Dynamic Representation Learning for Large-Scale Attributed Networks

Zhijun Liu
Yantai University
Yantai, China
liuzhijun9503@126.com

Chao Huang
JD Finance America Corporation
Mountain View, USA
chaohuang75@gmail.com

Yanwei Yu*
Ocean University of China
Qingdao, China
yuyanwei@ouc.edu.cn

Peng Song
Yantai University
Yantai, China
pengsongseu@gmail.com

Baode Fan
Yantai University
Yantai, China
fanbaodeyt@163.com

Junyu Dong
Ocean University of China
Qingdao, China
dongjunyu@ouc.edu.cn

ABSTRACT

Network embedding, which aims at learning low-dimensional representations of nodes in a network, has drawn much attention for various network mining tasks, ranging from link prediction to node classification. In addition to network topological information, there also exist rich attributes associated with network structure, which exerts large effects on the network formation. Hence, many efforts have been devoted to tackling attributed network embedding tasks. However, they are also limited in their assumption of static network data as they do not account for evolving network structure as well as changes in the associated attributes. Furthermore, scalability is a key factor when performing representation learning on large-scale networks with huge number of nodes and edges. In this work, we address these challenges by developing the DRLAN–Dynamic Representation Learning framework for large-scale Attributed Networks. The DRLAN model generalizes the dynamic attributed network embedding from two perspectives: First, we develop an integrative learning framework with an offline batch embedding module to preserve both the node and attribute proximities, and online network embedding model that recursively updates learned representation vectors. Second, we design a recursive pre-projection mechanism to efficiently model the attribute correlations based on the associative property of matrices. Finally, we perform extensive experiments on three real-world network datasets to show the superiority of DRLAN against state-of-the-art network embedding techniques in terms of both effectiveness and efficiency. The source code is available at: https://github.com/ZhijunLiu95/DRLAN.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Computing methodologies** → **Learning latent representations**.

*Yanwei Yu is the corresponding author.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411945

KEYWORDS

Network representation learning; large-scale attributed networks; dynamic networks; sparse random projection

ACM Reference Format:
Zhijun Liu, Chao Huang, Yanwei Yu, Peng Song, Baode Fan, and Junyu Dong. 2020. Dynamic Representation Learning for Large-Scale Attributed Networks. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411945

1 INTRODUCTION

Network embedding has received great attention owing to its ability in learning low-dimensional representations for nodes in a network for a variety of real-world mining applications, such as link prediction [2, 34], node classification [6, 8], network reconstruction [1], spatial data learning [25, 31] and user behavior modeling [7, 13, 32]. In these applications, the core of network representation learning methods is preserving the proximity of nodes based on the topological structural information of a network, without involving hand-engineering domain-specific features [8]. In recent years, due to the availability and usefulness of rich attribute information (*e.g.*, user profiles in social media or author affiliations in academic networks) in real-world network analysis scenarios, attributed network embedding [3, 8] has become a promising solution to improve the performance of network representation learning. These studies indicate that the network structure formulation is highly correlated with the associated node attributes. In particular, these models represent a vertex as a learnable embedding vector with the joint consideration of network topological structure and node attributes.

Despite the effectiveness of existing attributed network embedding methods [3, 9, 14, 23], most of them miss an important issue of network representation – network structures and node attributes would evolve over time, and thus the assumption of node proximity and attribute consistencies does not hold any more. In such cases, the node relations and the associated attribute information are no longer static. Therefore, it is crucial to take the above two-dimensional dynamics (*i.e.*, both the network structure and associated attributes) into consideration for more accurate representation learning on attributed networks.

One possible solution to deal with dynamic embedding learning scenarios of attributed networks, is to run batch algorithms until

all evolving data are collected. Nevertheless, the batch algorithms will either need to operate on the data from the first timestamp to the current one (which is very costly on large-scale data), or ignore some previously inferred results and run from scratch on a sliding recent time window (which does not exploit all available data). As such, these static network embedding models are not well-suited for dynamic attribute network embedding. While there exists a handful of recent study that considers dynamic information in attributed network embedding [11], this method is limited to its inefficiency issue. In this work, we strive to develop an effective and efficient network representation learning method which explicitly and jointly encodes dynamic network topological and attributed signals on large-scale network datasets.

There are several key challenges that remain to be solved, in order to realize the dynamic representation learning for large-scale attributed networks. In particular, *First*, in light of these limitations of existing network embedding methods, perform representation learning for large-scale networks with the goal of jointly preserving global structure contexts and attributed information, remains a significant challenge. *Second*, the variation of network structure and node attributes raises the challenge of capturing the correlations among nodes in a timely manner. *Third*, it is important and challenging to operate the representation learning framework on the time-evolving network data and update the learned embedding vectors on the fly.

To tackle the aforementioned challenges, we develop a new framework (*i.e.*, DRLAN)–Dynamic Representation Learning method for large-scale Attributed Networks. To be more specific, we first develop a sparse random projection module to learn node representations as the main embedding space, where both the network structure and attribute proximities can be well preserved. To capture the dynamic network signals from both the topological and semantic dimensions, we propose an online network embedding model that recursively updates learned representation vectors based on the new network information. Towards scalable modeling of attribute similarities, a recursive pre-projection mechanism is developed to efficiently estimate the correlations among node attributes based on the associative property of matrix product. We further perform complexity analysis of the developed DRLAN framework. The results of this work are important since they enable the representation learning on attributed networks on the fly even for large-scale network data.

We highlight the key contributions of this work as follows:

- We present DRLAN: a dynamic representation learning framework to learn network embeddings on large-scale attributed networks. Our model is capable of modeling the time-evolving network characteristics (*i.e.*, network structural and attributed information) for dynamic attributed network embedding.
- Our developed DRLAN model is an integrative learning framework with the offline learning phase to generate node embeddings by discovering a latent node attribute and network structure guided learning subspace, and an online learning phase to deal with dynamic network characteristics by incrementally update node embeddings in a timely manner.
- In the main embedding space, we develop a recursive pre-projection mechanism to efficiently model similarities among node attributes on the fly in the view of dynamic network structures. We also analyze the time complexity of DRLAN framework.
- We conduct extensive experiments for various representative network mining tasks (*i.e.*, node classification, link prediction, and network reconstruction) on three real-world networks. Experimental results demonstrate the effectiveness and efficiency of our DRLAN framework in learning representation for dynamic large-scale attributed networks, as compared to state-of-the-art network embedding techniques.

2 RELATED WORK

Network Embedding. Recently, static network embedding methods can be classified into two categories: one for plain networks where only topological structure information is utilized for embedding, e.g., DeepWalk [20] employs a truncated random walk on network then utilizes the word embedding to learn latent representations of the network structure information. Node2vec [5] further extends DeepWalk by adding the flexibility in exploring network structure. LINE [24] tactfully designs an optimized objective function that preserves first-order and second-order proximities to learn network structure representations. NetSMF [21] leverages theories from spectral sparsification to efficiently sparsify the dense NetMF matrix implicitly factorized by DeepWalk, which can support large-scale network embedding learning.

Another one for non-plain networks such as complex information network which considers not only topological structure information but also auxiliary information of nodes, e.g., attributes or types of nodes. Matrix factorization models are proposed to learn representations by considering both structure and node attribute information in [29]. Gao et al. [3], Tu et al. [27] and Meng et al. [17] use deep learning to learn a joint feature representation for attributed networks. LGCN [4] transforms graph data into grid-like structures and then applies regular convolutional operations to learn graph representation. In addition to node attributes, node labels are also taken into consideration in [8]. NetHash [28] employs the randomized hashing technique to encode shallow trees to preserve information about large-scale attributed network. FeatWalk [9] utilizes a similarity-based random walks among heterogeneous attributed network to extract the local structure proximity and attribute proximity. However, such methods cannot be applied to dynamic environments.

