BehaviorNet: A Fine-grained Behavior-aware Network for Dynamic Link Prediction

MINGYI LIU, Harbin Institute of Technology, China
ZHIYING TU, Harbin Institute of Technology, China and Science & Technology on Integrated Infomation System Laboratory, Institute of Software Chinese Academy of Sciences, China
TONGHUA SU, Harbin Institute of Technology, China
XIANZHI WANG, University of Technology Sydney, Australia
XIAOFEI XU, Harbin Institute of Technology, China
ZHONGJIE WANG, Harbin Institute of Technology, China

Dynamic link prediction has become a trending research subject because of its wide applications in web, sociology, transportation, and bioinformatics. Currently, the prevailing approach for dynamic link prediction is based on graph neural networks, in which graph representation learning is the key to perform dynamic link prediction tasks. However, there are still great challenges because the structure of graphs evolves over time. A common approach is to represent a dynamic graph as a collection of discrete snapshots, in which information over a period is aggregated through summation or averaging. This way results in some fine-grained time-related information loss, which further leads to a certain degree of performance degradation. We conjecture that such fine-grained information is vital because it implies specific behavior patterns of nodes and edges in a snapshot. To verify this conjecture, we propose a novel fine-grained behavior-aware network (BehaviorNet) for dynamic network link prediction. Specifically, BehaviorNet adapts a transformer-based graph convolution network to capture the latent structural representations of nodes by adding edge behaviors as an additional attribute of edges. GRU is applied to learn the temporal features of given snapshots of a dynamic network by utilizing node behaviors as auxiliary information. Extensive experiments are conducted on several real-world dynamic graph datasets, and the results show significant performance gains for BehaviorNet over several state-of-the-art (SOTA) discrete dynamic link prediction baselines. Ablation study validates the effectiveness of modeling fine-grained edge and node behaviors.

CCS Concepts: • **Information systems** → **Social networks**; • **Computing methodologies** → **Neural networks**.

Additional Key Words and Phrases: Link prediction, Dynamic networks, Graph Neural Networks, Dynamic Representation Learning

1 INTRODUCTION

Dynamic network is a natural and powerful tool to delineate complex systems such as social networks[59], finance[13], bioinformatics[27], and knowledge graphs[6, 53]. Dynamic link prediction is a typical task in

Authors' addresses: Mingyi Liu, liumy@hit.edu.cn, Harbin Institute of Technology, 92 West Dazhi Street, Nangang District, Harbin, Heilongjiang, China, 150001; Zhiying Tu, tzy_hit@hit.edu.cn, Harbin Institute of Technology, 2 West Wenhua Road, Weihai, Shandong, China, 264209 and Science & Technology on Integrated Infomation System Laboratory, Institute of Software Chinese Academy of Sciences, Beijing, China, 264209; Tonghua Su, thsu@hit.edu.cn, Harbin Institute of Technology, 92 West Dazhi Street, Nangang District, Harbin, Heilongjiang, China, 150001; Xianzhi Wang, xianzhi.wang@uts.edu.au, University of Technology Sydney, Sydney, Australia; Xiaofei Xu, xiaofei@hit.edu.cn, Harbin Institute of Technology, 92 West Dazhi Street, Nangang District, Harbin, Heilongjiang, China, 150001; Zhongjie Wang, rainy@hit.edu.cn, Harbin Institute of Technology, 92 West Dazhi Street, Nangang District, Harbin, Heilongjiang, China, 150001.

1559-1131/2023/1-ART $15.00
https://doi.org/10.1145/3580514

dynamic network analysis that is intended to forecast future links of a certain network based on its historical information[65].

For a given dynamic network, there are two approaches to represent the dynamics: continuous and discrete[43]. Discrete approaches model the dynamic network as a chronological sequence of snapshots that are generated by aggregating temporal information at different time intervals. On the other hand, continuous approaches do not use temporal aggregations but they model the dynamic network as an interaction flow. Fig. 1 shows some examples of discrete networks and continuous networks and their transformations. Although continuous networks contain richer information and methods based on continuous networks could achieve better performance, most research focuses on discrete networks because continuous interaction information is costly or impossible to obtain in real-world applications. Actually, continuous networks are special cases of discrete networks if the time interval is close to the infinitesimal. In this paper, we focus on discrete dynamic network link prediction and we try to impose the advantages of continuous networks that retain richer information to discrete networks by using fine-grained temporal information.

The key challenge for dynamic link prediction is how to represent nodes as low-dimensional vectors that capture their structural and temporal information, which is called dynamic representation learning. Discrete dynamic representation learning approaches follow the same paradigm, i.e., constructing a structural encoder (e.g., GCN[61], GAT[51]) followed by a temporal encoder (e.g., RNN[41], Attention[50]). This paradigm is often criticized by continuous representation learning methods because the loss of temporal information would lead to performance degradation [28, 43]. Causes of such temporal information loss can be classified into two types:

(1) Real-world restrictions make it impossible to observe specific temporal information at the time of observation (from the top layer to middle layer in Fig. 1). For example, some news websites only give the release date of news, and when multiple news released on the same day are observed, the order of these news is lost. This kind of temporal information loss cannot be avoided and makes continuous representation learning methods not available.
(2) The aggregation operation in forming snapshots results in a loss of temporal information (from the middle layer to bottom layer in Fig. 1). The aggregation operation is necessary as the observed fine-grained information is very sparse and thus it is difficult to be directly captured. This type of temporal information loss can be avoided, but unfortunately existing methods ignore it.

We argue that *fine-grained temporal information is valuable and should be effectively used in an appropriate manner because this fine-grained information can reflect certain patterns of node and edge behaviors.*

Therefore, the following question needs to be answered: *is it possible to extend the existing discrete dynamic network representation learning architecture to make it capable of dealing with such node behaviors and edge behaviors to improve dynamic link prediction performance?*

To answer this question, we propose a novel fine-grained behavior-aware network called BehaviorNet. Different from existing approaches, when aggregating interactions within a time interval to form a snapshot, BehaviorNet cuts the interval into a fixed number of equal fine-grained time intervals. Edge behavior vectors and node behavior vectors are generated from these fine-grained intervals as auxiliary attributes of the aggregated snapshot. Edge behaviors are more responsive to the interaction patterns on the local structure of the dynamic network, while node behaviors are more responsive to the patterns of the nodes themselves in the temporal order. BehaviorNet is designed with a transformer-based graph convolution network as a structural encoder where edge behaviors are processed as edge attributes, followed by a GRU-based temporal encoder where node behaviors are processed as auxiliary information to handle these two different types of behaviors. Extensive experiments have been conducted on six standard dynamic datasets and the results show that BehaviorNet shows significantly better performance than SOTA methods. Furthermore, we demonstrate the benefits of using fine-grained information through ablation study. Main contributions are summarized as follows:

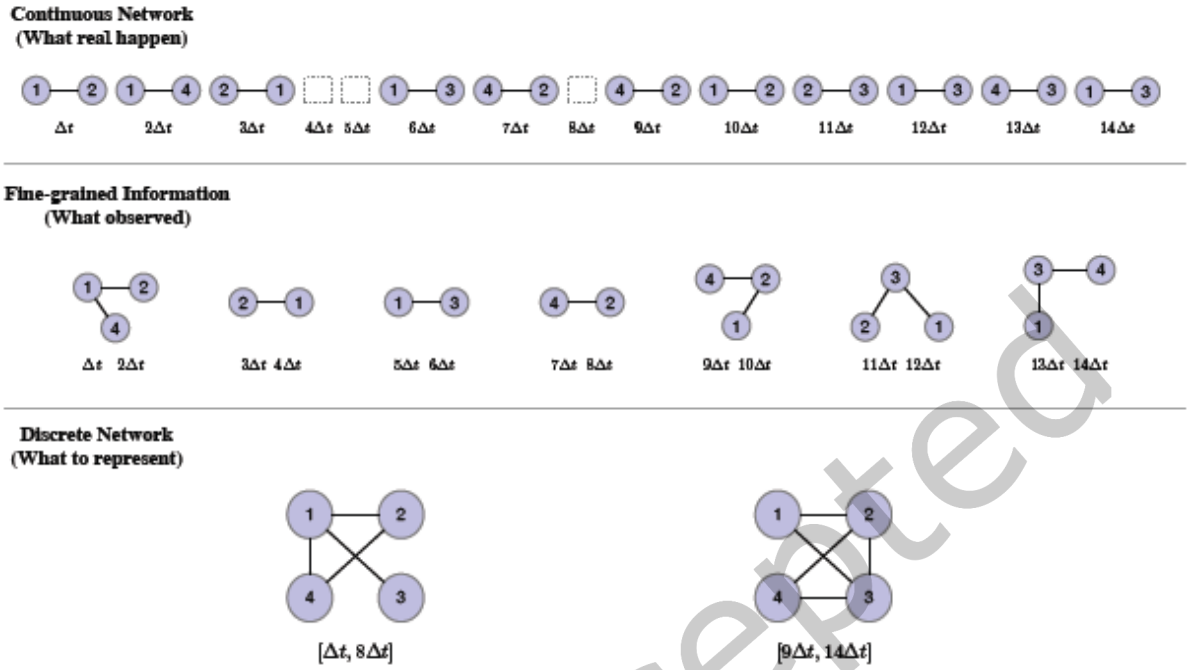


Fig. 1. Example of continuous network and discrete networks, and transformations between them

(1) We demonstrate that under the conditions where continuous timing information is unavailable or costly to obtain, the loss of fine-grained temporal information brought by the coarse aggregation forming snapshots, which is common in existing discrete dynamic graph networks, leads to degradation of model performance. To address this issue, we introduce node behaviors and edge behaviors to express fine-grained temporal information from different perspectives.
(2) We propose a novel fine-grained behavior-aware neural network named BehaviorNet for dynamic network link prediction. Compared with existing discrete dynamic graph networks, BehaviorNet can preserve fine-grained information through node behaviors and edge behaviors to enhance structural and temporal information.
(3) We evaluate BehaviorNet and some SOTA methods using six standard dynamic datasets: HYPERTEXT, ENRON, EMAIL-DNC, FB-MESSAGES, UCI and SOC. The results show that the superior performance for BehaviorNet over SOTA methods in dynamic link prediction tasks.
(4) We discuss parameter sensitivity of BehaviorNet and give some suggestions for using BehaviorNet in practice and future research.

The remainder of this paper is organized as follows. Section 2 discusses related works. Section 3 presents related preliminaries. Section 4 gives a detailed interpretation of BehaviorNet. Section 5 are the experimental results, as well as detailed analysis and discussion. Section 6 is the conclusion.

2 RELATED WORK

Existing link prediction methods can be roughly divided into two categories[2]: one is classical similarity-based methods and the other is feature learning based methods.

2.1 Similarity-based methods

Similarity-based methods assume that the more similar two nodes are, the more likely it is that a link exists between them. The key of these methods is the choice of a similarity measure function. The similarity measure function can be subdivided into global similarity measure functions, local similarity measure functions and quasi-local similarity measure functions[52].

Global similarity measure functions use the topological information of the entire network to measure the similarity of nodes, such as Katz Index, SimRank, Random Walk, etc. Because the global similarity measure functions have to deal with the entire network, they cannot be computed in parallel and the time complexity of these methods is quadratically related to the size of the network, so these methods are not suitable for large-scale networks.

Local similarity measure functions use the the local topological information to measure the similarity of nodes, such as Commmon Neighbors, Jaccard Index, adamic-adar, etc. Compared with global similarity measure functions, local similarity have worse performance but its low computational cost allows it to be used in large-scale networks.

Quasi-local similarity measure functions trade off the global similarity measure functions and local similarity measure functions to obtain better link prediction performance and maintain low computational cost, such as Local Random Walk, FriendLink, etc.

Overall, similarity-based methods adopt heuristic algorithms, they are usually very efficient and easy to implement, but the assumption that similar nodes are more likely to be linked does not hold in many scenarios, and these methods do not take the time-varying features into account very adequately.

2.2 Feature learning based methods

As we mentioned in Section 1, the key of feature learning based dynamic network link prediction is representation learning on networks. In this section, we review representation learning on static networks and dynamic networks.

