Jointly Learning Representations of Nodes and Attributes for Attributed Networks

ZAIQIAO MENG, University of Glasgow
SHANGSONG LIANG, Sun Yat-sen University
XIANGLIANG ZHANG, King Abdullah University of Science and Technology
RICHARD MCCREADIE and IADH OUNIS, University of Glasgow

Previous embedding methods for attributed networks aim at learning low-dimensional vector representations only for nodes but not for both nodes and attributes, resulting in the fact that node embeddings cannot be directly used to recover the correlations between nodes and attributes. However, capturing such correlations by embeddings is of great importance for many real-world applications, such as attribute inference and user profiling. Moreover, in real-world scenarios, many attributed networks evolve over time, with their nodes, links, and attributes changing from time to time. In this article, we study the problem of jointly learning low-dimensional representations of both nodes and attributes for static and dynamic attributed networks. To address this problem, we propose a Co-embedding model for Static Attributed Networks (CSAN), which jointly learns low-dimensional representations of both attributes and nodes in the same semantic space such that their affinities can be effectively captured and measured, and a Co-embedding model for Dynamic Attributed Networks (CDAN) to dynamically track low-dimensional representations of nodes and attributes over time. To obtain effective embeddings, both our co-embedding models, CSAN and CDAN, embed each node and attribute with means and variances of Gaussian distributions via variational auto-encoders. Our CDAN model formulates the dynamic changes of a dynamic attributed network by aggregating perturbation features from the nodes' local neighborhoods as well as attributes' associations such that the evolving patterns of the given network can be tracked. Experimental results on real-world networks demonstrate that our proposed embedding models outperform state-of-the-art non-dynamic and dynamic embedding models.

CCS Concepts: • **Information systems → Data mining**; • **Computing methodologies → Neural networks**;

Additional Key Words and Phrases: Attributed network, network embedding, dynamic embedding, variational auto-encoder

ACM Reference format:
Zaiqiao Meng, Shangsong Liang, Xiangliang Zhang, Richard McCreadie, and Iadh Ounis. 2020. Jointly Learning Representations of Nodes and Attributes for Attributed Networks. *ACM Trans. Inf. Syst.* 38, 2, Article 16 (January 2020), 32 pages.
https://doi.org/10.1145/3377850

This work was partially supported by the National Natural Science Foundation of China (NSFC No. 61906219 and NSFC No. 61828302). Zaiqiao Meng, Richard McCreadie and Iadh Ounis have been partially funded by the European Community's Horizon 2020 research and innovation programme under grant agreement no. 779747.

Authors' addresses: Z. Meng, R. McCreadie, and I. Ounis, University of Glasgow; emails: zaiqiao.meng@gmail.com, {richard.mccreadie, iadh.ounis}@glasgow.ac.uk; S. Liang (corresponding author), Sun Yat-sen University; email: liangshangsong@gmail.com; X. Zhang, King Abdullah University of Science and Technology; email: xiangliang.zhang@kaust.edu.sa.

1046-8188/2020/01-ART16 $15.00
https://doi.org/10.1145/3377850

1 INTRODUCTION

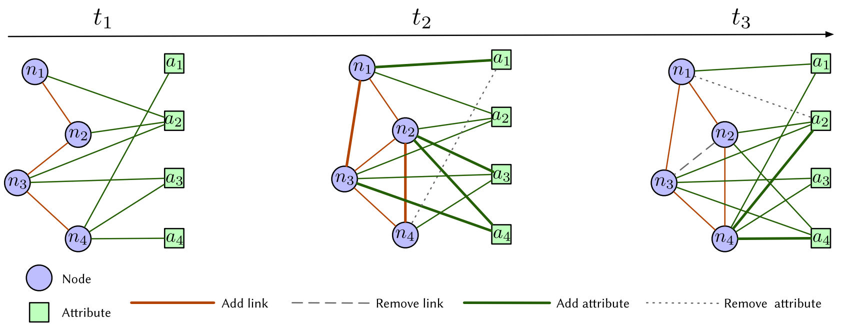
Network embedding techniques that learn low-dimensional vector representations of nodes in a network have been increasingly attracting attention in recent years. Performances of various network-based applications, such as node classification [42, 47], node clustering [3], community detection [14, 53], and link prediction [30], have been shown to be significantly improved via utilizing embeddings of nodes inferred by these network embedding techniques.

Attributed networks constituting one category of the most important networks are ubiquitous in a myriad of important domains, ranging from social media networks to academic networks, where rich attributes, e.g., ages of the users and journals/conferences the authors published at, that describe the properties of the nodes are available. A number of algorithms of embedding attributed networks have been proposed to learn low-dimensional vector representations of nodes via the leverage of the structure and/or the attributes information of the networks [15, 18, 28, 31, 55]. By doing so, node embeddings of attributed networks can be effectively obtained, while the structure and/or the attribute information of the networks can be embedded into these vector representations, which in turn help to enhance the performance of many down-stream applications [15, 18, 27, 33].

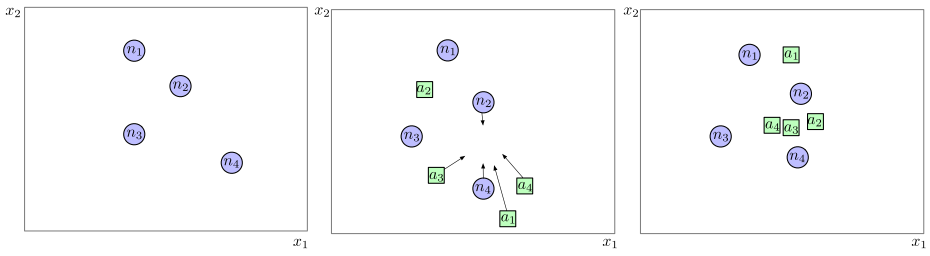
However, these existing embedding methods focus on mapping an attributed network into representation vectors (i.e., embeddings) for nodes only but not for both nodes and attributes, resulting in the fact that node embeddings cannot be directly used to recover the correlations (or affinities) between nodes and attributes. However, recovering such correlations between nodes and attributes by embeddings is of great importance to the success of many tasks, where the relationships between nodes and attributes need to be directly quantitatively measured, such as connecting users and keywords in user profiling [32], annotating users with tags in attribute inference [13, 56], and characterizing experts by topics in expert finding [20]. Moreover, a vast majority of existing methods only focus on embedding problems for **static** networks, ignoring the fact that some real-world networks such as social networks and biological networks are typically **dynamic**, where the edges and nodes' attributes are evolving over time. Encoding a dynamic network by the embeddings of different points in time, however, is crucial to understand mechanisms for how the networks evolve over time and to make good predictions for the future. In this article, we refer those networks with both network structure and node attributes changing over time as **dynamic attributed networks**.

To illustrate these disadvantages of the existing methods, we provide a toy dynamic attributed network with three snapshots in Figure 1. One can represent this toy attributed network by node embeddings based on an existing embedding method (e.g., GCN [28]), as shown in Figure 1(b). Although the network structure and the attribute information can be easily embedded into these node embeddings, it is still nontrivial to recover the correlations between nodes and attributes by these node embeddings. In contrast, when embedding both nodes and attributes in a same embedding space, as shown in Figure 1(c) and Figure 1(d), we can directly recover the correlations between nodes and attributes by the distance function (e.g., dot product) of these two types of embeddings. In addition, as the toy network changes over time, we can see that the positions of nodes and attributes change in the embedding space (Figure 1(c) and Figure 1(d)). How to learn dynamic embedding capturing the evolving trend from the observed network changes (time step t_1 and t_1) to predict links and attributes for the future (time step t_3) remains unsolved.

To alleviate the aforementioned problems, we introduce two models—one for static attributed networks, the Co-embedding model for Static Attributed Networks, abbreviated as CSAN, which aims to learn low-dimensional vector representations of both attributes and nodes for a static



(a) A toy dynamic attributed network



(b) Node embeddings (c) Node and attribute embeddings at time t_1 (d) Node and attribute embeddings at time t_2

Fig. 1. Given a toy dynamic attributed network with four nodes and four attributes (a), existing methods map the network into the embeddings space of nodes only (b). Our proposed CSAN model maps the network into an embedding space with both nodes and attributes (c). The learned representation encodes the correlations between nodes and attributes so it can be directly recovered by standard distance functions (e.g., dot product). Our proposed CDAN learns dynamic embeddings for each time step while the evolving trends can be captured from the observed network changes (time step t_1 and t_2) so it can predict links and attributes for the future (time step t_3).

attributed network in the same semantic space such that their affinities can be effectively captured and measured, and another for dynamic attributed networks, the Co-embedding model for Dynamic Attributed Networks, abbreviated as CDAN, which aims to learn low-dimensional vector representations of both attributes and nodes for a dynamic attributed network dynamically such that the evolving patterns of the given network can be tracked. To effectively infer the embeddings of both nodes and attributes in an attributed network, we use the variational auto-encoder to infer the embeddings of both nodes and attributes and represent them by means of Gaussian distributions, with the corresponding variances measuring the uncertainty of the inferred embeddings. Unlike previous network embedding models that only represent nodes, both our co-embedding models obtain embeddings of both nodes and attributes in a unified manner, and our CDAN model is able to track low-dimensional representations of nodes and attributes over time. Specifically, our CSAN model encodes a static attributed network into Gaussian embeddings of both nodes and attributes by the variational auto-encoder so that the affinities between nodes and attributes and the uncertainty of the embeddings can be effectively preserved. Our CDAN model further formulates the dynamic changes of a dynamic attributed network by the first-order and higher-order perturbations between two consecutive time steps of original dynamic network. Consequently, the dynamics of nodes, edges, and attributes in the network can be effectively captured by the embeddings of nodes and attributes.

The node and attribute embeddings obtained in a unified manner in our models can benefit not only **node-oriented** network problems (e.g., node classification and link prediction), but also **attribute-oriented** problems (e.g., predicting the values of attributes of nodes). More importantly, the common semantic embedding space provides a simple but effective solution to the **user profiling** problem, as the relevance of users (nodes) and keywords (attributes) can be directly measured, e.g., by cosine similarity, or a dot product. By utilizing the embedding results of our CDAN model, we are able to, for instance, reconstruct the network topology and predict the attributes of the nodes in the future.

To verify the effectiveness of our proposed models, we conduct experiments on several real-world network datasets and aim to seek answers to the following core research question: Can our CSAN and CDAN models obtain more informative and better quality representations than the previous network embedding models? Our experimental results demonstrate that CSAN and CDAN yield excellent performances in attribute inference and user profiling tasks, which the previous network embedding models cannot deal with, while they also show high competitive performances in link prediction and node classification tasks compared with both the state-of-the-art network embedding models and dynamic network embedding models.

Our contributions in this article can be summarized as follows:

(1) We propose two novel network representation learning models to jointly learn the low-dimensional representations of both nodes and attributes for static and dynamic attributed networks, respectively. Instead of training a distinct embedding vector for each node, our models return the embedding vectors of both nodes and attributes in the same semantic space, such that the affinities between nodes and attributes can be effectively measured.
(2) To learn effective embeddings of both nodes and attributes in attributed networks, we propose a customized variational auto-encoder in our models, which consists of an inference model, aiming at encoding features of nodes and attributes into Gaussian distributions, and a generative model, which reconstructs both real and binary weighted edges and attributes. With the nature property of Gaussian embeddings obtained by the encoder, our models innately capture the uncertainty of representations.
(3) We model the dynamic changes of a dynamic attributed network between two time steps through aggregation functions that learn to aggregate changes propagated from the lower-order perturbation of the nodes' local neighborhoods as well as attributes' associations.
(4) We conduct extensive experiments on 10 real-world static and dynamic attributed networks to verify the effectiveness of our proposed models by addressing four graph mining tasks, i.e., node classification, link prediction, attribute inference, and user/expert profiling. Experimental results show that our models are able to learn informative and high-quality representations and significantly outperform the state-of-the-art methods.

This article is a substantially extended version of our previous paper [37], where we proposed a static model for co-embedding attributed networks. The main extension of this article are twofold. First, we propose a new model for embedding dynamic attributed networks by aggregating perturbation features from nodes' and attributes' local neighbourhoods and associations such that the evolving patterns of the given network can be tracked. Second, we conducted more experiments to demonstrate the advances of both the static model proposed in our previous paper [37] and the dynamic model proposed in this extension, reported and analyzed the experimental results with more refined tables and figures. For learning representation of attributed network in a semi-supervised environment, please refer to Reference [38].

The remainder of the article is organized as follows. In Section 2 we briefly review the related work. Section 3 introduces the notations, provides the preliminary of variational auto-encoders, and formally defines the problems to be addressed. Section 4 and Section 5 present the details of our proposed models, CSAN and CDAN, respectively. Section 6 describes the experimental setup. Section 7 presents the experimental results and analysis. Finally, we conclude the article in Section 8.

2 RELATED WORK

Our work relates to the topic of network embedding that learns distributed representation for various networks, to handle which many different approaches have been proposed. Existing methods for network embedding can be classified into three categories based on the characteristics of the real-world networks, namely, the plain network embedding methods, the attributed network embedding methods, and the dynamic network embedding methods. In this section, we briefly review the related literature in these three domains.

2.1 Plain Network Embedding

Early works of network embedding [3, 14, 42, 47, 51] mainly focus on plain networks, in which only the topological structure information (i.e., only nodes and edges) can be utilized as input for learning latent representations. These methods try to represent a plain network with a lower-dimension space where the proximities between nodes and the local or global geometry structure of the network can be preserved. For instance, approaches such as DeepWalk [42] and LINE [47] are the classical exploration pioneers of network embedding, inspired by word embedding [39]. These methods generate paths or edges as training context based on random walks or edge sampling methods and then feed them into the Skip-Gram model to unsupervisedly learn node representations, where the negative sampling trick [39] can be used to enhance the efficiency. A lot of following works (e.g., node2vec [14] and PTE [46]) are extended from them, and some theoretical work have proved that these models can be implicitly approximated by matrix factorization [43]. However, these methods only use some shallow layer neural networks to train the Skip-Gram model, which cannot capture the highly non-linear network structure. To deal with this issue, Wang et al. [51] proposed a structural deep network embedding method to capture the highly non-linear network structure and preserve the global and local structure of the network. Wang et al. [53] integrated the community structure of network into result embeddings to preserve both of the microscopic and community structures. However, based on the limited input information from a plain network, these methods can only encode a network by preserving three kinds of observations, i.e., the first proximity represented by edges between nodes, the second proximity shared neighbourhood structures of the nodes, and community patterns observed in networks, while they ignored the fact that attributes of nodes can also provide useful information for encoding nodes as vectors.

