.520	0.565	0.592	0.608	0.642	0.653	0.526	0.567	0.575	0.645	0.651	0.654	0.620	0.625	0.630
DANE	0.309	0.366	0.451	0.086	0.189	0.316	0.208	0.281	0.414	0.057	0.155	0.294	0.697	0.759	0.786	0.701	0.760	0.787
ASNE	0.353	0.395	0.428	0.178	0.280	0.338	0.234	0.269	0.310	0.155	0.221	0.258	0.676	0.697	0.703	0.663	0.686	0.693
STNE	0.488	0.624	0.673	0.398	0.560	0.638	0.319	0.437	0.488	0.248	0.377	0.417	0.546	0.575	0.583	0.470	0.517	0.534
ARGA	0.477	0.784	0.808	0.407	0.761	0.797	0.312	0.639	0.675	0.250	0.583	0.605	0.407	0.673	0.680	0.306	0.678	0.685
ARVGA	0.529	0.808	0.821	0.474	0.783	0.812	0.341	0.721	0.736	0.280	0.647	0.660	0.400	0.762	0.781	0.221	0.754	0.775
ANRL	0.673	0.747	0.758	0.622	0.709	0.732	0.696	0.735	0.746	0.609	0.679	0.684	0.707	0.742	0.759	0.705	0.742	0.760
CoANE	**0.767**	**0.818**	**0.840**	**0.737**	**0.787**	**0.824**	**0.723**	**0.744**	**0.759**	**0.628**	**0.680**	**0.696**	**0.825**	**0.842**	**0.851**	**0.816**	**0.836**	**0.847**

We mark the rank-1 *and* rank-2 *models in* bold *and* underline*, respectively, in the following tables.*

4.2 Main Results

Node Label Classification. For node classification, we randomly select training and testing samples by varying the percentage of the training set in 5, 20, and 50 percent, and employ one-versus-rest logistic regression classifier with L2 regularization (by following the common-used settings [7]). We utilize Macro-F1 and Micro-F1 as the evaluation metrics, in which higher scores indicate better performance. The results are exhibited in Tables 2 and 3. It can be clearly observed that CoANE consistently leads to the best performance among 11 competing methods across five datasets and two metrics. The improvement of CoANE also keeps stable with various training percentages. Such results prove the usefulness of modeling context co-occurrence (i.e., latent social circles), which is not considered in baselines. Typical models node2vec, ASNE, LINE, and STNE cannot produce higher scores as they cannot well depict mutually-correlated nodes by distinguishing the contribution of links. The autoencoder in DANE is hard to capture the correlation between node attributes and graph structure, so it cannot keep competing scores. Besides, GAE, VGAE, GraphSAGE, and ANRL consider neighborhood aggregation that preserves the local connectivity surrounded each node. Therefore, their scores are higher than other methods, and are closer to our CoANE. However, the discriminator adopted by ARGA and ARVGA built upon GAE/VGAE-based models can improve the performance, but make scores a bit unstable. These outstanding performances of CoANE also demonstrates the effectiveness of its three-way objective.

Node Clustering. We also conduct experiments to examine whether the embeddings generated by CoANE can produce effective node clustering. We employ K-means algorithm, along with the embedding vectors of nodes, to perform clustering. The number K of clusters is given by the number of ground-truth labels. The normalized mutual information (NMI) is used as the evaluation metric. Higher NMI scores imply better performance. Nodes with the same labels are treated as clusters. The results are summarized in Table 4 (right). Additional results on four networks in WebKB data are presented in Table 5. We can apparently find that the proposed CoANE significantly outperforms state-of-the-art methods across five datasets and four networks for WebKB dataset. We can find that the competing methods have lower and unstable scores in all datasets. It implies that modeling attribute-context matrices of specific contexts can better depict the representational features for the target node, which cannot be well captured by node2vec, GAE, VGAE, GraphSAGE, and ANRL. Besides, LINE and ASNE that model lower-order neighbors are hard to distill deeper relationships among nodes. STNE only preserves local features and is hard to encode high-order neighborhoods that can be organized as latent social circles. Moreover, although ARGA and ARVGA further train the discriminators to enhance the preservation of graph topology, their models cannot well capture the meaningful neighborhood that is composed by both graph structure and node features.