2.2.1 Representation learning on static network. Matrix factorization based approaches[26, 46, 57] obtain node embeddings by exploring the spectral graph properties to perform dimensionality reduction. They have been proved to be effective in learning node embeddings. However, it is required to load the entire matrix into RAM for factorization, which requires a large amount of memory and computational resources, or even infeasible. Therefore, they cannot be directly applied to large-scale networks.

Random walk based approaches[9, 22, 35] are proposed to improve scalability. By performing fixed-length random walks, a network can be transformed into a collection of node sequences, in which skip-gram is utilized to lean node embeddings by maximizing the likelihood of co-occurrence in these sequences.

In recent years, deep learning-based approaches become more and more popular due to their improved embedding performance. Kipf et al.[15] first proposed graph convolution network (GCN) which utilizes a *message passing* mechanism to obtain node embeddings. After GCN, many *message passing* based graph neural network (GNN) architectures have been proposed, including simple graph convolution network (SGC)[54], GraphSAGE[10], graph attention network (GAT)[51], APPNP[16], FastGCN[1], EGNN[7], and GraphTransformer[42, 60].

2.2.2 Representation learning on dynamic network. Dynamic networks can be represented in both continuous and discrete forms, as discussed in Section 1.

Most existing continuous dynamic network embedding approaches can be classified into temporal random walks based approaches[24, 28, 33, 34], RNN based approaches[20, 25, 36], and temporal point processes based approaches[11, 18, 48]. Recently, some new continuous dynamic network embedding paradimgs are emerging. For example, HVGNN[44] use hyperbolic space and variational autoencoder to better model the hierarchical of graph structure and uncertainty in evolution. As discussed in Section 1, it is impractical to sense network

changes in the smallest time unit (e.g., second, day) in some real-world scenarios, and they cannot be applied to snapshots that lack precise time information.

Existing discrete dynamic network embedding approaches can be classified into two categories[39]: *temporal smoothness* approaches and *recurrent* approaches.

Temporal smoothness approaches are built on the assumption that the evolution of the network is smooth, and they guarantee the stability of embedding across consecutive time-steps[64, 67]. Zhu et al.[66] proposed a generalized eigen perturbation to incrementally update the results of generalized SVD, which is designed to preserve the high-order proximity, to incorporate the changes of dynamic networks. Goyal et al.[8] explored graph-autoencoders to incrementally build the embeddings of a snapshot from embeddings of the snapshot at the previous step. However, these methods cannot capture long-term changes in the network structure. More importantly, they lead to oversmoothing, thus cannot handle nodes that exhibit widely varying evolutionary behaviors.

Recurrent approaches normally combine GNNs and recurrent architectures, and they capture temporal dynamics by remaining hidden states[23, 40, 63]. Chen et al.[2] embedded GCN into LSTM cells to better integrate structural and temporal information. Pareja et al.[32] proposed EvolveGCN which uses the RNN to update the weights in the GCN model at each time step. Sankar et al.[39] proposed the joint self-attention on structural and temporal domains to better capture the structural and dynamic information at the same time.

BehaviorNet is a *recurrent* approach, and unlike existing methods, it preserves fine-grained behavior information during the aggregation operation to generate snapshots. **It is important to note that the fine-grained information used in BehaviorNet also belongs to a time interval, but such time interval is significantly finer-grained than the time interval of a snapshot.**

3 PRELIMINARIES

In this section, we introduce some preliminaries. We reserve bold capital letters for matrices and bold lowercase letters for vectors. We omit some superscripts and subscripts in case they do not lead to ambiguity. The symbols commonly used in this paper are listed in Appendix A for reference.

Definition 1 (Dynamic Networks). *A dynamic network is a sequence of discrete snapshots* $\mathcal{G} = \{G^{(1)}, G^{(2)}, \ldots, G^{(T)}\}$, *where* $G^{(t)} = \{\mathcal{V}, \mathcal{E}^{(t)}, \mathbf{A}^{(t)}\}$ *denotes a weighted undirected graph at time* t. $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ *denotes a set of nodes shared by all snapshots and* $\mathcal{E}^{(t)} = \{e_{i,j}^{(t)}\}$ *is the aggregated temporal links within fixed timespan* $[t-\tau, t]$, *where* τ *is a positive constant. Each edge* $e_{i,j}^{(t)} \in \mathcal{E}^{(t)}$ *is associated with a weight* $\mathbf{A}_{i,j}^{(t)} > 0$ *in the adjancy matrix* $\mathbf{A}^{(t)}$.

Definition 2 (Fine-grained Behavior in a Snapshot). *For a given snapshot* $G^{(t)}$ *generated within the timespan* $[t-\tau, t]$, *the timespan can be split into* K *equal length fine-grained time intervals. Then edge fine-grained behavior between nodes* i *and* j *can be denoted as* $\mathbf{e}_{i,j} = \{e_{i,j,1}, e_{i,j,2}, \ldots, e_{i,j,K}\}$, *where* $e_{i,j,k}$ *is the number of interactions between nodes* i *and* j *within the* k*-th fine-grained time intervals. Node fine-grained behavior* $\mathbf{b}_i^{(t)}$ *is an aggregation of its associated edges behavior and can be denoted as* $\mathbf{B}^{(t)} = \{\mathbf{b}_i^{(t)} | \sum_{j \in \mathcal{V}} \mathbf{e}_{i,j}^{(t)}, i \in \mathcal{V}\}$.

With Definition 1 and Definition 2, we can support the simulation of 5 types of fine-grained behaviors, including adding nodes, adding edges, dropping edges, node feature changes and edge feature changes. Currently, BehaviorNet does not support simulating dropping nodes.

For example, in Fig. 1, each snapshot is generated within $8\Delta t$, and this timespan is consist of 4 fine-grained time intervals. Then the edge behavior between node 1 and node 2 in the first snapshot is $\mathbf{e}_{1,2} = \{1, 1, 0, 0\}$, and the node behavior of node 1 is $\mathbf{b}_1 = \{2, 1, 1, 0\}$.

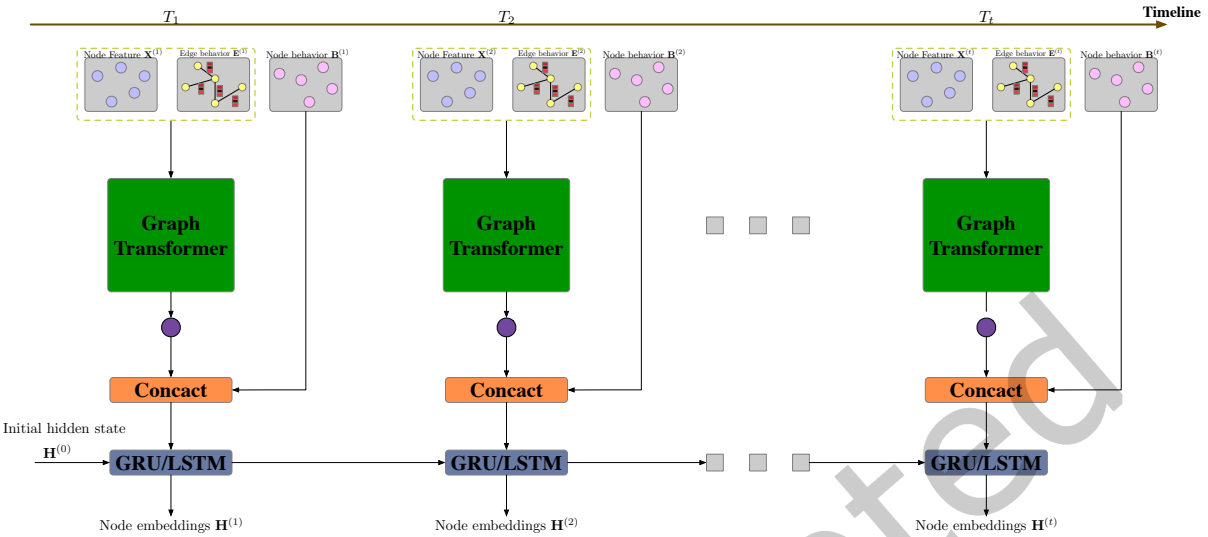


Fig. 2. The overall architecture of the fine-grained behavior aware network (BehaviorNet).

Definition 3 (Dynamic Link Prediction). *Given a sequence of snapshots with length T, $\mathcal{G} = \{G^{(t-T)}, G^{(t-T+1)}, \ldots, G^{(t)}\}$. The dynamic link prediction aims to predict the existence of an edge in the next time step $t+1$ based on the historical information of $\mathcal{G}$.*

4 THE PROPOSED METHOD

4.1 Overall architecture

Shown in Fig. 2, BehaviorNet consists of a structural encoder and a temporal encoder. The structural encoder is a variant of GraphTransformer, which can process edge behavior by treating it as an edge attribute. The temporal encoder is a recurrent network (e.g., LSTM, GRU), where node behaviors are processed and the evolution of nodes can be captured. Specifically, for the dynamic link prediction task, the fully-connected layers are used as a decoder. Note that the decoder part is not shown in Fig. 2 to keep the clarity.

4.2 Generation of fine-grained behavior features

As described in Definition 2, original edge behaviors can be represented as a K-dimension vector. Such representation is appropriate when the edge appears multiple times in the given timespan $[t-\tau, t]$ and is dispersed over K fine-grained time intervals.

However, we find that in some cases, edge behavior patterns are few in number and most edge behaviors are concentrated on one pattern. Such sparse features are challenging to extract valuable behavioral pattern information in the structural encoder because of overfitting. To tackle this problem, we treat the edge behavior vector as a function and use the moments of the function to characterize the behavior vector. Particularly, in this paper, we care about when the interaction is happened $\mathbf{k} = \{\frac{k}{K} | \mathbf{e}_k \neq 0, 0 \leq k \leq K\}$ and use the first three orders of moments to denote the transformed edge behavior:

$$\mathbf{e}' = \left[\mathbb{E}[\mathbf{k}^0], \mathbb{E}[\mathbf{k}^1], \mathbb{E}[\mathbf{k}^2]\right] \tag{1}$$

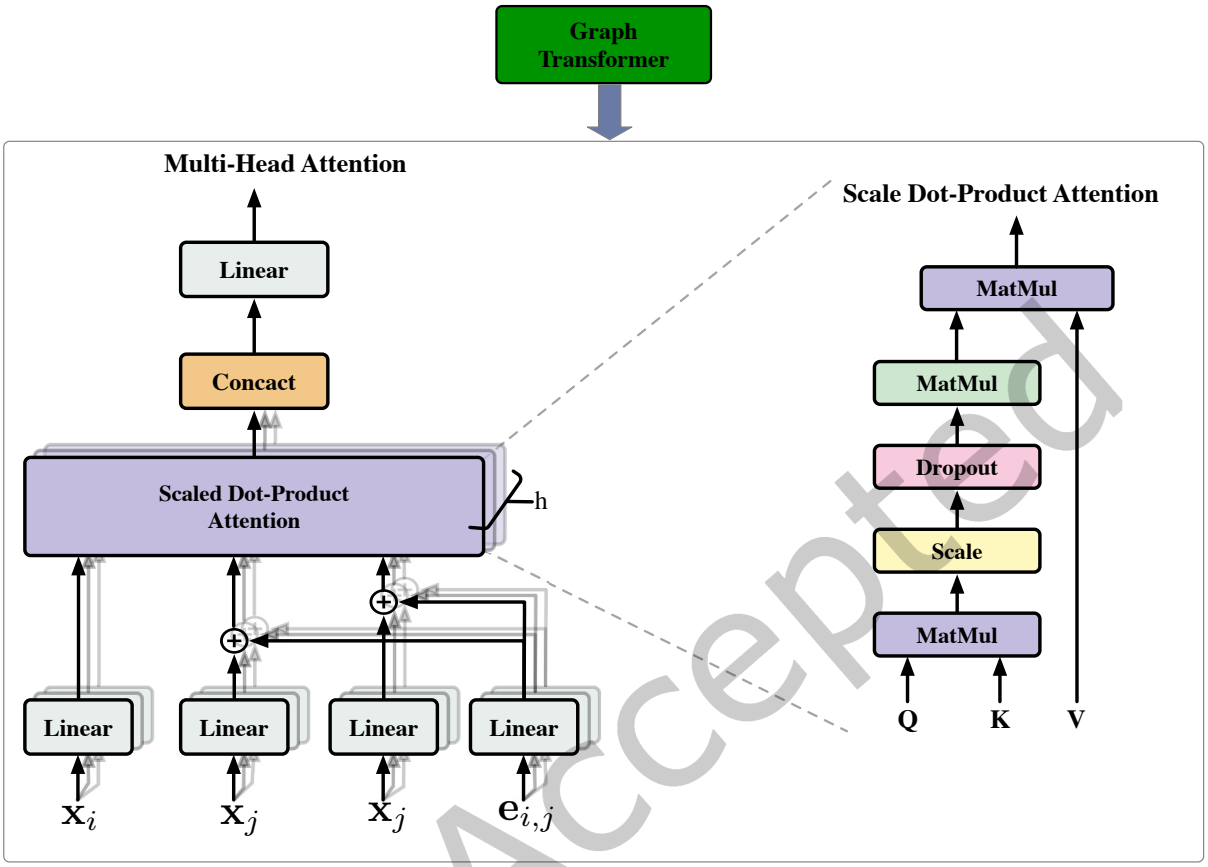


Fig. 3. The architecture of the Graph Transformer layer.