2.2 Attributed Network Embedding

Attributed networks are ubiquitous in many real-world networks, such as social networks and scientific collaboration networks, to handle which a lot of recent works [15, 18, 28, 31, 55] have been proposed for embedding attributed networks, where not only the topological structure information but also auxiliary information, e.g., the attribute/content information of nodes, are taken into account for embedding. The rich available of auxiliary information of attributed network makes the embedding methods more promising to achieve better representations by taking a full advantage of both the topological structure and node attributes of the attributed networks [15, 18, 28, 52]. For instance, the TADW model [55] is an attributed network embedding model that incorporates DeepWalk and associated text features into the matrix factorization framework. Huang et al. [18] employed the graph Laplacian technique to learn the joint embedding from the topological structure and attributes. Zhang et al. [57] and Gao and Huang [11] proposed customized deep neural network architectures, called ANRL and DANE, respectively, to learn node embeddings while capturing the underlying high non-linearity in both the topological structure and attributes. Other approaches obtain embeddings for non-plain networks with other auxiliary information, such as labels [19], text contents [55], and edge types [41], in addition to node attributes. All experimental results of these papers showed that combining different types of auxiliary information, rather than using only the topological features, can provide different insights for the embedding of nodes and help capture rich patterns in many real-world networks.

Borrowing the concept of a convolution filter for the signal processing, Kipf and Welling [28] proposed a graph convolutional neural network model (GCN) for attributed networks by devising a specialized graph convolution operation over nodes and their neighbours. GCN properly interoperates attributed feature of nodes by the neighbourhood aggregation procedure, and it also can be easily applied to the semi-supervised learning tasks [38]. Many of other approaches broadly follow the graph convolution operation (or the "neighbourhood aggregation" scheme) and have been shown their effectivenesses [5, 15, 54]. For instance, Hamilton et al. [15] proposed the GraphSAGE model that learns node representations by sampling and aggregating features from a node's local neighbourhood. To adapt to local neighbourhood properties and tasks, Xu et al. [54] explored the jumping knowledge networks architecture that flexibly leverages different neighbourhood for each node ranges to obtain better structure-aware representation. A variational auto-encoder architectural neural network extension of GCN also can be found in Reference [27], which is referred to as the Variational Graph Auto-Encoder (GAE). Our work in this article follows the framework of GAE, but our proposed models further extended it to the co-embedding situation that affinities between nodes and attributes need to be captured, and the dynamic situation such that the evolving patterns of the given network need be tracked.

Recently, few approaches embedded nodes by distributions (e.g., Gaussian distribution) and captured the uncertainty of the embeddings [2, 6, 17]. Inspired by previous work of learning Gaussian word embeddings [50], He et al. [17] proposed a knowledge graph embedding model that represents each entity/relation of knowledge graphs as a Gaussian distribution. Dos Santos et al. [6] studied heterogeneous graphs for node classification using Gaussian embeddings. In terms of embedding for attributed networks, Bojchevski and Günnemann [2] embedded each node as a Gaussian distribution according to the energy-based loss of personalized ranking formulation. We tackle the attributed network embedding problem by modeling both nodes and attributes as Gaussian distributions, which permit the measuring of the uncertainty of both embeddings by the variances of the distributions.

2.3 Dynamic Network Embedding

Capturing dynamic changes by embeddings of entities for many real-time-evolving datasets has recently attracted increasing attention. Kim et al. [22] and Hamilton et al. [16] modelled semantic changes of words by splitting the data into separate time bins and training the model by some word embedding models, such as word2vec [39]. Esteban et al. [9] proposed multi-way neural network architecture for modelling the temporal evolution of knowledge graphs and the evolution of associated events and signals. Trivedi et al. [49] proposed the Know-Evolve model to learn dynamic knowledge graph embeddings by modelling the occurrence of a fact (edge) as a multivariate point process whose intensity function is modulated by the score for that fact computed based on the learned entity embeddings. Modelling temporal information by embeddings of real datasets have shown to be beneficial for a number of temporal applications, such as dynamic user profiling [32, 49] and temporal summarization [35, 36].

To deal with dynamic networks, a lot of dynamic network embedding approaches have been proposed. For instance, Zhu et al. [59] proposed a temporal latent space model for dynamic networks to model the temporal link probability of node pairs for the link prediction task. Zhou et al. [58] presented a model to track the evolution patterns of triads for plain networks by their so-called triadic closure process. Li et al. [31] built an online model based on an offline model to maintain the freshness of embedding on nodes and attributes by leveraging matrix perturbation theory. Du et al. [8] modified the Skip-gram framework to enable it to learn embeddings for dynamic plain networks. But it is still nontrivial to extend their dynamic Skip-gram framework to the dynamic attributed networks and the co-embedding situation. Zuo et al. [60] proposed a temporal network embedding model based on Hawkes process that takes the full historical neighborhood formation process into account.

However, node embeddings obtained by previous work cannot be directly used to reconstruct (or recover) associations between nodes and attributes. They may need to further train another model to recover such associations, while in our model, with the learned embeddings of both nodes and attributes, we can directly recover the correlations between nodes and attributes by dot product of these two types of embeddings. In this article, we also tackle the problem of embedding dynamic attributed networks by dynamically modeling both nodes and attributes as Gaussian distributions, which consequently allows to measure the uncertainty of both embeddings by the variances of the distributions.

3 PRELIMINARIES

In this section, we introduce the notations used across the whole article, review the traditional variational auto-encoders and formally define the research problem.

3.1 Notations

In this article, scalars are denoted by normal letters (e.g., the total number of nodes in the network is denoted by a normal alphabet, N), while sets are denoted by calligraphy typeface letters (e.g., the set of nodes in the network: $\mathcal{V}$). Matrices are denoted by bold uppercase letters (e.g., an adjacency matrix $\mathbf{A}$ serving for the network). The ith row of a matrix, e.g., $\mathbf{A}$, is denoted with a subscript, e.g., $\mathbf{A}_i$. Vectors are represented by bold lowercase letters or uppercase letters with a subscript (e.g., $\mathbf{a}$ or $\mathbf{A}_i$). A vector with a subscript represents a scalar element of that vector (e.g., $\mathbf{a}_i$ or $\mathbf{A}_{ij}$). The transpose of a matrix, e.g., $\mathbf{A}$, is represented by $\mathbf{A}^{\mathrm{T}}$.

Definition 1 (Attributed Network (AN)). Let $\mathcal{V}$ and $\mathcal{A}$ be a set of nodes and attributes, respectively, in a graph/network. An attributed network is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathbf{A}, \mathbf{X})$ with $N = |\mathcal{V}|$ nodes and $F = |\mathcal{A}|$ attributes, where $\mathbf{A} \in \mathbb{R}^{N\times N}$ and $\mathbf{X} \in \mathbb{R}^{N\times F}$ are the adjacency matrix and the attribute matrix of nodes to collect all the edges (node-to-node connections) and attribute associations (node-to-attribute associations), with $\mathbf{A}_i$ and $\mathbf{X}_i$ being the ith row vectors of $\mathbf{A}$ and $\mathbf{X}$, respectively.

Let $\mathcal{G}^{(t)} = (\mathcal{V}, \mathcal{A}, \mathbf{A}^{(t)}, \mathbf{X}^{(t)})$, be an undirected attributed network at time t, then the *Dynamic Attributed Network* can be defined as follows.

Definition 2 (Dynamic Attributed Network (DAN)). A dynamic attributed network $\mathcal{G}^*$ is represented as a sequence of attributed networks, i.e., $\mathcal{G}^* = (\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \dots, \mathcal{G}^{(T)})$, recording the dynamic changes of edges and node attributes by each $\mathcal{G}^{(t)}$ with the time steps t evolving from 1 to T, where T is the number of time stamps.

We note that our dynamic attributed networks are conceptualized as discrete time series of networks on fixed sets of nodes $\mathcal{V}$ and attributes $\mathcal{A}$, which is the same as the previous work

Table 1. Main Notations Used in This Paper

Symbol	Description
$\mathcal{G}$	an attributed network
$\mathcal{G}^*$	an dynamic attributed network
$\mathcal{V}$	set of nodes
$\mathcal{A}$	set of attributes
$\mathcal{E}n$	set of edges
$\mathcal{E}a$	set of node-attribute associations
$\mathcal{E}n^o$	set of the observed edges
$\mathcal{E}a^o$	set of the observed node-attribute associations
$N = \lvert\mathcal{V}\rvert$	number of nodes
$F = \lvert\mathcal{A}\rvert$	number of attributes
D	dimension of latent variables
$\mathbf{A} \in \mathbb{R}^{N\times N}$	adjacency matrix of nodes
$\mathbf{X} \in \mathbb{R}^{N\times F}$	attribute matrix for nodes
$\mathbf{Z}n \in \mathbb{R}^{N\times D}$	latent representation matrix for all the *nodes*
$\mathbf{Z}a \in \mathbb{R}^{F\times D}$	latent representation matrix for all the *attributes*
$\mathbf{F}n \in \mathbb{R}^{N\times(N+F)}$	input features of all the *nodes*
$\mathbf{F}a \in \mathbb{R}^{F\times N}$	input features of all the *attributes*

[30, 31, 60]. For a node i, $\mathbf{A}_i^{(t)}$ and $\mathbf{X}_i^{(t)}$ can be regarded as its features; therefore, we use $\mathbf{F}n^{(t)} = \mathbf{con}(\mathbf{A}^{(t)}, \mathbf{X}^{(t)})$ to represent the input features of all nodes with $\mathbf{F}n_i^{(t)}$ being the features of node i, where $\mathbf{con}(\cdot)$ is the *concatenate* function. Similarly, we use $\mathbf{F}a^{(t)} = \mathbf{X}^{T(t)}$ to represent the input features of all the attributes with $\mathbf{F}a_a^{(t)}$ being the features of attribute a, where $\mathbf{X}^{T(t)}$ is the transpose of $\mathbf{X}^{(t)}$. Table 1 summarizes the main notations used all thorough this article.

3.2 Variational Auto-encoders

Variational Auto-Encoders (VAEs) are expressive latent variable models that can be used to learn complex probability representations from training data. A basic VAE model [25, 26, 45] comprises two computational neural networks: an **inference model** $f_\phi(\cdot)$ that acts as an encoder network mapping the given observations into latent variables, and a **generative model** $g_\theta(\cdot)$ that acts as a decoder network performing the reverse mapping from latent variables to each observed data point.

Specifically, the objective of VAE is to optimize a recognition model that allows to learn useful representations from an efficient approximate posterior inference of the latent variable given an observed data, while the expected reconstruction error can be reduced. The learning process is to minimize the following Evidence Lower BOund (ELBO) [1], $\mathcal{L}_{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X})$, on the marginal likelihood of the observed variables $\mathbf{X}$ [26]:

$$\begin{aligned}\log p(\mathbf{X}) &\geq \mathbb{E}_{q_\phi}[p_\theta(\mathbf{X} \mid \mathbf{Z})] - D_{KL}(q_\phi(\mathbf{Z} \mid \mathbf{X}) \parallel p_\theta(\mathbf{Z})) \\ &\triangleq \mathcal{L}_{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}),\end{aligned} \tag{1}$$

where $\mathbf{Z}$ denotes the latent representation vectors of the corresponding input entities $\mathbf{X}$, and $D_{KL}(\cdot \parallel \cdot)$ is the Kullback-Leibler divergence (KL-divergence). By using the so-called reparameterization trick [26], this model can be easily trained via a Stochastic Gradient Variational Bayes (SGVB) estimator [23]. The choice of encoder $f_\phi(\cdot)$ and decoder $g_\theta(\cdot)$ may vary across different tasks. There have been many proposed variations of VAEs [21, 25, 27, 40, 45], which have been extensively studied and applied in various tasks such as semi-supervised classification [25], clustering [21, 34], and image generation [7]. The GAE [27] model proposed by Kipf and Welling is an example of VAEs, which learns representations for attributed networks. However, it learns embeddings for nodes only but not for both nodes and attributes.

3.3 Problem Definition

With the terminologies described above, we formally define the problem of co-embedding an attributed network as follows:

Problem 1 (Static Attributed Network Co-embedding (SANE)). *Given an attributed network $\mathcal{G}$, the goal is to learn a mapping function Ξ in an unsupervised manner, which satisfies the following:*

$$\mathcal{G} \xrightarrow{\Xi} \mathrm{Z}n, \mathrm{Z}a, \tag{2}$$

such that both the structure and attribute information of the network can be preserved as much as possible by $\mathrm{Z}n \in \mathbb{R}^{N \times D}$ and $\mathrm{Z}a \in \mathbb{R}^{F \times D}$, respectively, with $\mathrm{Z}n_i \in \mathrm{Z}n$, $\mathrm{Z}a_j \in \mathrm{Z}a$, N, F, and D being the embeddings of the ith node and the jth attribute to be inferred, the number of nodes and attributes, and the size of the dimension of the embeddings, respectively.

Then, the problem of embedding a dynamic attributed network is defined as follows:

Problem 2 (Dynamic Attributed Network Embedding (DANE)). *Given a DAN, i.e., $\mathcal{G}^* = (\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(T)})$, the goal is to learn a mapping function Ξ to obtain low-dimensional representation vectors for both nodes and attributes:*

$$\mathcal{G}^* \xrightarrow{\Xi} \mathrm{Z}n^*, \mathrm{Z}a^*,$$

where $\mathrm{Z}n^* = (\mathrm{Z}n^{(1)}, \ldots, \mathrm{Z}n^{(T)})$ and $\mathrm{Z}a^* = (\mathrm{Z}a^{(1)}, \ldots, \mathrm{Z}a^{(T)})$, with $\mathrm{Z}n^{(t)} \in \mathbb{R}^{N \times D}$ and $\mathrm{Z}n^{(t)} \in \mathbb{R}^{F \times D}$ representing the latent representation matrices for all the nodes and attributes, respectively, at time step t, and D being the dimension of latent representations. By doing so, both the structure and attribute information of the network at each time step can be preserved as much as possible.