Besides static networks, how to embed dynamic networks where structures and attributes change over time also attracts research attention. DLNE [33] updates previous embeddings by considering the changes of the networks. DHPE [38] and DANE [11] propose to use matrix perturbation to handle the changes of edges or attribute values. However, they are at least quadratic time complexity with respect to the number of nodes, thus it is difficult to apply these methods to dynamic large-scale attributed networks. Recently, TIMERS [36] proposes to optimally set the restart time to reduce error accumulation of incremental SVD in time for dynamic networks. DynamicTriad [37] uses triadic closure process to capture the network dynamics in topological structure. However, these methods only capture the dynamic structural changes for plain networks. CTDNE [18] uses temporal random walk to capture the continuous time dynamics of the network. MMDNE [16] captures

micro and macro changes in the network through temporal attention point process and the number of network edges. DySAT [22] and DyRep [26] employ self-attention network and deep temporal point process to capture the structural neighborhood and temporal dynamics. Such methods based on deep neural networks can learn structural representations of nodes over dynamic networks, but they are all time-consuming, thus these methods cannot handle dynamic large-scale networks that are updated quickly and frequently in a timely manner.

Random Projection. Random projection is motivated by the Johnson-Lindenstrauss lemma [10], which uses a random matrix with unit Euclidean column norms to find a lower-dimensional subspace that approximately preserves the distances between all pairs of data points in the original space. However, performing such a projection, while conceptually simple, is non-trivial, especially in a real-world environment. In order to speed up the efficiency of random projection, Li et al. [12] proposes to use the very sparse projection matrix with i.i.d. entries, which promotes the implementation of memory and computation efficient algorithms. Recently, some works [35] aim to capture structure information of network by a variant of Gaussian random projection. However, these methods do not consider how to embed node attributes, thus cannot be directly applied to our problem of dynamic attributed network embedding.

3 PROBLEM DEFINITION

In this section, we introduce key notations used in this paper and then formally define the studied problem.

3.1 Notations

The main symbols used in this paper are listed in Table 1. We use normal lowercase characters to denote scalars (e.g., a), bold lowercase characters to denote vectors (e.g., $\mathbf{a}$), bold uppercase alphabets to denote matrices (e.g., $\mathbf{A}$). We use dot to denote the matrix product of two matrices, e.g., $\mathbf{A} \cdot \mathbf{B}$. The transpose of matrix $\mathbf{A}$ is represented as $\mathbf{A}^T$. $\mathbf{I}$ denotes the identity matrix.

Table 1: Main symbols and their definitions.

Notations	Definitions
$\mathbf{A}_{(t)}$	adjacency matrix of network structure at time step t
$\Delta\mathbf{A}_{(t)}$	changes of adjacency matrix at time step t
$\mathbf{X}_{(t)}$	attribute information for nodes at time step t
$\Delta\mathbf{X}_{(t)}$	changes of attribute matrix at time step t
$\mathbf{S}$	attribute similarity matrix of node
$\mathbf{R}$	random projection matrix
$\mathbf{U}_{(t)}$	embedding representation matrix at time step t
d	dimension of embedding representations
p	order for structure proximity
q	order for attribute proximity
s	sparsity of random projection matrix
Δn	number of added nodes
Δk	number of added attributes

3.2 Problem Formulation

Given a time-stamped attributed network, we aim to provide online representation for every snapshot in time. Let $\mathbf{u}_{(t)} = \{u_1, u_2, ..., u_n\}$ denote a set of n nodes in the attributed network $\mathcal{G}_{(t)}$ at time step t. We use the adjacency matrix $\mathbf{A}_{(t)} \in \mathbb{R}^{n \times n}$ to represent the network structure of $\mathcal{G}_{(t)}$. In addition, we assume that nodes are affiliated with k-dimensional attributes $\mathbf{f} = \{f_1, f_2, ..., f_k\}$ and $\mathbf{X}_{(t)} \in \mathbb{R}^{n \times k}$ denotes the node attribute matrix. At the following time step, the attributed network is characterized with both topology and content drift such that new/old edges may be added/deleted, and node attribute values and the number of attributes could also change. We use $\Delta\mathbf{A}$ and $\Delta\mathbf{X}$ to denote the network and attribute changes between two consecutive time step t and time step t+1, respectively. Following the settings of [11], we split dynamic attributed embedding problem into two sub-problems. First, we build an offline model to embed the static attributed network to a low-dimensional vector space, where structure and attribute proximities between nodes are preserved. Second, we propose a dynamic model to update the embedding of nodes corresponding to the changes in the attribute network at following time steps. These two sub-problems are summarized as follow:

Problem 1. *The offline model of DRLAN at time step t_0: Given the network adjacency matrix $\mathbf{A}_{(t_0)}$ and node attribute matrix $\mathbf{X}_{(t_0)}$, the offline model aims to output attributed network embedding $\mathbf{U}_{(t_0)}$.*

Problem 2. *The online model of DRLAN at time step t+1: Given network adjacency matrix $\mathbf{A}_{(t+1)}$ and node attribute matrix $\mathbf{X}_{(t+1)}$, and intermediate embedding result $\mathbf{U}_{(t)}$ at time step t, the online model aims to output attributed network embedding $\mathbf{U}_{(t+1)}$ based on $\mathbf{U}_{(t)}$.*

4 THE PROPOSED FRAMEWORK – DRLAN

In this section, we first present an offline learning module that learns embedding representation for attributed networks in a static setting to tackle Problem 1. Then we propose an online learning module that dynamically updates consensus embeddings on the fly with changes in the network to tackle Problem 2. At the end, we analyze the computational complexity of our model.

4.1 DRLAN: Offline Learning Phase

Network topology structures and node attributes present different information about nodes of attributed networks in different perspectives. In general, either of these information could be sparse and noisy, presenting great challenges to learn embedding representation [11]. Fortunately, the previous work has shown that high-order proximities are essential to mitigate the network structure sparsity in finding better representation, which can be formulated as a polynomial function of the adjacency matrix [30]. Rich node attributes could further mitigate the problem. In this paper, we aim to preserve high-order structure proximity and attribute similarity in the embedding matrix with the following objective function:

$$\begin{aligned} &\min_{\mathbf{U}} \|\mathbf{H}\mathbf{H}^T - \mathbf{U}\mathbf{U}^T\|_2 \\ &\mathbf{H} = \beta(\alpha_0\mathbf{I} + \alpha_1\mathbf{A} + \cdots + \alpha_p\mathbf{A}^p) \\ &\quad + (1-\beta)(\theta_0\mathbf{I} + \theta_1\mathbf{S} + \cdots + \theta_q\mathbf{S}^q), \end{aligned} \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{n \times d}$ is a low-dimensional representation matrix, d is the embedding dimension size, $\mathbf{H}$ is the proximity matrix of high-order network structure and attribute proximities in attributed network, $\mathbf{A}^p$ is p-order structure proximity, $\mathbf{S}^q$ is q-order attribute

proximity, and $\alpha_0, \alpha_1, \ldots, \alpha_p, \theta_0, \theta_1, \ldots, \theta_q$, and β are predefined weights.

Then, as discussed in [19], the objective function can be solved by the generalized Singular Value Decomposition (GSVD) method. However, GSVD is computationally expensive and thus not suitable for large-scale attributed networks. To optimize the objective function in Eq. (1), a simple but powerful method is random projection. The pairwise similarity of high dimensional data can be preserved effectively by random projection [12, 35]. There are variants of random projection techniques are applied in manifold learning area. The sparse random projection we use is one of the variations, which achieves a significant speedup with little loss in accuracy. Formally, let projection matrix $\mathbf{R} \in \mathbb{R}^{n \times d}$ with *i.i.d* entries drawn from:

$$r_{ji} = \sqrt{s} \begin{cases} 1 & \text{with prob. } \frac{1}{2s} \\ 0 & \text{with prob. } 1 - \frac{1}{s} \\ -1 & \text{with prob. } \frac{1}{2s} \end{cases}, \tag{2}$$

where $s = \sqrt{n}$ or $s = \frac{n}{\log n}$ [12]. With $s = \sqrt{n}$, one can achieve a $\sqrt{n}$-fold speedup because only $\frac{1}{\sqrt{n}}$ of the data need to be processed. Since the multiplications with $\sqrt{s}$ can be delayed, no floating point arithmetic is needed.

