Motif-Preserving Dynamic Attributed Network Embedding

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

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

ABSTRACT

Network embedding has emerged as a new learning paradigm to embed complex network into a low-dimensional vector space while preserving node proximities in both network structures and properties. It advances various network mining tasks, ranging from link prediction to node classification. However, most existing works primarily focus on static networks while many networks in real-life evolve over time with addition/deletion of links and nodes, naturally with associated attribute evolution. In this work, we present **M**otif-preserving **T**emporal **S**hift **N**etwork (MTSN), a novel dynamic network embedding framework that simultaneously models the local high-order structures and temporal evolution for dynamic attributed networks. Specifically, MTSN learns node representations by stacking the proposed TIME module to capture both local high-order structural proximities and node attributes by *motif-preserving encoder* and temporal dynamics by *temporal shift operation* in a dynamic attributed network. Finally, we perform extensive experiments on four real-world network datasets to demonstrate the superiority of MTSN against state-of-the-art network embedding baselines in terms of both effectiveness and efficiency. The source code of our method is available at: https://github.com/ZhijunLiu95/MTSN.

CCS CONCEPTS

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

KEYWORDS

Network embedding, graph neural networks, dynamic networks

ACM Reference Format:
Zhijun Liu, Chao Huang, Yanwei Yu, and Junyu Dong. 2021. Motif-Preserving Dynamic Attributed Network Embedding. In *Proceedings of the Web Conference 2021 (WWW'21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3442381.3449821

*Yanwei Yu is the corresponding author.

WWW'21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449821

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 [4, 26, 52], node classification [12, 15], network reconstruction [38], and recommendation [13, 50]. The core of network embedding is to preserve the proximities of nodes based on the topological structures of a network. 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 scenarios, attributed network embedding [8, 15] has become a promising solution to improve the performance of network representation learning. It aims to learn latent feature representations of nodes, such that the network topological structure and node attributes can be jointly preserved in the learned embeddings.

Despite the effectiveness of existing attributed network embedding methods [8, 16, 23, 43], these works are generally designed for static graphs. In real-world scenarios, however, many networks are inherently dynamic where network structures and node attributes constantly evolve over time. Thus, the assumption of node proximity and attribute consistencies does not hold any more. For example, the friendship between users in social network platforms can evolve over time [35], and users' interactions over different items could change dramatically, since user's preferences are dynamic by nature [31]. We term this kind of networks with the evolution of both network structure and node attribute information as *dynamic attributed networks*, which not only refer to the creation and deletion of edges and nodes in the network, but also the change of weights on the edges, as well as the evolution of node attributes.

Performing representation learning on the dynamic attributed networks is of great importance to network mining tasks, yet it is very challenging due to the complex evolution patterns of network structure and node attributes. Therefore, it is crucial to account for the above two-dimensional dynamics of graph structural context and attribute information, which needs dedicated efforts. To address this challenge, there exist several attempts proposed to study the dynamic attributed network embedding [22, 40, 41, 48]. For example, DANE [22] learns the representation of network through the eigenvalue decomposition of each time snapshot, and uses the matrix disturbance theory to update the representations in an incremental manner. Recently, TGAT [48] and DySAT [41] extend the graph attention network (GAT) [46] to dynamic scenarios, with the utilization of temporal self-attention mechanism to capture

temporal evolution in network structures. EvolveGCN [40] uses recurrent neural networks (RNN) to model the temporal changes of the network, and expand the graph convolutional network (GCN) into a dynamic graph learning framework. However, it requires large amounts of training data to outperform even static methods and scale poorly with an increase of time steps.

Additionally, the aforementioned approaches ignore the local high-order structural patterns that evolve in dynamic networks – *network motif*. Inspired by the effectiveness of network motifs, recent efforts propose to incorporate the network motifs into the static graph representation settings to improve the quality of learned node embeddings [18, 28]. Nevertheless, how to embed network motifs into the graph representation learning in a dynamic environment, so as to capture the evolution of network characteristics and dynamic structural relations, still remains a significant challenge.

To tackle the aforementioned challenges, we develop a **M**otif-preserving **T**emporal **S**hift **N**etwork (MTSN), which is a dynamic graph neural network (GNN) framework that integrates network motif features into the dynamic attributed network representation. First, we propose a *Motif-Preserving Encoder* (MPE) to model network topology and network motif information with the incorporation of node attributes into node representations. Furthermore, we propose a *temporal shift mechanism* along with the temporal dimension to effectively capture the evolution patterns of dynamic networks. More specifically, we integrate the *temporal shift mechanism* with *motif-preserving encoder* into a convolution-based network, which serve as the base architecture of MTSN . Experimental results on four real-world dynamic networks show that our model significantly outperforms several strong baselines (7.3% AUC gain and 7.1% F1 gain on average) in dynamic link prediction task. We also demonstrate the benefits of the collaboration of temporal shift meahcnism and motif-preserving encoder through an ablation study. We further visualize the motif weights to illustrate the capability of our developed MTSN framework in capturing and differentiating local high-order structures on different datasets.

We highlight the key contributions of this work as follows:

- We propose an efficient and effective dynamic graph neural network, which designs temporal shift mechanism to endow GNN with the capability of capturing network temporal evolution without heavy computational burden.
- We integrate motif features of network into dynamic network representation to capture the local high-order structural proximities within the neighborhood at different levels and scales.
- We conduct extensive experiments on four real-world datasets to verify the effectiveness and efficiency of our proposed model MTSN in dynamic link prediction task.

2 RELATED WORK

2.1 Dynamic Network Embedding

Existing dynamic network representation learning methods fall into two broad categories: *discrete-time* methods [25, 32, 33, 36, 42, 47, 51] and *continuous-time* methods [29, 37, 48, 57]. In particular, *Discrete-time* methods divide a dynamic network into multiple time snapshots to learn node dynamic representations. For example, DANE [22], TIMERS [53] and DHPE [56] aim to learn network representation through the eigenvalue decomposition of snapshot-specific adjacency matrix based on the matrix disturbance theory and incremental matrix decomposition method. In [5, 7, 54, 55], the random walk strategy is utilized to capture network dynamics. In addition, DRLAN [27] recursively updates the learned representation vectors based on the new network information from the current time slot by incremental matrix projection for dynamic attributed networks. However, it does not consider the temporal evolution behaviours on dynamic networks. DynGEM [19] utilizes deep auto-encoders to incrementally generate the representation of a dynamic graph by only using the data from the last snapshot. DynAERNN [9] uses a DNN which is composed of dense network and recursive layers on multiple historical snapshots to capture the network temporal evolution. To model the temporal dependencies of dynamic network, [6, 34, 44] improve the graph convolutional network to inject the graph learning process with the time-aware relationships in dynamic networks. EvolveGCN [40] applies the GCN for each time snapshot to obtain the low-dimensional representations of nodes, and uses RNN to update the parameters of message passing during the graph convolution operations at each time snapshot, without the direct resort of node embeddings. DySAT [41] uses GAT [46] to capture structural neighborhood information and extends the self-attention mechanism to model temporal evolution across multiple time snapshots. In [2, 17, 20, 30], they study the representation learning on dynamic heterogeneous networks.