Link Prediction. For each dataset, we randomly choose 70, 10, and 20 percent edges as the training, validation, and testing sets, respectively. While training the link prediction model, we randomly sample the same number of non-existing links as negative instances, and ensure the negative instances are not replicated in both sets. Then we employ logistic regression as the classifier. We follow the settings of node2vec [7]: using the Hadamard product of embedding vectors to generate the feature vector of each node pair. The area under the ROC curve (AUC) is utilized as the evaluation metric. The results are shown in Table 4 (left). We can find that the embeddings generated by CoANE outperform those generated by most of the competing methods across five datasets. Such promising results exhibit the effectiveness of modeling a more precise attribute-context correlation by the proposed context generator and convolutional mechanism. That is, competing methods via lower-order proximities (i.e., LINE and ASNE) are obviously not able to distinguish loosely-connected nodes. node2vec cannot incorporate node attributes into embedding learning. Besides, STNE preserves the first-order neighbors and requires more dimensions to encode higher-order latent correlation between nodes, which results in worse performance. Since GAE and VGAE have better information aggregation between network structure and node attributes, they produce higher scores than GraphSAGE and DANE. The discriminator in ARGA/ARVGA and the joint structure-attribute modeling in ANRL make them have

TABLE 3
Macro and Micro F1 Scores for Node Label Classification of WebKB, and Flickr Datasets

Dataset	WebKB						Flickr					
Training ratio	5%	20%	50%	5%	20%	50%	5%	20%	50%	5%	20%	50%
Method	Macro F1			Micro F1			Macro F1			Micro F1		
node2vec	0.448	0.473	0.491	0.169	0.166	0.207	0.437	0.489	0.506	0.400	0.476	0.496
LINE	0.455	0.478	0.500	0.142	0.143	0.166	0.257	0.303	0.328	0.236	0.288	0.317
GAE	0.478	0.478	0.491	0.131	0.129	0.144	0.243	0.251	0.272	0.181	0.195	0.213
VGAE	0.449	0.490	0.530	0.204	0.220	0.270	0.287	0.312	0.347	0.234	0.274	0.314
GraphSAGE	0.483	0.522	0.563	0.183	0.202	0.254	0.145	0.158	0.170	0.098	0.123	0.142
DANE	0.472	0.483	0.511	0.146	0.148	0.182	0.160	0.205	0.233	0.135	0.195	0.228
ASNE	0.451	0.486	0.489	0.151	0.150	0.176	0.395	0.457	0.489	0.362	0.440	0.477
STNE	0.432	0.476	0.487	0.169	0.156	0.200	0.251	0.282	0.301	0.222	0.264	0.281
ARGA	0.434	0.483	0.528	0.152	0.192	0.254	0.155	0.189	0.213	0.131	0.168	0.201
ARVGA	0.431	0.514	0.559	0.166	0.226	0.286	0.159	0.109	0.128	0.095	0.022	0.043
ANRL	0.494	0.512	0.590	0.198	0.190	0.310	0.215	0.286	0.330	0.196	0.278	0.324
CoANE	**0.553**	**0.597**	**0.683**	**0.268**	**0.296**	**0.396**	**0.482**	**0.544**	**0.589**	**0.436**	**0.518**	**0.573**

TABLE 4
AUC Scores for Link Prediction (Left); NMI Scores for Node Clustering (Right)

Task	Link Prediction					Node Clustering				
Method\Dataset	Cora	Citeseer	Pubmed	WebKB	Flickr	Cora	Citeseer	Pubmed	WebKB	Flickr
node2vec	0.896	0.901	0.927	0.684	0.748	0.367	0.149	0.273	0.058	0.165
LINE	0.632	0.626	0.754	0.664	0.648	0.052	0.005	0.003	0.074	0.088
GAE	0.921	0.934	0.947	0.507	0.903	0.374	0.198	0.228	0.007	0.109
VGAE	0.923	0.949	**0.975**	0.639	0.914	0.361	0.157	0.275	0.092	0.131
GraphSAGE	0.757	0.836	0.744	0.700	0.502	0.382	0.305	0.147	0.128	0.037
DANE	0.663	0.768	0.869	0.635	0.901	0.021	0.032	0.148	0.083	0.015
ASNE	0.571	0.586	0.792	0.448	0.848	0.073	0.005	0.165	0.078	0.111
STNE	0.846	0.885	0.880	0.670	0.913	0.207	0.068	0.038	0.069	0.081
ARGA	0.941	0.966	0.920	0.614	0.925	0.452	0.181	0.211	0.092	0.066
ARVGA	0.927	0.972	0.877	0.765	**0.926**	0.530	0.381	0.244	0.104	0.108
ANRL	0.871	0.965	0.769	0.752	0.601	0.391	0.407	0.099	0.132	0.014
CoANE	**0.947**	**0.982**	0.969	**0.784**	**0.926**	**0.544**	**0.435**	**0.313**	**0.180**	**0.211**