Next, we take the high-order proximities as a multi-layer proximity network that each layer represents a different order of proximity, and we can obtain the representation $\mathbf{U}$ by randomly projecting the proximity matrix $\mathbf{H}$ into a low-dimensional subspace:

$$\begin{aligned} \mathbf{U} =& \mathbf{H} \cdot \mathbf{R} \\ =& \beta(\alpha_0 \mathbf{I} + \alpha_1 \mathbf{A} + \cdots + \alpha_p \mathbf{A}^p) \cdot \mathbf{R} \\ &+ (1-\beta)(\theta_0 \mathbf{I} + \theta_1 \mathbf{S} + \cdots + \theta_q \mathbf{S}^q) \cdot \mathbf{R} \\ =& \beta \mathbf{U}^{\mathbf{A}} + (1-\beta)\mathbf{U}^{\mathbf{X}}, \end{aligned} \tag{3}$$

where $\mathbf{U}^{\mathbf{A}}$ and $\mathbf{U}^{\mathbf{X}}$ is structure and attribute representation, respectively.

Inspired by [35], we can improve the efficiency by the associative property of matrix multiplication to reuse previously computed matrix products $\mathbf{A}^i \cdot \mathbf{R}$ to avoid explicit calculations of higher order proximity matrices $\mathbf{A}^{i+1} \cdot \mathbf{R}$:

$$\mathbf{U}^{\mathbf{A}} = \alpha_0 \mathbf{Y}_0 + \alpha_1 \mathbf{Y}_1 + \cdots + \alpha_p \mathbf{Y}_p \tag{4}$$

where $\mathbf{Y}_0 = \mathbf{R}$ and $\mathbf{Y}_i = \mathbf{A} \cdot \mathbf{Y}_{i-1}, \forall 1 \le i \le p$.

It is worth noting that the attribute similarity matrix $\mathbf{S}$ usually is not a sparse matrix so both its storage and computation are very time-consuming. All pairwise proximities in $\mathbf{S}$ can be computed as $\mathbf{X} \cdot \mathbf{X}^{\mathbf{T}}$, at the cost of time $O(n^2 k)$, which is often prohibitive for large n and k, in large-scale network. To solve this problem, we design a recursive pre-projection mechanism to avoid explicit calculation of the enormous matrix $\mathbf{S}$ by leveraging the associative property of matrix. For simplicity, we use the simplest inner product to describe our method:

$$\begin{aligned} \mathbf{U}^{\mathbf{X}} &= (\theta_0 \mathbf{I} + \theta_1 \mathbf{S} + \cdots + \theta_q \mathbf{S}^q) \cdot \mathbf{R} \\ &= \theta_0 \mathbf{I} \cdot \mathbf{Z}_0 + \theta_1 \mathbf{X} \cdot \mathbf{Z}_1 + \cdots + \theta_q \mathbf{X} \cdot \mathbf{Z}_q \end{aligned} \tag{5}$$

where $\mathbf{Z}_0 = \mathbf{R}$, $\mathbf{Z}_1 = \mathbf{X}^T \cdot \mathbf{R}$, and $\mathbf{Z}_i = \mathbf{X}^{\mathbf{T}}(\mathbf{X} \cdot \mathbf{Z}_{i-1}), \forall 2 \le i \le q$. The matrix $\mathbf{Z}_i$ is the projection of attribute matrix $\mathbf{X}$ in low dimensional space on i-th order proximity, which preserves all pairwise distances of $\mathbf{X}$. Then we can indirectly obtain the representation vector with the core features of attribute similarity matrix $\mathbf{S}^i$ by $\mathbf{X} \cdot \mathbf{Z}_i$. Finally, we achieve a substantial cost reduction for computing $\mathbf{S} \cdot \mathbf{R}$ from $O(n^2 k + n^2 d)$ to $O(2nkd)$. As a result, to obtain a consensus embedding representation from $\mathbf{H}$, we could take the high-order structure proximity and attribute similarity together. Algorithm 1 shows the pseudo-code of our offline model.

Algorithm 1 DRLAN: Offline Learning Phase

Input: Adjacency matrix $\mathbf{A}$, attribute matrix $\mathbf{X}$, dimension d, orders p and q, weights $\alpha_0, \alpha_1, \ldots, \alpha_p, \theta_0, \theta_1, \ldots, \theta_q$, hyper-parameter β.
Output: Representation results $\mathbf{U}$

```
Generate R ∈ R^{n×d} by Eq. (2)
Y_0 ← R
Z_0 ← R, Z_1 ← X^T R
for i in 1 : p do
    Calculate Y_i = A · Y_{i−1}
end for
for i in 2 : q do
    Calculate Z_i = X^T · (X · Z_{i−1})
end for
Calculate U^A using Eq. (4)
Calculate U^X using Eq. (5)
Calculate U = βU^A + (1 − β)U^X
```

4.2 DRLAN: Online Learning Phase

In general, attributed networks often exhibit high dynamics. For example, in social networks, social relations are continuously evolving, and user preference characteristics may also evolve accordingly. The changes may be passed on to their friends, and have a profound impact on network information. But the only way of the existing offline methods handle this situation is to recalculate the entire representation. When the data is large (e.g., 100 million users), this is obviously not feasible. Therefore, it is critical to build an efficient online embedding algorithm which can directly update the existing representation on the fly for the large-scale attributed networks.

Next, we present our online learning model for dynamic attributed networks. Here, we divide the dynamic changes of attributed networks into two categories: structure changes and attribute changes.

4.2.1 **Structure Changes.** The structure changes can be divided into the changes of edges and the changes of nodes. For changes of edges, we only need to update the representation from the previous representation. Formally, we denote the changes in the adjacency matrix as $\Delta\mathbf{A}$ and the changes in $\mathbf{U}^{\mathbf{A}}$ as $\Delta\mathbf{U}^{\mathbf{A}}$. From Eq. (3), we only need to update the structure part of the representation by an increment projection:

$$\begin{aligned} \mathbf{U}^{\mathbf{A}}{}_{(t+1)} &= \mathbf{U}^{\mathbf{A}}{}_{(t)} + \Delta\mathbf{U}^{\mathbf{A}}{}_{(t)} \\ &= [\alpha_0 \mathbf{I} + \alpha_1(\mathbf{A}_{(t)} + \Delta\mathbf{A}_{(t)}) + \cdots + \alpha_p(\mathbf{A}_{(t)} + \Delta\mathbf{A}_{(t)})^p]\mathbf{R} \\ &= \alpha_0(\mathbf{Y}_0 + \Delta\mathbf{Y}_0) + \alpha_1(\mathbf{Y}_1 + \Delta\mathbf{Y}_1) + \cdots + \alpha_p(\mathbf{Y}_p + \Delta\mathbf{Y}_p) \\ \Rightarrow \Delta\mathbf{Y}_\mathbf{i} &= \mathbf{A}_{(t)} \cdot \Delta\mathbf{Y}_{i-1} + \Delta\mathbf{A}_{(t)} \cdot \mathbf{Y}_{i-1} + \Delta\mathbf{A}_{(t)} \cdot \Delta\mathbf{Y}_{i-1} \end{aligned} \tag{6}$$

Here we can also use the associative property of matrix multiplication to improve efficiency.

Nodes in the network may also be added or deleted. As for deleted nodes, it can be treated equivalently as the changes of edges by deleting all edges of the deleted nodes. For newly added nodes, we first add some empty nodes (i.e., no edge is connected to any other nodes) to make the dimension of the matrices match. Specifically, we denote Δn as the number of added nodes. For the projection matrix $\mathbf{R}$, we can generate an additional *i.i.d* random matrix $\hat{\mathbf{R}} \in \mathbb{R}^{\Delta n \times d}$ by Eq. (2) and concatenate it with the current projection matrix to gain the new projection matrix $\mathbf{R}_{(t+1)} \in \mathbb{R}^{(n+\Delta n)\times d}$. For other matrix $\mathbf{A}$ and $\mathbf{I}$ we add Δn all-zero rows and columns to complement the dimension. Then we can update added edges to those newly added nodes using Eq. (6).