The main purpose of Eq. 1 is to preserve the distribution properties of the original edge behavior $\mathbf{e}$ using dense vectors. There could be many transformation methods for edge behaviors, but exploring the best transformation method is not the major task of this paper. For convenience, when the transformation of edge behaviors is needed, we uniformly use Eq. 1 considering its simplicity and effectiveness.

In the following sections, if not explicitly stated, the edge behaviors we refer to are transformed by Eq. 1 by default and denoted as $\mathbf{e}$. Node behaviors do not have a transformation step. They remain as an aggregation of the original edge behaviors, as described in Definition 2.

4.3 Structural encoder

The structural encoder is a variant of graph transformer[42] with taking into account the case of edge behaviors, and this architecture is illustrated in Fig. 3.

In practice, given a set of initial node features $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times D_n}$ and edge behavior $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times D_e}$, where D_n is the initial node features dimension and D_e is the edge behavior dimension. The structural encoder obtain a new set of node representations $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times F}$ with F dimensions through a message passing (neighborhood aggregation) scheme:

$$\mathbf{z}_i = \underbrace{\mathbf{W}_s \mathbf{x}_i}_{\text{self propagation}} + \overbrace{\sum_{j \in \mathcal{N}(i)} \alpha_{i,j} (\mathbf{W}_V \mathbf{x}_j + \mathbf{W}_E \mathbf{e}_{i,j})}^{\text{neighborhood aggregation}} \tag{2}$$

The first term in Eq. 2 is *self propagation*, which is designed to preserve the features of the node itself. This allows the output new node representation to have structural and semantic information when the initial node vector is generated from additional attributes (e.g., description of entities in the knowledge graph). $\mathbf{W}_s \in \mathbb{R}^{F \times D_n}$ is a trainable parameter. The second term in Eq. 2 is *neighborhood aggregation*, which aims to grab local structural information of node i by aggregating information from its immediate neighbors. Edge behavior is used in Eq. 2 to enhance structural information. Attention coefficients $\alpha_{i,j}$ are computed as follows:

$$\alpha_{i,j} = \frac{< \mathbf{W}_Q \mathbf{x}_i, \mathbf{W}_K \mathbf{x}_j + \mathbf{W}_E \mathbf{e}_{i,j} >}{\sum_{u \in \mathcal{N}(i)} < \mathbf{W}_Q \mathbf{x}_i, \mathbf{W}_K \mathbf{x}_u + \mathbf{W}_E \mathbf{e}_{i,u} >} \tag{3}$$

where $< \mathbf{q}, \mathbf{k} >= \exp(\frac{\mathbf{q}^T \mathbf{k}}{\sqrt{F}})$ is exponential scale dot-product function. $\mathbf{W}_Q \in \mathbb{R}^{F \times D_n}$ and $\mathbf{W}_K \in \mathbb{R}^{F \times D_n}$ are two learnable parameters that used to transform the source node feature $\mathbf{x}_i$ and target node feature $\mathbf{x}_j$ into query vector and key vector. Additionally, edge behavior $\mathbf{e}_{i,j}$ is encoded by $\mathbf{W}_E \in \mathbb{R}^{F \times D_e}$ and added into key vector as auxiliary information. $\mathbf{W}_V \in \mathbb{R}^{F \times D_n}$ also is a trainable matrix that used to transform node feature $\mathbf{x}_j$ into value vector.

To model the *multi-faceted* variations in dynamic networks, we follow DySAT[39] to introduce multi-head attention in the graph transformer. We use C headers to model C different facets, and concatenation is used to compute output representation:

$$\mathbf{z}_i = \text{Concat}(\mathbf{z}_i^1, \mathbf{z}_i^2, \ldots, \mathbf{z}_i^C) \tag{4}$$

A single graph transformer layer can only capture first-order proximity. One can capture high-order proximity by stacking more layers. However, experience shows that deep graph neural networks always encounter the problem of oversmoothing[58, 62]. There are some tricks that can be used to alleviate the oversmoothing problem, such as gated residual connection[3, 21, 42], but in our experiments, we found that these tricks are not stable, and the performance of shallow networks is no worse or even better than that of deep networks. In this paper, we stack two graph transformer layers to capture second-order proximity and without using gated residual connection between layers.

4.4 Temporal encoder

The temporal encoder layer is used to capture the temporal evolutionary patterns in a dynamic network. The input of the temporal encoder layer is a sequence of node representations $\{\mathbf{Z}^{(t-T)}, \ldots, \mathbf{Z}^{(t)}\}$ obtained by structural encoder and a sequence of node behaviors $\{\mathbf{B}^{(t-T)}, \ldots, \mathbf{B}^{(t)}\}$ at different time steps, where $\mathbf{B} \in \mathbb{R}^{|\mathcal{V}| \times K}$.

We first conduct a mixture node feature to capture both its structural information and its fine-grained behavior pattern information by concatenating node representation and node behavior:

$$\mathbf{N}^{(t)} = \text{Concat}(\mathbf{Z}^{(t)}, \mathbf{B}^{(t)}) \tag{5}$$

The mixture node features are packed together across time to get the input representation $\mathbf{N} \in \mathbb{R}^{|\mathcal{V}| \times T \times (F+K)}$. The output of the temporal encoder is a new node representation sequences at each time step $\{\mathbf{H}^{(t-T)}, \mathbf{H}^{(t-T+1)}, \ldots, \mathbf{H}^{(t)}\}, \mathbf{H}^{(t)} \in \mathbb{R}^{|\mathcal{V}| \times F'}$. Noted only the last time step node representation $\mathbf{H}^{(t)}$ will be used for dynamic link prediction.

Gated Recurrent Unit (GRU)[4] is employed in the temporal encoder:

$$\mathbf{C}^{(t)} = \text{sigmoid}(\mathbf{W}_{nc}\mathbf{N}^{(t)} + \mathbf{W}_{hc}\mathbf{H}^{(t-1)} + \mathbf{a}_C) \tag{6}$$

$$\mathbf{R}^{(t)} = \text{sigmoid}(\mathbf{W}_{nr}\mathbf{N}^{(t)} + \mathbf{W}_{hr}\mathbf{H}^{(t-1)} + \mathbf{a}_R) \tag{7}$$

$$\tilde{\mathbf{H}}^{(t)} = \tanh(\mathbf{W}_{nh}\mathbf{N}^{(t)} + \mathbf{R}^{(t)} * (\mathbf{W}_{hh}\mathbf{H}^{(t-1)} + \mathbf{a}_H)) \tag{8}$$

$$\mathbf{H}^{(t)} = (\mathbf{1} - \mathbf{R}^{(t)})\tilde{\mathbf{H}}^{(t)} + \mathbf{R}^{(t)}\mathbf{H}^{(t-1)} \tag{9}$$

where $\mathbf{C}^{(t)}$, $\mathbf{R}^{(t)}$, $\tilde{\mathbf{H}}^{(t)}$ are the reset, update, and new gates, respectively. $\mathbf{W}_{n\cdot} \in \mathbb{R}^{F' \times (F+K)}$ and $\mathbf{W}_{h\cdot} \in \mathbb{R}^{F' \times F'}$ are trainable parameters. $\mathbf{a}$ is bias. $*$ is the Hadamard product. Noted we set the initial hidden representation $\mathbf{H}^{(0)} = \mathbf{0}$.

4.5 Dynamic link prediction decoder and loss function

The dynamic link prediction decoder predict the existence of edge $e_{i,j}$ at time step $t+1$ based on historical node representations $\mathbf{h}_i^{(t)}$ and $\mathbf{h}_j^{(t)}$. We first concatenate these two vectors and then apply a two-layer MLP, where ReLU is used as activation function between fully connected layers and sigmoid function is applied on the last layer output, to obtain the edge existence probability:

$$\hat{y}_{i,j}^{(t+1)} = \text{MLP}(Concat(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)})) \tag{10}$$

We optimize a binary cross-entropy loss function:

$$\mathcal{L} = \sum_{i \in \mathcal{V}} \left(\sum_{j \in \mathcal{N}^{(t+1)}(i)} -\log(\hat{y}_{i,j}^{(t+1)}) - \sum_{j' \in \mathcal{P}^{(t+1)}(i)} \log(1 - \hat{y}_{i,j'}^{(t+1)}) \right) \tag{11}$$

where $\mathcal{P}^{(t+1)}(i)$ is a negative sampling of non-neighboring nodes of i. Empirically, we set $|\mathcal{P}^{(t+1)}(i)| = 5|\mathcal{N}^{(t+1)}(i)|$[48] to balance the positive and negative samples.

4.6 Complexity Analysis

For the generation of fine-grained behavior features, we only need to iterate over all edges in each snapshot, this operation can be processed just once before the data is loaded, and the time consumed can be neglected during the model training.

For the structural encoder, the time complexity of the graph transformer layers is $O(Tl|\mathcal{E}|_{max})$, where l is the number of stacked graph transformer layers and $|\mathcal{E}|_{max}$ is the maximal edge number of all snapshots. As we can parallel process each each snapshot in the time window, the time complexity can be reduce to $O(l|\mathcal{E}|_{max})$.

For the temporal encoder, the time consumption is mainly from GRU, whose time complexity is $O(|\mathcal{V}| \times (F + K + F'))$.

Overall, the time complexity of BehaviorNet is linear to the edge number or the node number (depends on the graph density), which indicates BehaviorNet can be applied to large-scale graphs.

5 EXPERIMENTS

In this section, we present a comprehensive set of experiments to demonstrate the effectiveness of BehaviorNet.

5.1 Datasets

We use six publicly available benchmark datasets for experiments. All datasets were collected by Rossi et al.[37] and placed on the website http://networkrepository.com. The datasets are described as follows:

Table 1. Basic statistics of six datasets.

	#Nodes	#Edges	#Samples (Train/Val/Test)
HYPERTEXT	113	20,815	6/2/1
ENRON	151	43,007	6/2/1
EMAIL-DNC	1,484	29,208	6/2/1
FB-MESSAGES	1,595	44,273	6/2/1
UCI	1,899	42,979	6/2/1
SOC	3,704	23,797	6/2/1

- **HYPERTEXT**[12]. HYPERTEXT is a face-to-face proximity human contact network. The dataset was collected from the wireless devices carried by the participants of the corresponding events during the ACM Hypertext 2009 conference. If two people have a conversation during a certain time interval, then a corresponding edge is generated.
- **ENRON**[17]. ENRON is an email network. Each node represents an employee of ENRON, and a link occurs each time denotes an email sent from one employee to another.
- **EMAIL-DNC**[29]. EMAIL-DNC is an email network in the 2016 Democratic National Committee email leak. Nodes in the network correspond to persons in the dataset. An edge in the dataset denotes that a person has sent an email to another person.
- **FB-MESSAGES**[30]. FB-MESSAGES is a Facebook-like social network originating from an online community for students at the University of California, Irvine. Nodes in the network correspond to users in the dataset. An edge in the dataset denotes that one user has sent/received at least one message from another users.
- **UCI**[31]. UCI is an online community of students from the University of California, Irvine. Edges in the UCI indicate messages sent or received between users.
- **SOC**[19]. SOC is a who-trusts-whom network of people who trade using Bitcoin on the *Bitcoin Alpha*[1].

