






Sparse-Dyn: Sparse dynamic graph multirepresentation learning via event-based sparse temporal attention network

Yan Pang¹  | Ai Shan¹ | Zhen Wang²  |
Mengyu Wang³  | Jianwei Li⁴ | Ji Zhang⁵  |
Teng Huang¹ | Chao Liu⁶ 

¹Institute of Artificial Intelligence and Blockchain, Guangzhou University, Guangzhou, Guangdong, China

²Research Center for Big Data Intelligence, Zhejiang Lab, Hangzhou, Zhejiang, China

³Department of Information Science and Technology, Osaka University, Osaka, Japan

⁴Department of Computer Science, San Jose State University, San Jose, California, USA

⁵School of Mathematics, Physics and Computing, University of Southern Queensland, Toowoomba, Queensland, Australia

⁶Department of Electrical Engineering, University of Colorado Denver, Denver, Colorado, USA

Correspondence

Chao Liu, Department of Electrical Engineering, University of Colorado Denver, Denver, CO 80204, USA.
Email: chao.liu@ucdenver.edu

Abstract

Dynamic graph neural networks (DGNNs) have been widely used in modeling and representation learning of graph structure data. Current dynamic representation learning focuses on either discrete learning which results in temporal information loss, or continuous learning which involves heavy computation. In this study, we proposed a novel DGNN, sparse dynamic (Sparse-Dyn). It adaptively encodes temporal information into a sequence of patches with an equal amount of temporal-topological structure. Therefore, while avoiding using snapshots which cause information loss, it also achieves a finer time granularity, which is close to what continuous networks could provide. In addition, we also designed a lightweight module, Sparse Temporal Transformer, to compute node representations through structural neighborhoods and temporal dynamics. Since the fully connected attention conjunction is simplified, the computation cost is far lower than the current state-of-the-art. Link prediction experiments are conducted on both continuous and discrete graph data sets. By comparing several state-of-the-art graph embedding baselines, the experimental

results demonstrate that Sparse-Dyn has a faster inference speed while having competitive performance.

KEYWORDS

adaptive data encoding, dynamic graph neural network, link prediction, sparse temporal transformer

1 | INTRODUCTION

Dynamic graph neural networks (DGNNs) have seen a notable surge of interest with the encouraging technique for learning complicated systems of relations or interactions over time. Since DGNNs append an additional temporal dimension to accumulate the variation of embedding or representations, they are powerful tools to employ in diverse fields, such as social media,^{1,2} dynamic social networks,³ bioinformatics,⁴ knowledge bases,⁵ brain neuroscience,⁶ protein–protein interaction networks,⁷ recommendation system,^{8,9} image processing,^{10–12} remote sensing,^{13–15} information safety,^{16–18} reinforcement Learning,^{19,20} and so forth.

To deal with the complicated time-varied graphs, it is necessary and crucial to preprocess the raw dynamic graph representations, which record all continuous evolution of the graph over time, such as node emerging/disappearing and link addition/deletion.^{21–25} Current researches^{26–33} refine the raw dynamic representations to two main branches: dynamic continuous and dynamic discrete graphs. The raw representations are projected to a single two-dimensional (2D) temporal graph for the former, storing the most information in graph evolution. However, the corresponding dynamic continuous networks are considerably complicated, which involves heavy computation.²⁵ For discrete graphs, the structural representations are sampled to graph snapshots at regular time intervals, such as 1 day, over time. Although the developing networks are easier than the continuous ones, the temporal information is lost much more.²⁴ We hope to find an efficient way to encode the raw dynamic graph representations, which can alleviate the temporal information loss and simplify the evolved network in future representation learning. Thus, one of our primary contributions, Adaptive Data Encoding (ADE), is proposed to adequately project the temporal information into a sequence of event-based patches with equal amounts of temporal-topological structural patterns to avoid information loss.