Additionally, *Continuous-time* methods do not divide the entire network into snapshot-specific sub-networks. Instead, they learn from edges with fine-grained time granularity. For instance, CTDNE [37] designs a temporal random walk method to capture the dynamic evolution characteristics of nodes. HTNE [57] and MMDNE [29] use time sequence point process to capture the influence of historical time slots on the current one, then use attention mechanism to differentiate the influence of different neighbors. DyRep [45] uses two time scales of deep sequential point process models to capture the interactive changes in dynamic networks. TGAT [48] adapts the attention-based graph neural network in dynamic scenarios, and designs a temporal self-attention mechanism to aggregate the temporal neighboring context from nodes with different time granularities to obtain node dynamic representations. Different from those methods, our proposed MTSN framework .

2.2 Motif-based Network Embedding

Many static network embedding methods focus on modeling the high-order local structure pattern (network motif) into node representation. For example, [21] uses the multiple motif adjacency matrices to expand the receptive field of node's neighbors, and then uses attention mechanism to identify the influential neighboring nodes for feature aggregations. RUM [49] proposes MotifWalk and MotifRe-weighting strategies to learn both motif-based and node-based representations, so as to capture the high-level structural relations and individual properties of each node. GraLSP [18] captures local network structure through anonymous walks. Although it considers the complex motif types, this method has heavy computational cost and is limited by its poor scalability. GraphSTONE [28] builds a graph topic model through anonymous walks and graph anchor LDA, to improve the model efficiency of GraLSP. It uses the

structural topics to guide aggregation process of node features, with the goal of generating more powerful representations. However, none of these methods consider how to integrate network motifs in dynamic networks. Among those approaches, MTNE [14] considers to use Hawkes process to model the network motif of three nodes for the evolution process in a dynamic network. However, when MTNE uses historical network motif information, it regards the entire historical network information as the search scope for network motif construction, which results in two edges that occurred at a relatively long time being considered to constitute an open triad. This not only leads to the large number of open triads calculations, but also mixes some open triads that do not conform to the real scene, which eventually affects the quality of the representations.

3 PROBLEM DEFINITION

In this section, we formally define the problem of dynamic attributed network embedding. A dynamic attributed network is a series of network snapshots from different time steps, *i.e.*, $\mathbb{G} = \{\mathcal{G}^1, \mathcal{G}^2, \ldots, \mathcal{G}^T\}$, where T denotes the number of time steps. Each network snapshot $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t, \mathbf{X}^t)$ is a weighted undirected graph which is consisted of a node set $\mathcal{V}^t$, a edge set $\mathcal{E}^t$, a weighted adjacency matrix $\mathbf{A}^t$, and a node attribute matrix $\mathbf{X}^t$.

Dynamic Attributed Network Embedding. Dynamic attributed network embedding aims to learn latent representation $\mathbf{Z}^t \in \mathbb{R}^{|\mathcal{V}^t| \times d}$ for all nodes in $\mathcal{G}^t$ at individual time step $t = \{1, 2, \ldots, T\}$, where d represents the embedding dimensionality ($d << |\mathcal{V}^t|$). Here, $\mathbf{Z}^t$ collectively preserves the local high-order structural proximities, node attribute features, and temporal evolutionary behaviours in a dynamic attributed network up to time step t.

4 METHODOLOGY

In this section, we present the details of our neural network model MTSN (as depicted in Figure 1). There are two key components in the framework: *First*, in order to capture the local high-order topology proximities of network, we design a novel graph neural network – *Motif-Preserving Encoder* (MPE). MPE performs the feature learning with the consideration of both network adjacency matrix and network motifs, so as to obtain the representation of the high-order similarities of the network on each snapshot. *Second*, in order to efficiently capture the temporal evolutionary patterns in dynamic networks, we design an efficient temporal convolution model – *Temporal shIft based on Motif-preserving Encoder* (TIME). TIME simulates the one-dimensional convolution operation over the representations across different historical time snapshots along with the time dimension through the temporal shift mechanism.

Our key innovation lies in simultaneously capturing the local high-order structural proximities by motif-preserving encoder and the temporal evolution patterns with the temporal shift mechanism in dynamic node representations by stacking TIME layers.

4.1 Motif-Preserving Encoder

The adjacency matrix of the network can only represent the direct proximity of nodes in the network. Most random walks and graph neural networks-based methods only reply on the adjacency matrix to encode graph connections, and thus cannot well capture the local high-order topology proximities. However, different types of network motifs can reflect the proximity between nodes in the network under different high-order connection modes. In addition to the network structures between node pairs in the adjacency matrix, we further introduce the local high-order structures within the neighborhood nodes contained in network motifs. Then, the local high-order topology proximities from these two aspects are encoded together to learn node representations.

The input of this module is the graph snapshot $\mathcal{G}^t \in \mathbb{G}$ at the t-th time step, and the motif matrices $\{\mathcal{M}_i^t | i = 1, 2, \ldots, |M|\}$ generated from $\mathcal{G}^t$, where M is the set of motifs and $|M|$ is the number of motif types. Specifically, in motif matrix, $(\mathcal{M}_i^t)_{u,v} = j$ indicates that the edge between nodes u and v participates in the i-th motif type j times, that is, it participates in establishing j different i-type motifs. The output is the hidden representation $\mathbf{H}^t \in \mathbb{R}^{n \times d}$ where n is the number of nodes in $\mathcal{G}^t$ and d is the representation dimensionality, such that $\mathbf{H}^t$ preserves both the local high-order node proximities and attribute features in the individual graph snapshot $\mathcal{G}^t$.

Motif-Preserving Encoder (MPE) first uses the Parameterized Graphlet Decomposition (PGD) technique [1] to extract the motif matrices for each graph snapshot. PGD only takes a few seconds to count network motifs over large-scale network with hundreds of millions of edges. The time complexity to obtain motif matrix of all network motifs with 2,3,4-nodes is $O(|M| \cdot \Delta \cdot T_{max}) + O(|M| \cdot \Delta \cdot S_{max})$ where $|M|$ is the number of motif types, Δ is the maximum degree in the network, T_{max} is the maximum number of triangles incident to an edge and $T_{max} \ll \Delta$ for sparse graphs, and S_{max} is the maximum number of stars incident to an edge and $S_{max} \leq \Delta$. More specifically, we select eight undirected motif types in this paper as shown in the bottom-center part of Figure 1, which cover all undirected network motifs within 4 nodes. By doing so, all local structures among any 4 (or less than 4) nodes can be captured in MPE. Then, we perform a weighted sum of the eight motif matrices to obtain a combined motif matrix $\mathbb{M}^t$ that contains multi-level network structure proximities of nodes as follows:

$$\mathbb{M}^t = \sum_{i=1}^{|M|} \alpha_i \mathcal{M}_i^t \tag{1}$$

where α_i is the learnable weight for the i-th motif type.