The basic statistics of these datasets are summarized in Table 1. For convenience, all datasets are divided into 12 equal-length snapshots, and each snapshot is internally divided into 50 fine-grained time intervals, which means that the value of K is set to 50. The pseudo code for generating snapshots is given in Appendix B. The training/validation/testing split is carried out along the time dimension. We use continuous 4 snapshots as one sample, where the first 3 snapshots are used as input, and the last snapshot is used as the target, equal to set time length $T = 3$.

5.2 Baselines

We employ the following dynamic network link prediction methods as baselines:

- **CTGCN**[23]: CTGCN proposes a k-core based temporal graph convolutional network, which can preserve both local connective proximity and global structural similarity while simultaneously capturing network dynamics.
- **DyGrAE**[45]: DyGrAE uses gated graph neural networks to learn the topology of snapshots and applies an encoder-decoder framework to process temporal information.

[1]http://www.btc-alpha.com

- **DySAT**[39]: DySAT computes node representation through joint self-attention along structural neighborhood dimension and temporal dynamic dimension, and uses multi-head attention to model multi-faceted variations in network structures.
- **EvolveGCN**[32]: EvolveGCN is a dynamic variant of GCN, which captures the dynamism of the network sequence by using an RNN to evolve the GCN parameters. There are two versions of EvolveGCN: EvolveGCN-H and EvolveGCN-O.
- **GCLSTM**[2]: GCLSTM proposes an end-to-end model with a GCN embedded LSTM. GCN is applied to capture the local structure, and LSTM is adopted as the main framework to learn temporal features of all snapshots of a dynamic network.

We use one-hot degree-based node feature matrices for all baselines and our proposed BehaviorNet as the initial node representation. We adopt the Adam optimizer[14] with learning rate of 0.001 and weight decay of 10^{-5}. For a fair comparison, the dimensionality of final node embedding F' is set to 128 for all methods and stack two layer structure information encoder. We set them according to the suggested values in the corresponding papers for the other parameters involved in baselines, Appendix C provide more implement details. It should be noted that when comparing with the baselines, we don't search for the optimal hyperparameters in BehaviorNet. The optimization of hyperparameters in BehaviorNet will be discussed in Section. 5.7.

5.3 Dynamic link prediction results

Follow EvolveGCN [32], evaluation metrics used in this paper include mean average precision (MAP), mean reciprocal rank (MRR) and HITS@5. All methods were performed in 10 independent experiments on all datasets and we report the mean results in Table. 2. Noted BehaviorNet is the version that uses the edge behavior transformation as described in Section 4.2 and BehaviorNet-NT is the version that does not use.

From Table 2, we find that overall our methods BehaviorNet and Behavior-NT are significantly better than the other baselines in all metrics, while DySAT and GCLSTM perform the worst. However, under dataset UCI, GCLSTM obtained the optimal performance on all metrics, but noted that all methods performed poorly on both UCI and FB-MEAASGES datasets, and the performance gap between all methods on these two datasets is actually small. On the other four datasets, BehaviorNet and its variants significantly outperformed the other baselines, especially on the EMAIL-DNC dataset, BehaviorNet improved **0.0129**, **0.097** and **0.1030** on HIT@5, MAP and MRR, respectively, over DyGrAE, which obtained the best performance among all baselines.

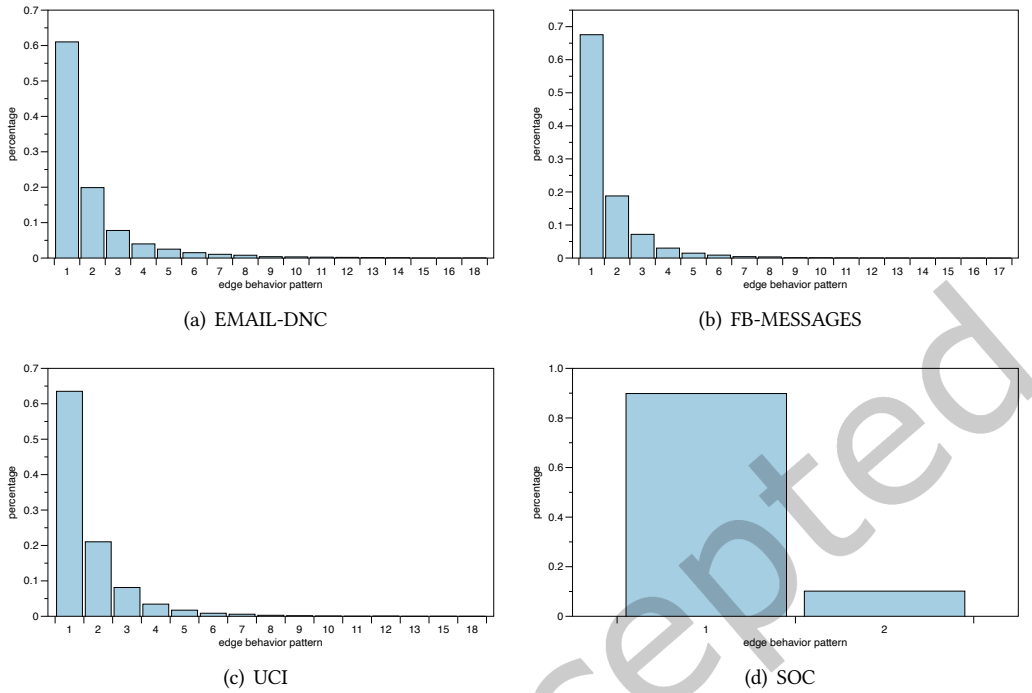
CTGCN and DyGrAE outperform the remaining baselines overall. CTGCN achieves this by introducing additional k-core graphs to enhance the representation of structural information, while DyGrAE is achieved by using a bidirectional LSTM encoder-decoder architecture to enhance the temporal information. BehaviorNet enhances structural and temporal information through edge behavior and node behavior, respectively.

Comparing BehaviorNet and BehaviorNet-NT, we find that BehaviorNet outperforms BehaviorNet-NT on EMAIL-DNC, FB-MESSAGES and UCI. While BehaviorNet-NT performs better on SOC. The main reason for this is the diversity and balance difference in edge behavior patterns in the dataset, as described in Section 4.2. To further demonstrate this point, we performed statistics on datasets. The results are presented in Fig. 4, where the x-axis is the different edge behavior patterns denoted by the number of non-zero elements in the edge behavior vector and y-axis is the percentage of each behavior pattern. As we can see that The edge behavior patterns in the SOC are few and concentrated, while the other three datasets are diverse and unbalanced. These phenomena prove our point in Section 4.2 that sparse features are challenging to extract valuablebehavioral pattern information in the structural encoder because of overfitting.

In addition, we also compare the number of model params for each model, the results are shown in Fig. 5. DySAT has the lowest and most stable number of parameters, which is almost not affected by the size of the dataset, at the cost of the worst performance of all the baselines in most cases. CTGCN, DyGrAE and EvolveGCN

Table 2. Dynamic network link prediction performances. The best result is marked in **bold** and the second best result is marked with underline. * indicates that BehaviorNet or BehaviorNet-NT significantly outperforms the best baseline based on one-tail *t-test* ($p\text{-}value < 0.05$)

		CTGCN	DyGrAE	DySAT	EvolveGCN-H	EvolveGCN-O	GCLSTM	BehaviorNet	BehaviorNet-NT
EMAIL-DNC	HITS@5	0.1227±0.0022	0.1710±0.0027	0.0740±0.0004	0.0475±0.0000	0.0475±0.0000	0.0859±0.0000	**0.1839*** ± 0.0002	0.1740±0.0024
	MAP	0.1546±0.0005	0.2086±0.0000	0.0878±0.0002	0.0504±0.0000	0.0469±0.0000	0.1132±0.0000	**0.3056*** ± 0.0050	0.2095±0.0003
	MRR	0.1831±0.0008	0.2530±0.0000	0.1368±0.0003	0.0668±0.0000	0.0635±0.0000	0.1705±0.0000	**0.3533*** ± 0.0065	0.2735* ± 0.0002
FB-MESSAGES	HITS@5	0.0281±0.0000	0.0171±0.0000	0.0111±0.0000	0.0317±0.0000	0.0317±0.0000	0.0152±0.0000	**0.0320*** ± 0.0000	0.0299±0.0000
	MAP	**0.0374**±0.0000	0.0262±0.0000	0.0163±0.0000	0.0295±0.0000	0.0297±0.0000	0.0260±0.0000	0.0315±0.0000	0.0259±0.0000
	MRR	0.0627±0.0000	0.0452±0.0000	0.0269±0.0000	0.0561±0.0000	0.0563±0.0000	0.0448±0.0000	**0.0641*** ± 0.0000	0.0469±0.0000
UCI	HITS@5	0.0287±0.0000	0.0168±0.0000	0.0224±0.0000	0.0293±0.0000	0.0317±0.0000	**0.0451**±0.0000	0.0335±0.0000	0.0116±0.0000
	MAP	0.0402±0.0000	0.0244±0.0000	0.0288±0.0000	0.0343±0.0000	0.0346±0.0000	**0.0430**±0.0000	0.0357±0.0000	0.0234±0.0000
	MRR	0.0689±0.0000	0.0446±0.0000	0.0495±0.0000	0.0608±0.0000	0.0614±0.0000	**0.0757**±0.0000	0.0593±0.0000	0.0388±0.0000
ENRON	HITS@5	0.0912±0.0004	0.0981±0.0001	0.0628±0.0000	0.0862±0.0000	0.0862±0.0000	0.0791±0.0000	**0.1042*** ± 0.0001	0.1012±0.0001
	MAP	0.1803±0.0006	0.1709±0.0001	0.1346±0.0000	0.1444±0.0000	0.1449±0.0000	0.1327±0.0000	0.1772* ± 0.0001	**0.1840*** ± 0.0001
	MRR	0.2996±0.0022	0.3342±0.0005	0.2268±0.0007	0.2880±0.0000	0.2860±0.0000	0.2593±0.0000	**0.3704*** ± 0.0013	0.3486* ± 0.0002
HYPERTEXT	HITS@5	0.0682±0.0000	0.0577±0.0000	0.0570±0.0000	0.0805±0.0000	0.0861±0.0000	0.0280±0.0000	**0.0878*** ± 0.0001	0.0805±0.0000
	MAP	0.1516±0.0000	0.1466±0.0000	0.1553±0.0000	0.1871±0.0000	0.1853±0.0000	0.1313±0.0000	0.1713±0.0000	**0.1890*** ± 0.0000
	MRR	0.1995±0.0000	0.2998±0.0017	0.2497±0.0000	0.3109±0.0000	0.3098±0.0000	0.2079±0.0000	0.3434* ± 0.0012	**0.3510*** ± 0.0000
SOC	HITS@5	0.0747±0.0000	0.0468±0.0000	**0.0757**±0.0000	0.0518±0.0000	0.0518±0.0000	0.0667±0.0001	0.0747±0.0000	0.0753±0.0000
	MAP	0.1089±0.0000	0.0800±0.0000	0.0812±0.0000	0.0818±0.0000	0.0817±0.0000	0.0729±0.0000	0.1070±0.0000	**0.1128*** ± 0.0000
	MRR	0.1842±0.0000	0.1124±0.0000	0.1406±0.0000	0.1237±0.0000	0.1235±0.0000	0.1131±0.0000	0.1800±0.0000	**0.1908*** ± 0.0000
Overall	HITS@5	0.0689	0.0679	0.0505	0.0545	0.0558	0.0533	**0.0860**	0.0788
	MAP	0.1122	0.1094	0.0840	0.0879	0.0872	0.0865	**0.1380**	0.1241
	MRR	0.1663	0.1815	0.1384	0.1510	0.1501	0.1452	**0.2284**	0.2083