DGNNs extract and analyze patterns for graph learning along temporal dimension on the refined dynamic temporal graphs. In this step, it is crucial to have an efficient and powerful network under specific tasks. Some researches^{28,32,33} utilize recurrent neural networks (RNNs) to scrutinize representations on the sequence of dynamic graphs. However, RNN-based DGNNs are more time-consuming and inadequately handle sequential time-dependent embedding with increasing moments of time steps. Since the transformer-based approaches^{26,27,34} adaptively designate divergent and interpretable attention to past embedding over time, the performance is better than RNN-based DGNNs on the long-time duration. However, because the standard transformer (ST)³⁵ contains fully connected attention conjunction, which causes the heavy computation on a time-dependent sequence.³⁶ We hope to simplify and effectively convey temporal information along the time dimension and achieve acceptable performance under inductive and transductive link prediction tasks. Thus, a lightweight module, Sparse Temporal Transformer (STT), is proposed to compute the temporal information with far lower costs by simplified sparse attention conjunction under both graph tasks.

Figure 1 illustrates the overall architecture of the sparse dynamic (Sparse-Dyn), which contains three main components: ADE for adaptive encoding of the raw dynamic continuous graph-based data, graph structural attention (GSA) for investigation of local structural patterns on patches, and STT for graph evolution capture of global temporal patterns over time.

To evaluate our proposed model, we conduct experiments on two continuous data sets under inductive link prediction tasks. The experiments demonstrate that Sparse-Dyn outperforms state-of-the-art networks on the continuous graph data sets. In addition, we also designed an abbreviated version of Sparse-Dyn with only GSA and STT. This abbreviated version is then utilized to learn the representation of the discrete data sets under both inductive and transductive link prediction tasks. Further experiments show that this version is still faster, more efficient, and has higher accuracy on four discrete dynamic graph data sets.

The contributions in this paper are summarized as follows:

- The Sparse-Dyn is proposed to trade off the accuracy and efficiency on both dynamic continuous and discrete representations under link prediction tasks.
- We recommend a new approach, ADE, to preprocess the raw dynamic graph representations. The ADE can alleviate the information loss in the process and simplify the evolved network in future representation learning.
- We propose a lightweight temporal self-attentional module called STT. The STT-based Sparse-Dyn can substantially reduce the computation by comparing RNN-based and ST-based solutions on both continuous dynamic graph data sets.
- The abbreviated version of Sparse-Dyn, comprising only GSA and STT, can also be utilized on the discrete dynamic graph data sets. The experiments consistently demonstrate superior performance for Sparse-Dyn over state-of-the-art approaches under inductive and transductive link prediction tasks.

The structure of this paper is organized as follows: Section 2 presents a review of the latest work on dynamic graph data sets and networks. Section 3 identifies critical terminology and fundamental concepts used in this paper. Section 4 describes the detailed mathematical process of Sparse-Dyn. The experiment design and results are discussed in Section 5. Finally, Section 6 presents our conclusion.

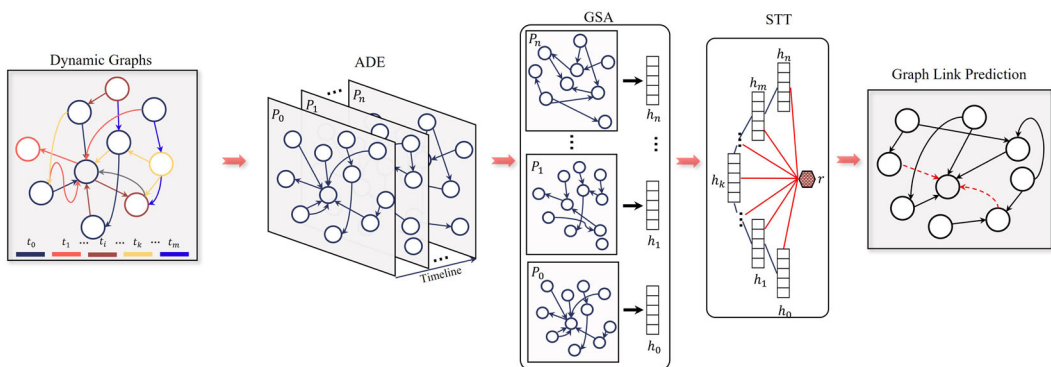


FIGURE 1 Overall architecture of Sparse-Dyn includes three main parts: Adaptive Data Encoding (ADE), Graph Structural Attention (GSA), and Sparse Temporal Transformer (STT) [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/ai.22967)]