In MPE, we uses the simplified GCN to aggregate neighbor node features to generate the node representation. In particular, we first learn two representations for nodes based on the adjacency matrix $\mathbf{A}$ and the combined motif matrix $\mathbb{M}$, respectively. Finally, these two representations are integrated to obtain the final node embedding. We first conduct the embedding transformation as $\mathbf{H}_{(0)}^t = \mathbf{X}^t \cdot \mathbf{W}$. The message passing process in MPE, $\mathbf{H}_{(l)}^t = MPE(\mathbf{A}^t, \mathbb{M}^t, \mathbf{H}_{(l-1)}^t)$, can be formally represented as below:

$$\begin{aligned} \mathbf{H}_A^t &= \mathbf{A}^t \cdot \mathbf{H}_{(l-1)}^t, \\ \mathbf{H}_M^t &= \mathbb{M}^t \cdot \mathbf{H}_{(l-1)}^t, \\ \mathbf{H}_{(l)}^t &= \mathbf{H}_A^t + \beta \mathbf{H}_M^t, \end{aligned} \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{f \times d}$ is a trainable linear transformation weight matrix, and weight β indicates the learnable importance of motif-based

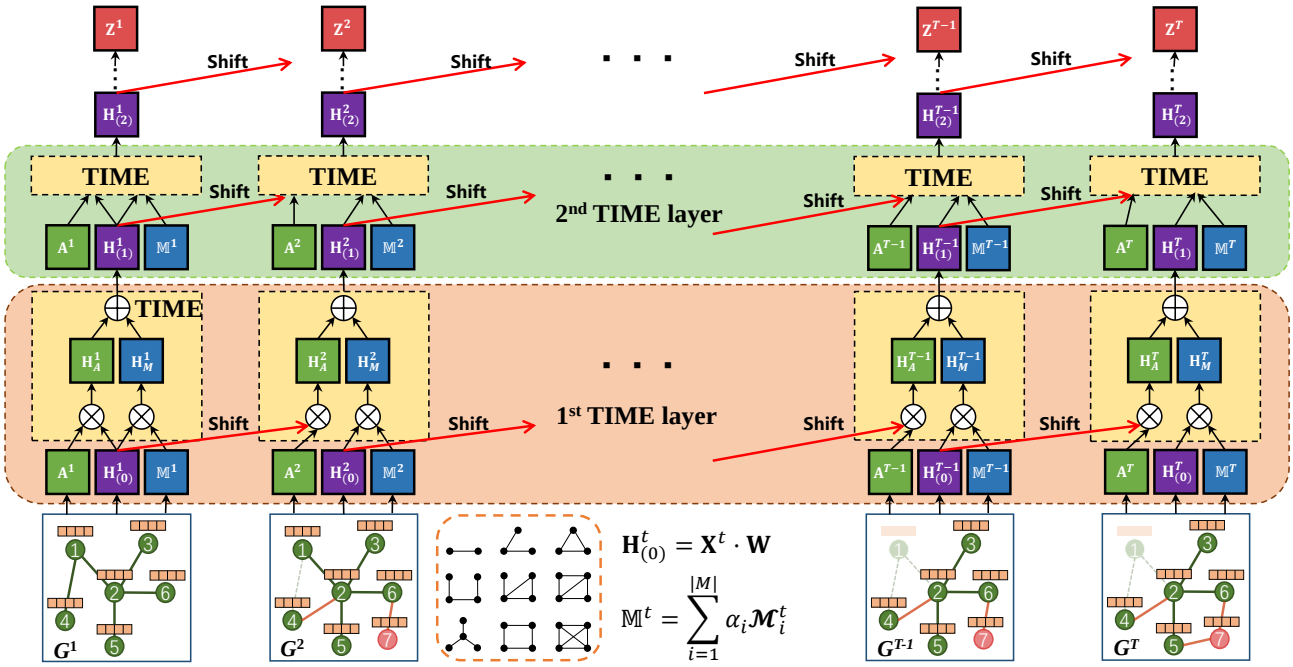


Figure 1: The overview of the proposed Motif-preserving Temporal Shift Network (MTSN) framework.

network structure in generating node representations. By stacking multiple MPE layers, our framework could capture the high-order structural relationships among nodes in each graph snapshot through the embedding propagation across graph layers.

4.2 TIME: Temporal Shift based on MPE Layer

Although MPE can obtain motif-preserving node representations for individual network snapshot, the temporal evolutionary patterns in a dynamic network has not captured in the node representations. Inspired by the Temporal Shift Mechanism (TSM) introduced in [24], we develop a temporal shift module to model the impact of temporal factors for node representations over the dynamic network. Specifically, we use a 1-D convolution operation to capture the temporal evolution of dynamic networks. For brevity, we use a 1-D convolution with the kernel size of 3 as an example. Suppose the weight of the convolution is $W_s = (w_1, w_2, w_3)$, and the input is the hidden representation H_v^t of node v, which is a 1-D vector. The convolution operation $Y_v^t = \text{Conv}(W_s, H_v^t)$ can be represented as: $Y_v^t = w_1 H_v^{t-1} + w_2 H_v^t + w_3 H_v^{t+1}$ where t is the current time step. In general, the temporal dependency modeling consists of two key steps: *shift* and *multiply-accumulate*. We shift the input snapshot-specific node embedding H_v^t by -1, 0, +1 step and multiply by w_1, w_2, w_3 respectively, and then sum up to generate the representation Y_v^t. Formally, the *shift* operation is given as follows:

$$(H_v^t)^{-1} = H_v^{t-1}, (H_v^t)^0 = H_v^t, (H_v^t)^{+1} = H_v^{t+1} \tag{3}$$

and the *multiply-accumulate* operation is formally represented as:

$$Y_v^t = w_1 (H_v^t)^{-1} + w_2 (H_v^t)^0 + w_3 (H_v^t)^{+1} \tag{4}$$

By considering that our multiply-accumulate operation is computationally expensive, due to the involved multiplications. To effectively captures the temporal evolution of network in time dimension without significantly increasing computational cost, we optimize our TIME component with the following operation: since the number of nodes in each graph snapshot may be different, we only shift the features of the nodes that already existed in the previous snapshots to the current snapshot.

$$\begin{aligned} \mathbf{H}_A^t &= \mathbf{A}^t \cdot (\mathbf{H}_{(l-1)}^t)^{-1} = \mathbf{A}^t \cdot \mathbf{H}_{(l-1)}^{t-1}, \\ \mathbf{H}_M^t &= \mathbb{M}^t \cdot (\mathbf{H}_{(l-1)}^t)^{-1} = \mathbb{M}^t \cdot \mathbf{H}_{(l-1)}^{t-1}, \\ \mathbf{H}_{(l)}^t &= \mathbf{H}_A^t + \beta \mathbf{H}_M^t, \end{aligned} \tag{5}$$

where $(\cdot)^{-1}$ is the shift operation, *i.e.*, $(\mathbf{H}^{t_1})^{-1} = \mathbf{H}^{t_1}$. The shift operation along with the temporal dimension allows the model to capture the temporal characteristics nonlinearly. By incorporating this component into our MTSN framework, the model can expand the receptive field in the time dimension and capture the dynamic evolution characteristics of the network structures.

Figure 2 shows an example of the temporal shift operation. Specifically, the number of nodes in the network may be different at each time step (*e.g.*, $n^{t_1}, n^{t_2}, n^{t_3}$). For simplicity, we assume that the number of nodes at each time step increases (*i.e.*, $n^{t_1} < n^{t_2} < n^{t_3}$). $\{H_{(1)}^1, H_{(1)}^2, H_{(1)}^3\}$ in the first layer are the latent representations obtained after one layer. We shift the representations obtained from the previous time step to replace the representations of the current time step. For those nodes which do not exist at the previous time, we maintain their current representations. The new latent representations after the shift operation are used as the input of the second layer to be fed into next convolution operation.