4 CO-EMBEDDING NODES AND ATTRIBUTES

To address the SANE problem, we propose **CSAN**, a novel model for **C**o-embedding **S**tatic **A**ttributed **N**etworks based on VAE [26], to obtain the Gaussian embeddings of both nodes and attributes. Figure 2 provides an overview of our CSAN model. In the following subsections, we describe this model in detail.

4.1 Evidence Lower Bound in CSAN

To obtain embeddings of both nodes and attributes (i.e., $\mathrm{Z}n$ and $\mathrm{Z}a$) for an attributed network $\mathcal{G}$ in an unsupervised manner, we first define an objective of maximizing the evidence lower bound on the observed adjacency matrix $\mathbf{A}$ and the attribute matrix $\mathbf{X}$ of $\mathcal{G}$. Using Jensen's inequality, the logarithmic marginal likelihood of $\mathbf{A}$ and $\mathbf{X}$ for a given attributed network $\mathcal{G}$ can be represented as:

$$\begin{aligned}\log p(\mathbf{A}, \mathbf{X}) &= \log \int_{\mathrm{Z}n} \int_{\mathrm{Z}a} p_\theta(\mathbf{A}, \mathbf{X}, \mathrm{Z}n, \mathrm{Z}a) \frac{q_\phi(\mathrm{Z}n, \mathrm{Z}a \mid \mathbf{A}, \mathbf{X})}{q_\phi(\mathrm{Z}n, \mathrm{Z}a \mid \mathbf{A}, \mathbf{X})} \mathrm{d}\mathrm{Z}n\, \mathrm{d}\mathrm{Z}a \\ &\geq \mathbb{E}_{q_\phi(\mathrm{Z}n, \mathrm{Z}a \mid \mathbf{A}, \mathbf{X})}\left[\log \frac{p_\theta(\mathbf{A}, \mathbf{X}, \mathrm{Z}n, \mathrm{Z}a)}{q_\phi(\mathrm{Z}n, \mathrm{Z}a \mid \mathbf{A}, \mathbf{X})}\right],\end{aligned} \tag{3}$$

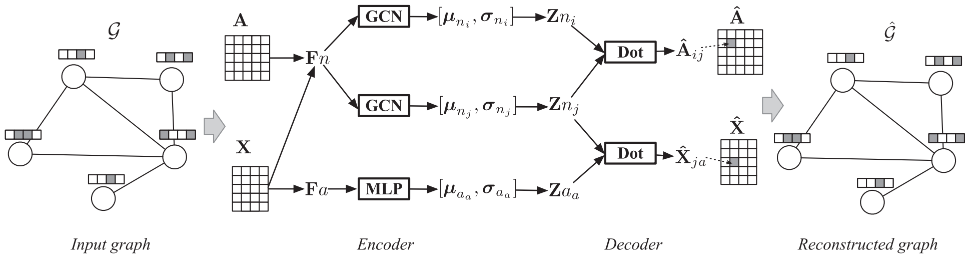


Fig. 2. The architecture of our proposed CSAN model. The model takes the adjacency matrix A and the attribute matrix X as input and outputs Gaussian distributions with means and variances as latent embeddings for all nodes and attributes of the network, respectively. The two neural network models, i.e., the inference model $f_{\boldsymbol{\phi}}$ and the generative model $g_{\boldsymbol{\theta}}$, act as the probabilistic encoder and the probabilistic decoder, respectively.

where $q_{\boldsymbol{\phi}}(\mathbf{Z}n, \mathbf{Z}a \mid \mathbf{A}, \mathbf{X})$, abbreviated as $q_{\boldsymbol{\phi}}$ for convenient discussion, is the *variational posterior* to approximate the true posterior $p(\mathbf{Z}n, \mathbf{Z}a \mid \mathbf{A}, \mathbf{X})$. Here, we assume that $q_{\boldsymbol{\phi}}$ is a mean-field distribution, which can be factorized as:

$$q_{\boldsymbol{\phi}}(\mathbf{Z}n, \mathbf{Z}a \mid \mathbf{A}, \mathbf{X}) = \prod_{i \in \mathcal{V}} q_{\boldsymbol{\phi}_1}(\mathbf{Z}n_i \mid \mathbf{F}n_i) \prod_{a \in \mathcal{A}} q_{\boldsymbol{\phi}_2}(\mathbf{Z}a_a \mid \mathbf{F}a_a), \tag{4}$$

where $\mathbf{F}n_i = \mathbf{con}(\mathbf{A}_i, \mathbf{X}_i)$ and $\mathbf{F}a_a = \mathbf{X}_i^{\mathrm{T}}$ are the input features of the nodes and attributes, respectively, and $\boldsymbol{\phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2]$ are trainable parameters shared across all the nodes and attributes, respectively. In Equation (4), $q_{\boldsymbol{\phi}_1}(\mathbf{Z}n_i \mid \mathbf{F}n_i)$ and $q_{\boldsymbol{\phi}_2}(\mathbf{Z}a_a \mid \mathbf{F}a_a)$ are referred to as the *inference model* (or the *encoder*), since given the observed data, i.e., A and X, they aim at producing variational posterior distributions over the possible values of the latent embeddings $\mathbf{Z}n$ and $\mathbf{Z}a$. Hence the encoders encode from the input features of nodes and attributes into embeddings of nodes and attributes, respectively.

The joint distribution $p_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{X}, \mathbf{Z}n, \mathbf{Z}a)$ can be represented as:

$$p_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{X}, \mathbf{Z}n, \mathbf{Z}a) = p(\mathbf{Z}n)p(\mathbf{Z}a) \prod_{(i,j) \in \mathcal{E}n^o} p_{\boldsymbol{\theta}_1}(\mathbf{A}_{ij} \mid \mathbf{Z}n_i, \mathbf{Z}n_j) \prod_{(i,a) \in \mathcal{E}a^o} p_{\boldsymbol{\theta}_2}(\mathbf{X}_{ia} \mid \mathbf{Z}n_i, \mathbf{Z}a_a), \tag{5}$$

where $\mathbf{A}_{ij} \in \mathbf{A}$, $\mathbf{X}_{ia} \in \mathbf{X}$ are weights for the observe edge $\mathcal{E}n^o(i,j)$ and attribute $\mathcal{E}a^o(i,a)$ in the training instances, respectively, and $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2]$ are trainable parameters that are shared across all the edges and attributes, respectively. Similarly, we refer to $p_{\boldsymbol{\theta}_1}(\mathbf{A}_{ij} \mid \mathbf{Z}n_i, \mathbf{Z}n_j)$ and $p_{\boldsymbol{\theta}_2}(\mathbf{X}_{ia} \mid \mathbf{Z}n_i, \mathbf{Z}a_a)$ as the *generative model* (or the *decoder*), since, given the latent embeddings $\mathbf{Z}n$ and $\mathbf{Z}a$, they produce a distribution over the possible corresponding values of observed edges and attributes. Hence the decoders decode from the result embeddings of both nodes and attributes into distributions (e.g., Bernoulli) of adjacency matrix and attribute matrix, respectively.

Substituting Equation (4) and Equation (5) into Equation (3), Equation (3) can be represented as:

$$\begin{aligned}\log p(\mathbf{A}, \mathbf{X}) \geq & \sum_{(i,j) \in \mathcal{E}n^o} \mathbb{E}_{q_{\boldsymbol{\phi}}} \left[\log p_{\boldsymbol{\theta}_1}(\mathbf{A}_{ij} \mid \mathbf{Z}n_i, \mathbf{Z}n_j)\right] + \sum_{(i,a) \in \mathcal{E}a^o} \mathbb{E}_{q_{\boldsymbol{\phi}}} \left[\log p_{\boldsymbol{\theta}_2}(\mathbf{X}_{ia} \mid \mathbf{Z}n_i, \mathbf{Z}a_a)\right] \\ & - D_{KL}(q_{\boldsymbol{\phi}_1}(\mathbf{Z}n \mid \mathbf{F}n) \parallel p(\mathbf{Z}n)) - D_{KL}(q_{\boldsymbol{\phi}_2}(\mathbf{Z}a \mid \mathbf{F}a) \parallel p(\mathbf{Z}a)) \\ & \triangleq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{A}, \mathbf{X}),\end{aligned} \tag{6}$$

where $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{A}, \mathbf{X})$ is the ELBO of the log-marginal likelihood of the observed variables A and X.

The KL-divergence terms in Equation (6) can be interpreted as the regularizer encouraging the approximate posteriors to be close to the priors $p(\mathbf{Z}n)$ and $p(\mathbf{Z}a)$, while the expectation terms are regarded as the expected negative reconstruction error loss. Often, analytical solutions of expectations w.r.t. the variational posterior are intractable in general cases, but we can reduce the problem of estimating the gradient w.r.t. the parameters of the posterior distribution to a simpler problem of estimating the gradient w.r.t. parameters of a deterministic function, which is called the *reparameterization* trick [26].

Similarly to the standard VAE [26], we assume all of the priors of latent variables and the variational posterior distributions to be Gaussian:

$$p(\mathrm{Z}n_i) = p(\mathrm{Z}a_a) = \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{7}$$

$$q_{\boldsymbol{\phi}_1}(\mathrm{Z}n_i \mid \mathrm{F}n_i) = \mathcal{N}\left(\boldsymbol{\mu}_{n_i}, \boldsymbol{\sigma}^2_{n_i}\mathbf{I}\right), \tag{8}$$

$$q_{\boldsymbol{\phi}_2}(\mathrm{Z}a_a \mid \mathrm{F}a_a) = \mathcal{N}\left(\boldsymbol{\mu}_{a_a}, \boldsymbol{\sigma}^2_{a_a}\mathbf{I}\right), \tag{9}$$

where $\boldsymbol{\mu}_{n_i}$, $\boldsymbol{\mu}_{a_a}$ and $\boldsymbol{\sigma}^2_{n_i}$, $\boldsymbol{\sigma}^2_{a_a}$ are means and variances of node embeddings and attributed embeddings to be learned, respectively.

Since we assume that the priors and the variational posteriors are Gaussian distributions, the KL-divergence terms in Equation (6) have analytical forms [26]. By using the SGVB estimator and the reparameterization trick [26], we can directly derive from the Monte Carlo estimates of these expectation terms by the following estimators of this model:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathrm{A}, \mathrm{X}) = &\sum_{(i,j)\in\mathcal{E}n^o} \mathbb{E}_{q_\phi}\left[\sum_{l=1}^{L} \log p_{\boldsymbol{\theta}_1}\left(\mathrm{A}_{ij} \,\middle|\, \mathrm{Z}n_i{}^{(l)}, \mathrm{Z}n_j{}^{(l)}\right)\right] \\ &+ \sum_{(i,a)\in\mathcal{E}a^o} \mathbb{E}_{q_\phi}\left[\sum_{l=1}^{L} \log p_{\boldsymbol{\theta}_2}\left(\mathrm{X}_{ia} \,\middle|\, \mathrm{Z}n_i{}^{(l)}, \mathrm{Z}a_a{}^{(l)}\right)\right] + \mathcal{L}_{KL},\end{aligned} \tag{10}$$

where $\mathrm{Z}n_i{}^{(l)}$, $\mathrm{Z}n_j{}^{(l)}$, and $\mathrm{Z}a_a{}^{(l)}$ are the deterministic embedding vectors given both means and variances of their distributions and the newly introduced auxiliary variable $\boldsymbol{\epsilon}^{(l)}$, which is distributed as $\mathcal{N}(0, \mathbf{I})$ (i.e., $\mathrm{Z}n_i^{(l)} = \boldsymbol{\mu}_{n_i} + \boldsymbol{\sigma}^2_{n_i} \odot \boldsymbol{\epsilon}^{(l)}$, $\mathrm{Z}a_a^{(l)} = \boldsymbol{\mu}_{a_a} + \boldsymbol{\sigma}^2_{a_a} \odot \boldsymbol{\epsilon}^{(l)}$); $\bullet\,|_d$ denotes the dth element of $\bullet$; D is the size of dimension of the latent vectors; L is the number of deterministic samples; $\odot$ indicates the element-wise product; $\mathcal{E}n^o$ and $\mathcal{E}a^o$ are the given observed edges and attribute associations; and $\mathcal{L}_{KL}$ is the analytical KL loss. As shown in Equation (10), by a differentiable transformation of an auxiliary "noise" variable $\boldsymbol{\epsilon}^{(l)}$ [26], we can reparameterize the latent Gaussian embeddings $q_{\boldsymbol{\phi}_1}(\mathrm{Z}n \mid \mathrm{F}n)$ and $q_{\boldsymbol{\phi}_2}(\mathrm{Z}a \mid \mathrm{F}a)$ into $\mathrm{Z}n^{(l)}$ and $\mathrm{Z}a^{(l)}$, which are deterministic and can be differentiated efficiently with respect to their means and variances using the backpropagation algorithm. Hence, the vanilla gradient-based optimization techniques can be used to train the model end to end.

4.2 Optimization

To optimize the objective in Equation (10), we apply two neural network models, i.e., the *inference model* with trainable parameters $\boldsymbol{\phi}$ and the *generative model* with trainable parameters $\boldsymbol{\theta}$, for the probabilistic encoder and probabilistic decoder, respectively, to perform gradient ascent for learning all the model parameters.

Inference model. To encode from observation variables to Gaussian embeddings, we apply a two-layer GCN [28] and a two-layer Multilayer Perceptron (MLP), to map the adjacency matrix $\mathbf{A}$ and attribute matrix $\mathbf{X}$ into their corresponding means and variations of Gaussian embeddings for nodes and attributes, respectively. We apply GCN and not other neural network models for the nodes' inference, since GCN is an efficient variant of Convolutional Neural Networks on graphs that applies graph-based convolution receiving features from the relevant neighbors around each node across the entire network. In particular, the two-layer GCN is defined as:

$$\begin{aligned} \mathbf{H}_n^{(1)} &= \mathrm{ReLU}\left(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_n^{(0)}\right), \\ \left[\boldsymbol{\mu}_n, \boldsymbol{\sigma}_n^2\right] &= \tilde{\mathbf{A}}\mathbf{H}_n^{(1)}\mathbf{W}_n^{(1)}, \end{aligned} \tag{11}$$

where $\boldsymbol{\mu}_n$ and $\boldsymbol{\sigma}_n^2$ are the means and variances of the learned Gaussian embeddings for **nodes** (see Equation (8)), $\mathrm{ReLU}(\cdot) = max(0, \cdot)$ is the Rectified Linear Unit activation function, $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ being $\mathcal{G}$'s degree matrix, and $\boldsymbol{\phi}_1 = [\mathbf{W}_n^{(0)}, \mathbf{W}_n^{(1)}]$ are trainable weights for the node inference layers, respectively.