2 | RELATED WORK

2.1 | Graph representations

Graph representations can generally be categorized into two distinct levels: static and dynamic. The former includes only structural information, while the latter includes another critical parameter: time. The raw dynamic representation contains node interactions³⁷ and time-stamped edges,²⁴ where instantaneous events are recorded into the raw graph representations, such as the creation and removal of nodes and edges.

Current researches mainly focus on two branches to preprocess the raw representations: dynamic continuous and discrete graphs.^{21,22,24,25} The dynamic continuous graphs store the most information by projecting the raw representations to a 2D temporal graph, which is also a specific static graph appended with temporal information.²³ However, the corresponding networks are complicated because they have to extract temporal information at each moment, which is the common issue of the dynamic continuous graphs.²⁵

For the dynamic discrete graphs, current research^{22,23} group graph embedding with a certain temporal granularity over time. The discrete graphs include equal time intervals, which can be represented with multiple snapshots along temporal dimension.³⁸ Because the temporal information is sampled at the discrete moments, such as 1 day/month, to several graph snapshots, the discrete representation is less complicated than a continuous representation.²¹ However, this kind of graph tracking method causes more processing information loss. Also, the distribution of events or interactions among different temporal windows is not homogeneous, which leads to an imbalance of temporal information among divergence graph snapshots.

We hope to find an efficient temporal encoding approach to alleviating the temporal information loss and simplify the evolved network in future representation learning. Thus, an event-based ADE module is proposed to adaptively encode the temporal information and determine the optimum number of temporal patches on the time dimension. Unlike traditional representations, our temporal patches contain an equal amount of temporal-topological structural patterns, which is fair and effective for future representation learning.

2.2 | Dynamic graph neural network

Since dynamic graph representations append a time dimension on the static ones, the RNN-based DGNNs^{31,39} are considered to summarize temporal information over time. However, the computation of RNN-based DGNNs is expensive because RNNs need a large amount of graph data for training. Moreover, it scales poorly on a long-time temporal dimension.²⁶ To solve this issue, the transformer-based DGNNs^{26,27} are introduced to deal with the temporal information along the time dimension. Temporal graph attention (TGAT)²⁷ introduces temporal constraints on neighborhood aggregation methods and utilizes a TGAT layer to aggregate temporal-topological features on the continuous graph data sets. Dynamic graphs via self-attention (DySAT)²⁶ generates a dynamic representation by joint self-attention of both structural and temporal information on discrete graph data sets. However, the common problem is that their computation is enormous on a long temporal sequence due to the fully connected attention conjunction of the ST. A proper dynamic graph network should achieve the desired trade-off between accuracy and efficiency under graph representation learning tasks. In Sparse-Dyn, we designed a lightweight module, STT. The information is only conveyed among 1-hop neighbors

and the relay node. Experiments show that such a module significantly reduces the inference time and still performs better than the state-of-the-art approaches on both continuous and discrete representations.

3 | PRELIMINARIES

This section illustrates the terminology and preliminary knowledge. Table 1 denotes various terminologies.

Definition 1 (Dynamic graph neural network). In general, a dynamic graph, $DG = (A, X; T)$, contains two main aspects: structural patterns (A, X) and temporal information T .

Definition 2 (Graph link prediction). Given DG , the task is to estimate the link status among nodes by analyzing the aggregated information on both nodes at one moment.