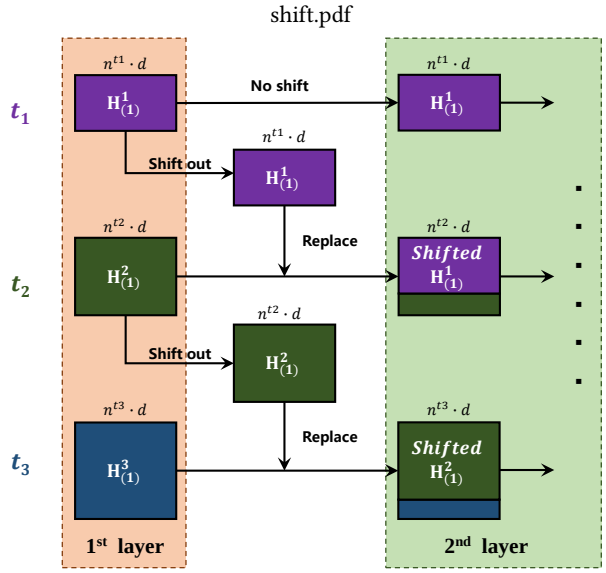


Figure 2: An example of temporal shift operation

In practice, we find that simultaneously performing shift operations on adjacency matrix and motif matrix cannot provide high performance nor efficiency. To be specific, a complete shift operation may cause most of the network structures in the current snapshot to be covered by historical information, resulting in the loss of network structures, and thus a significant decrease in algorithm performance for embedding network into latent representations.

To address the performance degradation issue caused by the complete shift operation, we propose a *partial shift* strategy in our TIME component to improve the model performance. In our model implementation, we evaluate a variety of shift combination methods, and observe that our proposed method can achieve the best performance only when the adjacency matrix is used for shifting, *i.e.*, without shift on the motif matrices. This suggests that the shift operation on adjacency matrix can better capture temporal evolution of dynamic networks, while the shift operation on motif matrix may harm the modeling of network temporal evolution. Hidden representations of previous snapshot are fused with the attribute matrix in our temporal shift mechanism during the layer-specific information propagation paradigm, which yields better performance compared with the selection of motif matrix. The potential reason lies in the attribute-aware adjacent matrix reflects the overall graph structure information, while the time-specific node interactive patterns are preserved with the motif-based node connections. We believe that the representation based on motif matrix can provide more local structural information on each independent snapshot, rather than the temporal characteristics in a dynamic network. Therefore, we use a partial shift operation, that is, shift operation is only performed based on adjacency matrix A in TIME layer. In addition, in order to alleviate the overfitting phenomenon, we add a dropout operation to each output. We re-define the convolution operation $\mathbf{H}^t_{(l)} = TIME(\mathbf{A}^t, \mathbb{M}^t, \mathbf{H}^t_{(l-1)})$ in TIME layer to replace

$$\begin{aligned}
\mathbf{H}^t_{(l)} = MPE(\mathbf{A}^t, \mathbb{M}^t, \mathbf{H}^t_{(l-1)}):& \\
\mathbf{H}^t_A &= dropout(\mathbf{A}^t \cdot (\mathbf{H}^t_{(l-1)})^{-1}) \\
&= dropout(\mathbf{A}^t \cdot \mathbf{H}^{t-1}_{(l-1)}), \\
\mathbf{H}^t_M &= dropout(\mathbb{M}^t \cdot \mathbf{H}^t_{(l-1)}), \\
\mathbf{H}^t_{(l)} &= \mathbf{H}^t_A + \beta \mathbf{H}^t_M,
\end{aligned} \tag{6}$$

where $(.)^{-1}$ is the shift operation and $(\mathbf{H}^t)^{-1} = \mathbf{H}^{t-1}$.

Accordingly, by stacking multiple TIME layers, our neural architecture MTSN can be represented as below:

$$\begin{aligned}
\mathbf{H}^t_{(0)} &= \mathbf{X}^t \cdot \mathbf{W}, \\
\mathbf{H}^t_{(l)} &= TIME(\mathbf{A}^t, \mathbb{M}^t, \mathbf{H}^t_{(l-1)}),
\end{aligned} \tag{7}$$

where $\mathbf{H}^t_{(l)}$ is the output of the l-th TIME layer, $l = 1, 2, \ldots, L$, and L is the number of layers. We endow our proposed dynamic graph neural network with the capability of handling node attributes, by projecting them into latent representations as node initialized embeddings. Hence, the node attributes could be incorporated into our MTSN to generate time-aware representations of nodes. By stacking temporal shift operation multiple times, our model can fully integrate the characteristics of the historical time steps with the network structure of the current time step, increasing the model's ability to capture the dynamic evolution characteristics of the network. It worth noting that MTSN not only encodes the motif-based graph relational structures in the individual time-aware snapshot, but also captures the graph structure evolving patterns through our designed temporal shift mechanism. In particular, the aggregated motif-aware node representations are incorporated into the graph relation encoder of next snapshot while preserving the high-order connectivity during the embedding propagation across graph layers. Therefore, our motif-based dynamic graph neural network is different from static motif-based aggregation methods.

4.3 Model Learning

In this section, we present the objective function that captures the dynamic structural evolution into node representations to train our model. Inspired by DySAT [41], we use the dynamic representation of a node v at time step t, Z^t_v, to preserve local proximity around v at time step t. In particular, we use a binary cross-entropy loss at each time step to increase the similarities between the node representations appearing in the same fixed-length random walks:

$$\begin{aligned}
\mathcal{L} = \sum_{t=1}^{T} \sum_{v \in \mathcal{V}^t} \Big(& \sum_{u \in \mathcal{N}^t_{walk}(v)} -\log\left(\sigma(< Z^t_u, Z^t_v >)\right) \\
& -w_n \sum_{u' \in Neg^t(v)} \log\left(1 - \sigma(< Z^t_{u'}, Z^t_v >)\right)\Big)
\end{aligned} \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid function, $<,>$ can be any vector similarity measure function (*e.g.*, inner product operation), $\mathcal{N}^t_{walk}(v)$ is the set of nodes that co-occur with v in fixed-length random walk. $Neg^t(v)$ is a negative sampling *w.r.t.* node v in snapshot $\mathcal{G}^t$, and w_n denotes the negative sampling ratio, which is a tunable hyper-parameter to balance the positive and negative samples. Notice that $\mathbf{W}$, $\{\alpha_i | i = 1, 2, \ldots, |M|\}$, and β in our model are all learnable parameters during our training phase.

Algorithm 1 shows the pseudo-code of our model. The complexity of linear transformation in line 3 is $O(T \cdot |\mathcal{V}| \cdot f \cdot d)$, and the complexity of calculating Eq. (7) in line 7 is $O(|\mathcal{E}| \cdot d + |M| \cdot |\mathcal{E}_M| + |\mathcal{E}_M| \cdot d)$, where f is the number of node attributes, d is the embedding dimensionality, $|\mathcal{V}|$ is the maximum number of nodes, $|\mathcal{E}|$ is the maximum number of edges in the dynamic network, $|\mathcal{E}_M|$ is the maximum number of edges in the motif matrices and $\mathcal{E}_M \subset \mathcal{E}$. Thus, the complexity of each iteration from line 4 to line 11 is $O(L \cdot T \cdot |\mathcal{E}| \cdot d + L \cdot T \cdot |M| \cdot |\mathcal{E}_M| + L \cdot T \cdot |\mathcal{E}_M| \cdot d)$. As a result, the overall time complexity is $O(T \cdot |\mathcal{V}| \cdot f \cdot d + L \cdot T \cdot |\mathcal{E}| \cdot d)$, *i.e.*, TIME is linear with respect to the number of network edges. At the end, we use the output of the last layer from our last snapshot as the representation of the dynamic network up to the current time step.