The two-layer MLP for inferring Gaussian embeddings of attributes are defined as:

$$\begin{aligned} \mathbf{H}_a^{(1)} &= \tanh\left(\mathbf{X}^{\mathrm{T}}\mathbf{W}_n^{(0)} + \mathbf{b}^{(0)}\right), \\ \left[\boldsymbol{\mu}_a, \boldsymbol{\sigma}_a^2\right] &= \mathbf{H}_a^{(1)}\mathbf{W}_n^{(1)} + \mathbf{b}^{(1)}, \end{aligned} \tag{12}$$

where $\boldsymbol{\mu}_a$ and $\boldsymbol{\sigma}_a^2$ are the means and variances of the learned embeddings for **attributes** (see Equation (9)), $\mathbf{b}$ is the bias, $\tanh(\cdot)$ is the tangent activation function, and $\boldsymbol{\phi}_2 = [\mathbf{W}_n^{(0)}, \mathbf{W}_n^{(1)}, \mathbf{b}^{(0)}, \mathbf{b}^{(1)}]$ are trainable weights for the attribute inference layers. We let $\boldsymbol{\phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2]$ be the packed trainable parameters in the inference model.

After having obtained all the means and variances for all the Gaussian embeddings of nodes and attributes, the reparameterization trick [26] is applied to transform the latent Gaussian random variables into the deterministic variables $\mathbf{Z}n$ and $\mathbf{Z}a$, which are differentiable and capable to propagate the gradient between the inference model and the generative model.

Generative model. The generative model aims to decode the deterministic latent embeddings $\mathbf{Z}n$ and $\mathbf{Z}a$ into generative random variables, where the original input edges and attributes can be reconstructed. Specifically, given the embeddings of two nodes i and j, we first compute $\boldsymbol{\mu}_{\mathcal{E}_n(i,j)}$ and $\boldsymbol{\sigma}^2_{\mathcal{E}_n(i,j)}$ according to:

$$\left[\boldsymbol{\mu}_{\mathcal{E}_n(i,j)}, \boldsymbol{\sigma}^2_{\mathcal{E}_n(i,j)}\right] = g_{\theta_1}(\mathbf{Z}n_i, \mathbf{Z}n_j), \tag{13}$$

where g_{θ_1} is a neural network for reconstructing edges. Then, an observed edge can be generated by the following:

(1) For real-valued edges,

$$p_{\theta_1}(\mathbf{A}_{ij} \mid \mathbf{Z}n_i, \mathbf{Z}n_j) = \mathcal{N}\left(\boldsymbol{\mu}_{\mathcal{E}_n(i,j)}, \boldsymbol{\sigma}^2_{\mathcal{E}_n(i,j)}\mathbf{I}\right). \tag{14}$$

(2) For binary edges,

$$p_{\theta_1}(\mathbf{A}_{ij} \mid \mathbf{Z}n_i, \mathbf{Z}n_j) = \mathrm{Ber}(\boldsymbol{\mu}_{\mathcal{E}_n(i,j)}). \tag{15}$$

Here $\mathcal{N}(\boldsymbol{\mu}_{\mathcal{E}_n(i,j)}, \boldsymbol{\sigma}^2_{\mathcal{E}_n(i,j)}\mathbf{I})$ and $\mathrm{Ber}(\boldsymbol{\mu}_{\mathcal{E}_n(i,j)})$ are multivariate Gaussian distribution and Bernoulli distribution parametrized by $\boldsymbol{\mu}_{\mathcal{E}_n(i,j)}$, $\boldsymbol{\sigma}^2_{\mathcal{E}_n(i,j)}\mathbf{I}$ and $\boldsymbol{\mu}_{\mathcal{E}_n(i,j)}$, respectively.

Similarly, given the embeddings of node i and attribute a, we first compute $\boldsymbol{\mu}_{\mathcal{E}_a(i,a)}$ and $\boldsymbol{\sigma}^2_{\mathcal{E}_a(i,a)}$ by:

$$\left[\boldsymbol{\mu}_{\mathcal{E}_a(i,a)}, \boldsymbol{\sigma}^2_{\mathcal{E}_a(i,a)}\right] = g_{\theta_2}(\mathbf{Z}n_i, \mathbf{Z}a_a), \tag{16}$$

where g_{θ_2} is a neural network for reconstructing attributes. Then an observed attribute association $(i, a) \in \mathcal{E}_a$ can be generated by the following process:

(1) For real-valued attributes,

$$p_{\theta_2}(\mathbf{X}_{ia} \mid \mathbf{Z}n_i, \mathbf{Z}a_a) = \mathcal{N}\left(\boldsymbol{\mu}_{\mathcal{E}_a(i,a)}, \boldsymbol{\sigma}^2_{\mathcal{E}_a(i,a)}\mathbf{I}\right). \tag{17}$$

(2) For binary attributes,

$$p_{\theta_2}(\mathbf{X}_{ia} \mid \mathbf{Z}n_i, \mathbf{Z}a_a) = \mathrm{Ber}(\boldsymbol{\mu}_{\mathcal{E}_a(i,a)}). \tag{18}$$

We let $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2]$ be the packed trainable parameters in the generative model. Since all the edges and attributes in our experimental datasets are binary valued, we implement model g simply by inner product between latent variables, i.e.,

$$\begin{aligned} g_{\theta_1}(\mathbf{Z}n_i, \mathbf{Z}n_j) &= \mathrm{sig}\left(\mathbf{Z}n_i^{\mathrm{T}}\mathbf{Z}n_j\right), \\ g_{\theta_2}(\mathbf{Z}n_i, \mathbf{Z}a_a) &= \mathrm{sig}\left(\mathbf{Z}n_i^{\mathrm{T}}\mathbf{Z}a_a\right), \end{aligned} \tag{19}$$

where sig$(\cdot)$ is the sigmoid function. Our experimental results will later show that such a simple implementation produces better results than other common networks, such as the MLP, and outperforms state-of-the-art baselines in many graph mining tasks such as link prediction, node classification and attribute inference. The training procedure for our CSAN model is summarized in Algorithm 1.

ALGORITHM 1: CSAN training algorithm.

```
1: repeat
2:     Learn $\boldsymbol{\mu}_n, \boldsymbol{\sigma}_n, \boldsymbol{\mu}_a, \boldsymbol{\sigma}_a$ according to Equation (11) and Equation (12)
3:     Sample $\mathbf{Z}n$, $\mathbf{Z}a$ according to Equation (8) and Equation (9) with noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4:     Compute ELBO according to Equation (10)
5:     Update $\boldsymbol{\theta}$, $\boldsymbol{\phi}$ by gradient back-propagation
6: until Convergence of the parameters
```

Time complexity analysis. As we can see in Algorithm 1, the layerwise propagation rule for both the encoder and decoder networks costs the most to the time complexity of our algorithm, while the two-layer GCN network [28] has the highest computational complexity in the network propagation flow. We use early stopping strategy to reduce the update epochs, i.e., we premature stopping if the evaluation performance of different task on the validation data does not increase for 10 successive epochs. The time complexity of the two-layer GCN network for one epoch boils down to 2 sparse-dense-matrix multiplications for a cost of $O(|\tilde{\mathbf{A}}^{+}|(H + D + F))$, where $|\tilde{\mathbf{A}}^{+}|$ denotes the number of nonzero entires in the Laplacian matrix and H is the dimension of the first hidden layer. Empirically, our CSAN costs around 10 s and 290 s per 10 epochs on the Cora and Pubmed [44] datasets, respectively, for training with Inter i7 3.60-GHz CPU machines.

5 DYNAMIC CO-EMBEDDING MODEL

To address our DANE problem, we extend our CSAN model to propose **CDAN**. CDAN also makes use of latent Gaussian random variables to represent both nodes and attributes, and it is further extended to make allow the tracking of the dynamic changes of networks. Figure 3 provides an overview of this model. In the following, we describe our CDAN model in detail.

5.1 Evidence Lower Bound in CDAN

Our CDAN model is, again, an unsupervised generative model that learns latent variables by maximizing a bound of average marginal log-likelihood of observations. We herein first derive our objective function from the logarithmic marginal likelihood according to the basic framework of

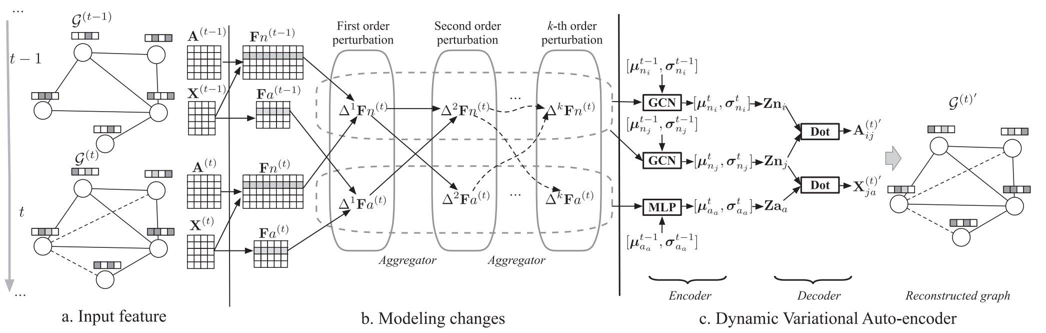


Fig. 3. Overview of our CDAN model. Given the adjacency matrices and the attribute matrices between two consecutive time steps $(t-1)$ and t, our CDAN model formulates the first-order perturbation and other higher-order perturbations according to their node features $\mathbf{F}n^{(t-1)}$ and $\mathbf{F}n^{(t)}$ and attribute features $\mathbf{F}a^{(t-1)}$ and $\mathbf{F}a^{(t)}$ and then takes them as input into a dynamic variational auto-encoder and outputs Gaussian embeddings for all nodes and attributes of the network, respectively.

VAE [26]. Given the adjacency matrix $\mathbf{A}^{(t)}$ and the attribute matrix $\mathbf{X}^{(t)}$ of a DAN at time step t, the marginal likelihood, by using Jensen's inequality, can be written as:

$$\log p(\mathbf{A}^{(t)}, \mathbf{X}^{(t)}) \geq \mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{Z}n^{(t)}, \mathbf{Z}a^{(t)}, \mathbf{A}^{(t)}, \mathbf{X}^{(t)})] - \mathbb{E}_{q_\phi}[\log q_\phi(\mathbf{Z}n^{(t)}, \mathbf{Z}a^{(t)} \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)})], \tag{20}$$

where $q_\phi(\mathbf{Z}n^{(t)}, \mathbf{Z}a^{(t)} \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)})$, abbreviated as q_ϕ, is the *variational posterior* to approximate the true posterior $p(\mathbf{Z}n^{(t)}, \mathbf{Z}a^{(t)} \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)})$ with ϕ being the parameters needed to be trained. Here we assume q_ϕ to be a mean-field distribution and can be factorized as:

$$q_\phi(\mathbf{Z}n^{(t)}, \mathbf{Z}a^{(t)} \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)}) = \prod_{i \in \mathcal{V}} q_{\phi_1}\left(\mathbf{Z}n_i^{(t)} \,\middle|\, \mathbf{F}n_i^{(t)}\right) \prod_{a \in \mathcal{A}} q_{\phi_2}\left(\mathbf{Z}a_a^{(t)} \,\middle|\, \mathbf{F}a_a^{(t)}\right), \tag{21}$$

where $q_{\phi_1}(\mathbf{Z}n_i^{(t)} \mid \mathbf{F}n_i^{(t)})$ and $q_{\phi_2}(\mathbf{Z}a_a^{(t)} \mid \mathbf{F}a_a^{(t)})$, referred to as the *encoders* with $[\phi_1, \phi_2] = \phi$ being the parameters, are stochastic embeddings of nodes and attributes. Here we assume all of the variational posteriors to be Gaussian:

$$q_{\phi_1}\left(\mathbf{Z}n_i^{(t)} \,\middle|\, \mathbf{F}n^{(t)}\right) = \mathcal{N}\left(\mu_{n_i}, \sigma_{n_i}^2 \mathbf{I}\right), \tag{22}$$

$$q_{\phi_2}\left(\mathbf{Z}a_a^{(t)} \,\middle|\, \mathbf{F}a^{(t)}\right) = \mathcal{N}\left(\mu_{a_a}, \sigma_{a_a}^2 \mathbf{I}\right). \tag{23}$$

The joint distribution $p_\theta(\mathbf{Z}n^{(t)}, \mathbf{Z}a^{(t)}, \mathbf{A}^{(t)}, \mathbf{X}^{(t)})$ of Equation (20) can be represented as:

$$p_\theta(\mathbf{Z}n^{(t)}, \mathbf{Z}a^{(t)}, \mathbf{A}^{(t)}, \mathbf{X}^{(t)}) = p(\mathbf{Z}n^{(t)})\, p(\mathbf{Z}a^{(t)}) \prod_{(i,j) \in \mathcal{E}n^o} p_{\theta_1}\left(\mathbf{A}_{ij}^{(t)} \,\middle|\, \mathbf{Z}n_i^{(t)}, \mathbf{Z}n_j^{(t)}\right)$$
$$\prod_{(i,a) \in \mathcal{E}a^o} p_{\theta_2}\left(\mathbf{X}_{ia}^{(t)} \,\middle|\, \mathbf{Z}n_i^{(t)}, \mathbf{Z}a_a^{(t)}\right), \tag{24}$$

where $p_{\theta_1}(\mathbf{A}_{ij}^{(t)} \mid \mathbf{Z}n_i^{(t)}, \mathbf{Z}n_j^{(t)})$ and $p_{\theta_2}(\mathbf{X}_{ia}^{(t)} \mid \mathbf{Z}n_i^{(t)}, \mathbf{Z}a_a^{(t)})$, referred to as the *decoders* with $\theta = [\theta_1, \theta_2]$ being the parameters to be learned, are the likelihood of the observations $\mathbf{A}_{ij}^{(t)}$ and $\mathbf{X}_{ia}^{(t)}$ in the network, respectively. We assume both the prior of latent variables $p(\mathbf{Z}n^{(t)})$ and $p(\mathbf{Z}a^{(t)})$ to be Gaussian, such that we have:

$$p(\mathbf{Z}a^{(t)}) = p(\mathbf{Z}n^{(t)}) = \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{25}$$

Substituting Equation (21) and Equation (24) into Equation (20), we have:

$$\begin{aligned}\log p(\mathbf{A}^{(t)}, \mathbf{X}^{(t)}) \geq & \sum_{(i,j)\in\mathcal{E}n^o} \mathbb{E}_{q_\phi}\left[\log p_{\theta_1}\left(\mathbf{A}_{ij}^{(t)} \,\middle|\, \mathbf{Z}n_i^{(t)}, \mathbf{Z}n_j^{(t)}\right)\right] \\ & + \sum_{(i,a)\in\mathcal{E}a^o} \mathbb{E}_{q_\phi}\left[\log p_{\theta_2}\left(\mathbf{X}_{ia}^{(t)} \,\middle|\, \mathbf{Z}n_i^{(t)}, \mathbf{Z}a_a^{(t)}\right)\right] \\ & - D_{KL}\left(q_{\phi_1}\left(\mathbf{Z}n^{(t)} \,\middle|\, \mathbf{F}_n^{(t)}\right) \parallel p(\mathbf{Z}n^{(t)})\right) - D_{KL}\left(q_{\phi_2}\left(\mathbf{Z}a^{(t)} \,\middle|\, \mathbf{F}_a^{(t)}\right) \parallel p(\mathbf{Z}a^{(t)})\right) \\ \triangleq & \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{A}^{(t)}, \mathbf{X}^{(t)}), \end{aligned} \tag{26}$$

where $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{A}^{(t)}, \mathbf{X}^{(t)})$ is the ELBO of the log-marginal probability of the observations $\mathbf{A}^{(t)}$ and $\mathbf{X}^{(t)}$.