(a) EMAIL-DNC (b) FB-MESSAGES

(c) UCI (d) SOC

Fig. 4. Edge behavior patterns distribution on EMAIL-DNC, FB-MESSAGES, UCI and SOC

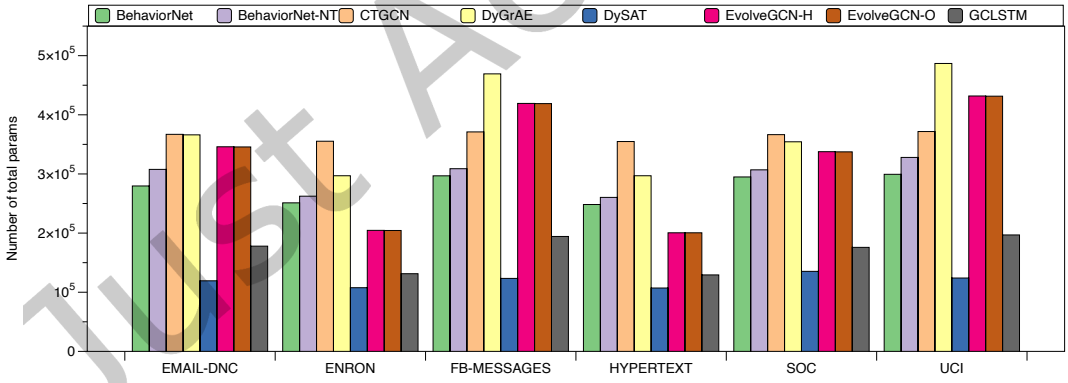
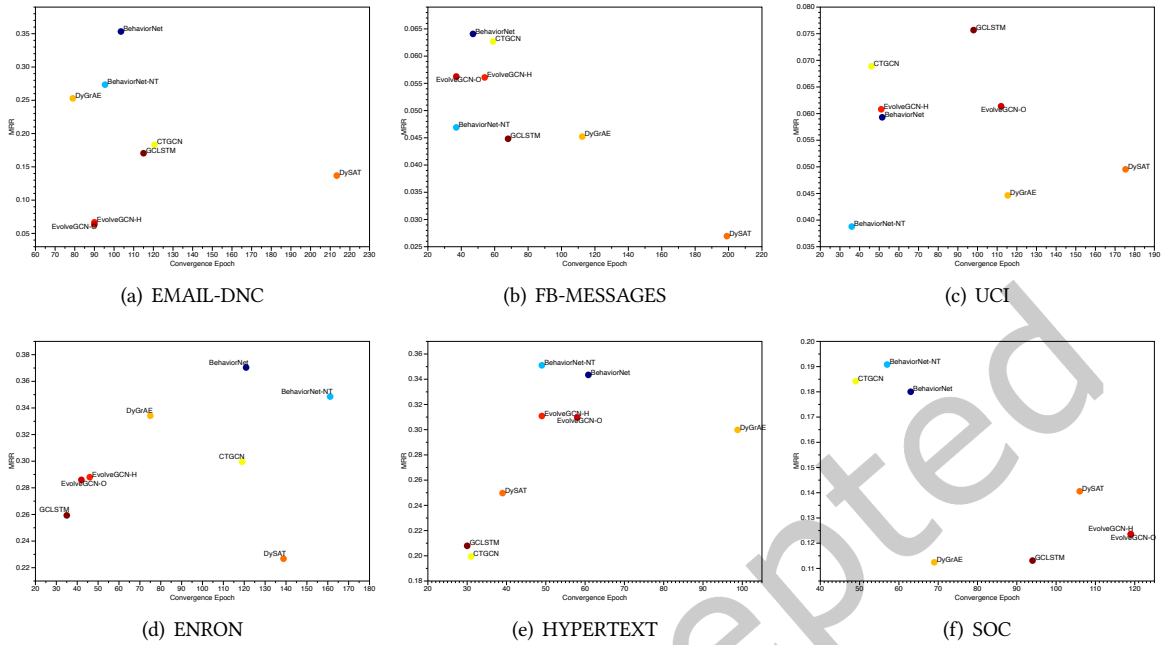


Fig. 5. Comparison of the number of model params for each model on each dataset.

typically require more parameters and are sensitive to the dataset size. The number of parameters in BehaviorNet is intermediate between all baselines and is insensitive to the size of the dataset, which indicates that BehaviorNet is scalable. Additionally, as most models (except DySAT) use RNN-like structure as temporal encoder, which needs to maintain a node number $|\mathcal{V}|$ related hidden embeddings. Thus, these models have difference in number of params on different datasets.



(a) EMAIL-DNC (b) FB-MESSAGES (c) UCI

(d) ENRON (e) HYPERTEXT (f) SOC

Fig. 6. Comparison of the number of epochs to converge of different models on the six datasets.

5.4 Model convergence comparison

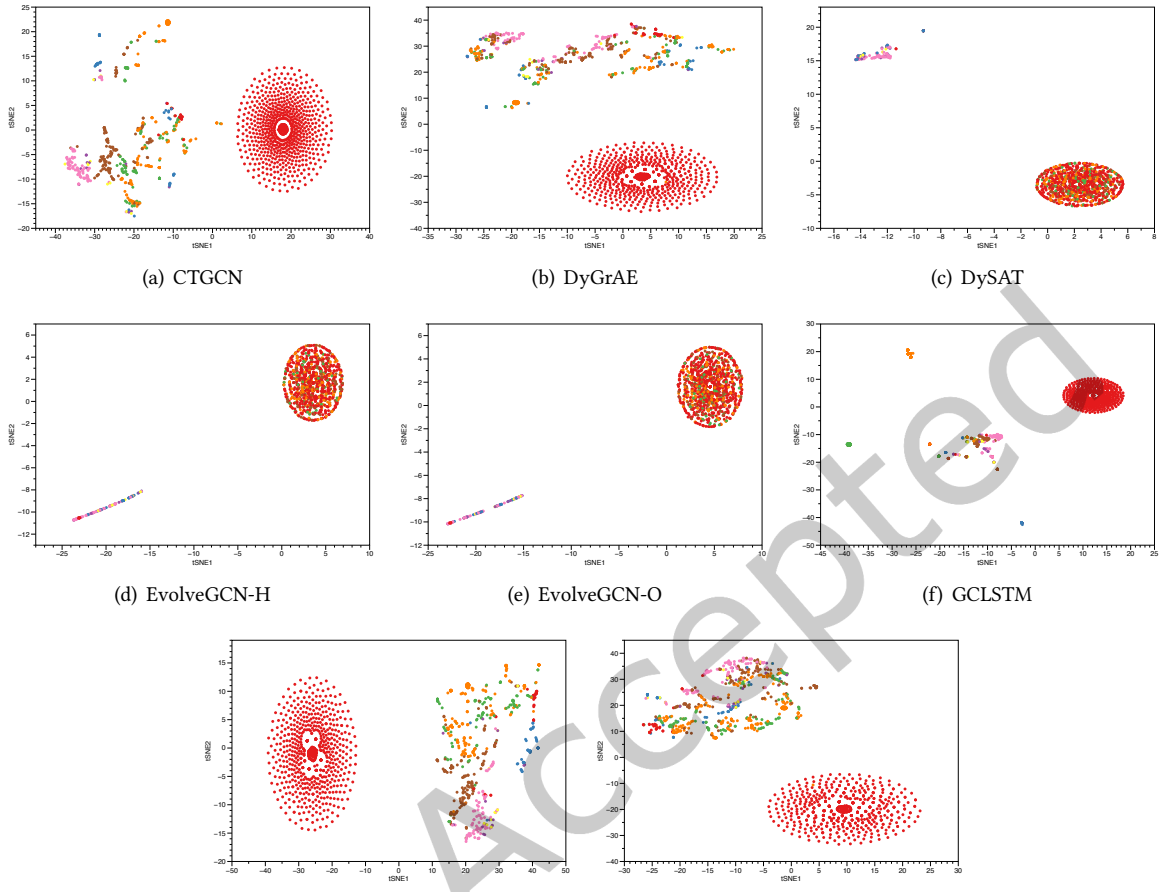
We compare the convergence speed of different models in this section. Since our experiments were conducted on a shared server with many other running jobs, the model's running time per epoch fluctuated greatly, so we chose the number of epochs of convergence for comparison rather than the running time per epoch.

Fig. 6 shows the convergence epochs of different models on the six datasets, where the horizontal axis indicates the number of epochs to converge and the vertical axis indicates the MRR value. DySAT usually requires more epochs to converge than other methods, except on the smallest HYPERTEXT dataset. The other methods perform differently on different datasets. BehaviorNet and its variant BehaviorNet-NT generally require more training epochs to converge than baselines on small-scale networks but converge faster than most baselines on large-scale datasets. It should be emphasized that BehaviorNet performs significantly better than baselines. We also observed that BehaviorNet-NT always needs fewer convergence epochs than BehaviorNet, which is caused by the fact that BehaviorNet-NT and BehaviorNet have the same architecture, but BehaviorNet-NT uses more hidden units for handling edge behavior (Eq. 2 and Eq. 3)[55].

5.5 Visualization

In this section, we explore the differences in the node representations obtained by different models through visualization. We utilize t-distributed stochastic neighbor embedding (tSNE) [49] to project node representations of a dynamic network from a high dimension into a 2D space. The visualization results are presented in Fig. 7.

From Fig. 7, we can find that EvolveGCN (Fig. 7(d) and Fig. 7(e)) and DySAT (Fig. 7(c)) fail to separate the inactive nodes (**000**) from the active nodes, while other methods make a clear distinction between these nodes. This is the reason why EvolveGCN and DySAT perform much lower than other models on EMAIL-DNC dataset.



(a) CTGCN (b) DyGrAE (c) DySAT

(d) EvolveGCN-H (e) EvolveGCN-O (f) GCLSTM

(g) BehaviorNet (h) BehaviorNet-NT

Fig. 7. tSNE visualization of node embeddings on the EMAIL-DNC dataset. We use a vector $\mathbf{v}$ of length T to represent the historical interaction of a node, where $\mathbf{v}_t = 0$ means that it did not interact with other nodes at time step t and $\mathbf{v}_t = 1$ means it interacted with other nodes at time step t. The color assignments in the figure are: **000**, **001**, **010**, **011**, **100**, **101**, **110**,**111**

Despite GCLSTM (Fig. 7(f)) distinguishes inactive nodes from active nodes, the difference between inactive nodes is not captured and the different types of active nodes are not distinguished. While CTGCN (Fig. 7(a)) and DyGrAE (Fig. 7(b)) are much improved compared to GCLSTM, they still do not distinguish well for nodes with activity at the most recent time step (**101**, **001** and **111**). The main reason is that CTGCN enhances on structural information by k-core graph, and DyGrAE enhances on temporal information by BiLSTM. BehaviorNet enhances both structural and temporal information of dynamic graph by edge behavior vector and node behavior vector, so that BehaviorNet is capable to distinguishing all type active nodes clearly, as shown in Fig. 7(h). The results demonstrate that BehaviorNet can capture behavior patterns in dynamic networks, which highly augments the expressive power of BehaviorNet.