relatively better performance. Nevertheless, without the learning of latent social circles, all competing methods cannot achieve stable performance in all datasets. Our CoANE can produce higher NMI scores in most of the five datasets.

Summary. In summary, our CoANE can mostly outperform eleven competing methods on three tasks across five datasets, i.e., CoANE performs the best in 39 out of 40 cases.[5] In the only one case (1 out of 40 cases) that CoANE cannot work the best (i.e., the link prediction task on Pubmed dataset), CoANE's performance is still quite close to the best competing methods. We find that most competitive models have the neighborhood aggregation mechanism (e.g., VGAE and ARVGA) because generating embeddings by aggregating information from neighbors can better fuse graph structure and node attributes. The superiority of our CoANE comes from not only neighborhood aggregation (by the convolutional mechanism), but the finer-grained modeling of neighborhoods in terms of context co-occurrences, i.e., the latent social circles. In other words, the merit of CoANE mainly lies in the convolutional mechanism that captures both intra-hop and inter-hop feature correlation, and the comprehensive design of loss function to better preserve structural and semantic knowledge in node embeddings. In addition, CoANE leads to the best time efficiency while the strong baselines VGAE and ARGA require more training time. Last, we have realized that CoANE cannot work the best for predicting links in the graph with extremely high sparsity (i.e., Pubmed whose density is 0.002). We think the reason is that high sparsity makes the latent social circles less evidential during training.

4.3 Model Analysis

Embedding Visualization. To further present the embedding results of the proposed CoANE, we visualize the node representation using t-SNE [29]. The plots allow us to see the global property of embeddings and understand whether nodes with the same labels are close enough in the embedding space. Note that due to the page limit, we only show the results of three competing methods using Cora dataset. The visualization is shown in Fig. 3, in which nodes are colored according to their labels. It can be apparently found that our CoANE can lead to more compact and well-separated clusters than VAGE and ARVGA. Although ARNL can also generate a clear separation of groups, our CoANE can further attract those possessing the same labels to be

5. 40 cases = 5 datasets × 8 task settings, in which 8 tasks consists of 3 training percentages (5, 20, 50 percent) for node label classification with 2 metrics (Macro F1 and Micro F1), node clustering (NMI), and link prediction (AUC).

TABLE 5
NMI for Clustering on WebKB Networks

Method\Dataset	Cornell	Texas	Washington	Wisconsin
node2vec	0.066	0.070	0.044	0.053
LINE	0.066	0.093	0.085	0.051
GAE	0.002	0.000	0.027	0.000
VGAE	0.086	0.081	0.103	0.096
GraphSAGE	0.105	0.157	0.140	0.111
DANE	0.067	0.087	0.118	0.061
ASNE	0.066	0.094	0.103	0.047
STNE	0.071	0.088	0.065	0.052
ARGA	0.086	0.093	0.099	0.091
ARVGA	0.091	0.094	0.128	0.101
ANRL	0.114	0.116	0.167	0.131
CoANE	**0.191**	**0.200**	**0.181**	**0.148**

much closer and push those with different labels farther away from each other. Such visualization also unveils why CoANE can achieve better performance on both supervised and unsupervised tasks.

Sensitivity Analysis. We analyze three hyperparameters and training runtime in CoANE: (a) length of contexts, (b) number of sampled sequences for random walk, and (c) embedding dimension. For (a) and (b), we use WebKB and compare with node2vec, and consider CoANE without the attribute preservation for the analysis. For (c), we apply link prediction for CoANE in various embedding dimensionalies, and present the results of training and test sets.