5.2 Modeling Features over Time

The objective of Equation (26) still serves as a static learning model, since the embeddings are inferred by the static features of observations at time t. To capture the evolving patterns of a dynamic attributed network, we introduce $\Delta\mathbf{F}n^{(t)}$ and $\Delta\mathbf{F}a^{(t)}$, i.e., the feature perturbation of nodes and the feature perturbation of attributes, to represent the feature changes of nodes and attributes by the perturbation of the observed adjacency matrix and the attribute matrix between time steps $(t-1)$ and t.

First-order perturbation. The intuition to model features over time is that the direct changes of the adjacency matrix and attribute matrix of nodes must be preserved. To achieve this, we introduce the **first-order perturbation** to represent the direct changes of features for both nodes and attributes, respectively, between two consecutive time steps. Formally, the first-order perturbation for node i and attribute a between time t and $(t-1)$, i.e., $\Delta^1\mathbf{F}n_i^{(t)}$ and $\Delta^1\mathbf{F}a_a^{(t)}$, is defined as:

$$\begin{aligned}\Delta^1\mathbf{F}n_i^{(t)} &= \tanh\left(\mathbf{W}n^1 \cdot \left(\mathbf{F}n_i^{(t)} - \mathbf{F}n_i^{(t-1)}\right)\right), \\ \Delta^1\mathbf{F}a_a^{(t)} &= \tanh\left(\mathbf{W}a^1 \cdot \left(\mathbf{F}a_a^{(t)} - \mathbf{F}a_a^{(t-1)}\right)\right),\end{aligned} \tag{27}$$

where $\mathbf{W}n^1 \in \mathbb{R}^{(N+F)\times H_p}$ and $\mathbf{W}a^1 \in \mathbb{R}^{N\times H_p}$ are weight matrices with $H_p < N, H_p < F$ being the dimension of perturbation feature, respectively. If no changes of the network are observed between $t-1$ and t, then $\Delta^1\mathbf{F}n_i^{(t)} = \mathbf{0}$ and $\Delta^1\mathbf{F}a_a^{(t)} = \mathbf{0}$.

However, in practice, the feature perturbation of a node (or an attribute) not only directly affects the representation of the node (or attribute) itself but also indirectly affects the embeddings of other nodes and attributes by the propagation of perturbation. For instance, a change occurring on node u's feature will result in a change on u's embedding vector, which also indirectly affects the embedding on all of u's neighbors and attributes. These changes will continue to propagate to the embeddings of other associated nodes and attributes and result in a corresponding adjustment to maintain the overall equilibrium of the proximities among the embeddings of all the nodes and attributes. Therefore, we further introduce the ***k*th-order perturbation** that captures the indirect changes, which are propagated from the lower-order perturbation.

The *k*th-order perturbation. To formulate higher-order perturbation, the key issue is how to aggregate a node's (or an attribute's) feature changes from its local neighborhoods (i.e., its associated nodes and attributes). For this purpose, we apply the *mean aggregation function* to learn the aggregation of the lower-order perturbations from a node's (or an attribute's) local neighborhoods, so that any direct feature changes of a node (or an attribute) can propagate along the edge connections and attribute associations to the higher-order perturbations of its corresponding neighboring nodes and attributes.

The *aggregation architectures* have been used in the literature [15] to aggregate neighboring feature and output a new feature vector for each node. There are many choices for the aggregation functions, such as mean, max pooling, and LSTM aggregation [15], but in this article, we only consider the *mean* aggregation as the mean-based aggregation is a rough, linear approximation of a localized spectral convolution [12, 15]. In particular, the *mean aggregation* $\mathbf{agg}^k(\cdot)$ for the kth-order perturbation in our article is defined as:

$$\mathbf{agg}^k(\cdot) = \mathrm{ReLU}(\mathbf{W}^k \cdot \mathbf{mean}(\cdot)), \tag{28}$$

where $\mathbf{W}^k \in \mathbb{R}^{H_p \times H_m}$ is the weight matrix with H_m being the dimension of aggregation output and $\mathbf{mean}(\cdot)$ is an operation that computes the mean of each dimension. For a node, the higher-order perturbation should consider the lower-order perturbation of both its neighbors and attributes, as the feature of each node is associated with neighbors and attributes. Hence, the kth-order perturbation vector for node i, i.e., $\Delta^k \mathbf{F}n_i^{(t)}$, is defined as:

$$\begin{aligned}
\Delta^k \mathbf{F}n_i^{(t)} &= \tanh\left(\mathbf{W}n^k \cdot \mathbf{con}\left(\mathbf{h}_1^{k(t)}, \mathbf{h}_2^{k(t)}\right)\right),\\
\mathbf{h}_1^{k(t)} &= \mathbf{agg}\left(\left\{\Delta^{(k-1)}\mathbf{F}n_j^{(t)} \,\middle|\, \forall (i,j) \in \mathcal{E}_n^{(t)} \cup \mathcal{E}_n^{(t-1)}\right\}\right),\\
\mathbf{h}_2^{k(t)} &= \mathbf{agg}\left(\left\{\Delta^{(k-1)}\mathbf{F}a_a^{(t)} \,\middle|\, \forall (i,a) \in \mathcal{E}_a^{(t)} \cup \mathcal{E}_a^{(t-1)}\right\}\right),
\end{aligned} \tag{29}$$

where $\mathbf{W}n^k \in \mathbb{R}^{(2\times H_m)\times H_p}$ is the weight matrix. Similarly, for an attribute, the higher-order perturbation is aggregated by the lower-order perturbation of its associated nodes, thus the kth-order perturbation for an attribute a, i.e., $\Delta^k \mathbf{F}a_a^{(t)}$, is defined as:

$$\begin{aligned}
\Delta^k \mathbf{F}a_a^{(t)} &= \tanh\left(\mathbf{W}a^k \cdot \mathbf{h}_3^{k(t)}\right),\\
\mathbf{h}_3^{k(t)} &= \mathbf{agg}\left(\left\{\Delta^{(k-1)}\mathbf{F}n_j^{(t)} \,\middle|\, \forall (j,a) \in \mathcal{E}_a^{(t)} \cup \mathcal{E}_a^{(t-1)}\right\}\right),
\end{aligned} \tag{30}$$

where $\mathbf{W}a^k \in \mathbb{R}^{H_m \times H_p}$ is the weight matrix. Practically speaking we found that our approach could achieve high performance with $k = 2$. Hence, we let

$$\Delta \mathbf{F}n_i^{(t)} = \mathbf{con}\left(\Delta^1 \mathbf{F}n_i^{(t)}, \Delta^2 \mathbf{F}n_i^{(t)}\right), \tag{31}$$

$$\Delta \mathbf{F}a_a^{(t)} = \mathbf{con}\left(\Delta^1 \mathbf{F}a_a^{(t)}, \Delta^2 \mathbf{F}a_a^{(t)}\right), \tag{32}$$

with $\mathbf{W} = [\mathbf{W}n^1, \mathbf{W}a^1, \mathbf{W}^1, \mathbf{W}n^2, \mathbf{W}a^2, \mathbf{W}^2]$ being all the weight matrices.

Let us now focus on ELBO in Equation (26). The KL-divergence terms of Equation (26) can be interpreted as a regularizer encouraging the approximate posterior to be close to the prior, while the expectation terms are regarded as an expected negative reconstruction error loss. Since we assume the priors and the variational posteriors to be Gaussian distributions, the two KL-divergence terms of Equation (26) have analytical forms. However, for the expectation terms, $\mathbf{Z}n^{(t)}$ and $\mathbf{Z}a^{(t)}$ are random variables and directly sampling for their deterministic variables is a non-continuous operation and has no gradient. Similarly to our CSAN, we can also use the reparameterization trick [26] by introducing an auxiliary noise variable $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ such that $\mathbf{Z}n^{(t)}$ and $\mathbf{Z}a^{(t)}$ becomes deterministic: $\mathbf{Z}n^{(t)} = \boldsymbol{\mu}_n^{(t)} + \boldsymbol{\sigma}_n^{2(t)} \odot \boldsymbol{\epsilon}$ and $\mathbf{Z}a^{(t)} = \boldsymbol{\mu}_a^{(t)} + \boldsymbol{\sigma}_a^{2(t)} \odot \boldsymbol{\epsilon}$. This deterministic variable can be differentiated efficiently by using the backpropagation algorithm, and thus we can train the model by gradient-based optimization techniques. By applying the SGVB estimator and the reparameterization trick [26], the ELBO in Equation (26) can be written as:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathrm{A}^{(t)}, \mathrm{X}^{(t)}) = \sum_{(i,j)\in\mathcal{E}n^o} \mathbb{E}_{q_\phi}\left[\sum_{l=1}^{L} \log p_{\boldsymbol{\theta}_1}\left(\mathrm{A}_{ij}^{(t)} \,\middle|\, \mathrm{Z}n_i^{(t,l)}, \mathrm{Z}n_j^{(t,l)}\right)\right] + \sum_{(i,a)\in\mathcal{E}a^o} \mathbb{E}_{q_\phi}\left[\sum_{l=1}^{L} \log p_{\boldsymbol{\theta}_2}\left(\mathrm{X}_{ia}^{(t)} \,\middle|\, \mathrm{Z}n_i^{(t,l)}, \mathrm{Z}a_a^{(t,l)}\right)\right] + \mathcal{L}_{KL}. \tag{33}$$

5.3 Optimization

In this subsection, we describe the specific neural networks for the encoders and decoders in Equation (33).

Inference Model. It is natural to assume that the embeddings evolve smoothly over two consecutive time steps, instead of being totally reconstructed at each time step. Therefore, the encoder neural network model (Equation (22) and Equation (23)) requires modification from the standard VAE's to encoder the changes of observation variables over time. Given the Gaussian embeddings of node i at time step $(t-1)$, our inference algorithm of encoder seeks to iteratively update the parameters $\boldsymbol{\mu}_{n_i}^{(t)}$ and $\boldsymbol{\sigma}_{n_i}^{2(t)}$ of the Gaussian embedding at time step t by using a neural network g parameterized by $\boldsymbol{\mu}_{n_i}^{(t)}$, $\boldsymbol{\sigma}_{n_i}^{2(t-1)}$, and $\Delta\mathrm{F}n_i^{(t-1)}$. More specifically, to obtain the Gaussian embeddings for a node i, we have:

$$\begin{aligned}\boldsymbol{\mu}_{n_i}^{(t)} &= g_{\boldsymbol{\phi}_1}\left(\boldsymbol{\mu}_{n_i}^{(t-1)}, \Delta\mathrm{F}n_i^{(t)}\right),\\ \boldsymbol{\sigma}_{n_i}^{2(t)} &= g_{\boldsymbol{\phi}_1}\left(\boldsymbol{\sigma}_{n_i}^{2(t-1)}, \Delta\mathrm{F}n_i^{(t)}\right).\end{aligned} \tag{34}$$

Similarly, given attribute a's embedding at time $(t-1)$, its Gaussian embedding at time t is

$$\begin{aligned}\boldsymbol{\mu}_{a_a}^{(t)} &= g_{\boldsymbol{\phi}_2}\left(\boldsymbol{\mu}_{a_a}^{(t-1)}, \Delta\mathrm{F}a_a^{(t)}\right),\\ \boldsymbol{\sigma}_{a_a}^{2(t)} &= g_{\boldsymbol{\phi}_2}\left(\boldsymbol{\sigma}_{a_a}^{2(t-1)}, \Delta\mathrm{F}a_a^{(t)}\right).\end{aligned} \tag{35}$$

Here $g_{\boldsymbol{\phi}_1}$ and $g_{\boldsymbol{\phi}_2}$ are all implemented by full connected networks.