5.6 Ablation Study

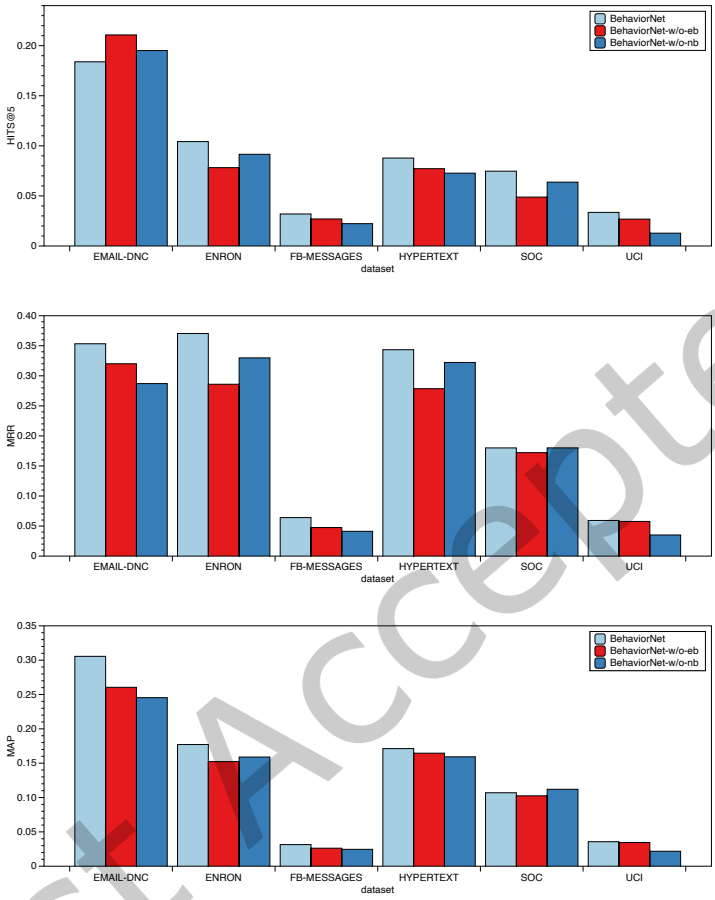


Fig. 8. Performance comparison for behavior vector ablation study.

BehaviorNet unifies two kinds of fine-grained behavioral information that contribute to its effectiveness in predicting future links in dynamic networks. We give insights on each kind of behavioral information and how it is indispensable to BehaviorNet by conducting an ablation study. We perform the following tests:

- **BehaviorNet-w/o-eb.** In this variant, we make Eq. 2 and Eq. 3 edge behavior independent (i.e. remove $\mathbf{W}_E\mathbf{e}_{i,j}$).
- **BehaviorNet-w/o-nb.** In this variant, we make Eq. 5 node behavior independent (i.e. remove $\mathbf{B}^{(t)}$).

The results are shown in Fig. 8. We can observe that BehaviorNet consistently outperforms BehaviorNet-w/o-eb and BehaviorNet-w/o-nb in all cases, except for the HIT@5 metric in the EMAIL-DNC dataset, where BehaviorNet is slightly worse than BehaviorNet-w/o-eb and BehaviorNet-w/o-nb. These experiments illustrate that fine-grained behavioral information can indeed improve the performance of dynamic link prediction.

Additionally, we also find that the removing edge behaviors lead to a significant performance drop for small networks (ENRON, HYPERTEXT) and sparse network (SOC), since these networks are mainly inadequate representations of structural information. While for denser networks (EMAIL-DNC, FB-MESSAGES, UCI), the removing node behaviors has a greater impact, because 1) temporal information is the main bottleneck in these networks; 2) The edge behaviors patterns inside these networks are much complex, and the transformation method introduced in Section. 4.2 not able to fully capture these edge behavior patterns. How to better transform the edge behaviors will become a priority in our future research.

BehaviorNet split the snapshot in τ into K equal intervals to generate behavior vectors, here we test a variant call **BehaviorNet-K** which directly use a more fine grained time interval $\frac{\tau}{K}$ for the overall sequence without introducing the behaviors.

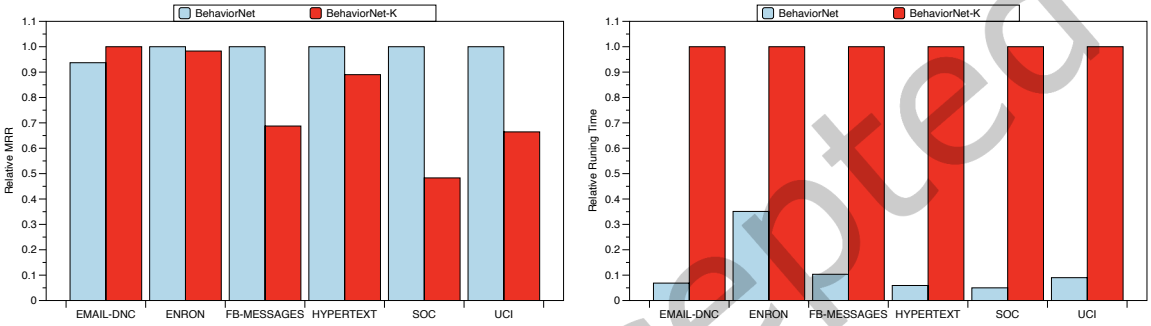


Fig. 9. Performance comparison for fine grained snapshots ablation study.

The results are shown in Fig. 9. Compared to BehaviorNet-K, BehaviorNet is more effective and efficient, especially on large graph(FB-MESSAGES, UCI and SOC). The main reasons are as follows:

(1) **Effectiveness**. When snapshots are generated using fine grained time intervals, each snapshot is very sparse. It is hard for structural encoder to capture the high-order structural information on such sparse snapshot.
(2) **Efficiency**. Using fine grained time intervals will generate more snapshots (K times) and further more training samples ($T - 3 \longrightarrow T \times K - 3$). This dramatic increase in samples leads to an increase in training time.

5.7 Parameter Sensitivity

The proposed BehaviorNet involves a number of parameters that may affect its performance. In this section we discuss the effects of different parameters, noting that when we discuss a certain parameter, we may uniformly modify the other parameters of the model to make it work correctly (e.g. node dimension and head number). In most case we only test the model performance on HYPERTEXT and EMAIL-DNC with 5 independent experiments to save the computation time.

5.7.1 Node Embedding Dimension F'. On the ENRON and HYPERTEXT dataset we select the node embedding dimension F' from {8, 16, 24, 32, 50, 64, 100, 128}, while on the other datasets we choice the node embedding dimension F' from {8, 16, 32, 64, 128, 256, 384, 512}. The dynamic link prediction results are shown in Fig. 10. From the figure we can find that higher node embedding dimension does not necessarily bring better performance, which is also consistent with the conclusion that only smaller nodes need to be selected to represent dimensionality in graph-related tasks in existing studies[47].

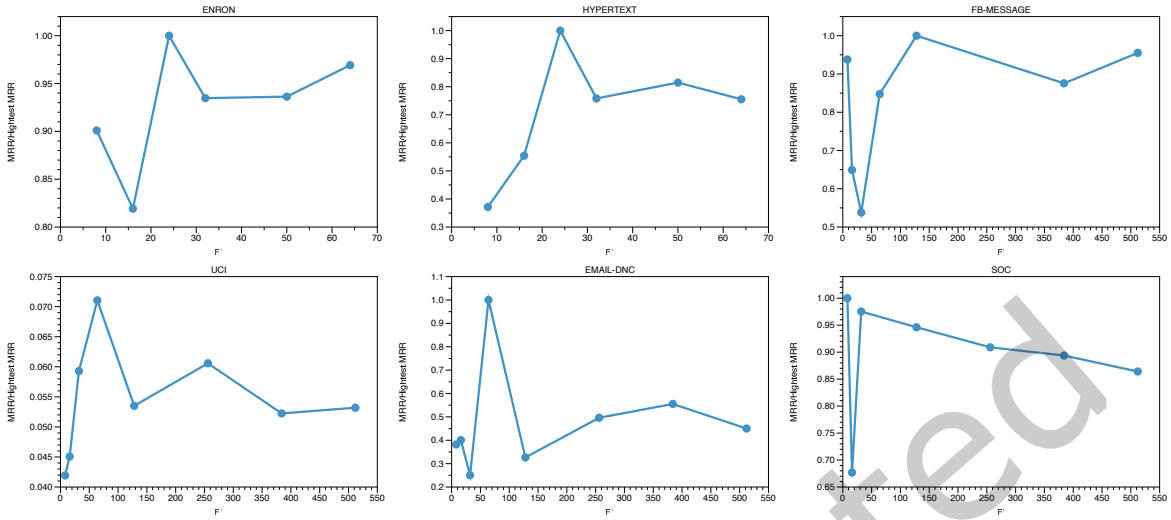


Fig. 10. Average MRR score with respect to the node embedding dimension F' on six datasets.

We also found an interesting phenomenon that the model achieves the best performance and far outperforms the other dimensions when the node embedding dimension F' is close to $\sqrt{4|\mathcal{V}|}$ on all datasets except for the SOC dataset, but the performance is also competitive in the SOC dataset when the node embedding dimension F' is close to $\sqrt{4|\mathcal{V}|}$. For example, the beset performance is achieved when $F' = 24$ on the HYPERTEXT dataset, where $\sqrt{4|\mathcal{V}|} = 21.45$. While the best performance is achieved when $F' = 64$ on the EMAIL-DNC dataset, where $\sqrt{4|\mathcal{V}|} = 77.05$.

The above experiments illustrate that BehaviorNet is sensitive to the node embedding dimension F', where a small dimension can lead to a significant performance degradation, while a large dimension can also lead to performance degradation and increase the required computational resources. Based on our experience, $\sqrt{4|\mathcal{V}|}$ can be used to guide the selection of the appropriate F'. It should be noted that this finding is only an empirical finding, and we may need more theories or more generalized scenarios to validate the correctness/formulation of this finding in our further work.

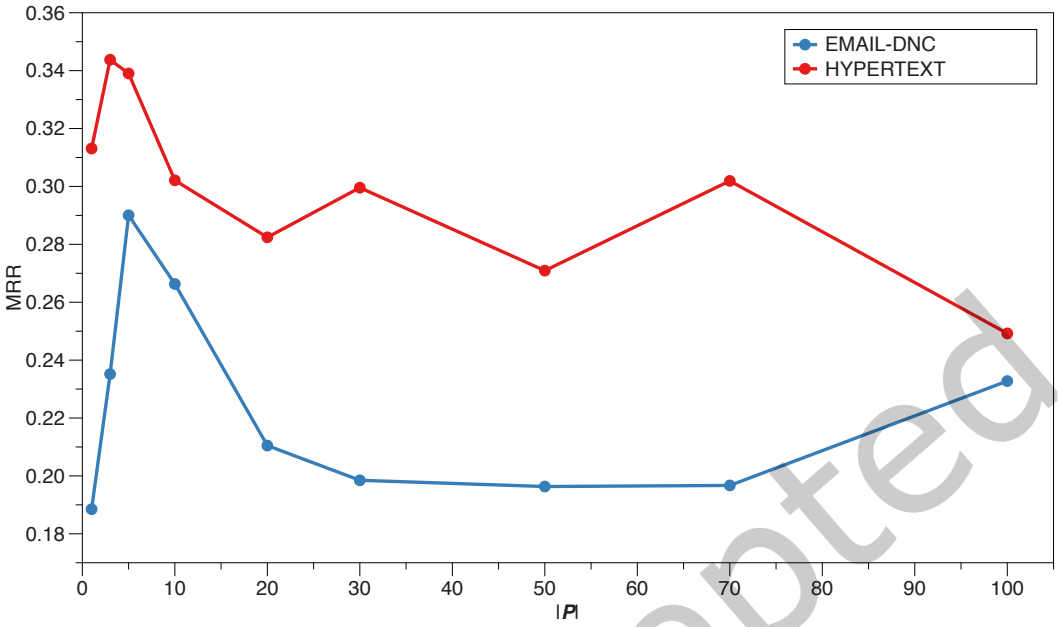


Fig. 11. Average MRR score with respect to the number of negative samples $|\mathcal{P}|$ on the HYPERTEXT and EMAIL-DNC dataset.

5.7.2 Number of Negative Sampling $|\mathcal{P}|$. This experiments focuses on analyzing the effect of choosing different ratios of non-neighboring nodes in Eq. 10. And we test the cases where the number of negative samples is {1,3,5,10,20,30,50,70,100} times that of positive samples.

As shown in Fig. 11, we can see that BehaviorNet is extremely sensitive with respect to the number of negative samples. The suitable range for the number of non-neighboring nodes is $3|\mathcal{N}| \leq |\mathcal{P}| \leq 5|\mathcal{N}|$. BehaviorNet cannot get enough information to distinguish non-existing edges form exist edges when $|\mathcal{P}|$ is too small, while when $|\mathcal{P}|$ is too large will lead to data imbalance and "overwhelming" the model with negative samples. Both of these cases lead to performance drops.

Moreover, there is an unavoidable problem with the negative sampling: it very difficult to guarantee random sampling of information-rich negative samples. How to overcome the negative impact of random negative sampling will be a important part of our further work. One possible solution is to use contrastive learning and adapt the negative-sampling-free contrastive loss function used for the node classification task[47, 56] to the dynamic link prediction task.