(a) *Length of contexts.* The length of contexts represents the size of neighbors. Larger sizes would result in high computation cost. We show how the context length affects the performance of link prediction (AUC) and node clustering (NMI) in Fig. 4a. The stable results in both AUC and NMI inform us that as the size of neighbors gets higher (i.e., larger context length), the performance can be kept and does not change too much. The context length $= 3$ may be enough. We think the reason is that local information is enough for link prediction and node clustering.

(b) *Number of sampled sequences.* In the random walk, the number of sampled node sequences can affect the quality of embeddings because fewer sampled sequences provide less neighborhood information. By varying the number of sampled sequences, as shown in Fig. 4b, we compare the performance of link prediction (AUC) between node2vec and CoANE using WebKB data. We can find that node2vec needs at least two sampled sequences for achieving stable performance. As for CoANE, requiring only one sampled sequence can lead to earlier stable results.

(c) *Dimension of embeddings.* We show how the dimension of embeddings affects the performance of link prediction. The results shown in Fig. 4c demonstrate that a bit higher dimensionality can lead to better performance in both training and testing. The performance gets stable when the dimensionality is larger than 150, implying that most information on network structure and node attributes is preserved.

Runtime Analysis. We present runtime analysis to understand the time efficiency of competing methods using the larger-scale Pubmed data. We conduct link prediction and report the AUC scores (y-axis) in validation and testing, and the training time in seconds for each epoch (x-axis) for CoANE and two stronger competing methods, VGAE and ARGA. The

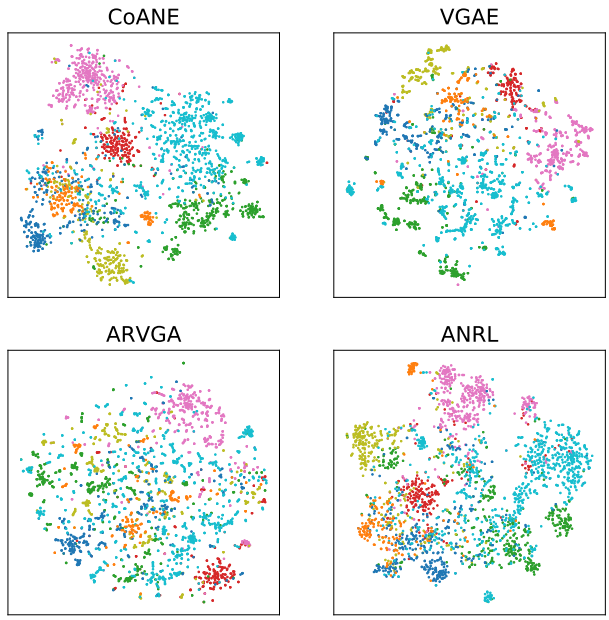


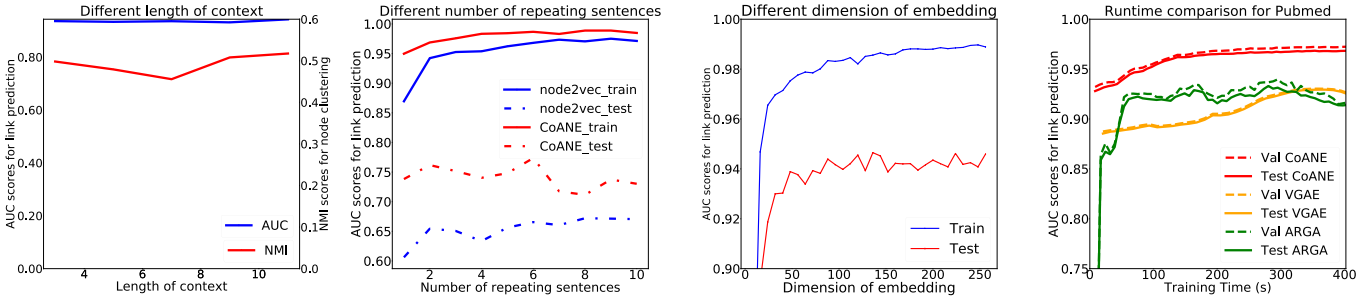
Fig. 3. Visualizing various approaches for Cora data.