Definition 3 (Inductive learning). Given DG , DGNNs can only investigate the graph information from the beginning to the moment $T - 1$. The analyzed representations are utilized to predict the future links at T . This task is prevalent and crucial since it makes predictions on unseen nodes and links in the future.

Definition 4 (Transductive learning). Given DG , DGNNs can observe all nodes from the beginning to the end, T . The models learn representation and make the predictions at each snapshot or moment.

4 | METHODS

The Sparse-Dyn consists of three components connected serially, ADE, GSA, and STT, as shown in Figure 1. To make the distribution of the events uniformly along a temporal dimension and alleviate the disturbance from irrelevant messages to crucial ones in the future

TABLE 1 General notations

Notations	Description	Notations	Description
DG	Dynamic graphs	S	Snapshots
A	Adjacency matrix	X	Node feature vectors
E	Edges/connections	N	Nodes/objects
e_{um}	Edge between nodes u and m	u	Center node
h	Structural representation	E	Embedding/token representation
R	Frequency of events	W	Learnable parameters
PE	Position embedding	p	Position-aware structural representation
c	Context information	z	Time-dependent structural representation
r	Relay representation	L	Loss function

graph learning, ADE adequately encodes the temporal information into a sequence of patches with an equal amount of temporal-topological structural patterns by investigating the frequency of events. The GSA module extracts the local structural representations with a self-attention layer on each temporal patch. The learned time-dependent structural representations are sent to the lightweight module, STT, to capture the global graph evolution along the temporal dimension.

4.1 | Adaptive data encoding

Since dynamic continuous graph stores almost events over time, it can be regarded as a particular static graph with an additional temporal dimension. Current researches^{28,40} project the topological graph structures and node features from the raw representation to a 2D temporal continuous graph for future representation learning. However, the data processing is low-efficiency because they pay much attention to the inoperative information for the target node.²³ The continuous representation of these networks is far more complicated.²⁵ In contrast, the discrete dynamic graph simplifies the data processing by sampling the structural representations to graph snapshots at regular time intervals. The developing discrete networks are less complicated than the continuous ones.²⁵ However, the temporal information is lost much more.

We hope to find an efficient way to encode raw dynamic representations and reduce the information loss in this processing. Thus, an ADE is proposed to adaptively encode temporal information into a set of event-based temporal patches with an equal amount of temporal-topological structure. While avoiding the use of snapshots which causes information loss, it also achieves a finer time granularity close to what a dynamic continuous graph could provide.

Figure 2 compares three encoding approaches: raw graph representation without encoding, uniform data encoding (UDE), and ADE. On the basis of the raw time-dependent representations with superabundant details, the UDE separates the graph patterns into several patches by the same temporal intervals, such as 1 day, and so forth. In this case, the events have happened irregularly along the time dimension. It means the distribution of temporal-topological patterns is not homogeneous of the encoded patches. The computation on some patches is expensive because it takes more to deal with the complicated patch structure; meanwhile, the calculation is light on those with fewer events. Thus, it is inefficient to process the patches with different amounts of structural patterns in parallel computing.

Instead of separating by a regular interval, the raw data are divided into N event-based patches along a temporal dimension by ADE, as shown in Figure 3. In this case, each patch contains an equal amount of temporal-topological patterns, which is beneficial to improve the efficiency of feature extraction on N event-based patches in parallel. To distribute the same event to each patch, an effortless way is to assign only one event on each patch, which means the total number of patches is enormous ($N \rightarrow \infty$). Alternatively, all events are projected to only one patch ($N = 1$), just like a static graph. However, the computational time of both approaches is heavy. In the previous approach, the network consumes too much on massive patches with low entropy, while it takes a long time to deal with the only patch with all events in the second approach. Thus, it is crucial to balance the total number of patches, N , and the quality of structural embedding of each patch along the time dimension. We utilize a cost function of ADE to adaptively determine N and the embedding of patches in Equations (1) and (2).