4.2.2 **Attribute Changes.** The changes of node attributes also can be divided into the changes of attributes and the changes of nodes. Formally, we denote the changes in the attribute matrix as $\Delta\mathbf{X}$ and the changes in $\mathbf{U}^{\mathbf{X}}$ as $\Delta\mathbf{U}^{\mathbf{X}}$. From Eq. (3), we only need to update the attribute part of the representation:

$$\begin{aligned}
\mathbf{U}^{\mathbf{X}}{}_{(t+1)} &= \mathbf{U}^{\mathbf{X}}{}_{(t)} + \Delta\mathbf{U}^{\mathbf{X}}{}_{(t)} \\
&= [\theta_0\mathbf{I} + \theta_1(\mathbf{S}_{(t)} + \Delta\mathbf{S}_{(t)}) + \dots \\
&\quad + \theta_q(\mathbf{S}^q_{(t)} + (\Delta\mathbf{S}_{(t)})^q)]\mathbf{R} \qquad (7) \\
\Rightarrow \Delta\mathbf{U}^{\mathbf{X}}{}_{(t)} &= (\theta_1\Delta\mathbf{S}_{(t)} + \dots + \theta_q(\Delta\mathbf{S}_{(t)})^q)\mathbf{R} \\
\Delta\mathbf{S}_{(t)} &= (\mathbf{X}_{(t)} \cdot \Delta\mathbf{X}^T_{(t)} + \Delta\mathbf{X}_{(t)} \cdot \mathbf{X}^T_{(t)} + \Delta\mathbf{X}_{(t)} \cdot \Delta\mathbf{X}_{(t)})
\end{aligned}$$

Here we find the computational complexity of computing $\Delta\mathbf{S}_{(t)} \cdot \mathbf{R}$ is $O(3d_X n + 3n^2 d)$, where d_X is the number of non-zero entries in the sparse matrix $\Delta\mathbf{X}_{(t)}$, which is much higher than $O(2nkd)$ of computing $\mathbf{S}_{(t+1)} \cdot \mathbf{R}_{(t+1)}$ in Eq. (5). Besides, the formula in Eq. (5) can naturally handle the change in the number of attributes. So instead of using Eq. (7) to update the attribute representation, we use Eq. (5) to directly calculate the new attribute representation.

Moreover, for the changes about the number of nodes, we can first update the projection matrix $\mathbf{R}$ through the previously mentioned process in the structure changes. Then, we only need to replace the attribute matrix $\mathbf{X}_{(t)}$ in Eq. (5) with the new attribute matrix $\mathbf{X}_{(t+1)}$. In this way, we can obtain the attribute representation $\mathbf{U}^X_{(t+1)}$ directly. Algorithm 2 shows the pseudo-code of our online model.

4.3 Computational Complexity Analysis

We next theoretically analyze the computational complexity of our proposed algorithms.

Lemma 4.1. *The time complexity of the proposed offline embedding algorithm is $O(pmn + qnkd + pnd)$, where n and k are the number of nodes and attributes of attributed network, m is non-zero entries in the adjacency matrix, p and q are the orders of high-order structure and attribute proximity, and d is the embedding dimension size.*

Proof. As shown in Algorithm 1, the complexity of generating projection matrix (line 1) is $O(nd)$. The complexity of performing a sparse projection (line 3) is $O(knd)$. Then each iteration of computing i-th order structure proximity (lines 4-6) is $O(mn)$. And each iteration of recursive pre-projection (lines 7-9) is $O(nkd)$. After that, the complexity of calculating $\mathbf{U}^{\mathbf{A}}$ (line 10), $\mathbf{U}^{\mathbf{X}}$ (line 11), and $\mathbf{U}$ (line 12) are $O(pnd)$, $O(qnkd)$, and $O(nd)$, respectively. Therefore, the overall time complexity of Algorithm 1 is $O(pmn+qnkd+pnd)$. □

Algorithm 2 DRLAN: Online Learning Phase

Input: Adjacency matrix $\mathbf{A}_{(t)}$, dynamic changes of adjacency matrix $\Delta\mathbf{A}_{(t)}$, attributes matrix $\mathbf{X}_{(t+1)}$ at time step $t+1$, previous representation results $\mathbf{U}^{\mathbf{A}}{}_{(t)}$
Output: Updated representation results $\mathbf{U}_{(t+1)}$

1: **if** $\Delta\mathbf{A}_{(t)}$ includes Δn new nodes **then**
2: Generate an sparse random projection $\hat{\mathbf{R}} \in \mathbb{R}^{\Delta n \times d}$
3: Concatenate $\hat{\mathbf{R}}$ with $\mathbf{R}$ to obtain $\mathbf{R}_{(t+1)}$
4: Add Δn all-zero rows in $\mathbf{U}^{\mathbf{A}}{}_{(t)}$
5: **end if**
6: $\mathbf{Y}_0 \leftarrow \mathbf{R}_{(t+1)}$
7: $\mathbf{Z}_0 \leftarrow \mathbf{R}_{(t+1)}$, $\mathbf{Z}_1 \leftarrow \mathbf{X}^T_{(t+1)}\mathbf{R}_{(t+1)}$
8: **for** i in $1:p$ **do**
9: Calculate $\mathbf{Y}_i$ using Eq. (6)
10: **end for**
11: **for** i in $2:q$ **do**
12: Calculate $\mathbf{Z}_i = \mathbf{X}^T_{(t+1)} \cdot (\mathbf{X}_{(t+1)} \cdot \mathbf{Z}_{i-1})$
13: **end for**
14: Calculate $\Delta\mathbf{U}^{\mathbf{A}}{}_{(t+1)}$ using Eq. (6)
15: Calculate $\mathbf{U}^{\mathbf{X}}{}_{(t+1)}$ using Eq. (5)
16: Calculate $\mathbf{U}_{(t)} = \beta(\mathbf{U}^{\mathbf{A}}{}_{(t+1)} + \Delta\mathbf{U}^{\mathbf{A}}{}_{(t+1)}) + (1-\beta)(\mathbf{U}^{\mathbf{X}}{}_{(t+1)})$

Lemma 4.2. *The time complexity of the proposed online embedding algorithm at time step t is $O(qn_{(t)}kd+(\Delta n+pd_A)d)$, Where Δn is the number of added nodes of adjacency matrix $\mathbf{A}_{(t)}$, $n_{(t)} = n_{(t-1)} + \Delta n$, and d_A is the number of non-zero entries in the sparse matrices $\Delta\mathbf{A}_{(t)}$.*

Proof. By Algorithm 2, the complexity of updating the sparse random projection matrix (lines 1-5) is $O(\Delta nd)$, and the complexity of performing a sparse projection (line 7) is $O(kn_{(t)}d)$. The complexity of each iteration for i-th order increment structure proximity (lines 8-10) is $O(d_A d)$. Each iteration of recursive pre-projection for i-th order attribute proximity (lines 11-13) is $O(n_{(t)}kd)$. Finally, the complexity of line 14, line 15, and line 15 are $O(pd_A d)$, $O(qn_{(t)}kd)$, and $O(n_{(t)}d)$, respectively. Therefore, the overall time complexity of Algorithm 2 is $O(qn_{(t)}kd + (\Delta n + pd_A)d)$. □

As can be shown, $\Delta\mathbf{A}$ and $\Delta\mathbf{X}$ are often very sparse, that is, added nodes and attributes are usually quite small, thus d_A and Δn are very small, meanwhile we have $qn_{(t)}kd \ll pmn$ (here $n_{(t)}$=n). Based on the above analysis, the proposed online embedding is more efficient than re-running the offline method repeatedly for dynamic large-scale attributed networks.

Table 2: The Statistics of Datasets

Dataset	BlogCatalog	Flickr	DBLP
# Nodes	5,196	7,575	781,108
# Edges	171,743	239,738	4,191,677
# Attributes	8,189	12,047	160,648
# Labels	6	9	10
# Time steps	10	10	10

5 EXPERIMENTS

In this section, we conduct experiments to evaluate the effectiveness and efficiency of the proposed DRLAN method for dynamic attributed network representation. In particular, we attempt to answer the following three questions: (1) *Efficiency*: Can the proposed DRLAN be faster than other attributed network representation and dynamic representation learning methods in learning the representations for nodes? (2) *Effectiveness*: Can DRLAN perform better than other methods over dynamic graphs in traditional network mining tasks, e.g., node classification, link prediction, and network reconstruction? (3) *Sensitivity*: What are the impacts of hyper-parameter β, orders p, q, dimension d and the sparsity s of projection matrix on DRLAN?