Algorithm 1 The Learning Process of MTSN

Input: Adjacency matrix set $\{\mathbf{A}^1, \mathbf{A}^2, \ldots, \mathbf{A}^T\}$, attribute matrix set $\{\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^T\}$, motif type set M, and the number of graph neural layers L.
Output: Representations $\mathbf{Z}^T$

1: Use PGD to generate motif matrices $\mathcal{M}$ for each network snapshot;
2: $\mathcal{N}^T_{walk} = RandomWalk(\{\mathbf{A}^1, \mathbf{A}^2, \ldots, \mathbf{A}^T\})$;
3: Perform a linear transformation: $\mathbf{H}^t_0 = \mathbf{X}^t \cdot \mathbf{W}$;
4: **for** $iter = 1, \ldots, max_{iter}$ **do**
5: **for** l in $1 : L$ **do**
6: **for** t in $1 : T$ **do**
7: Calculate $\mathbf{H}^t_{(l)}$ by Eq. (7)
8: **end for**
9: **end for**
10: Use $\mathcal{N}^T_{walk}$ to optimize model by Eq. (8)
11: **end for**
12: Output: $\mathbf{Z}^T \leftarrow \mathbf{H}^T_{(L)}$

5 EXPERIMENTS

In this section, we perform experiments on several real-world datasets to evaluate the performance of our proposed MTSN framework. We first describe the experimenntal settings and then present the evaluation results as compared to state-of-the-art baselines.

5.1 Datasets

We conduct extensive experiments on four public real-world datasets. UCI [39] data is from a publicly available communication network, in which the links denote messages sent between peer users on an online social network platform. The time period is from Apr. 2004 to Aug. 2004 and the corresponding resolution of time snapshots is set as ten days. MovieLens (ML-10M) [10] consists of user-tag interactions where the links connect user with the tags they applied on certain movies. The period of MovieLens data spans from Dec. 2005 to Dec. 2009 and the corresponding time resolution of partitioned graph snapshots is set as ninety days. Epinions[1] shows the trust relationships between users. In this data, the edge indicates the trust relations between these two connected users. The time span of the dataset is from 2003 to 2011, and the attribute is the user's registration year and the degree of interest over different categories of products. Alibaba dataset[2] has two node types, user and item, and includes four types of edges between users and items. We use the click edge type in the experiment, and the time span is 11 days from June 10, 2019 to June 20, 2019. For this data, the attributes of users includes gender, age group, education level, career, income and life stage, and the attributes of product contains the first-level category, the second-level category, brand and price. Since some of the baselines cannot scale to the whole graph on Epinions and Alibaba datasets, we evaluate model performance on two sampled datasets from Epinions and Alibaba, respectively. Additionally, we also sample a large dataset from Epinions, denoted by Epinions-L, to evaluate our model in a large-scale dynamic network embedding scenario. The statistics of these datasets are summarized in Table 1.

Table 1: Statistics of Datasets

Dataset	#nodes	#edges	#features	#time steps
UCI	1,809	16,822	/	12
ML-10M	20,537	43,760	/	9
Epinions	9,398	231,537	44	9
Epinions-L	69,236	280,180	44	9
Alibaba	5,640	53,049	19	11

5.2 Baselines

In our evaluation, we compare our MTSN against the following graph learning baselines with different model structures:

- **GraLSP** [18] - GraLSP is a graph neural network model which explicitly incorporates local structural patterns into the neighborhood aggregation through random walk strategy.
- **GraphSTONE** [28] - GraphSTONE is built upon the graph convolutional network that utilizes topic models of graphs, such that the structural topics could capture indicative graph structures broadly from a probabilistic aspect.
- **dynGEM** [19] - dynGEM utilizes deep auto-encoders to incrementally generate representation of a dynamic graph based on the data only from the last snapshot.
- **dynAE** and **dynAERNN** [9] - They are DNN with dense layers and DNN composed of dense layers with recurrent neural component to capture temporal evolution, respectively.
- **MMDNE** [29] - MMDNE microscopically models the formation process of network structure with a temporal attention point process, and macroscopically constrains network structure to obey a certain evolutionary pattern with dynamic learning.
- **CTDNE** [37] - CTDNE incorporates the temporal information into the graph embedding based on the random walk, so as to learn time-dependent network representation for continuous-time dynamic networks with temporal dependencies.
- **TGAT** [48] - TGAT employs the self-attention and functional time encoding technique to aggregate the temporal features as well as the underlying interactions.

[1] https://cse.msu.edu/~tangjili/trust.html

[2] https://tianchi.aliyun.com/competition/entrance/231719/information/

Table 2: AUC and F1 score of dynamic link prediction on four datasets

Dataset	UCI		ML-10M		Epinions		Alibaba	
Method	AVG AUC	AVG F1	AVG AUC	AVG F1	AVG AUC	AVG F1	AVG AUC	AVG F1
GraLSP	0.559	0.536	0.629	0.586	0.807	0.752	0.575	0.547
GraphSTONE	0.557	0.545	0.631	0.601	0.838	0.754	0.587	0.557
dynGEM	0.541	0.604	0.694	0.654	0.560	0.636	0.671	0.629
dynAE	0.582	0.615	OOT	OOT	0.556	0.663	0.645	0.615
dynAERNN	0.536	0.550	OOT	OOT	0.532	0.644	0.584	0.564
MMDNE	0.724	0.705	0.741	0.681	0.792	0.784	0.856	0.793
CTDNE	0.565	0.557	0.650	0.642	0.532	0.523	0.546	0.540
TGAT	0.640	0.602	0.630	0.603	0.868	0.799	0.763	0.708
DySAT	0.728	0.668	0.870	0.799	0.910	0.847	0.815	0.770
TSN	0.847	0.766	0.903	0.821	0.919	0.862	0.871	0.832
MTSN	**0.859**	**0.777**	**0.916**	**0.862**	**0.931**	**0.879**	**0.886**	**0.842**

OOT: Out Of Time (72 hours). The best results are shown in bold, and the second best results are underlined.

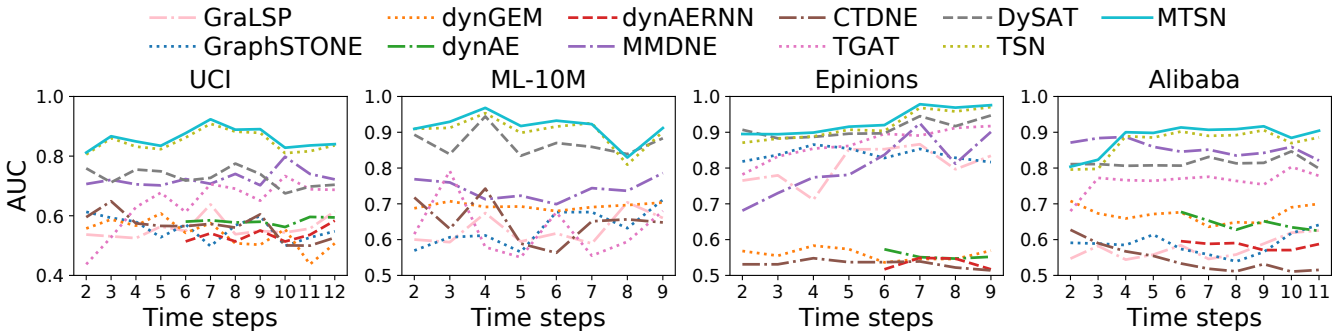


Figure 3: Evaluation results on dynamic link prediction in terms of AUC.