experiment is performed under Google Cloud Platform whose computing environment contains 8 vCPU with 30 GB memory and 1 NVIDIA Tesla K80 (12 GB GDDR5). The results are exhibited in Fig. 4d. We can clearly find that both VGAE and ARGA require more training time to reach their converged performance. It is because VGAE needs to indirectly aggregate the higher-order context neighbors, which results in taking more time to capture latent features. With the benefit of the discriminator, ARGA gains the improvement of effectiveness and efficiency. However, ARGA ignores the aggregation of related neighbors and cannot find all latent links. CoANE leads to high AUC scores with fast convergence (in only one epoch) in terms of training time. It is because CoANE can extract more representative contexts and utilize multiple filters that efficiently can learn important features.

4.4 Discussion on CoANE Superiority

Experimental results show that CoANE leads to better and stable performance than existing solutions. The fundamental reason is the modeling of latent *social circles* by the proposed multi-channel 1-D convolutional layer and the information preservation in CoANE. It can be further divided into two parts. First, CoANE can directly learn how subsets of neighbors sharing similar attributes can shape the potential *social circles*, which cannot be well captured by joint structure-attribute learning-based models. Second, CoANE can give different distributions of weights to neighbors in the same hop and different hops away from the target node, which cannot be achieved by graph autoencoder-based models. Therefore, CoANE allows more flexibility and provides more deliberate learning in trainable parameters for fine-grained social circle modeling. The details are summarized as below.

Joint structure-attribute learning-based models, including DANE [4], ASNE [16], and ANRL [35], first independently encode respective information (either graph structure or node attributes), then learn the correlation between structure and attributes, and have two prediction tasks: preserve the target/neighbor node(s) and reconstruct the attributes. However, in



(a) context length (b) #sampling walks (c) embedding dimension (d) runtime in seconds

Fig. 4. (a)-(c) Sensitivity analysis for (a) length of contexts, (b) number of sampled sequences for random walk, (c) embedding dimension. (d) Runtime analysis.

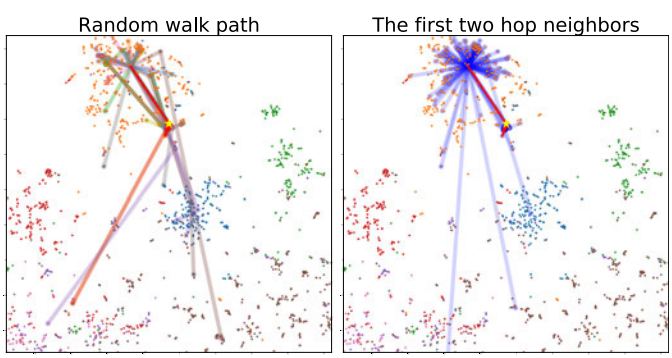
this way, the direct interactions between graph structure and node attributes cannot be captured. These methods simply treat node attributes as independent input, instead of viewing individual node attributes together with graph neighborhood. Therefore, they cannot learn the latent *social circles* of the target node represented by how some neighbors possess similar attributes.

Graph autoencoder models, including GAE/VGAE [13] and ARGA/ARVGA [23], can simultaneously model network structure and node attributes. Besides, ANRL [35] fuses the technologies of jointly feature learning and graph aggregation to preserve more information. However, their recursive scanning of same-hop neighbors from the first to higher order cannot capture the target node's *social circles* (e.g., such as "CS dept", "family", and "labmates"), which can span across different orders of neighbors. In other words, all neighbors in the same order are treated as having equal importance. They cannot distinguish same-hop neighbors from each other.

4.5 Discussion of Information Preserved by CoANE

We discuss the effectiveness with different designs of CoANE by considering its three main components, including (a) random walk, (b) convolutional mechanism, and (c) objective function. We apply link prediction on Cora dataset with the same settings as the previous evaluation, and exhibit the performance in the corresponding cases.