5.1 Experimental Setting

5.1.1 **Datasets.** Three publicly available real-world datasets, i.e., BlogCatalog, Flickr, and DBLP[1], are employed in the experiments. BlogCatalog and Flickr are generated from static attributed networks, and have been used in previous research [8, 11]. In order to simulate the dynamic changes of the network, we randomly add 1% new nodes and 1% new attributes, while randomly delete 1% existing edges and 1% existing attributes at each time step. DBLP dataset is extracted from DBLP public bibliography data. Specifically, a citation network is built for the papers published before 2014 from 10 main research areas in computer science, including AI, computer networks, information security, database, etc. We treat each paper as a node, each citation as a direct edge from one paper to its reference. Bag-of-words model is applied on the paper title and venue to obtain the attribute information, and the major area the papers are published in is considered as label ground truth. We divide the DBLP network into 10 time steps from 2005-2014 based on the publishing time of papers. The detailed statistics of these datasets are summarized in Table 2.

5.1.2 **Baselines.** To evaluate the performance of our proposed DRLAN, we compare it with three categories of methods: plain network embedding methods, attributed network embedding methods and dynamic network embedding methods. Four plain network embedding methods are:

- DeepWalk [20]: DeepWalk preforms random walks on network and uses the skip-gram model to learn representation.
- node2vec [5]: node2vec extends DeepWalk by proposing a flexible neighbor discovery method in random walks to capture richer structure information.
- LINE [24]: LINE is one of state-of-the-art method for large-scale plain network embedding using edge-sampling method. It is able to preserve both first-order and second-order proximities between the nodes.
- RandNE [35]: RandNE is also a state-of-the-art plain network embedding method for billion-scale networks. It learns representation preserving high-order network structural proximities by iterative Gaussian random projection.

Three state-of-the-art attributed network embedding methods are:

- LANE [8]: LANE incorporates label information into attributed network representation. However, label is very important for node classification task, so we use the variations of LANE which only leverages the attributed network, without the help of label.
- CAN [17]: CAN learns representations of both attributes and nodes in the same semantic space such that the affinities between them can be captured.
- FeatWalk [9]: FeatWalk is a state-of-the-art attributed network embedding method. It uses random walks sampled from heterogeneous feature matrix to encode multiple node features into unified node vector representations.

Four dynamic network embedding methods are:

- CTDNE [18]: CTDNE uses temporal random walk to capture the continuous time dynamics of the network, learning a time dependent network representation for continuous-time dynamic networks.
- TIMERS [36]: TIMERS is a state-of-the-art network embedding method for dynamic networks. It develops a lower bound of SVD minimum loss for dynamic networks and uses the bound to monitor the margin between reconstruction loss of incremental updates and the minimum loss in SVD model.
- MMDNE [16]: MMDNE microscopically models the formation process of network structure with a temporal attention point process, and macroscopically constrains the network structure to obey a certain evolutionary pattern with a dynamics equation.
- DANE [11]: DANE learns representation from eigenvector decomposition of the Laplace matrix and employs first-order matrix perturbation theory to dynamically update the representation.

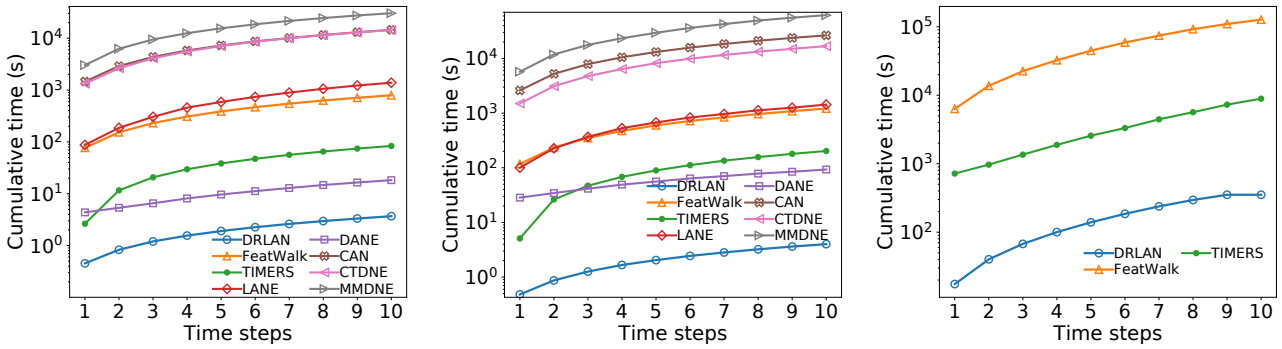
We implemented all the baselines by the code released by original authors. We evaluate the efficiency of our method on a machine with Intel Xeon E5-2660 (2.2GHz) CPU and 80GB memory. For all the methods, we uniformly set the dimension as $d = 128$ unless stated otherwise, and other parameters of all baselines are tuned to be optimal. In addition, all of the pre-defined weights in our model can be efficiently tuned at the end of the algorithm without re-running the model. It is important to note that DeepWalk, node2vec, LINE, LANE, CAN and FeatWalk can only handle static networks. For a fair comparison, we rerun these methods at each time step.

5.2 Results and Analysis

5.2.1 **Efficiency Study.** To compare the efficiency of different methods, we first compare DRLAN with the attributed network embedding baselines (i.e., LANE, DANE, CAN and FeatWalk) and state-of-the-art dynamic network embedding (i.e., TIMERS, CTDNE and MMDNE). Because DeepWalk, node2vec, LINE and RandNE can only handle plain networks and calculating attribute information is very time-consuming, we do not include these methods for fairness. Meanwhile, LANE and DANE are not able to handle the large-scale network DBLP due to the out of memory issue, and CAN, CTDNE and MMDNE cannot get results on DBLP in an acceptable time (i.e., 36 hours). Hence, we do not present their experimental results on Figure 1(c) and subsequent experiments on DBLP.

As we can observe from Figure 1, DRLAN is much faster than all baseline methods. To be more specific, for example, DRLAN is $6,676\times$ and $11,910\times$ faster than state-of-the-art dynamic embedding MMDNE on moderate-scale BlogCatalog and Flickr, respectively. DRLAN significantly outperforms deep neural network model CAN by $3,213\times$ and $5,426\times$ on BlogCatalog and Flickr networks. DRLAN

[1] https://www.aminer.cn/billboard/citation (V7 version is used)



(a) BlogCatalog (b) Flickr (c) DBLP

Figure 1: Cumulative running time comparison.

is faster than FeatWalk by 170×, 244× and 310× on BlogCatalog, Flickr and DBLP, respectively. This indicates that DRLAN is significantly more efficient than state-of-art methods as the network scale expands. DRLAN is 5× and 23× faster than DANE on BlogCatalog and Flickr, however, DANE is not able to handle large-scale attributed network DBLP due to its huge memory requirements and time consumption for eigen-decomposition.

This efficiency experiment confirms that our proposed DRLAN is faster than other attributed and dynamic network embedding methods on large-scale networks, and is able to efficiently handle the large-scale attributed networks.

5.2.2 **Node Classification.** We further evaluate the effectiveness of representations on the task of node classification. We split the embeddings of all nodes via 5-fold cross-validation, using 80% of nodes as training set and the rest of nodes for the testing. Then, an one-vs-all logistic regression classifier implemented by LibLinear[2] is trained using the representations of labeled nodes in the training set, and then tested on the labeled nodes of testing set. Similar to previous studies [3, 20], we use two measurements Macro-F1 and Micro-F1 to evaluate the performance. The whole process is repeated 10 times and the average performance on every time step is reported in Tables 3, 4 and 5.

The following observations can be made from the tables: (1) All attributed network embedding methods generally perform better than the plain network embedding methods. The main reason is that attributed network embedding methods can capture the attribute information of nodes, which effectively alleviates the sparsity and noise of network structure. Especially for node classification task, the help of attributes would be more significant. It can be observed that in the more sparse DBLP network, those methods considering attribute (i.e., FeatWalk and DRLAN) is more effective. Additionally, TIMERS performs worst on DBLP network. This is because DBLP network is very sparse (average degree is only 10.73), and SVD based embedding is not suitable for very sparse networks. (2) DRLAN achieves the best results on the large-scale network DBLP while slightly better performance than FeatWalk on BlogCatalog and Flickr. This may be because FeatWalk uses a random walk sampling based method to learn the features of attributed network. As the scale of the network increases, the sampling based method will be very time-consuming and difficult to accurately learn the essential features from the sampled walks. In contrast, DRLAN does not require a complicated sampling process, and it can learn the representation through direct projection of the network, which allows it to directly obtain the core information of the attributed network and naturally adapt to large-scale attributed networks.