- **DySAT** [41] - DySAT is a neural architecture that learns node representations to capture dynamic graph structural evolution by using self-attention to aggregate the structural neighborhood information and temporal dynamics.

In our performance comparison, as GraLSP and GraphSTONE can only deal with static networks, for a fair comparison, we run these methods from scratch at each time step, so as to reflect the temporal information. MMDNE, CTDNE and TGAT are continuous-time dynamic network embedding methods. dynGEM, dynAE, dynAERNN, DySAT and MTSN are all discrete-time dynamic network embedding methods. In addition to the above state-of-the-art baselines, we also design a variant **TSN** of our model to verify the effectiveness of our proposed motif-preserving encoder, by removing the network motifs in our graph structure learning and only use adjacency matrix to learn node representation in TIME layers.

5.3 Experimental Setting

Link prediction is a very important task in real-world network mining scenarios, *e.g.*, the interactions between users and items in e-commerce platforms for making accurate recommendation [11, 31]. Following DySAT [41], the evaluation focuses on the dynamic link prediction. Using the node embeddings at time step t, dynamic link prediction predicts the connections between nodes at the next time step $t+1$. This task has been widely used in evaluating the quality of dynamic node representations to predict the temporal evolution of graph structures [9, 19, 41]. We use the commonly used evaluation metrics in link prediction, *i.e.*, the Area Under the ROC Curve (AUC) and the F1 score as our performance measurement.

In our experiments, we use 20% links in the network at the next time step as the validation set to tune model hyperparameters. We randomly sample another 20% links for training, and use the remaining 60% for testing. In our prediction layer, we train a logistic regression classifier to predict whether there exist a link between each node pair in the testing set. We set embedding dimensionality d as 128 for all the methods for a fair comparison. We use the source code released by authors for baseline evaluation. Specifically, for all random walk based methods, we set window size as 6. In addition, the number of negative sampling, walks and walk length as 8, 100 and 10, respectively. For dynAE and dynAERNN, we set lookback to 4, that is, 4 snapshots before the current time are used as the historical information. For our MTSN, we set the number of graph neural layers L to 3, dropout rate as 0.9, and max_{iter} as 30. We evaluate the efficiency of all methods on a machine with Intel Xeon E5-2660 (2.2GHz) CPU and 80GB memory.

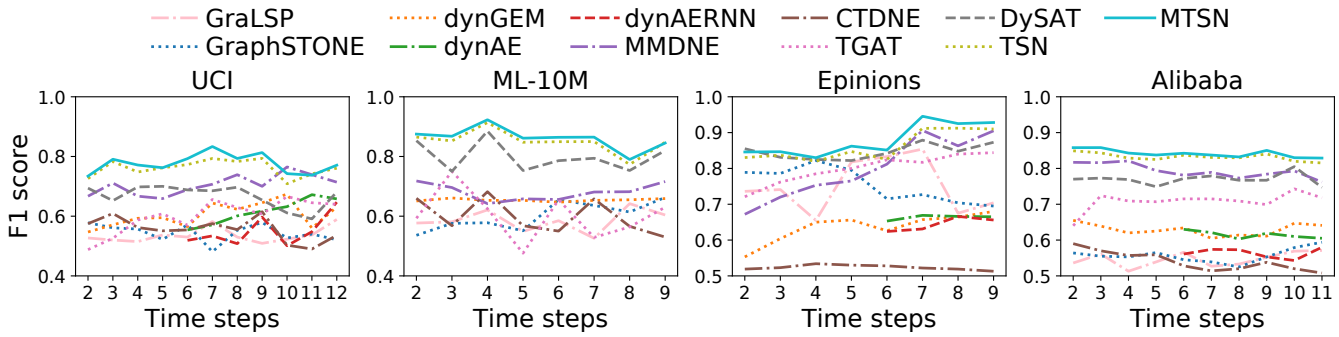


Figure 4: Evaluation results on dynamic link prediction in terms of F1-score.

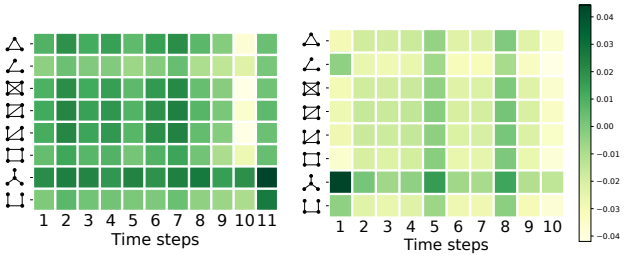
Table 3: Dynamic link prediction results on Epinions-L

Method	AVG AUC	AVG F1
CTDNE	0.696	0.650
TSN	**0.887**	**0.852**

5.4 Dynamic Link Prediction

First, we evaluate the model performance of our MTSN as compared to other state-of-the-art baselines in dynamic link prediction task. Considering that both dynAE and dynAERNN require a certain amount of historical information to train, we set the parameter look-back as 4. Hence these methods do not output results on the first 4 time steps. Experimental results are reported in Table 2. As we can see, our MTSN significantly outperforms all baseline methods in terms of both AUC and F1 score on four datasets. More specifically, experimental results indicate that MTSN achieves average gains of 7–8% AUC and F1 score in comparison to the best performed baseline across all datasets. Considering that the performance gain in link prediction reported in some network representation learning methods [18, 28, 29] is usually around 2-3%, this performance improvement achieved by our MTSN is significant. Among various compared methods, DySAT performs the best, because it attempts to learn node representations by adopting the self-attention along with two dimensions of structural neighborhoods and temporal dynamics. However, our MTSN consistently outperforms DySAT by 2-18% AUC and 4-16% F1 score across all datasets, which suggests the rationality of our designed motif-preserving dynamic graph neural network architecture. In addition, our model variant TSN performs better than compared baselines, which validates the effectiveness of our developed temporal shift mechanism for modeling temporal dependencies in dynamic network.

Furthermore, we report the evaluation results of all approaches in terms of AUC and F1 score at each time step in Figure 3 and Figure 4. From these results, we have the following observations: First, the performance of the static methods, *i.e.*, GraLSP and GraphSTONE, perform worse than most dynamic methods, but they performs better than the dynamic network embedding methods such as dynGEM, dynAE, dynAERNN, MMDNE and CTDNE on Epinions dataset. This potential reason is that these dynamic methods cannot leverage the attribute information included in the Epinions dataset.