5.7.3 Temporal History Window Size T. In this section, we analyze how temporal history window size T affects BehaviorNet's performance. We test the values of T from 1 to 8, and the results are shown in Fig. 12.

We can find that the performance first increases as T increases, but there is a sharp drop in the model performance when T is too large. The initial increase is due to the richer temporal information brought by the increase of T. When T is too large, it introduces too old historical information that does not provider valid complementary information for link prediction, or even introduces noise. An appropriate temporal history window size is $3 \leq T \leq 5$.

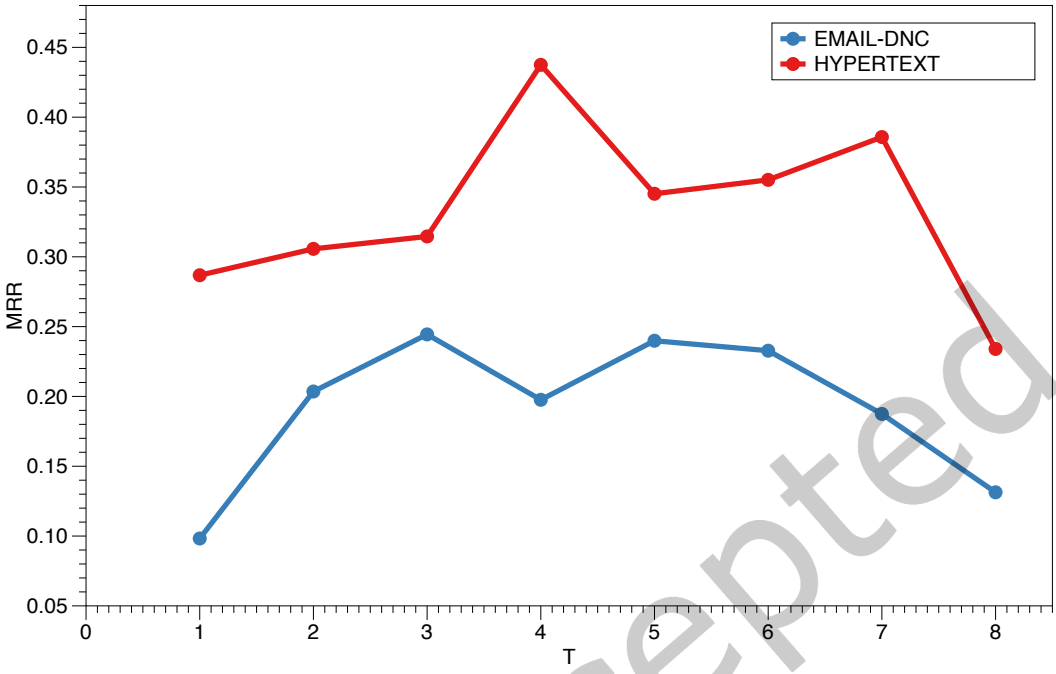


Fig. 12. Average MRR score with respect to the temporal history window size T on the HYPERTEXT and EMAIL-DNC dataset.

This phenomenon could also inspire some existing dynamic graph representation learning tasks, especially self-supervised dynamic network representation learning methods that use all historical information (from 0 to t): more history information is not necessarily better, and by picking the right temporal history time window size can lead to performance improvements.

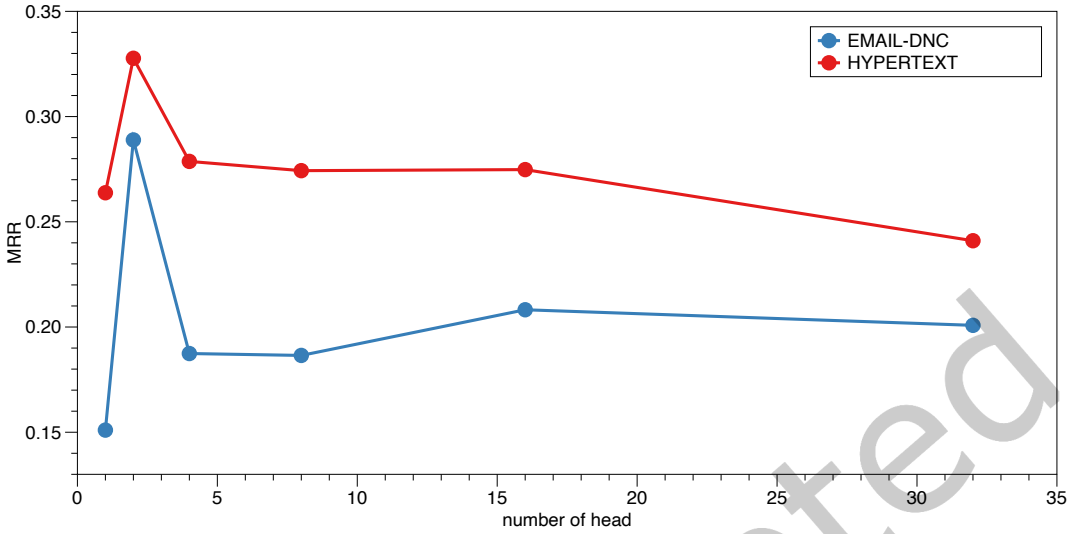


Fig. 13. Average MRR score with respect to the number of attention heads in the Graph Transformer on the HYPERTEXT and EMAIL-DNC dataset.

5.7.4 Attention Heads C. Finally, we analyze the benefits of multifaceted modeling in structural encoder via attention heads C, we vary the number of attention heads in Graph Transformer in structural encoder in the range {1, 2, 4, 8, 16, 32}, while keeping the output dimension fixed to 128 for fairness. The results are shown in Fig. 13.

From Fig. 13, we can observe that BehaviorNet benefits from suitable attention head numbers. The performance reaches its maximum when C is taken as 2. Further increase in the number of attention heads will make the performance drop sharply, which is different from DySAT[39], where the more attention heads there are, the higher the performance.

6 CONCLUSIONS

In this paper, we propose a fine-grained behavior- aware network for dynamic link prediction called BehaviorNet. The entire BehaviorNet consist of a structural encoder and a temporal encoder, where the structural encoder is implemented by a list of graph transformers and the temporal encoder is implemented by GRU. BehaviorNet can capture both edge behavior and node behavior to enhance structural and temporal information. Experimental results on several real-world dynamic networks demonstrate the effectiveness and efficiency of BehaviorNet. The ablation study proves that edge and node behaviors do have their unique values. We also discuss edge behavior transformation methods, parameter sensitivity of BehaviorNet, and find many challenges to be solved in the future, such as better and more general representation of edge behavior, negative sampling methods.

ACKNOWLEDGMENTS

The research in this paper is partially supported by the National Key Research and Development Program of China (No 2021YFB3300700), the National Natural Science Foundation of China (61832004, 61802089, 61832014) and the Fund project of Key Laboratory of space-based integrated information system (HLJGXQ20210701004).

A SYMBOLS

Table 3 list frequently-used symbols used in this paper and their meanings. We omit some superscripts and subscripts for clarity.

Table 3. Symbols used in this paper

Symbol	Description
$\mathcal{G}$	A sequence of discrete snapshots
$.^{(t)}$	Time t
G	A snapshot
$\mathcal{V}$	A set of shared nodes
$\mathcal{E}$	Edges in snapshot G
u, v, i, j	A node
$e_{i,j}$	An edge between node i and j
$\mathbf{A}$	Adjancy matrix
$\mathbf{e}_{i,j}$, $\mathbf{e}'_{i,j}$	Edge behavior vector of edge $e_{i,j}$
$\mathbf{b}_i$	Node behavior vector of node i
K	Number of fine-grained time intervals in a snapshot
$\mathbf{X}$	Initial node features
$\mathbf{Z}$	Node representations given by structural encoder
C	Head number in structural encoder
$\mathbf{H}$	Hidden representation in temporal encoder
$\mathbf{W}_\cdot, \mathbf{a}$	Trainable parameters
$\mathbf{C}$, $\mathbf{R}$, $\tilde{\mathbf{H}}$	The reset, update, and new gates.

B GENERATION OF SNAPSHOTS

Algorithm 1: Generation Algorithm for Snapshots

Input : Number of snapshot T ;
Temporal sorted interaction set $\mathcal{E} = \{(u, v, t)\}$;
Start time t_s and end time t_e of $\mathcal{E}$;
Number of fine-grained intervals K ;

Output : A set of snapshots $\mathcal{G}$;

$\mathcal{G} \leftarrow \{\}$
$\mathcal{N} \leftarrow \{\}$
for $t \leftarrow 0$ **to** T **do**
  $\mathcal{E}^{(t)} \leftarrow \{\}$; // The default value of edge $e_{u,v}^{(t)}$ in $\mathcal{E}^{(t)}$ is $(w = 0, \mathbf{e} = \mathbf{0})$
end
$\tau_s \leftarrow (t_s - t_e)/T$
$\tau_f \leftarrow \tau_s/K$
for ***each*** $(u, v, t) \in \mathcal{E}$ **do**
  $\mathcal{N} \leftarrow \mathcal{N} \cup \{u, v\}$
  $x_s = \lfloor \frac{t - t_s}{\tau_s} \rfloor$
  $x_f = \lfloor \frac{t - t_s - \tau_s x_s}{\tau_f} \rfloor$
  $e_{u,v,w}^{(x_s)} \leftarrow e_{u,v,w}^{(x_s)} + 1$
  $e_{u,v,\mathbf{e}}^{(x_s)} \leftarrow e_{u,v,\mathbf{e}}^{(x_s)} + \mathbb{I}(x_f)$ // $\mathbb{I}(\cdot)$ is indicator function with K dimension.
end
for $t \leftarrow 0$ **to** T **do**
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{G^{(t)} = (\mathcal{N}, \mathcal{E}^{(t)})\}$
end
return $\mathcal{G}$

The pseudo code for generating snapshots is given in Algorithm 1, where $\mathbb{I}(\cdot)$ is indicator function with K dimension:

$$\mathbb{I}_y(x) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{if } x \neq y. \end{cases} \tag{12}$$

C IMPLEMENT DETAILS

Table 4 show the initial one-hot degree-based node feature dimension D_n for each dataset.

Table 4. The Initial node feature **X** dimension on six datasets

	HYPERTEXT	ENRON	EMAIL-DNC	FB-MESSAGES	UCI	SOC
D_n	58	62	153	185	190	149

All experiments were conducted on the same TitanXP server with 12GB memory. We uniformly implemented all models using PyG[5], organized the experiments using hydra[2] and pytorch-lightning[3], and used W&B[4] to automatically collect log information including model size, running time, etc. The implementation details and detailed parameters for all baselines and BehaviorNet are listed as follows:

- **CTGCN**: We have reproduced it using PyG based on the official Pytorch code. We chose the variant CTGCN-S because it is insensitive to K-core numbers[23, Fig. 9]. Parameter values are: $max\_core\_num = 3$, $cgcn\_layer\_num = 2$.
- **DyGrAE**: We use the code provide by PyGT[38] with $conv\_num\_layers = 2$ and $conv\_aggr =$ mean.
- **DySAT**: We have reproduced it using PyG based on the official Tensorflow code. Based on discussion in [39, Fig. 5] and their official code, we stack 2 structural self-attention layer and set all self-attention layers (both structural and temporal) with 8 heads.
- **EvolveGCN**: We adopted the code provide by PyGT[38].
- **GCLSTM**: We use the code provide by PyGT[38] with stack 4 GCN layers and set Chebyshev polynomial in each GCN is 3[2, Section 4.1].
- **BehaviorNet**: We implement BehaviorNet using PyG. We stack 2 graph transformer layers and the head number in each graph transformer is 8 (directed use the best hyperparameters in DySAT.)

REFERENCES

[1] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=rytstxWAW

[2] Jinyin Chen, Xueke Wang, and Xuanheng Xu. 2021. GC-LSTM: graph convolution embedded LSTM for dynamic network link prediction. *Applied Intelligence* (2021), 1–16.