5.2.3 **Link Prediction.** Link prediction is an important task of network analysis. It aims to predict if there is an edge between two nodes. Following [15], we randomly hide 10% of the edges for testing. After learning representation on the rest of the network, we use the inner product between representation to present the similarity of two nodes. We rank pairs of nodes according to the similarity measures and evaluate the performance on the testing set. To judge the ranking quality, we employ the *area under the ROC curve* (AUC), which is widely used in machine learning community to evaluate a ranking list. The process is repeated 10 times and the average results are reported.

The results on three attributed networks are shown in Tables 6, 8 and 10. From the results, we can see that DRLAN significantly outperforms the baselines in all cases on all networks in link prediction task. Notably, compared to the pure structure-based methods, i.e., DeepWalk, node2vec, LINE, RandNE, CTDNE, TIMERS and MMDNE, DRLAN achieves a significant improvement at each time step. Additionally, DRLAN even outperforms deep learning method CAN on two moderate-scale networks BlogCatalog and Flickr. In addition to our method considering high-order proximities, Johnson-Lindenstrauss Lemma guarantees the loss accuracy of random projection. Moreover, DRLAN still significantly performs better than FeatWalk with the dynamic updates of the networks. The reason is the same as explained above, our DRLAN can update the representations preserving both high-order structure proximities and attribute similarities by direct increment projection at each new time step, while FeatWalk re-learns the representation from the sampled walks on the updated heterogeneous feature matrix.

This experiment also proves that the representation learned by DRLAN can effectively infer the dynamic network structure

[2]https://www.csie.ntu.edu.tw/~cjlin/liblinear/

Table 3: Micro-F1 (Mi) and Macro-F1 (Ma) scores of node classification on BlogCatalog

Time	1		2		3		4		5		6		7		8		9		10	
Method	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma
DeepWalk	.735	.759	.737	.758	.718	.742	.714	.732	.725	.744	.713	.730	.715	.729	.704	.717	.695	.710	.695	.705
Node2vec	.745	.762	.743	.764	.737	.750	.743	.758	.751	.769	.748	.757	.751	.736	.739	.729	.721	.715	.728	.720
LINE	.744	.770	.749	.768	.715	.734	.728	.746	.711	.731	.692	.708	.703	.718	.698	.714	.711	.723	.693	.700
RandNE	.741	.733	.729	.742	.734	.742	.730	.741	.732	.745	.729	.738	.725	.735	.717	.724	.715	.720	.714	.719
LANE	.924	.861	.935	.905	.918	.877	.771	.755	.900	.887	.915	.919	.855	.860	.923	.919	.906	.908	.914	.916
CAN	.831	.847	.823	.840	.832	.843	.823	.835	.817	.830	.817	.824	.808	.818	.814	.822	.805	.809	.815	.820
FeatWalk	.921	**.925**	.936	.921	.919	.923	**.917**	.920	**.923**	**.926**	**.921**	**.923**	.927	**.930**	.925	**.928**	**.918**	**.921**	.922	**.925**
CTDNE	.749	.775	.762	.787	.755	.778	.744	.769	.754	.776	.744	.766	.747	.754	.734	.744	.714	.727	.724	.741
TIMERS	.730	.617	.713	.600	.700	.599	.692	.623	.702	.671	.704	.694	.698	.702	.704	.706	.711	.719	.722	.729
MMDNE	.756	.641	.776	.799	.773	.796	.778	.797	.769	.787	.777	.790	.776	.784	.778	.788	.757	.770	.764	.779
DANE	.901	.842	.925	.892	.902	.881	.785	.740	.907	.877	.901	.886	.869	.895	.921	.903	.913	.909	.892	.917
DRLAN	**.928**	.924	**.942**	**.928**	**.920**	**.925**	.916	**.922**	.918	.922	.914	.919	**.934**	.912	**.932**	.916	.905	.909	**.927**	.910

Table 4: Micro-F1 (Mi) and Macro-F1 (Ma) scores of node classification on Flickr

Time	1		2		3		4		5		6		7		8		9		10	
Method	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma
DeepWalk	.552	.574	.559	.580	.556	.572	.561	.576	.565	.574	.571	.578	.556	.561	.564	.562	.555	.551	.568	.565
node2vec	.524	.551	.537	.546	.539	.552	.540	.545	.533	.540	.538	.558	.535	.531	.537	.542	.537	.525	.540	.534
LINE	.539	.558	.539	.553	.554	.567	.528	.542	.550	.562	.545	.553	.520	.525	.531	.529	.548	.542	.531	.525
RandNE	.559	.549	.564	.536	.555	.542	.568	.535	.555	.531	.558	.545	.569	.554	.563	.555	.554	.540	.561	.534
LANE	.818	.778	.808	.786	.739	.713	.766	.742	.769	.751	.774	.765	.775	.769	.775	.773	.788	.785	.769	.767
CAN	.707	.720	.720	.734	.704	.716	.716	.723	.696	.700	.705	.709	.698	.698	.717	.714	.722	.719	.716	.709
FeatWalk	.877	.883	.874	.877	.885	**.899**	.884	.888	.875	.877	.878	.880	.879	.880	.876	.877	**.881**	**.880**	.877	.876
CTDNE	.682	.658	.665	.648	.701	.674	.716	.689	.722	.699	.729	.702	.734	.734	.726	.737	.747	.739	.739	.729
TIMERS	.625	.597	.624	.592	.609	.580	.613	.587	.597	.618	.615	.595	.605	.585	.605	.584	.606	.586	.600	.586
MMDNE	.698	.673	.720	.699	.722	.690	.732	.710	.736	.711	.730	.710	.730	.708	.742	.746	.741	.740	.745	.735
DANE	.741	.741	.760	.760	.752	.752	.773	.773	.775	.775	.760	.760	.766	.766	.762	.762	.771	.771	.761	.761
DRLAN	**.891**	**.897**	**.884**	**.889**	**.886**	.889	**.886**	**.889**	**.881**	**.884**	**.886**	**.888**	**.883**	**.885**	**.880**	**.879**	.878	.877	**.879**	**.877**

Table 5: Micro-F1 (Mi) and Macro-F1 (Ma) scores of node classification on DBLP

Time	1		2		3		4		5		6		7		8		9		10	
Method	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma	Mi	Ma
DeepWalk	.647	.542	.601	.494	.576	.467	.548	.433	.532	.407	.506	.377	.500	.366	.508	.368	.511	.363	.511	.369
node2vec	.659	.579	.621	.544	.592	.463	.545	.423	.533	.409	.507	.376	.500	.362	.508	.371	.512	.369	.510	.361
LINE	.530	.390	.519	.382	.503	.345	.500	.338	.486	.313	.473	.287	.478	.301	.491	.310	.503	.316	.502	.318
RandNE	.630	.690	.659	.682	.653	.645	.650	.638	.686	.633	.673	.687	.678	.631	.691	.631	.633	.657	.650	.648
FeatWalk	.901	.885	.898	.883	.892	.870	.884	.862	.882	.856	.882	.857	.870	.847	.869	.838	.855	.816	.855	.815
TIMERS	.444	.422	.450	.428	.437	.417	.421	.403	.415	.383	.405	.382	.402	.372	.395	.375	.391	.370	.384	.372
DRLAN	**.945**	**.930**	**.943**	**.926**	**.953**	**.939**	**.955**	**.939**	**.954**	**.935**	**.942**	**.942**	**.963**	**.951**	**.951**	**.936**	**.925**	**.905**	**.917**	**.894**