(a) UCI dataset (b) Alibaba dataset

Figure 5: Motif weight on different datasets

However, the attribute information can provide useful knowledge to effectively improve the performance of dynamic link prediction on Epinions data. This has been demonstrated by the observation that TGAT and DySAT perform better than MMDNE on Epinions. Second, discrete-time methods (*e.g.*, DySAT and MTSN) outperform continuous-time methods (*e.g.*, MMDNE, CTDNE and TGAT) on most datasets (except Alibaba dataset). One possible explanation is that the continuous-time method can hardly capture the network evolution with significant structure variation among time steps. But the network structure changes between adjacent snapshots of UCI, ML-10M and Epinions are relatively large, while the resolution of each time step in Alibaba data is one day, thus the changes in the network structure are relatively small. This is why MMDNE performs better than other discrete-time methods on Alibaba. However, our method still achieves best on Alibaba data, because the local high-order structures in motifs that implies important correlation factors for potential links are embedded in our learned node dynamic representations. In addition, our model performs multiple temporal shift operations in time dimension, and can effectively capture accurate temporal evolution characteristics in networks with different time granularities.

Additionally, due to the heavy computational time and memory cost of most baseline methods (*e.g.*, more than 72 hours), we only evaluate our TSN and CTDNE on the large-scale Epinions-L dataset, and the results are reported in Table 3. From the results, we can see that our variant TSN still achieves better results than CTDNE in terms of both AUC and F1 score on large-scale Epinions-L dataset.

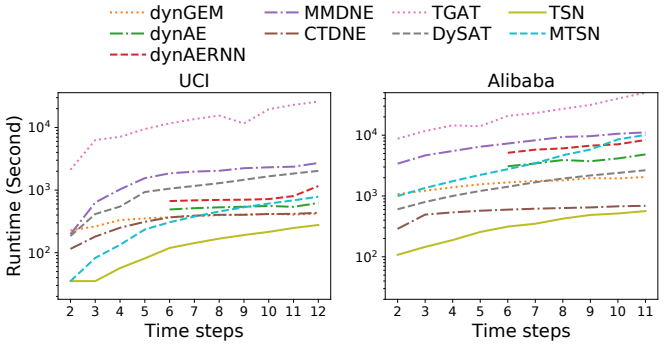


Figure 6: Runtime *w.r.t.* time step on UCI and Alibaba

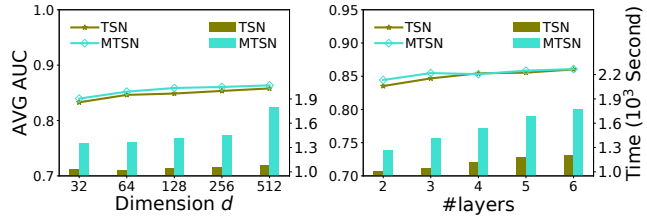


Figure 7: AUC and Runtime *w.r.t.* d and #layers L

5.5 Impact of Different Network Motifs

To understand the impact of motifs in generating node representations in dynamic network, we visualize the motif weights learned at each time step on UCI and Alibaba data in Figure 5. We can see that the importance of motifs for node embeding generation vary by datasets, and the importance of motifs at different time steps also show different patterns.

In communication network UCI, we can observe that users are more likely to interact with close friends in the same communities, and the dense local structures could helps us improve the prediction for future links. In E-commerce dataset Alibaba, the links in the network indicate the interactions between users and products. We can also observe that the four-star motif has the highest weight, but most other types of motifs have only a small effect. This observation is consistent with the characteristics of Alibaba as an E-commerce website without social functions. In addition, because users tend to click products that meet their preferences, four-star motif can help us find a collection of products with high similarity in link prediction. We can find that although the more general motif might be a subgraph of the other types of motif structures, network motifs with complex structures could preserve some node-specific unique topology characteristics, which is complementary to more general motif types. The above observations justify that our model can automatically capture and differentiate the different types of local high-order features in network at different time steps.

5.6 Efficiency and Parameter Sensitivity

Model Efficiency. In addition to the prediction accuracy, we further investigate the efficiency of our MTSN and the variant TSN and seven dynamic baselines on UCI and Alibaba datasets. Figure 6 shows the model efficiency evaluation results of different approaches. From Figure 6, it can be seen that the efficiency of our variant TSN is significantly better than all dynamic baselines, which suggests the good model scalability of our proposed framework in handling large-scale dataset. Recall that TSN can achieve much better prediction performance than state-of-the-art baselines. As we mentioned in the TIME layer section, the temporal shift operation can learn the dynamic evolution of the network at a very low computational cost. Our MTSN incorporates network motifs to obtain stronger representation performance with the increase of computation cost accordingly. Nevertheless, our MTSN is still more efficient than state-of-the-art methods: MMDNE and TGAT. In reality, if we pursue higher efficiency, we can use our variant TSN, which has highest efficiency, and also achieves better performance over state-of-the-art baselines. If we pursue high prediction performance, we can use our MTSN, which achieves state-of-the-art performance with an acceptable time consumption.

Parameter Sensitivity. We also evaluate the sensitivity of our proposed method MTSN and TSN *w.r.t.* different settings of embedding dimension d and the number of TIME layers L. Figure 7 shows the experimental results on UCI dataset. From the evaluation results, we find that the performance of our MTSN and TSN increases lightly with the larger embedding dimensionality. Similar observations for the effect of the number of layers. The model performance becomes stable when the number of layers reaches at 4. This is because that modeling the higher-order network structures with a more deep graph network may lead to the over-smoothing issue [3]. Nevertheless, we can notice see that the improvement in prediction performance of our model requires more computation cost. In practical scenarios, we can balance the performance and efficiency by flexibly choosing the model settings according to the requirements of network mining tasks.

6 CONCLUSION

In this paper, we present a motif-preserving dynamic attributed network embedding MTSN that captures the local high-order structures and temporal evolution for dynamic attributed networks. MTSN learns node dynamic representations by stacking TIME layers that simultaneously model both local high-order structural proximities and temporal dynamics for dynamic network embedding. Experiments on four real-world networks demonstrate that our model significantly outperforms state-of-the-art baselines in terms of both effectiveness and efficiency in dynamic link prediction. For future work, we are interested in inducing a more complex motif-aware sequential model to preserve the temporal dynamics of network structures for graph representation learning.

ACKNOWLEDGMENTS

This work is partially supported by the National Key Research and Development Program of China under grant No. 2018AAA0100602, the National Natural Science Foundation of China under grant Nos. 61773331, U1706218, 41927805 and 61976123, and Key Development Program for Basic Research of Shandong Province under grant No. ZR2020ZD44. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies. Corresponding author: Yanwei Yu.

REFERENCES

[1] Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, and Nick G. Duffield. 2015. Efficient Graphlet Counting for Large Networks. In *ICDM*. 1–10.

[2] Ranran Bian, Yun Sing Koh, Gillian Dobbie, and Anna Divoli. 2019. Network Embedding and Change Modeling in Dynamic Heterogeneous Networks. In *SIGIR*. 861–864.

[3] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI*, Vol. 34. 3438–3445.

[4] 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 *SIGKDD*. 1177–1186.

[5] Sam De Winter, Tim Decuypere, Sandra Mitrović, Bart Baesens, and Jochen De Weerdt. 2018. Combining temporal aspects of dynamic networks with Node2Vec for a more efficient dynamic link prediction. In *ASONAM*. 1234–1241.

[6] Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2019. Learning Dynamic Context Graphs for Predicting Social Events. In *SIGKDD*. 1007–1016.

[7] 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 *IJCAI*. 2086–2092.

[8] Hongchang Gao and Heng Huang. 2018. Deep Attributed Network Embedding. In *IJCAI*. 3364–3370.

[9] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. 2020. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems* 187 (2020), 104816.