(a) *Random walk*. We pay attention to the contribution of our random walk and the original neighbor selection. First, we randomly choose a node and show the coverage of its neighbors in the network by depicting their paths on the t-SNE embedding plots. In Fig. 5, the asterisk represents the chosen node, and red lines are the real edges in the network. The paths in various colors in Fig. 5a are contexts extracted by the random walk with the length of window $= 5$, and the paths in blue in Fig. 5b are the first two hop neighbors of the chosen node. We observe both of their regions cover most of the orange points but reach some nodes with other colors, especially for the random walk. This problem could be solved by paying different attention to each position in our CoANE model because these uncorrelated neighbors are located at the tail of the contexts. In addition, we can also find that the main region of the random walk paths (Fig. 5a) is more concentrating than the region of the first two hop neighbors (Fig. 5b). Such an effect implies that our random walk can help model better distinguish nodes that belong to the same clusters. Second, we aim at presenting how these two neighbor selection cases affect the performance of link prediction. For fair and simplified comparison, we set the context length $= 1$ for the random walk case, and consider the first-hop neighbors as another case for comparison. In addition, we make the average number of generated contexts for two cases as close as possible by repeatedly generating the contexts. Eventually, we have 17.5 and 22 contexts per node for the cases of random walk and the first-hop neighbors, respectively. The results are displayed in Fig. 6a (solid lines). The increment is obvious when using random walk contexts. Such a result indicates that the choice of nodes' neighbors is crucial for embedding learning.

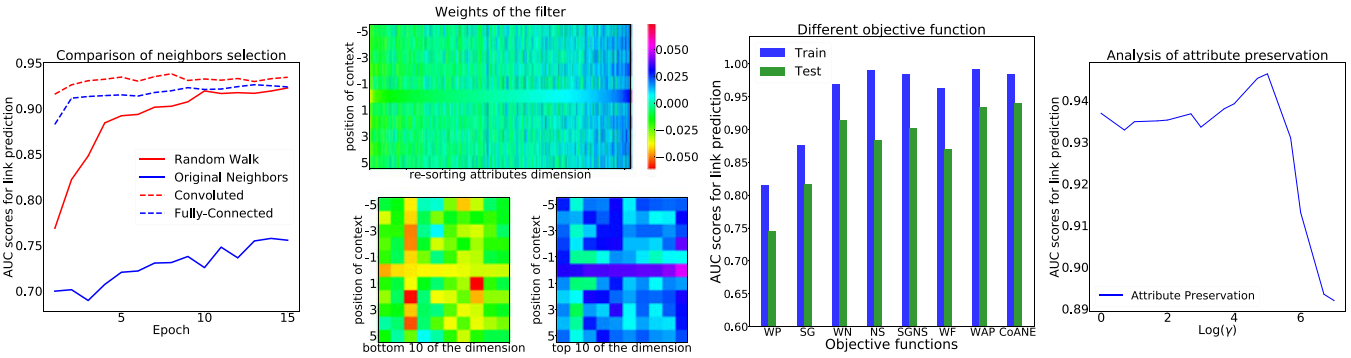


(a) Random walk paths (b) First two hop neighbors

Fig. 5. Analyzing the neighbor selection via Cora dataset.

(b) *Convolutional mechanism*. We discuss the preservation of positional information in the context by the learning layer comparison and the filter weights study. First, we compare the selection of feature extraction layer, including the convolution used by CoANE and the general mapping, i.e., fully connected (FC) layer. Applying the FC layer means that each node's features in the context are learned by the same parameters. The results are shown in Fig. 6a (dashed lines). The convolutional layer leads to better performance and faster convergence, compared to the FC layer. The results imply that the positional information of nodes in the context should be considered. Since the neighborhood's network topology can to some extent be depicted by positional information in the generated contexts, the convolution exactly offers a more compatible mechanism to extract diverse positional properties. That is, CoANE can capture node features and their relationships by using only one convolutional layer.

Furthermore, we exhibit the weights of filters to explain the effectiveness of CNN. Although we have no knowledge about the attributes in the datasets, we expect that filters can give



(a) Neighbor and extractor. (b) Visualizing filters. (c) Effect of objective functions (d) Attribute preservation.