[3] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and Deep Graph Convolutional Networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 1725–1735. http://proceedings.mlr.press/v119/chen20v.html

[4] Rahul Dey and Fathi M. Salem. 2017. Gate-variants of Gated Recurrent Unit (GRU) neural networks. In *IEEE 60th International Midwest Symposium on Circuits and Systems, MWSCAS 2017, Boston, MA, USA, August 6-9, 2017*. IEEE, 1597–1600. https://doi.org/10.1109/MWSCAS.2017.8053243

[5] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[6] Jibing Gong, Shen Wang, Jinlong Wang, Wenzheng Feng, Hao Peng, Jie Tang, and Philip S Yu. 2020. Attentional graph convolutional networks for knowledge concept recommendation in moocs in a heterogeneous view. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 79–88.

[7] Liyu Gong and Qiang Cheng. 2019. Exploiting Edge Features for Graph Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 9211–9219. https://doi.org/10.1109/CVPR.2019.00943

[8] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. 2018. DynGEM: Deep Embedding Method for Dynamic Graphs. *CoRR* abs/1805.11273 (2018). arXiv:1805.11273 http://arxiv.org/abs/1805.11273

[9] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 855–864. https://doi.org/10.1145/2939672.2939754

[10] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 1024–1034. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[2]https://hydra.cc

[3]https://www.pytorchlightning.ai

[4]https://wandb.ai/site

[11] Zhen Han, Yunpu Ma, Yuyi Wang, Stephan Günnemann, and Volker Tresp. 2020. Graph Hawkes Neural Network for Forecasting on Temporal Knowledge Graphs. In *Conference on Automated Knowledge Base Construction, AKBC 2020, Virtual, June 22-24, 2020*, Dipanjan Das, Hannaneh Hajishirzi, Andrew McCallum, and Sameer Singh (Eds.). https://doi.org/10.24432/C50018

[12] Lorenzo Isella, Juliette Stehlé, Alain Barrat, Ciro Cattuto, Jean-François Pinton, and Wouter Van den Broeck. 2011. What's in a crowd? Analysis of face-to-face behavioral networks. *Journal of theoretical biology* 271, 1 (2011), 166–180.

[13] Mansooreh Kazemilari and Maman Abdurachman Djauhari. 2015. Correlation network analysis for multi-dimensional data in stocks market. *Physica A: Statistical Mechanics and its Applications* 429 (2015), 62–75.

[14] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[15] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. (2017). https://openreview.net/forum?id=SJU4ayYgl

[16] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=H1gL-2A9Ym

[17] Bryan Klimt and Yiming Yang. 2004. The Enron Corpus: A New Dataset for Email Classification Research. In *Machine Learning: ECML 2004, 15th European Conference on Machine Learning, Pisa, Italy, September 20-24, 2004, Proceedings (Lecture Notes in Computer Science, Vol. 3201)*, Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi (Eds.). Springer, 217–226. https://doi.org/10.1007/978-3-540-30115-8_22

[18] Boris Knyazev, Carolyn Augusta, and Graham W. Taylor. 2021. Learning temporal attention in dynamic graphs with bilinear interactions. *PLOS ONE* 16, 3 (03 2021), 1–18. https://doi.org/10.1371/journal.pone.0247936

[19] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and V. S. Subrahmanian. 2018. REV2: Fraudulent User Prediction in Rating Platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek (Eds.). ACM, 333–341. https://doi.org/10.1145/3159652.3159729

[20] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 1269–1278. https://doi.org/10.1145/3292500.3330895

[21] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. 2019. DeepGCNs: Can GCNs Go As Deep As CNNs?. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 9266–9275. https://doi.org/10.1109/ICCV.2019.00936

[22] Juzheng Li, Jun Zhu, and Bo Zhang. 2016. Discriminative Deep Random Walk for Network Classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics. https://doi.org/10.18653/v1/p16-1095

[23] J. Liu, C. Xu, C. Yin, W. Wu, and Y. Song. 2020. K-Core based Temporal Graph Convolutional Network for Dynamic Graphs. *IEEE Transactions on Knowledge and Data Engineering* 01 (2020), 1–1. https://doi.org/10.1109/TKDE.2020.3033829

[24] Zhining Liu, Dawei Zhou, Yada Zhu, Jinjie Gu, and Jingrui He. 2020. Towards Fine-Grained Temporal Network Representation via Time-Reinforced Random Walk. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 4973–4980. https://aaai.org/ojs/index.php/AAAI/article/view/5936

[25] Yao Ma, Ziyi Guo, Zhaochun Ren, Jiliang Tang, and Dawei Yin. 2020. Streaming Graph Neural Networks. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 719–728. https://doi.org/10.1145/3397271.3401092

[26] Nagarajan Natarajan and Inderjit S. Dhillon. 2014. Inductive matrix completion for predicting gene-disease associations. *Bioinform.* 30, 12 (2014), 60–68. https://doi.org/10.1093/bioinformatics/btu269

[27] Duc Duy Nguyen and Guo-Wei Wei. 2019. AGL-Score: Algebraic Graph Learning Score for Protein-Ligand Binding Scoring, Ranking, Docking, and Screening. *J. Chem. Inf. Model.* 59, 7 (2019), 3291–3304. https://doi.org/10.1021/acs.jcim.9b00334

[28] 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 of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis (Eds.). ACM, 969–976. https://doi.org/10.1145/3184558.3191526

[29] Caleb C. Noble and Diane J. Cook. 2003. Graph-based anomaly detection. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, Lise Getoor, Ted E. Senator, Pedro M.

Domingos, and Christos Faloutsos (Eds.). ACM, 631–636. https://doi.org/10.1145/956750.956831
[30] Tore Opsahl and Pietro Panzarasa. 2009. Clustering in weighted networks. *Soc. Networks* 31, 2 (2009), 155–163. https://doi.org/10.1016/j.socnet.2009.02.002
[31] 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.
[32] 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 *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 5363–5370. https://aaai.org/ojs/index.php/AAAI/article/view/5984
[33] Hao Peng, Jianxin Li, Hao Yan, Qiran Gong, Senzhang Wang, Lin Liu, Lihong Wang, and Xiang Ren. 2020. Dynamic network embedding via incremental skip-gram with negative sampling. *Science China Information Sciences* 63, 10 (2020), 1–19.
[34] Hao Peng, Renyu Yang, Zheng Wang, Jianxin Li, Lifang He, S Yu Philip, Albert Y Zomaya, and Rajiv Ranjan. 2021. Lime: Low-cost and incremental learning for dynamic heterogeneous information networks. *IEEE Trans. Comput.* 71, 3 (2021), 628–642.
[35] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani (Eds.). ACM, 701–710. https://doi.org/10.1145/2623330.2623732
[36] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael M. Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. *CoRR* abs/2006.10637. arXiv:2006.10637 https://arxiv.org/abs/2006.10637
[37] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, Blai Bonet and Sven Koenig (Eds.). AAAI Press, 4292–4293. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9553
[38] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzman Lopez, Nicolas Collignon, and Rik Sarkar. 2021. PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 4564–4573.
[39] 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 '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*, James Caverlee, Xia (Ben) Hu, Mounia Lalmas, and Wei Wang (Eds.). ACM, 519–527. https://doi.org/10.1145/3336191.3371845
[40] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured Sequence Modeling with Graph Convolutional Recurrent Networks. In *Neural Information Processing - 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 11301)*, Long Cheng, Andrew Chi-Sing Leung, and Seiichi Ozawa (Eds.). Springer, 362–373. https://doi.org/10.1007/978-3-030-04167-0_33
[41] Alex Sherstinsky. 2020. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena* 404 (2020), 132306.
[42] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. 2021. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. (2021), 1548–1554. https://doi.org/10.24963/ijcai.2021/214
[43] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. 2020. Foundations and modelling of dynamic networks using Dynamic Graph Neural Networks: A survey. *CoRR* abs/2005.07496 (2020). arXiv:2005.07496 https://arxiv.org/abs/2005.07496
[44] Li Sun, Zhongbao Zhang, Jiawei Zhang, Feiyang Wang, Hao Peng, Sen Su, and S Yu Philip. 2021. Hyperbolic variational graph neural network for modeling dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4375–4383.
[45] Aynaz Taheri and Tanya Y. Berger-Wolf. 2019. Predictive temporal embedding of dynamic graphs. In *ASONAM '19: International Conference on Advances in Social Networks Analysis and Mining, Vancouver, British Columbia, Canada, 27-30 August, 2019*, Francesca Spezzano, Wei Chen, and Xiaokui Xiao (Eds.). ACM, 57–64. https://doi.org/10.1145/3341161.3342872
[46] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, and Mohammed Javeed Zaki (Eds.). ACM, 817–826. https://doi.org/10.1145/1557019.1557109
[47] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Velickovic, and Michal Valko. 2021. Bootstrapped Representation Learning on Graphs. *CoRR* abs/2102.06514 (2021). arXiv:2102.06514 https://arxiv.org/abs/2102.06514
[48] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=HyePrhR5KX
[49] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[51] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=rJXMpikCZ

[52] Hui Wang and Zichun Le. 2020. Seven-Layer Model in Complex Networks Link Prediction: A Survey. *Sensors* 20, 22 (2020). https://doi.org/10.3390/s20226560

[53] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Trans. Knowl. Data Eng.* 29, 12 (2017), 2724–2743. https://doi.org/10.1109/TKDE.2017.2754499

[54] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 6861–6871. http://proceedings.mlr.press/v97/wu19e.html

[55] Xiurui Xie, Hong Qu, Guisong Liu, Malu Zhang, and Jürgen Kurths. 2016. An efficient supervised training algorithm for multilayer spiking neural networks. *PloS one* 11, 4 (2016), e0150329.

[56] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. 2021. InfoGCL: Information-Aware Graph Contrastive Learning. *Advances in Neural Information Processing Systems* 34 (2021).

[57] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. 2015. Network Representation Learning with Rich Text Information. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, Qiang Yang and Michael J. Wooldridge (Eds.). AAAI Press, 2111–2117. http://ijcai.org/Abstract/15/299

[58] Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek F. Abdelzaher. 2020. Revisiting "Over-smoothing" in Deep GCNs. *CoRR* abs/2003.13663 (2020). arXiv:2003.13663 https://arxiv.org/abs/2003.13663

[59] Zhihu Yang, Zhi Li, and Long Wang. 2020. Evolution of cooperation in a conformity-driven evolving dynamic social network. *Appl. Math. Comput.* 379 (2020), 125251. https://doi.org/10.1016/j.amc.2020.125251

[60] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. 2019. Graph Transformer Networks. (2019), 11960–11970. https://proceedings.neurips.cc/paper/2019/hash/9d63484abb477c97640154d40595a3bb-Abstract.html

[61] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. 2019. Graph convolutional networks: a comprehensive review. *Computational Social Networks* 6, 1 (2019), 1–23.

[62] Lingxiao Zhao and Leman Akoglu. 2020. PairNorm: Tackling Oversmoothing in GNNs. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=rkecl1rtwB

[63] Li Zheng, Zhenpeng Li, Jian Li, Zhao Li, and Jun Gao. 2019. AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, 4419–4425. https://doi.org/10.24963/ijcai.2019/614

[64] Le-kui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic Network Embedding by Modeling Triadic Closure Process. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 571–578. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16572

[65] Tao Zhou. 2021. Progresses and Challenges in Link Prediction. *CoRR* abs/2102.11472 (2021). arXiv:2102.11472 https://arxiv.org/abs/2102.11472

[66] Dingyuan Zhu, Peng Cui, Ziwei Zhang, Jian Pei, and Wenwu Zhu. 2018. High-Order Proximity Preserved Embedding for Dynamic Networks. *IEEE Trans. Knowl. Data Eng.* 30, 11 (2018), 2134–2144. https://doi.org/10.1109/TKDE.2018.2822283

[67] Linhong Zhu, Dong Guo, Junming Yin, Greg Ver Steeg, and Aram Galstyan. 2016. Scalable Temporal Latent Space Inference for Link Prediction in Dynamic Social Networks. *IEEE Trans. Knowl. Data Eng.* 28, 10 (2016), 2765–2777. https://doi.org/10.1109/TKDE.2016.2591009