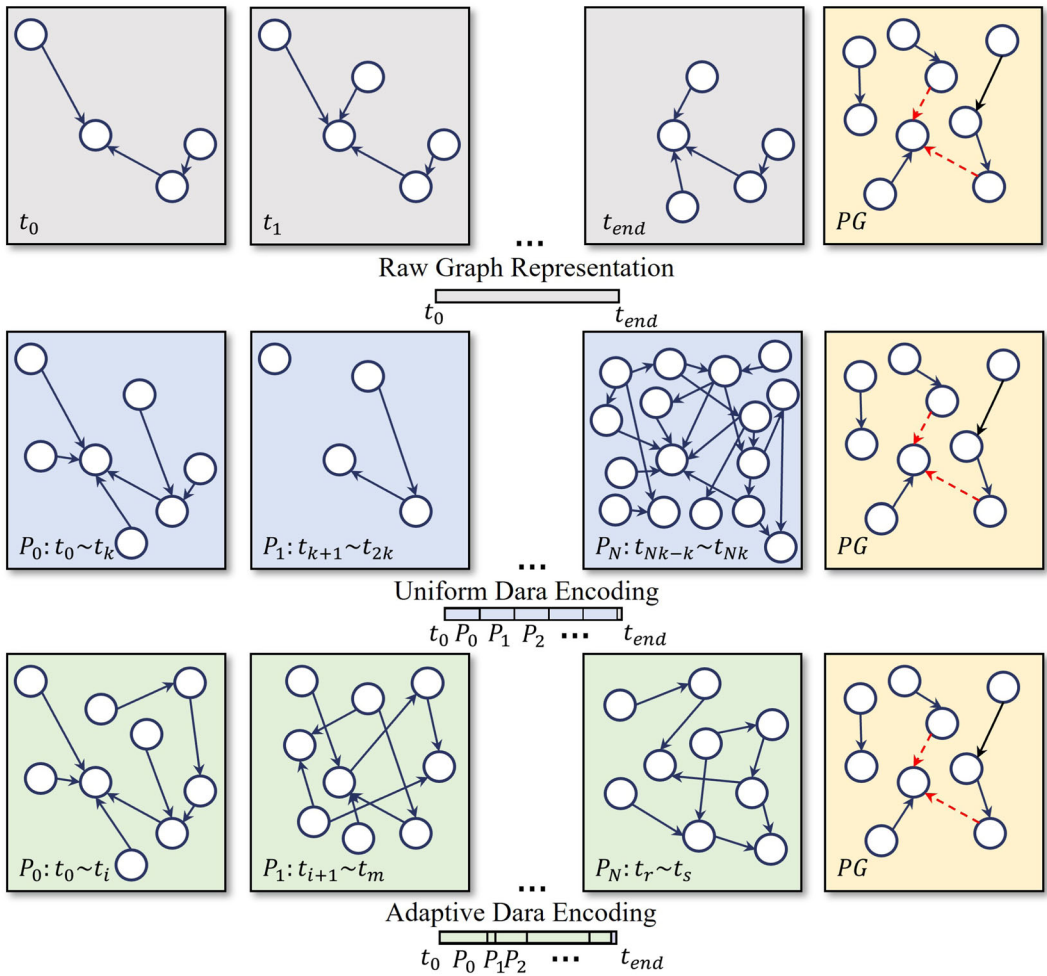


FIGURE 2 Different encoding approaches. Top row, raw graph representation; second row, uniform data encoding approach encodes the time-dependent representation by the same time interval uniformly; bottom row, adaptive data encoding approach adaptively encodes the whole raw representation to a sequence of event-based patches. Since each patch contains an equal amount of temporal-topological patterns, parallel computing is highly efficient. PG indicates the predicted graph. [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/terms-and-conditions)] [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/terms-and-conditions)]