Fig. 6. Analyzing each component in CoANE using Cora data.

similar weights on specific attributes between the central node and its neighbors (contexts). The learned weights of CNN filters are shown in Fig. 6b. In each subfigure, the x-axis represents the attribute dimensions, and the y-axis indicates the positions of the central node and its context nodes. Colors are used to display the weight values in the filters. To have a better observation, we sort the attributes dimensions of filters by the weights of central nodes. The top subfigure presents the weights of filters for all dimensions while the bottom subfigures show the weights in the bottom and the top 10 dimensions. We can clearly find that the attribute weights of midst nodes with higher weights (violet) are often accompanied by higher weights of their neighbors. Such results imply that filters concentrate on similar attributes and the positional information in the context. These findings become more obvious when looking into the bottom and the top 10 dimensions. Most neighbors of the midst have higher positive correlated weights; otherwise, and the rest have weights close to zero (cyan). In short, various filters are truly proficient in both searching features of their targets and giving weights according to the positions.

(c) *Objective function*. We present the contribution of modeling context co-occurrence by our likelihood functions. Eight cases are discussed: (1) CoANE without positive graph likelihood (WP) (i.e., set $L_{pos} = 0$), (2) using the general skip-gram model to replace positive graph likelihood (SG), i.e., simply computing dot product similarity for pairs of midst and neighbor, (3) CoANE without contextually negative sampling (WN) (i.e., set $L_{neg} = 0$), (4) using general negative sampling to replace contextually negative sampling (NS), i.e., simply computing dot product similarity for a fixed number of negative samples via uniformly random selection, (5) combining skip-gram model with negative sampling to replace CoANE's positive and negative loss (SGNS) (i.e., (2) + (4)), (6) CoANE using the original network data without the attribute information (WF), (7) CoANE without the attribute preservation (WAP), and (8) the complete CoANE. We compare their performance in training and testing AUC using link prediction task.

The results are shown in Fig. 6c. It can be clearly found that changing or removing any required components of CoANE brings performance damage. The effectiveness of each component of CoANE has been verified. By looking into the details, the worse AUC scores of WP and SG (i.e., without proper positive loss terms) prove the usefulness of our context co-occurrence matrices and positive graph likelihood. The worse AUC scores of WN and NS (i.e., original settings of negative sampling) imply that it is effective to have our contextually negative sampling, and higher training scores and lower testing scores may indicate the potential overfitting of WN and NS. Contextually negative sampling can better deal with overfitting. In addition, the AUC scores of combining skip-gram model and original negative sampling (SGNS) are not worse, comparing to the complete CoANE. We think it is because the model is still based on the proposed modeling of context co-occurrence that effectively distills features from network structure and node attributes. Besides, the significant AUC difference between WF and the complete CoANE shows the contribution of attributes in embedding learning and the prediction task.

For WAP, removing attribute preservation makes the model fit the training data well; nevertheless, some attributes can still benefit the model learning and improve the performance, as exhibited by the complete CoANE. We analyze the capability of attribute preservation by changing the attribute preservation controller γ in Eq. (4) based on the same experimental settings. We display the results in terms of AUC scores by varying $\log(\gamma)$ in Fig. 6d. It can be found that the curve is first increasing, and then goes down as the increment of $\log(\gamma)$. The reason is that much smaller attribute preservation does not affect the model learning; however, larger γ values would greatly dominate the embedding learning, i.e., focusing more on attribute preservation and weakening structure learning. In the setting with $\log(\gamma) = 5$ (i.e., $1e5$), the attribute preservation can make CoANE achieve better performance.

5 Conclusion

This paper proposes a novel context co-occurrence-aware attributed network embedding, CoANE. The main idea is to preserve three-fold information, the network structure, node attributes, and distilling the proper convolution between network structure and node attributes through specific contexts. When applying the embeddings generated by CoANE, we prove that the performance conducted on three essential network analysis tasks, including link prediction, node label classification, and node clustering, can get significantly and consistently boosted across five real datasets, comparing to state-of-the-art competing methods. Such results clearly exhibit the effectiveness of CoANE. A number of advanced analyses on filters' weights, parameter sensitivity, and model contributions provide a robust experimental study, and the results unfold where the superiority of CoANE comes from.

ACKNOWLEDGMENTS

This work was supported in part by the Ministry of Science and Technology (MOST), Taiwan under Grants 109-2636-E-006-017 (MOST Young Scholar Fellowship) and 109-2221-E-006-173, and in part by Academia Sinica under Grant AS-TP-107-M05.