Table 6: AUC scores of link prediction on BlogCatalog

Time	1	2	3	4	5	6	7	8	9	10
DeepWalk	.571	.565	.588	.604	.615	.619	.619	.616	.621	.618
node2vec	.563	.579	.586	.606	.615	.623	.622	.611	.620	.621
LINE	.588	.592	.583	.586	.583	.584	.586	.579	.577	.579
RandNE	.833	.840	.838	.834	.835	.832	.835	.831	.835	.838
LANE	.726	.702	.707	.727	.685	.660	.743	.714	.683	.713
CAN	.835	.839	.785	.857	.857	.798	.857	.802	.812	.857
FeatWalk	.635	.632	.631	.636	.636	.636	.642	.634	.640	.647
CTDNE	.776	.794	.790	.796	.807	.792	.800	.790	.800	.793
TIMERS	.707	.689	.704	.690	.710	.710	.720	.718	.708	.703
MMDNE	.840	.836	.833	.840	.841	.834	.830	.844	.840	.840
DANE	.736	.711	.694	.714	.708	.698	.745	.734	.715	.725
DRLAN	**.861**	**.856**	**.858**	**.859**	**.861**	**.863**	**.865**	**.867**	**.868**	**.875**

Table 7: AUC scores of network reconstru. on BlogCatalog

Time	1	2	3	4	5	6	7	8	9	10
DeepWalk	.498	.511	.506	.512	.517	.523	.528	.542	.554	.566
node2vec	.496	.501	.507	.511	.516	.521	.530	.536	.551	.564
LINE	.497	.496	.499	.499	.498	.502	.502	.502	.505	.501
RandNE	.863	.868	.867	.865	.876	.877	.883	.887	.890	.893
LANE	.685	.683	.750	.771	.657	.742	.764	.750	.725	.763
CAN	.834	.835	.790	.855	.838	.794	.856	.800	.808	.857
FeatWalk	.633	.634	.635	.636	.639	.640	.641	.640	.643	.645
CTDNE	.792	.792	.791	.794	.795	.795	.794	.792	.793	.796
TIMERS	.698	.686	.698	.702	.707	.705	.709	.709	.711	.713
MMDNE	.862	.869	.863	.875	.879	.876	.878	.879	.861	.872
DANE	.673	.669	.771	.691	.731	.692	.693	.701	.705	.713
DRLAN	**.905**	**.908**	**.909**	**.912**	**.912**	**.913**	**.914**	**.915**	**.917**	**.916**

Table 8: AUC scores of link prediction on Flickr

Time	1	2	3	4	5	6	7	8	9	10
DeepWalk	.682	.701	.699	.710	.747	.764	.775	.792	.795	.791
node2vec	.683	.685	.730	.749	.772	.793	.802	.805	.804	.807
LINE	.797	.798	.793	.783	.781	.784	.766	.765	.767	.769
RandNE	.840	.840	.842	.856	.855	.863	.853	.854	.855	.852
LANE	.692	.766	.756	.716	.740	.788	.728	.770	.722	.733
CAN	.858	.850	.858	.866	.858	.869	.870	.870	.869	.877
FeatWalk	.678	.679	.673	.681	.685	.705	.698	.703	.695	.696
CTDNE	.761	.778	.773	.779	.786	.784	.794	.809	.806	.805
TIMERS	.631	.612	.614	.600	.614	.621	.610	.619	.605	.616
MMDNE	.805	.812	.819	.815	.819	.820	.814	.817	.813	.822
DANE	.701	.775	.748	.705	.754	.797	.735	.764	.715	.720
DRLAN	**.862**	**.864**	**.867**	**.870**	**.874**	**.876**	**.880**	**.883**	**.886**	**.889**

Table 9: AUC scores of network reconstruction on Flickr

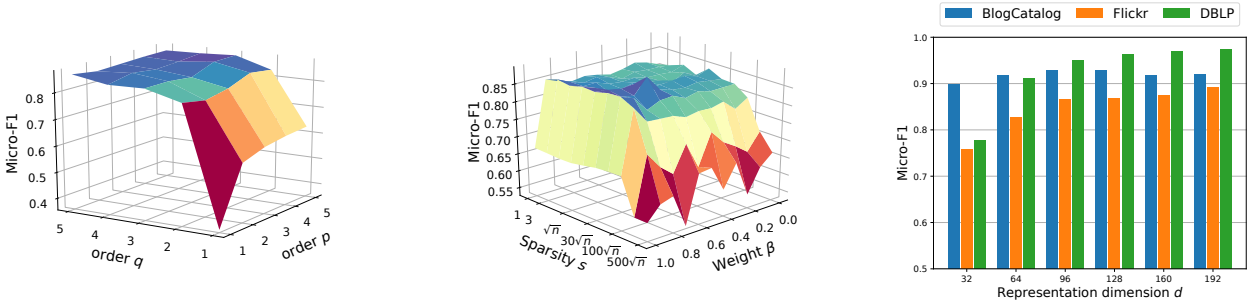
Time	1	2	3	4	5	6	7	8	9	10
DeepWalk	.440	.444	.450	.453	.449	.449	.427	.412	.416	.415
node2vec	.429	.435	.435	.431	.422	.408	.380	.370	.366	.373
LINE	.338	.343	.345	.350	.357	.361	.364	.367	.374	.376
RandNE	.841	.846	.847	.849	.846	.829	.837	.831	.841	.845
LANE	.666	.654	.708	.696	.680	.671	.688	.701	.682	.702
CAN	.756	.759	.721	.782	.769	.731	.790	.745	.752	.796
FeatWalk	.652	.655	.658	.661	.663	.668	.669	.672	.676	.678
CTDNE	.802	.802	.800	.796	.795	.799	.801	.809	.811	.805
TIMERS	.609	.605	.608	.601	.606	.610	.607	.608	.614	.609
MMDNE	.829	.833	.835	.839	.839	.836	.838	.839	.841	.842
DANE	.672	.661	.692	.701	.701	.688	.670	.681	.692	.711
DRLAN	**.858**	**.861**	**.864**	**.868**	**.871**	**.873**	**.877**	**.880**	**.883**	**.886**

Table 10: AUC scores of link prediction on DBLP

Time	1	2	3	4	5	6	7	8	9	10
DeepWalk	.701	.693	.678	.689	.686	.686	.702	.705	.713	.711
node2vec	.696	.689	.691	.681	.686	.694	.689	.690	.700	.706
LINE	.573	.558	.548	.557	.562	.548	.534	.526	.526	.524
RandNE	.793	.780	.810	.788	.798	.779	.758	.772	.778	.769
FeatWalk	.774	.773	.765	.765	.760	.757	.755	.751	.755	.745
TIMERS	.631	.667	.656	.625	.606	.590	.576	.573	.562	.561
DRLAN	**.892**	**.893**	**.881**	**.839**	**.880**	**.861**	**.828**	**.811**	**.838**	**.855**

Table 11: AUC scores of network reconstruction on DBLP

Time	1	2	3	4	5	6	7	8	9	10
DeepWalk	.729	.715	.697	.701	.704	.691	.705	.699	.687	.673
node2vec	.716	.704	.712	.702	.707	.705	.715	.707	.692	.681
LINE	.613	.565	.640	.678	.620	.606	.582	.563	.539	.551
RandNE	.796	.799	.771	.794	.751	.733	.754	.754	.768	.732
FeatWalk	.792	.790	.803	.813	.748	.728	.712	.716	.735	.730
TIMERS	.643	.659	.651	.605	.643	.589	.593	.563	.555	.587
DRLAN	**.893**	**.890**	**.891**	**.899**	**.870**	**.863**	**.844**	**.839**	**.826**	**.827**



(a) p and q on Flickr (b) β and s on Flickr (c) dimension d

Figure 2: The impacts of parameters p, q, β, dimension d, and the sparsity s of projection matrix on DRLAN

based on structure information and rich attribute information of the existing network.

5.2.4 **Network Reconstruction.** One basic objective for network representation is to reconstruct the network. We rank pairs of nodes in a similar way as link prediction. The top ranking pairs are used to reconstruct the network because large similarities indicate high probabilities of having edges. On DBLP network, the number of possible pairs of nodes $\frac{N(N-1)}{2}$ is too large to evaluate, following the setting in [15] we sample 1% of these pairs for evaluation.