Generative Model. The decoder model (Equation (24)) aims to decode the deterministic latent embeddings $\mathbf{F}n^{(t)}$ and $\mathbf{F}a^{(t)}$ into generative random variables of the original input edges and attributes. Since the adjacency matrices and attribute matrices of all experimental datasets are binary weighted, our encoder neural network is simply given by the inner product between latent variables. Specifically, given the embeddings of nodes, an observed edge $(i,j) \in \mathcal{E}_{\mathcal{V}}$ is generated by the following:

$$p_{\theta_1}\left(\mathrm{A}_{ij}^{(t)} \,\middle|\, \mathrm{Z}n_i^{(t)}, \mathrm{Z}n_j^{(t)}\right) = \mathbf{Ber}\left(\mathrm{sig}\left(\mathrm{Z}n_i^{(t)T}\mathrm{Z}n_j^{(t)}\right)\right), \tag{36}$$

where $\mathbf{Ber}(\cdot)$ is the Bernoulli distribution. Similarly, given embeddings of node i and attribute a, an observed attribute of node i, i.e., $(i,a) \in \mathcal{E}_a$, is generated by the following process:

$$p_{\theta_2}\left(\mathrm{X}_{ia}^{(t)} \,\middle|\, \mathrm{Z}n_i^{(t)}, \mathrm{Z}a_a^{(t)}\right) = \mathbf{Ber}\left(\mathrm{sig}\left(\mathrm{Z}n_i^{(t)T}\mathrm{Z}a_a^{(t)}\right)\right). \tag{37}$$

We note that our model can be easily extended to real-value weighted datasets by replacing the Bernoulli distribution with Gaussian distribution.

Based on the encoder and decoder neural networks, the gradient back-propagation based techniques then can be applied to optimize Equation (33) in an end-to-end manner. The training procedure for our CDAN model is summarized in Algorithm 2.

Time Complexity Analysis. As we discussed before, the mean aggregation function is a rough, linear approximation of GCN [12, 15]. But unlike GraphSAGE that only samples a small amount of

ALGORITHM 2: CDAN training algorithm.

```
1: for $t = 1, 2, \ldots, T$ do
2:     repeat
3:         Learn $\boldsymbol{\mu}_n^{(t)}, \boldsymbol{\sigma}_n^{2(t)}, \boldsymbol{\mu}_a^{(t)}, \boldsymbol{\sigma}_a^{2(t)}$ according to Equation (34) and Equation (35)
4:         Sample $\mathbf{Z}n^{(t)}$, $\mathbf{Z}a^{(t)}$ according to Equation (22) and Equation (23) with noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:         Compute ELBO according to Equation (33)
6:         Update $\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{W}$ by gradient back-propagation
7:     until Convergence of parameters
8: end for
```

neighbours, our mean aggregation function of CDAN aggregates all the neighbouring nodes and attributes, because we found in our experiments that neighborhood sample size of the best performance is almost same as the average number of neighborhoods. Therefore, the time complexity bottleneck of Algorithm 2 boils down to the mean aggregation function, which has a time complexity of $O(|\tilde{\mathbf{A}}^+|(H_m + H_p + D + F))$. Note that H_m, H_p, and D are the dimensions of the latent layers and embeddings, which are far less than the number of nodes. The total time complexity per epoch of the training procedures is $O(T|\tilde{\mathbf{A}}^+|(H_m + H_p + D + F))$, where T is the total time steps.

6 EXPERIMENTAL SETUP

In this section, we describe our experimental setup—the research questions (Section 6.1), the experimental datasets (Section 6.2), the baselines and the experimental settings (Section 6.3).

6.1 Research Questions

The research questions guiding the remainder of this article are as follows.

Research questions for evaluating the performance of CSAN:

(RQ1) Can the learned embeddings by our CSAN model show better performance than the state-of-the-art network embedding models in tasks of link prediction and node classification? (Section 7.1)

(RQ2) Can our CSAN perform better than other models in the tasks of attribute inference and user profiling, where capturing the affinities between nodes and attributes is crucial? (Section 7.2)

(RQ3) Can our CSAN model capture the affinities between nodes and attributes, and make the embedding results explainable? (Section 7.3)

Research questions for evaluating the performance of CDAN:

(RQ4) Can our CDAN model learn more effective representations for the dynamic attributed networks than the state-of-the-art non-dynamic and dynamic embedding models? (Section 7.4)

(RQ5) Can our CDAN model capture the dynamic affinities between nodes and attributes? (Section 7.5)

(RQ6) Can the learned embeddings of our CDAN model effectively capture the uncertainties for dynamic networks? (Section 7.6)

6.2 Datasets

We conduct experiments on the eight datasets, seven of which are the static attributed networks and one is the dynamic attributed network, with the statistics information provided in Table 2:

Table 2. Statistics of the Eight Datasets, Where the Density of Each Dataset Is Calculated by the Number of Edges Divided by the Total Potential Number of Edges between Any Two Nodes in the Network

Datasets	#Nodes	#Edges	#Attributes	#Labels	#Density	#Snapshot
Cora	2,708	5,429	1,433	7	0.15%	-
Citeseer	3,312	4,660	3,703	6	0.08%	-
Pubmed	19,717	88,651	500	3	0.05%	-
BlogCatalog	5,196	171,743	8,189	6	1.27%	10
Flickr	7,575	239,738	12,047	9	0.84%	10
Facebook	4,039	88,234	1,406	-	1.08%	-
DBLP-con	12,213	131,713	172	-	0.18%	-
DBLP-key	24,610	87,838	6,034	-	0.03%	14

- **Cora, Citeseer, Pubmed** [44]: The Cora, Citeseer, and Pubmed datasets are citation networks where nodes are publications and edges are citation links. Attributes of each node are bag-of-words representations of the corresponding publications.
- **BlogCatalog** [48]: This is a network of social relationships of bloggers from the BlogCatalog website, where nodes' attributes are constructed by keywords, which are generated by users as a short description of their blogs. The labels represent the topic categories provided by the authors.
- **Flickr** [19]: It is a social network where nodes represent users and edges correspond to friendships among users. The labels represent the interest groups of the users.
- **Facebook** [29]: This network is built from the profile and relation data of 10 users in Facebook by SNAP.[1] The attributes are constructed by their profiles.
- **DBLP-con**: This dataset is crawled from the DBLP public bibliography data.[2] We build a coauthor network extracted from publications in the top 172 computer science conferences (Tiers A and B from China Computer Federation[3]). We treat each author as a node, each collaboration between two authors in a publication as an edge. The 172 conferences are viewed as the attributes of each node.
- **DBLP-key**: This is a dynamic attributed network that is also the DBLP coauthor network extracted from the DBLP public bibliography data, but unlike the DBLP-con dataset, the attributes in this network are the keyterms[4] extracted from the article titles, and the time steps are the publication years.

In addition, we also introduce two synthetic dynamic networks from the static attributed networks, i.e., the **Flickr** and **BlogCatalog** networks, for evaluating our CDAN model, similarly to Reference [31]. Specifically, we generate a sequence of network snapshots $(\mathcal{G}_1, \ldots, \mathcal{G}_T)$ from the original static networks, where $\mathcal{G}_t$ is a sub-network sampling from the original network with a fixed size of edges and attribute associations removal from $\mathcal{G}_{t+1}$.

6.3 Baselines and Settings

We compare our two proposed models with three categories of methods: static network embedding methods, dynamic network embedding methods, and attribute inference methods.

[1]Available from: http://snap.stanford.edu/data/.
[2]Available from: http://dblp.uni-trier.de/xml/.
[3]Available from: http://www.ccf.org.cn/.
[4]Stopwords were removed and stemming preprocessing was applied.

The five static network embedding baseline methods are as follows:

- **DeepWalk (DW)** [42]: This is a classical embedding method for static plain networks that uses random walks as input to the Skip-gram model to learn node representations.
- **GAE** [27]: This model embeds an attributed network by using the variational auto-encoder, but it optimizes ELBO without considering the reconstruction error of attributes, and it learns embeddings for nodes only rather than for both nodes and attributes.
- **GraphSAGE** [15]: This is an attributed network embedding model that learns node representations by sampling and aggregating features from the nodes' local neighborhoods. In this article, we use its convolutional aggregation variant as our baseline.
- **AANE** [18]: It is an attributed network embedding model that learns node representations based on the decomposition of attribute affinities and the embedding difference between the connected nodes.
- **ANRL** [57]: It is a network embedding model that uses the neighbor enhancement auto-encoder and the attribute-aware skip-gram model to capture the node attributes and structural information of networks. We adopt one of its variants that uses the Weighted Average Neighbor function to construct its target neighbors, abbreviated as ANRL-WAN.

The three dynamic network embedding baseline methods are as follows:

- **DTriad** [58]: An embedding method for dynamic plain networks that applies the triad closure process to model the formation and evolution of dynamic networks.
- **DANE** [31]: This is an embedding method for dynamic attributed networks that uses an off-line Laplacian Eigenmaps-based model and the matrix perturbation theory to learn and update node embeddings.
- **CDAN1**: This is a variation of our proposed CDAN that only considers the first-order perturbation.

The three attribute inference baseline methods are:

- **SAN** [13]: This is a joint link prediction and attribute inference algorithm based on link features computed by Adamic-Adar and Low-Rank Approximation.
- **EdgeExp** [4]: This is an attribute inference algorithm that leverages a softmax function to solve both user attributes and relationship types.
- **BLA** [56]: This is a probabilistic model that iteratively learns user links and attributes, and leverages the data redundancy on each side as well as the mutual reinforcement between the two.

We implemented all the baselines using the codes released by the corresponding authors. The parameters of the baseline methods are tuned to be optimal. For example, in DeepWalk, we set the window size as 10, the walk length as 80 and the walks per node as 10. For GraphSAGE, we use the rectified linear units as the non-linearity and set the depth as 2 with neighborhood sample size being 10. For ANRL, the number of layers and dimensions are set as the same values as that in Reference [57].

We train our two models with 200 iterations by Adam [24] optimizer with the learning rate being 0.01. For our inference neural network, we use a 64-dimensional hidden layer and 32-dimensional latent variables in all experiments. For fair comparisons, the dimension of embedding for all methods is fixed to be 32. For our CDAN model, the dimension of perturbation feature H_p and the aggregation function output H_m are set to be 64 and 32, respectively. The dimension of embedding for CDAN and other embedding baselines are set to be 32 unless specifically stated. For g_{ϕ_1} and g_{ϕ_2} in the encoder of CDAN, we used a two-layer MLP networks composed of the tanh non-linear

Table 3. Link Prediction Performances

Method	Cora		Citeseer		Facebook		Pubmed		Flickr		BlogCatalog	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
AANE	.767	.720	.785	.765	.842	.834	.783	.754	.709	.697	.711	.714
GaphSAGE	.795	.763	.802	.791	.854	.846	.840	.829	.732	.728	.723	.702
ANRL-WAN	.832	.843	.867	.848	.935	.912	.918	.897	.724	.763	.762	.758
GAE	.914	.926	.908	.920	.980	.979	.944	.947	.828	.827	.821	.821
CSAN	**.985**†	**.984**†	**.950**†	**.958**†	**.988**†	**.986**†	**.980**†	**.977**†	**.914**†	**.922**†	**.837**†	**.837**†
Imporvement	7.77%	6.26%	4.63%	4.13%	0.82%	0.72%	3.81%	3.17%	10.39%	11.49%	1.95%	1.95%

The best and the second best performing runs per metric per dataset are denoted in bold and underlined, respectively. For each dataset, the last raw shows the improvements of CSAN over the best baseline algorithms, and the significant improvements over the best baseline algorithm are marked by † (paired t-test, $p < .05$).

mapping functions with a 64-dimensional hidden layer. In our experiments, we found that the number of samples, L, per edge and attribute association can be set to 1 as we take the full dataset as input in each iterator [26]. We determine the statistically significant differences using the two-tailed paired Student's t-test at a 95% confidence interval (i.e., $p < 0.05$).

7 RESULTS AND ANALYSIS

This section reports our experimental results. We first address **RQ1** by evaluating our CSAN model on two graph mining tasks, i.e., link prediction and node classification (Section 7.1). Then, to answer **RQ2**, we apply our CSAN model to address two additional tasks, i.e., attribute inference and user profiling, which cannot be solved by existing attributed network embedding methods as we have shown in the Introduction section (Section 7.2). After that, we visualize the DBLP_con network layout by arranging both the node and attribute embeddings on two-dimensional space and analysis the result for answering **RQ3**. For the evaluation of our CDAN model, we first address **RQ4** by evaluating our CDAN model on both the network reconstruction and network prediction tasks. Next, to answer (**RQ5**), we apply our CDAN model on the dynamic expert profiling task, which cannot be addressed by existing dynamic network embedding methods, since they cannot capture affinities between attributes and nodes. Finally, we visualize the variances of two-dimensional Gaussian embeddings in our DBLP_key network to discuss the uncertainty of the learned embeddings (**RQ6**).

7.1 Link Prediction and Node Classification

Link prediction, which aims to predict if there exists an edge between two nodes, is a typical task in networks analysis. Following those in References [27, 28], to evaluate the performance of our model, we randomly divide all edges into three sets, i.e., the training set (85%), the validating set (5%) and the testing set (10%). For negative instances (no edge between two nodes), we randomly sample an equal number of non-existing links. Both our CSAN and GAE [27] reconstruct both positive and negative edges during training, so we can directly obtain the probability of a link by the inner product of the embeddings between two nodes. For other baselines, we rank both positive and negative instances according to the cosine similarity between two nodes. For evaluation purposes, we employ area under the ROC curve (AUC) and average precision (AP) scores to evaluate the performance of link prediction, which are commonly used in the related literature [27, 56]. Higher values of AUC and AP indicate better performance.

Table 3 shows the link prediction performance of our CSAN model and the baselines on the six attributed networks. From Table 3, we can observe the followings. First, our CSAN model consistently performs better than any of the baseline models in this task. In Cora, Citeseer, Facebook,

Table 4. Node Classification Performance

Method	Cora		Citeseer		Pubmed		Flickr		BlogCatalog	
	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1
AANE	.715	.720	.617	.671	.779	.814	.596	.615	.597	.617
GaphSAGE	.747	.763	.594	.630	.809	.817	.647	.652	.626	.643
ANRL-WAN	.774	.634	**.671**	.713	**.846**	**.847**	.673	.697	**.656**	**.679**
GAE	.773	.788	.582	.638	.823	.828	.591	.655	.622	.636
CSAN	**.822** †	**.838**†	.642	**.720**†	.829	.834	**.692** †	**.701**†	.646	.653

The best and the second best performing runs per metric per dataset are marked in bold and underlined, respectively. In each dataset, significant improvements over the comparative methods are marked with † (paired t-test, $p < .05$).

and Pubmed datasets, CSAN can even achieve higher than 95% AUC and AP scores, which indicates that our inferred embeddings are able to predict the unobserved links with a high precision. This is mainly because our CSAN model optimizes a loss function consisting of the reconstruction error of all the edges. Second, our model obtains statistically significant improvements over the six datasets and two metrics (up to 10.39% and 11.49% on AUC and AP metrics in Flickr network) according to the paired t-test ($p < .05$), comparing with the second best model, namely the GAE model, which is also a variational auto-encoder based method but without embedding the attributes. Therefore, we can conclude that our co-embedding model is more effective than the models that learn only nodes embeddings. Third, the improvements of our CSAN model on the sparser networks (i.e., the Cora, Citesseer, Pubmed) are more obvious, comparing the relatively denser networks (i.e., the BlogCatalog and Facebook networks that have densities of 1.27% and 1.08%, respectively). This result suggests that our model can gain more improvements in sparse networks.