REFERENCES

[1] S. Abu-El-Haija, B. Perozzi, and R. Al-Rfou, "Learning edge representations via low-rank asymmetric projections," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.,"* 2017, pp. 1787–1796.
[2] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 833–852, May 2019.
[3] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 135–144.
[4] H. Gao and H. Huang, "Deep attributed network embedding," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3364–3370.
[5] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[6] M. Grbovic and H. Cheng, "Real-time personalization using embeddings for search ranking at airbnb," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 311–320.
[7] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.
[8] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034,.
[9] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2017, pp. 731–739.
[10] X. Huang, Q. Song, J. Li, and X. Hu, "Exploring expert cognition for attributed network embedding," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 270–278.
[11] X. Huang, Q. Song, Y. Li, and X. Hu, "Graph recurrent networks with attributed random walks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 732–740.
[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," , 2014, *arXiv:1412.6980*.
[13] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *in Proc. NIPS Workshop Bayesian Deep Learn.*, 2016.
[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.
[15] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 539–547.
[16] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2257–2270, Dec. 2018.
[17] J. Liu, Z. He, L. Wei, and Y. Huang, "Content to node: Self-translation network embedding," in *Proc.24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1794–1802.
[18] Y. Liu and Y.-F. Brook Wu, "Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 354–361.
[19] Z. Meng, S. Liang, J. Fang, and T. Xiao, "Semi-supervisedly co-embedding attributed networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6504–6513.
[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. Int. Conf. Learn. Representations*, 2013.
[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J.f Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
[22] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the Web," Stanford InfoLab, Tech. Rep. 1999–66, 1999.
[23] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 2609–2615.
[24] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.
[25] N. Sheikh, Z. Kefato, and A. Montresor, "gat2vec: representation learning for attributed graphs," *Computing*, vol. 101, no. 3, pp. 187–209, 2018.
[26] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.
[27] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *Proc. 6th Int. Conf. Data Mining*, 2006, pp. 613–622.
[28] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, "Deep recursive network embedding with regular equivalence," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2357–2366.
[29] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol.9, no. 86, pp. 2579–2605, 2008.
[30] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018.
[31] H. Wang *et al.*, "A united approach to learning sparse attributed network embedding," in *Proc. IEEE Int. Conf. Data Mining*, 2018, pp. 557–566.
[32] X. Xin, F. Yuan, X. He, and J. M. Jose, "Batch is not heavy: Learning word representations from all samples," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 1853–1862.
[33] W. Yu *et al.*, "Learning deep network representations with adversarially regularized autoencoders," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2663–2671.
[34] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "User profile preserving social network embedding," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 3378–3384.
[35] Z. Zhang *et al.*, "ANRL: Attributed network representation learning via deep neural networks," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3155–3161.

I-Chung Hsieh received the master's degree in science in statistical science from National Chung Cheng University, Chiayi, Taiwan, in 2017. Since 2018, he has been a research assistant with the Networked Artificial Intelligence Laboratory, National Cheng Kung University, Tainan, Taiwan. His research interests include privacy protection on graph, graph representation learning, data mining, and deep learning. His current research has been accepted and presented at the Conference and Workshop on Neural Information Processing Systems Graph Representation Learning 2019.

Cheng-Te Li (Member, IEEE) received the PhD degree from National Taiwan University, in 2013. He is currently an associate professor with the Institute of Data Science and Department of Statistics, National Cheng Kung University, Tainan, Taiwan. From 2014 to 2016, he was an assistant research fellow with The Research Center for Information Technology Innovation, Academia Sinica. He has authored and coauthored papers in top conferences, including Knowledge Discovery in Databases, World Wide Web, IEEE International Conference on Data Mining, Conference on Information and Knowledge Management , Special Interest Group on Information Retrieval, International Joint Conference on Artificial Intelligence, Annual Meeting of the Association for Computational Linguistics,Empirical Methods in Natural Language Processing, North American Chapter of the Association for Computational Linguistics, ACM Recommender Systems Conference, and ACM Multimedia Conference. His research interests include machine learning, deep learning, data mining, social networks and social media, recommender systems, and natural language processing.