We show the results on three attributed networks in Tables 7, 9 and 11. As we can see, DRLAN achieves the best performance among all methods on network reconstruction task. Although TIMERS considers restart of SVD to reduce error accumulation for dynamic networks in time, it still shows noticeable performance degradation at some time steps on large-scale DBLP network. Additionally, we find FeatWalk and DRLAN both show a performance degradation over time on DBLP network. The potential reason may be: on DBLP network, the attribute information of papers over a long time span may interfere the network reconstruction over time, and thus the structure of network is more important in this task. For example, a new published paper, which may be similar to a very old paper in title, may not cite this old paper. However, DRLAN still significantly outperforms FeatWalk at each time step. This is because DRLAN retains the powerful structure representation by explicitly considering the high-order structural proximities, which is more effective on sparse DBLP network.

In summary, the results of the three dynamic network mining tasks answer the question that our model can learn better node embedding with the dynamic updates of networks as compared to competitive network embedding methods.

5.2.5 **Parameter Sensitivity.** We now study the sensitivity of our proposed method with respect to the important parameters in terms of node classification task. Specifically, we analyze the effect of the structure proximity order p, attribute proximity order

q, sparsity s of random projection matrix, hyper-parameter β and embedding dimension d. The results are shown in Figure 2, results on other datasets are very similar, so we omit them. First, we vary p and q from 1 to 5 respectively. As p and q increase, high-order structure and high-order attribute information are added to $\mathbf{U}$. Figure 2(a) shows the results of DRLAN on Flickr with different p and q. We find that both high-order structure information and high-order information can significantly improve the performance of node representations. Second, we vary the hyperparameter β from 0 to 1 and sparsity s of random projection matrix from 1 to $500\sqrt{n}$. When $\beta = 0$, only attribute information is used to learn $\mathbf{U}$. When $\beta = 1$, only structure information is used to learn $\mathbf{U}$. The sparsity s denotes $1/s$ of the elements in the matrix $\mathbf{S}$ being preserved in the process of random projection. From the results in Figure 2(b), we observe that DRLAN achieves the best performance when $\beta = 0.6$ and $s = 30\sqrt{n}$, where the contributions of attribute information and structure information are balanced while preserving the appropriate information in the projection process. Finally, we vary d from 32 to 192. From Figure 2(c), we see that, as d increases from 32 to 192, the performance of DRLAN gradually increases. In particular, the experimental results on the BlogCatalog network achieve the best at $d = 128$, followed by slight fluctuations. From the results, we also find that DRLAN performs well under most parameter settings, which means that DRLAN has good robustness.

6 CONCLUSIONS

In this paper, we propose a novel dynamic representation learning method - DRLAN for large-scale attributed networks based on sparse random projection, which can capture the high-level non-linearity and preserve the high-order proximity both in the topological structures and node attributes. In particular, DRLAN can efficiently obtain the low-dimensional representations of nodes in attributed networks by sparse random projection and be able to update on the fly as the changes in network. Moreover, DRLAN does not require complex sampling processes in [9, 24] or intricacy matrix decomposition in [8, 11] to obtain representation. Extensive experiments on three real-world datasets validate the efficiency and effectiveness of DRLAN across various settings.

ACKNOWLEDGMENTS

This work is partially supported by the Fundamental Research Funds for the Central Universities under grant No. 201964022, the National Natural Science Foundation of China under grant Nos. 61773331, 61403328 and 61703360, and the Graduate Innovation Foundation of Yantai University under grant No. YDZD2021.

REFERENCES

[1] Diogo M Camacho, Katherine M Collins, Rani K Powers, James C Costello, and James J Collins. 2018. Next-generation machine learning for biological networks. *Cell* 173, 7 (2018), 1581–1592.
[2] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. 2018. PME: projected metric embedding on heterogeneous networks for link prediction. In *KDD*. ACM, 1177–1186.
[3] Hongchang Gao and Heng Huang. 2018. Deep Attributed Network Embedding.. In *IJCAI*. 3364–3370.
[4] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *KDD*. 1416–1424.
[5] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 855–864.
[6] Chao Huang, Baoxu Shi, Xuchao Zhang, Xian Wu, et al. 2019. Similarity-aware network embedding with self-paced learning. In *CIKM*. 2113–2116.
[7] Chao Huang, Xian Wu, Xuchao Zhang, Chuxu Zhang, Jiashu Zhao, Dawei Yin, and Nitesh V Chawla. 2019. Online purchase prediction via multi-scale modeling of behavior dynamics. In *KDD*. 2613–2622.
[8] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *WSDM*. 731–739.
[9] Xiao Huang, Qingquan Song, Fan Yang, and Xia Hu. 2019. Large-Scale Heterogeneous Feature Embedding. In *AAAI*. 3878–3885.
[10] William B Johnson and Joram Lindenstrauss. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics* 26, 189-206 (1984), 1.
[11] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, et al. 2017. Attributed network embedding for learning in a dynamic environment. In *CIKM*. 387–396.
[12] Ping Li, Trevor J Hastie, and Kenneth W Church. 2006. Very sparse random projections. In *KDD*. ACM, 287–296.
[13] Ruirui Li, Xian Wu, Xian Wu, and Wei Wang. 2020. Few-Shot Learning for New User Recommendation in Location-based Social Networks. In *WWW*. 2472–2478.
[14] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2018. Attributed social network embedding. *TKDE* 30, 12 (2018), 2257–2270.
[15] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications* 390, 6 (2011), 1150–1170.
[16] Yuanfu Lu, Xiao Wang, Chuan Shi, Philip S Yu, and Yanfang Ye. 2019. Temporal Network Embedding with Micro-and Macro-dynamics. In *CIKM*. 469–478.
[17] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-embedding Attributed Networks. In *WSDM*. ACM, 393–401.
[18] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference*. 969–976.
[19] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *KDD*. ACM, 1105–1114.
[20] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.
[21] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. 2019. NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization. In *WWW*. 1509–1520.
[22] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. In *WSDM*. 519–527.
[23] Yiwei Sun, Suhang Wang, Tsung-Yu Hsieh, Xianfeng Tang, and Vasant Honavar. 2019. Megan: A generative adversarial network for multi-view network embedding. In *IJCAI*. 3527–3533.
[24] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. 1067–1077.
[25] Xianfeng Tang, Boqing Gong, Yanwei Yu, Huaxiu Yao, Yandong Li, Haiyong Xie, and Xiaoyu Wang. 2019. Joint modeling of dense and incomplete trajectories for citywide traffic volume inference. In *WWW*. 1806–1817.
[26] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *ICLR*.
[27] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *ACL*. 1722–1731.
[28] Wei Wu, Bin Li, Ling Chen, and Chengqi Zhang. 2018. Efficient Attributed Network Embedding via Recursive Randomized Hashing. In *IJCAI*. 2861–2867.
[29] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. 2015. Network representation learning with rich text information. In *IJCAI*. 2111–2117.
[30] Cheng Yang, Maosong Sun, et al. 2017. Fast Network Embedding Enhancement via High Order Proximity Approximation.. In *IJCAI*. 3894–3900.
[31] Yanwei Yu, Xianfeng Tang, Huaxiu Yao, Xiuwen Yi, and Zhenhui Li. 2019. Citywide Traffic Volume Inference with Surveillance Camera Records. *IEEE Transactions on Big Data* (2019).
[32] Yanwei Yu, Hongjian Wang, and Zhenhui Li. 2018. Inferring mobility relationship via graph embedding. *IMWUT* 2, 3 (2018), 1–21.
[33] Yanwei Yu, Huaxiu Yao, Hongjian Wang, Xianfeng Tang, and Zhenhui Li. 2018. Representation learning for large-scale dynamic networks. In *DASFAA*. Springer, 526–541.
[34] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *KDD*. 793–803.
[35] Ziwei Zhang, Peng Cui, Haoyang Li, Xiao Wang, and Wenwu Zhu. 2018. Billion-scale Network Embedding with Iterative Random Projection. In *ICDM*. IEEE, 787–796.
[36] Ziwei Zhang, Peng Cui, Jian Pei, Xiao Wang, and Wenwu Zhu. 2018. Timers: Error-bounded svd restart on dynamic networks. In *AAAI*.
[37] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *AAAI*.
[38] Dingyuan Zhu, Peng Cui, Ziwei Zhang, Jian Pei, and Wenwu Zhu. 2018. High-order proximity preserved embedding for dynamic networks. *TKDE* 30, 11 (2018), 2134–2144.