Node classification is another classical task commonly used to evaluate the quality of the learned node embeddings. Similarly to previous studies [18, 57], we employ Micro-F1 and Macro-F1 as metrics to measure the performance of node classification. After the node representations are obtained, we randomly sample 20% labelled nodes to train an SVM classifier and the rest are used for testing. We repeat this process for 10 times and report the average performances on both Macro-F1 (Ma_F1) and Micro-F1 (Mi_F1). Table 4 shows the performance of our CSAN model and the used baselines on five labelled attributed networks. Note that we did not take embedding methods for plain networks as our baselines in the link prediction and node classification problems, as a large number of results show that embedding networks by leveraging both the network structure and the associated node attribute can achieve better performances in the majority of downstream applications compared to approaches that only consider the topological structure [15, 18, 27].

As shown in Table 4, our proposed CSAN model performs the best in the Cora and Flickr datasets and achieves the second-best performance in other datasets, compared with the state-of-the-art attributed network embedding baselines (i.e., AANE, GaphSAGE, ANRL-WAN, and GAE models). Meanwhile, when comparing with the GAE model, we observe consistent and statistically significant improvements by the CSAN model (paired t-test, $p < 0.05$) in node classification performance over all five datasets, and our CSAN model can obtain a 6.34% and 6.35% absolute increase over Ma_F1 and Mi_F1 metrics in the Cora dataset. The results demonstrate that our CSAN model can learn effective node representations of an attributed network for the node classification task. Note that there is no method in our experiments that can achieve consistently better performance than all the other methods over all the datasets in this task. Therefore, combining with the link prediction result (Section 7.1), we can conclude that our CSAN model is indeed more effective than other static baseline models in link prediction and node classification tasks, which validates RQ1.

Table 5. Attribute Inference Performance

Method	Cora		Citeseer		Facebook		Pubmed		Flickr		BlogCatalog	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
EdgeExp	.682	.690	.707	.714	.671	.687	.586	.576	.678	.685	.684	.744
SAN	.664	.672	.679	.675	.712	.723	.579	.572	.653	.660	.694	.710
BLA	.808	.801	.854	.876	.868	.830	.622	.602	.730	.769	.787	.792
CSAN	**.932**†	**.916**†	**.954**†	**.939**†	**.974**†	**.971**†	**.670**†	**.652**†	**.867**†	**.865**†	**.868**†	**.867**†
Improvement	15.35%	14.36%	11.71%	7.19%	12.21%	16.99%	7.72%	8.31%	18.77%	12.48%	10.29%	9.47%

The best and the second best runs per metric per dataset are marked in bold and underlined, respectively. For each dataset, the last raw shows the improvements of CSAN over the best baseline algorithms, and the significant improvements over the best baseline algorithm are marked with † (paired t-test, $p < .05$).

7.2 Attribute Inference and User Profiling

Attribute inference is a task that aims to predict the values of attributes of the nodes. Since using only the node embeddings cannot decode the affinities between nodes and attributes, the four baseline methods, i.e., AANE, GraphSAGE, ANRL-WAN, and GAE, cannot be applied in this task. Similarly to the link prediction task, we randomly divide all attribute values into three sets, i.e., the training set (85%), the validating set (5%), and the testing set (10%), where the the training set is used for training, the validating set is used for tuning parameters and iterations and the performance results are reported based on the testing set. We take other three state-of-the-art attribute inference algorithms, i.e., SAN [13], EdgeExp [4], and BLA [56], as our baselines for performance comparison. We employ the AUC and AP metrics to measure the attribute inference performance. Table 5 presents the attribute inference performance of our CSAN model and the baseline models in the six attributed networks. We can find that our CSAN model performs significantly better than all the baseline methods in all our datasets. The BLA method can always achieve the second best performance in this task. Moreover, the improvement between BLA and our CSAN model is always significant. This performance improvement can be explained by the fact that our CSAN model optimizes a loss function consisting of the reconstruction error of all the attributes.

It is worth noting that our CSAN model does not need any free parameters to trade the weights between the topological structure and attributes. This highlights another important feature of CSAN. It can achieve significant improvement in performances in both the link prediction and attribute inference tasks (see Table 3 and Table 5), without task-specific fine-tuning. Another fact validated by CSAN is that attribute inference can help to improve link prediction; that is, link prediction accuracy is further improved by first inferring missing attributes [13]. In addition, our CSAN as well as all the baseline methods show relatively poor performances on the Pubmed dataset. However, our CSAN model still scores the best. The facts making the attribute inference task in this dataset more challenging include: Pubmed has the largest node set but has a relatively small size of attribute set, and the ratio between the positive instances and the negative instances of nodes' attributes is only about 0.02, resulting in an unbalanced classification problem.

The user profiling task, also known as the expertise profiling task, aims at generating k keywords for profiling each user's expertise [32]. In academic social networks, authors' academic publications were often used to learn how personal research interests evolved [10] over time. In this setting, the authors' publications indicate their expertise. To validate whether the correlation between of users and conference can be reflected by the learned embeddings, we use the embeddings of nodes and attributes to obtain the most relevant attributes for each node as their expertise. Note that in this experiment, some metrics for user profiling (e.g., NDCG and MRR) and baselines (e.g.,

Table 6. User Profiling for Example Experts

Experts	Top-4 conferences
Michael I. Jordan	NIPS, UAI, ICML, ICASSP
Geoffrey E. Hinton	NIPS, ICML, ICCV, ICASSP
Yann LeCun	NIPS, ICCV, ICML, CVPR
Robert Endre Tarjan	FOCS, STOC, COLT, SODA
Sanjeev Arora	SODA, FOCS, ICALP, COLT

DUWE) [32] are not included to the experiment for comparison and discussion, but they are important measures for quantitatively evaluating the user profiling performance when the users and their expertise have some already known relevance assessments. However, the relevance scores or ground-truth top-k keywords for our attributed networks are not available. One viable direction for addressing this issue is to manually obtain relevance of the attributes to users by crowdsourcing, which is expensive, and hence we leave this evaluation for future work. In this article, we only evaluate our model in this task based on five renowned scholars. Specifically, we first co-embed the DBLP_con academic social network into embeddings of authors and conferences. Then, we calculate the cosine similarities between authors and conferences according to their embeddings. Finally, we rank the conferences by the similarities to each author and return the top-four conferences as the expertise for each author. Table 6 reports the profiling result for five renowned scholars. Among the five scholars, three of them (Michael I. Jordan, Geoffrey E. Hinton, and Yann LeCun) are well-known experts in the field of machine learning, and the other two (Robert Endre Tarjan and Sanjeev Arora) are established researchers in the field of theoretical computer science. According to Table 6, the three machine learning experts are profiled by NIPS, ICML, UAI, and so on, which are the top-tier machine learning conferences, and the other two experts are profiled by SODA, PODC, STOC, and so on, which are the top-tier theoretical computer science conferences. We note that our model takes binary-valued attributes as input but we can still effectively retrieve top-four relevant conferences based on the embeddings of authors and conferences. Our results also reveal that capturing affinities between nodes and attributes is essential for network embedding methods to characterize the node attributes.

7.3 Network Visualization

To qualitatively evaluate the result embeddings of our CSAN model by network visualization, we first obtain two-dimensional Gaussian embeddings for authors and attributes of the DBLP_con network. Then we plot the means and variances of the resulting embeddings into two-dimensional space. To be clear in the figure, we show the embeddings of only 200 authors who own the top 200 high H-index scores[5] in the dataset. The visualization result is shown in Figure 4, from which we can make the following observations:

(1) The representations of conferences show larger variances than those of authors in general. This is because the inference model for nodes takes both the adjacency matrix and the attribute matrix of nodes as input and uses the GCN [15, 28] to take the spectral convolutions of graphs into account during the training process, which is informative to describe a node in an attributed network. However, the inference model for attributes only takes the transposed adjacency matrix as input, which has relatively sparser features than the input features of nodes, resulting in larger variance and uncertainty in their representations.

[5]Please refer to http://www.guide2research.com/scientists/.

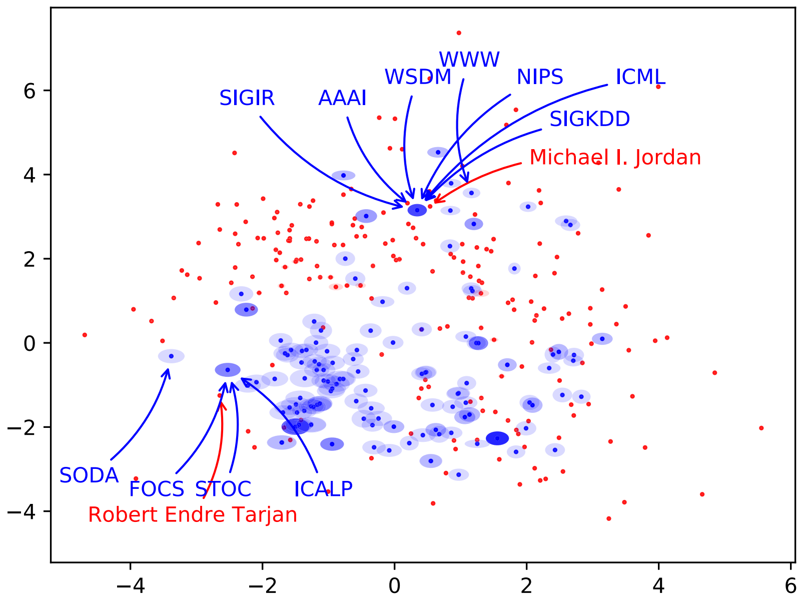


Fig. 4. Two-dimensional visualization of Gaussian embeddings of authors and conferences for DBLP_con dataset. The red nodes are the top 200 high H-index authors and the blue nodes are the 172 conferences in our DBLP_con dataset. The ellipsoids surrounding the nodes indicate variances in their embeddings. Some ellipsoids of conferences are in darker blue as these conferences are quite close and overlapped.

(2) Similar conferences have representations that are displayed closely in the same region on the two-dimensional plane. Specifically, as seen in Figure 4, SIGIR, AAAI, WSDM, WWW, NIPS, ICML, and SIGKDD, which are conferences related to machine learning and its applications, are all plotted at the top region of the plane, while SODA, FOCS, STOC, and ICALP, which are conferences of theoretical computer science, are all plotted at the bottom left region of the plane. This result indicates that our CSAN model effectively captures the similarities between attributes.

(3) From Figure 4, we can observe that author "Michael I. Jordan" is quite close to the seven machine learning conferences and author "Robert Endre Tarjan" is quite close to the four theoretical computer science conferences. It is well known that "Michael I. Jordan" is an expert in machine learning and "Robert Endre Tarjan" is an expert in theoretical computer science. Therefore, this result validates that our CSAN model can be used to effectively measure the affinities between authors and conferences, addressing RQ3.

7.4 Network Reconstruction and Prediction

We evaluate the ability of node representations in reconstructing the original network including links and attributes of the final time step T. We randomly select 10% of links and attribute associations as the positive testing set and randomly sample an equal number of non-existing links and associations as the negative testing set. Then, we rank both positive and negative instances according to the cosine similarity between their embeddings. Since existing network embedding methods that only learn node representations cannot reconstruct attributes of a DAN, we compare our CDAN1 and CDAN models with other two attribute inference baselines, i.e., SAN and BLA. We use the commonly used area under the ROC curve (AUC) and average precision (AP) metrics for performance evaluation. As shown in Table 7, our CDAN model consistently performs better

Table 7. Network Reconstruction Performance

	Method	Flickr		BlogCatalog		DBLP_key	
		AUC	AP	AUC	AP	AUC	AP
Link	**DW**	.694	.720	.666	.665	.598	.604
	GAE	.828	.827	.815	.811	.754	.731
	DTriad	.757	.763	.701	.697	.675	.684
	DANE	.795	.790	.739	.741	.710	.712
	CDAN1	.848	.845	.810	.809	.765	.767
	CDAN	**.895**†	**.909**†	**.857**†	**.845**†	**.801**†	**.809**†
Attribute	**SAN**	.653	.660	.694	.710	.679	.671
	BLA	.730	.769	.787	.792	.710	.710
	CDAN1	.828	.787	.816	.819	.809	.801
	CDAN	**.861**†	**.859**†	**.848**†	**.845**†	**.841**†	**.839**†

The best and second best performing run per metric per dataset is marked in bold and underlined, respectively. In each dataset, significant improvements over the comparative methods are marked with † (paired t-test, $p < .05$).

Table 8. Network Prediction Performance

	Method	Flickr		BlogCatalog		DBLP_key	
		AUC	AP	AUC	AP	AUC	AP
Link	**DW**	.680	.694	.650	.657	.581	.598
	VGAE	.818	.816	.806	.809	.708	.711
	DTriad	.732	.736	.685	.674	.664	.670
	DANE	.775	.778	.709	.711	.700	.702
	CDAN1	.835	.830	.797	.801	.738	.727
	CDAN	**.884**†	**.897**†	**.835**†	**.836**†	**.790**†	**.793**†
Attribute	**SAN**	.647	.652	.691	.694	.657	.658
	BLA	.722	.759	.779	.782	.700	.703
	CDAN1	.796	.778	.790	.796	.766	.769
	CDAN	**.845**†	**.844**†	**.833**†	**.832**†	**.825**†	**.826**†

The best and second best performing run per metric per dataset is marked in bold and underlined, respectively. In each dataset, significant improvements over the comparative methods are marked with † (paired t-test, $p < .05$).

than any of the baseline models. This result can be explained by the fact that our CDAN model optimizes a loss function consisting of the reconstruction error of the links and attributes. We also find that CDAN achieves a substantial gain over CDAN1, which validates the need for integrating perturbation features from the nodes' and attributes' local neighborhoods and associations.