[10] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[11] Chao Huang, Jiahui Chen, Lianghao Xia, Yong Xu, Peng Dai, Yanqing Chen, Liefeng Bo, et al. 2021. Graph-enhanced multi-task learning of multi-level transition dynamics for session-based recommendation. In *AAAI*.

[12] Chao Huang, Baoxu Shi, Xuchao Zhang, Xian Wu, et al. 2019. Similarity-aware network embedding with self-paced learning. In *CIKM*. 2113–2116.

[13] 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 *SIGKDD*. 2613–2622.

[14] Hong Huang, Zixuan Fang, Xiao Wang, Youshan Miao, and Hai Jin. 2020. Motif-Preserving Temporal Network Embedding. In *IJCAI*. 1237–1243.

[15] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *WSDM*. 731–739.

[16] Xiao Huang, Qingquan Song, Fan Yang, and Xia Hu. 2019. Large-Scale Heterogeneous Feature Embedding. In *AAAI*. 3878–3885.

[17] Di Jin, Mark Heimann, Ryan A Rossi, and Danai Koutra. 2019. node2bits: Compact Time-and Attribute-aware Node Representations for User Stitching. In *ECML PKDD*. 483–506.

[18] Yilun Jin, Guojie Song, and Chuan Shi. 2020. GraLSP: Graph Neural Networks with Local Structural Patterns. In *AAAI*. 4361–4368.

[19] Nitin Kamra, Palash Goyal, Xinran He, and Yan Liu. 2017. DynGEM: deep embedding method for dynamic graphs. In *IJCAI International Workshop on Representation Learning for Graphs (ReLiG)*.

[20] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *SIGKDD*. 1269–1278.

[21] John Boaz Lee, Ryan A Rossi, Xiangnan Kong, Sungchul Kim, Eunyee Koh, and Anup Rao. 2019. Graph convolutional networks with motif-based attention. In *CIKM*. 499–508.

[22] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed network embedding for learning in a dynamic environment. In *CIKM*. 387–396.

[23] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2018. Attributed social network embedding. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2257–2270.

[24] Ji Lin, Chuang Gan, and Song Han. 2019. Tsm: Temporal shift module for efficient video understanding. In *ICCV*. 7083–7093.

[25] Xi Liu, Ping-Chun Hsieh, Nick Duffield, Rui Chen, Muhe Xie, and Xidao Wen. 2019. Real-time streaming graph embedding through local actions. In *Companion Proceedings of WWW*. 285–293.

[26] Zhijun Liu, Chao Huang, Yanwei Yu, Baode Fan, and Junyu Dong. 2020. Fast Attributed Multiplex Heterogeneous Network Embedding. In *CIKM*. 995–1004.

[27] Zhijun Liu, Chao Huang, Yanwei Yu, Peng Song, Baode Fan, and Junyu Dong. 2020. Dynamic Representation Learning for Large-Scale Attributed Networks. In *CIKM*. 1005–1014.

[28] Qingqing Long, Yilun Jin, Guojie Song, Yi Li, and Wei Lin. 2020. Graph Structural-topic Neural Network. In *SIGKDD*. 1065–1073.

[29] 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.

[30] Wenjuan Luo, Han Zhang, Xiaodi Yang, Lin Bo, Xiaoqing Yang, Zang Li, Xiaohu Qie, and Jieping Ye. 2020. Dynamic Heterogeneous Graph Neural Network for Real-time Event Prediction. In *SIGKDD*. 3213–3223.

[31] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *SIGKDD*. 825–833.

[32] Jianxin Ma, Peng Cui, and Wenwu Zhu. 2018. DepthLGP: Learning Embeddings of Out-of-Sample Nodes in Dynamic Networks. In *AAAI*. 370–377.

[33] Lijia Ma, Yutao Zhang, Jianqiang Li, Qiuzhen Lin, Qing Bao, Shanfeng Wang, and Maoguo Gong. 2020. Community-aware dynamic network embedding by using deep autoencoder. *Information Sciences* 519 (2020), 22–42.

[34] Franco Manessi, Alessandro Rozza, and Mario Manzo. 2020. Dynamic graph convolutional networks. *Pattern Recognition* 97 (2020), 107000.

[35] Renny Márquez and Richard Weber. 2019. Overlapping community detection in static and dynamic social networks. In *WSDM*. 822–823.

[36] Zaiqiao Meng, Shangsong Liang, Xiangliang Zhang, Richard McCreadie, and Iadh Ounis. 2020. Jointly Learning Representations of Nodes and Attributes for Attributed Networks. *TOIS* 38, 2 (2020), 1–32.

[37] 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 WWW*. 969–976.

[38] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *SIGKDD*. 1105–1114.

[39] Pietro Panzarasa, Tore Opsahl, and Kathleen M Carley. 2009. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *Journal of the American Society for Information Science and Technology* 60, 5 (2009), 911–932.

[40] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B Schardl, and Charles E Leiserson. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *AAAI*. 5363–5370.

[41] 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 *ICDM*. 519–527.

[42] Uriel Singer, Ido Guy, and Kira Radinsky. 2019. Node Embedding over Temporal Graphs. In *IJCAI*. 4605–4612.

[43] 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.

[44] Aynaz Taheri, Kevin Gimpel, and Tanya Berger-Wolf. 2019. Learning to represent the evolution of dynamic graphs with recurrent models. In *Companion Proceedings of WWW*. 301–307.

[45] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *ICLR*.

[46] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[47] Yun Xiong, Yao Zhang, Hanjie Fu, Wei Wang, Yangyong Zhu, and S Yu Philip. 2019. Dyngraphgan: Dynamic graph embedding via generative adversarial networks. In *DASFAA*. 536–552.

[48] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *ICLR*.

[49] Yanlei Yu, Zhiwu Lu, Jiajun Liu, Guoping Zhao, and Ji-rong Wen. 2019. Rum: Network representation learning using motifs. In *ICDE*. 1382–1393.

[50] Yanwei Yu, Hongjian Wang, and Zhenhui Li. 2018. Inferring mobility relationship via graph embedding. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)* 2, 3 (2018), 1–21.

[51] Yanwei Yu, Huaxiu Yao, Hongjian Wang, Xianfeng Tang, and Zhenhui Li. 2018. Representation learning for large-scale dynamic networks. In *DASFAA*. 526–541.

[52] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *SIGKDD*. 793–803.

[53] Ziwei Zhang, Peng Cui, Jian Pei, Xiao Wang, and Wenwu Zhu. 2018. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. In *AAAI*. 224–231.

[54] Aakas Zhiyuli, Xun Liang, Yanfang Chen, and Xiaoyong Du. 2018. Modeling Large-Scale Dynamic Social Networks via Node Embeddings. *IEEE Transactions on Knowledge and Data Engineering* 31, 10 (2018), 1994–2007.

[55] Aakas Zhiyuli, Xun Liang, YanFang Chen, Peng Shu, and Xiaoping Zhou. 2018. Joint Learning of Evolving Links for Dynamic Network Embedding. In *AAAI*. 8189–8190.

[56] Dingyuan Zhu, Peng Cui, Ziwei Zhang, Jian Pei, and Wenwu Zhu. 2018. High-order proximity preserved embedding for dynamic networks. *IEEE Transactions on Knowledge and Data Engineering* 30, 11 (2018), 2134–2144.

[57] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding temporal network via neighborhood formation. In *SIGKDD*. 2857–2866.