The network prediction task is similar to the network reconstruction task, with the only difference being that it aims to use the embeddings of nodes and attributes at $T-1$ to predict links and attributes at the next time step T. Table 8 presents the performance results of the compared methods. From Table 8, we can observe that, as expected, our CDAN model with the second-order perturbation performs better than all the other baselines in terms of the link prediction and attribute inference tasks in all the three dynamic attributed networks, and the improvement over the best baseline is statistically significant. In particular, we see that even when using only the first-order perturbation in our CDAN model (i.e., CDAN1), it is still able to outperform the other two models (i.e., SAN and BLA) in the attribute inference task. It is worth mentioning that GAE

Table 9. Experts Profiling over Years in DBLP_key Network

Experts	2008	2012	2017
Michael I. Jordan	probabilist	bayesian	variat
	inferenc	sampl	mixtur
	process	process	inferenc
Geoffrey E. Hinton	handwritten	dropout	attend
	boltzmann	imag	imag
	brain	boltzmann	learn
Yann LeCun	featur	acceler	convolut
	energy-bas	convolut	featur
	recognit	featur	neural

obtains better performances on both the ACU and AP metrics than other two dynamic embedding baselines (i.e., DTriad and DANE), though it still underperform our CDAN in the link prediction task. Note that GAE model is an generative extension of the graph convolutional neural network [28] to learn distributional representation based on variational auto-encoders. This result suggests that the distributional representation learned by variational auto-encoders is able to capture/encode more information from the training instances. The temporal network prediction result indicates that our proposed CDAN model is able to learn dynamic embeddings capturing the evolving trend from the observed network perturbations to predict links and attributes for the future. The high performance of our model in both the network reconstruction and network prediction tasks validates RQ4 that our CSAN model can learn more effective representations for the dynamic attributed networks than the state-of-the-art non-dynamic and dynamic embedding models.

7.5 A Case Study of Dynamic Expert Profiling

We also evaluate our CDAN model by a case study of the expert profiling task, which aims to generate k keywords to describe the expertise of experts [32]. In academic social networks, the authors' publications are often used to learn how personal research interests evolve over time [10]. For the dynamic expert profiling task in our DBLP_key dataset, we use the keyterms of their publication titles (attributes) to describe the authors' expertise. Specifically, we first obtain the embeddings of authors and keyterms, and then we return the top-three keyterms for each author according to the cosine similarities of embeddings between authors and keyterms. Table 9 reports the profiling result for three well-known machine learning experts, i.e., Michael I. Jordan, Geoffrey E. Hinton, and Yann LeCun, over the years of 2008, 2012, and 2017.

From Table 9, we find that all of their expertise evolves over time, and our model can precisely find relevant keyterms for the expertise of these three experts. For instance, in 2008 Yann LeCun is profiled by "featur," "energy-bas," and "recognit," as there are several energy-based and feature learning models proposed by this author in that year, while in 2012 and 2017, the profiling evolved into "convolut" and "neural," since some of his works on convolutional neural network were published in the later years. Meanwhile, we can observe that the keyword "featur" always keeps in the top-three over the three years, which also can be validated by the fact that Yann LeCun always published papers about "feature extraction." For the other two experts, similar results can be observed by comparing the top-three keywords with their published papers. The result validates RQ5 that our CDAN model can effectively capture the dynamic affinities between nodes and attributes.

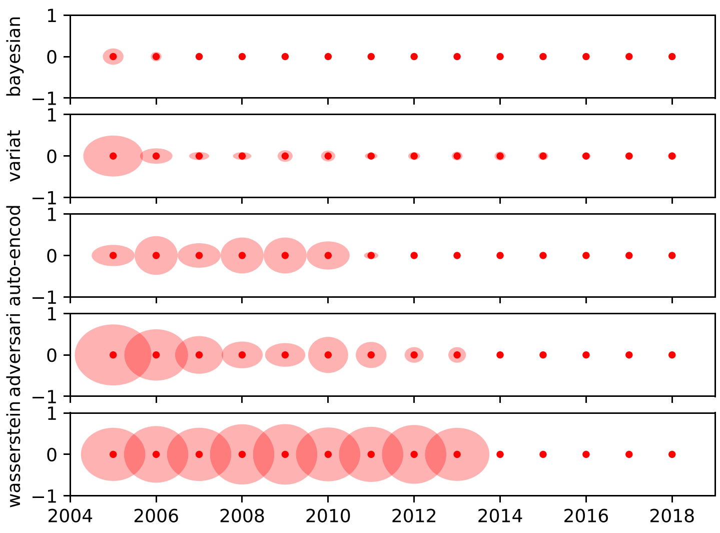


Fig. 5. Variances of Gaussian embeddings evolving over years for five attributes in DBLP_key network.

7.6 Visualizing the Uncertainty

To show the uncertainty captured by our dynamic embedding model, we visualize the variances of two-dimensional Gaussian embeddings in our DBLP_key network in Figure 5 by ellipsoids for five attributes (i.e., "bayesian," "variat," "auto-encod," "adversari," and "wasserstein") over 14 years. We can observe that the embeddings of "bayesian" show lower variances than other attributes over all 14 years, while "wasserstein" shows the largest variances. Note that our model represents nodes and attributes of a dynamic attributed network by Gaussian distributions, where the mean vectors indicate the position of the nodes on the embedding space and the variances reflect the uncertainty of these vectors. This result can be explained by the fact that "bayesian" has been consistently attracting enduring attention over all 14 years while the publications with the "wasserstein" keyword only appear in the recent 5 years of our dataset. As a consequence, the representation of "wasserstein" is more uncertain than that of "bayesian." The variances of embeddings for the other three attributes experience a certain decrease as the year increases, which also coincides with the popularity of the three keywords. In deed, this result also can be validated according to the analytical forms of the KL-divergence terms (between Gaussian posteriors and Normal Gaussian priors) in our loss, i.e., $\frac{1}{2}\sum_{j=1}^{J}(1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2)$, where the term $\log((\sigma_j)^2) - (\sigma_j)^2$ is a concave function (its maximum value can be obtained when $\sigma_j = 1$). Since all the variances are constrained to be less than or equal to 1 by our nonlinearity function of the inference network, if we keep all the other parameters fixed and optimize the KL loss w.r.t the variance σ_j,[6] then the received more updates by the training instances for minimizing the KL loss will lead to the reduction of the variances. Lower variances can make the position of nodes and attributes in the embedding space to be more certain, which means we can gain more confidence for the downstream learning tasks by using the embedding with lower variances. This result validates that our CDAN model effectively captures the dynamic uncertainty about the learned representation (RQ6).

[6]Note that only the KL terms of the loss function contain the variance parameters.

8 CONCLUSIONS

In this article, we have studied the problem of learning representations for static and dynamic attributed networks. Specifically, we have proposed a novel co-embedding algorithm, called CSAN, to deal with static attributed networks, and then extended it with our proposed CDAN model to deal with the dynamic attributed networks. The proposed CSAN model learns the low-dimensional representations of both nodes and attributes in the same semantic space, such that the affinities between the nodes and attributes of the networks can be effectively measured. To learn high-quality embeddings of both nodes and attributes in attributed networks, we have proposed a variational auto-encoder algorithm, which consists of an inference model for encoding the features of nodes and attributes into Gaussian distributions and a generative model for reconstructing both real and binary weighted edges and attributes. Through our customized dynamic VAE model, we also presented a novel dynamic attributed networks embedding model that learns low-dimensional Gaussian embeddings for both nodes and attributes over time such that the evolving patterns as well as the affinities between nodes and attributes can be captured. Our two models represent a network by Gaussian distributions of nodes and attributes, which innately represent the uncertainty property of the embeddings for the network.

In our experiments, we have evaluated the performances of CSAN and CDAN in comparison with effective baseline algorithms on 10 static/dynamic attributed networks. The experimental results demonstrated that our CSAN model is able to learn informative and high-quality representations of both nodes and attributes and significantly outperforms the state-of-the-art methods on several application tasks. Visualization on the DBLP data showed that our CSAN model effectively maps nodes and attributes into a unified semantic space, where the affinities between authors and conferences can be preserved. Experimental results on dynamic social networks demonstrated that our CDAN model achieves substantial gains in four applications compared to several state-of-the-art techniques and is able to capture the dynamic uncertainty.

As to the future work, we aim to extend our models to heterogeneous networks. How to effectively deploy the proposed dynamic embedding to dynamic networks with variable node and attribute sizes is also a key next-step in our investigation.

ACKNOWLEDGMENTS

We thank the reviewers for their valuable comments.

REFERENCES

[1] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. 2017. Variational inference: A review for statisticians. *J. Am. Statist. Assoc.* 112, 518 (2017), 859–877.

[2] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *Proceedings of the International Conference on Learning Representations*.

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 891–900.

[4] Deepayan Chakrabarti, Stanislav Funiak, Jonathan Chang, and Sofus A. Macskassy. 2014. Joint inference of multiple label types in large networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning, Volume 32*. II–874.

[5] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *Proceedings of the International Conference on Learning Representations*.

[6] Ludovic Dos Santos, Benjamin Piwowarski, and Patrick Gallinari. 2016. Multilabel classification on heterogeneous graphs with gaussian embeddings. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 606–622.

[7] Alexey Dosovitskiy and Thomas Brox. 2016. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*. 658–666.

[8] Lun Du, Yun Wang, Guojie Song, Zhicong Lu, and Junshan Wang. 2018. Dynamic network embedding: An extended approach for skip-gram based network embedding. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2086–2092.

[9] Cristóbal Esteban, Volker Tresp, Yinchong Yang, Stephan Baier, and Denis Krompaß. 2016. Predicting the co-evolution of event and knowledge graphs. In *Proceedings of the 2016 19th International Conference on Information Fusion*. 98–105.

[10] Yi Fang and Archana Godavarthy. 2014. Modeling the dynamics of personal expertise. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1107–1110.

[11] Hongchang Gao and Heng Huang. 2018. Deep attributed network embedding. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 3364–3370.

[12] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1416–1424.

[13] Neil Zhenqiang Gong, Ameet Talwalkar, Lester Mackey, Ling Huang, Eui Chul Richard Shin, Emil Stefanov, Elaine (Runting) Shi, and Dawn Song. 2014. Joint link prediction and attribute inference using a social-attribute network. *ACM Trans. Intell. Syst. Technol.* 5, 2 (2014), 27:1–27:20.

[14] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 855–864.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[16] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Vol. 1. 1489–1501.

[17] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 623–632.

[18] Xiao Huang, Jundong Li, and Xia Hu. 2017. Accelerated attributed network embedding. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. 633–641.

[19] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. 731–739.

[20] Xiao Huang, Qingquan Song, Jundong Li, and Xia Hu. 2018. Exploring expert cognition for attributed network embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 270–278.

[21] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. 2017. Variational deep embedding: An unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 1965–1972.

[22] Yoon Kim, Yi-I. Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*. 61.

[23] Diederik Kingma and Max Welling. 2014. Efficient gradient-based inference through transformations between bayes nets and neural nets. In *Proceedings of the International Conference on Machine Learning*. 1782–1790.

[24] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

[25] Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*. 3581–3589.

[26] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations*.

[27] Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. In *Proceedings of the NIPS Workshop on Bayesian Deep Learning*.

[28] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*. In *Proceedings of the International Conference on Learning Representations*.

[29] Jure Leskovec and Julian J. Mcauley. 2012. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems*. 539–547.

[30] Jundong Li, Kewei Cheng, Liang Wu, and Huan Liu. 2018. Streaming link prediction on dynamic attributed networks. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 369–377.

[31] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 387–396.

[32] Shangsong Liang, Xiangliang Zhang, Zhaochun Ren, and Evangelos Kanoulas. 2018. Dynamic embeddings for user profiling in Twitter. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1764–1773.

[33] Yupeng Luo, Shangsong Liang, and Zaiqiao Meng. 2019. Constrained co-embedding model for user profiling in question answering communities. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 439–448.

[34] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2016. Adversarial autoencoders. In *Proceedings of the Workshop, International Conference on Learning Representations (ICLR'16)*.

[35] Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2018. Automatic ground truth expansion for timeline evaluation. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 685–694.

[36] Richard McCreadie, Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. 2018. Explicit diversification of event aspects for temporal summarization. *ACM Trans. Inf. Syst.* 36, 3 (2018), 25.

[37] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-embedding attributed networks. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. 393–401.

[38] Zaiqiao Meng, Shangsong Liang, Jinyuan Fang, and Teng Xiao. 2019. Semi-supervisedly co-embedding attributed networks. In *Advances in Neural Information Processing Systems*. 6504–6513.

[39] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. 3111–3119.

[40] Eric Nalisnick and Padhraic Smyth. 2017. Stick-breaking variational autoencoders. In *Proceedings of the International Conference on Learning Representations*.

[41] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

[42] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 701–710.

[43] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 459–467.

[44] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Mag.* 29, 3 (2008), 93.

[45] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*. 3483–3491.

[46] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[47] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. 1067–1077.

[48] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 817–826.

[49] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *Proceedings of the 34th International Conference on Machine Learning, Volume 70*. 3462–3471.

[50] Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *International Conference on Learning Representations*. In *Proceedings of the International Conference on Learning Representations*.

[51] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1225–1234.

[52] Suhang Wang, Charu Aggarwal, Jiliang Tang, and Huan Liu. 2017. Attributed signed network embedding. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 137–146.

[53] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. 203–209.

[54] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning*.

[55] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. 2015. Network representation learning with rich text information. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 2111–2117.

[56] Carl Yang, Lin Zhong, Li-Jia Li, and Luo Jie. 2017. Bi-directional joint inference for user links and attributes on large social graphs. In *Proceedings of the 26th International Conference on World Wide Web Companion*. 564–573.

[57] Zhen Zhang, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang. 2018. ANRL: Attributed network representation learning via deep neural networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 3155–3161.

[58] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence.*

[59] L. Zhu, D. Guo, J. Yin, G. V. Steeg, and A. Galstyan. 2016. Scalable temporal latent space inference for link prediction in dynamic social networks. *IEEE Trans. Knowl. Data Engin.* 28, 10 (Oct. 2016), 2765–2777.

[60] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding temporal network via neighborhood formation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2857–2866.

Received April 2019; revised October 2019; accepted December 2019