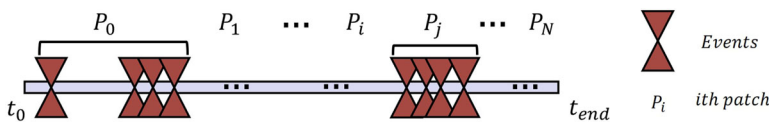


FIGURE 3 ADE adaptively encodes temporal information into a sequence of patches by events along the temporal dimension. Each temporal patch contains an equal amount of temporal-topological patterns with the others. ADE, Adaptive Data Encoding. [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/terms-and-conditions)]

$$\min(L_A) = \phi \left(E, \sum_{n=1}^N E_n \right) + \tau \log \frac{R}{\Delta}, \quad (1)$$

$$\phi(E_i - E_j) - \epsilon = 0, \quad (2)$$

where E is the embedding of the raw graph representations, and τ is a constant parameter. The R is the number of total events, and $N = \frac{R}{\Delta}$ is the number of patches. The ϕ is the Pearson Correlation Coefficient⁴¹ to measure the linear correlation of embedding between every two patches. The ϵ is a tiny constant parameter to ensure that each patch contains an equal amount of temporal-topological patterns.

Equation (2) restricts the difference of the structural embedding on each patch is tiny. In Equation (1), the first item is to minimize the difference between the quality of the final projected representations of all patches. The second item is encouraged to increase a considerable value of N . By minimizing the cost function of L_A , an optimum balance between N and E_i can finally be obtained.

4.2 | Graph structural attention

Once the raw graph representations are encoded to the optimal N time-dependent patches, extracting the local structural patterns on each encoded patch is next. The input of GSA is a set of node representations of the current patch. Inspired by graph attention (GAT),⁴² a node self-attention matrix, $\alpha_{um} = \text{softmax}(e_{um})$, is learned to determine the relevance between neighbors and the center node, u , on the time-dependent patch.

$$e_{um} = \sigma(A_{um} \cdot r[W_p X_u; W_p X_m]), \quad (3)$$

where e_{um} indicates the relevance of neighbor node m to the center node u . The σ is the exponential linear unit (ELU) activation function.⁴³ The A is the patch's adjacency matrix to indicate the current patch's linking relation, The γ indicates the self-attention mechanism, and W_p is the weight matrix of the patch p . The x_u and x_m are the node representation of the center node u and neighbor node m . Then, the activation function ψ is applied to get the nonlinear node representation of the current patch in Equation (4).

$$h_u = \psi \left(\sum_{m \in N_u} \alpha_{um} \cdot W_p X_m \right), \quad (4)$$

where ψ is the Gaussian Error Linear Unit (GELU)⁴⁴ for the final output representations, $h_u \in \mathbf{R}^d$ is the patch embedding, and d is the updated feature dimension.

The final step of GSA is to add the position embedding of time-dependent patches, which embed the absolute temporal position of each patch, as shown in Figure 4 Left. Thus, the output of GSA, p_i , contains both local structural patterns and temporal position information of the current patch.

4.3 | Sparse temporal transformer

The objective of STT is to gather the global evolution of structural patterns on each patch from GSA along the time dimension. Current transformer-based DGNNs^{26,27} utilize the ST to extract

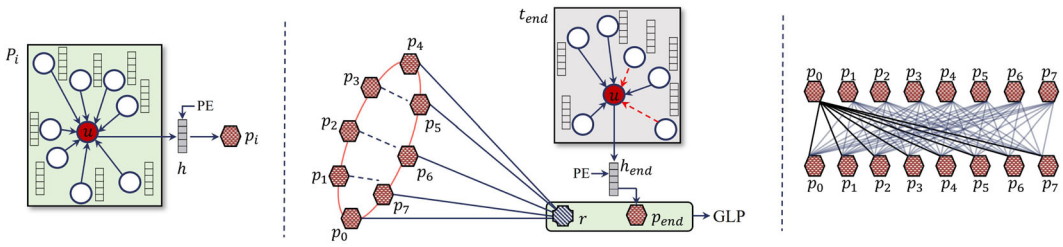


FIGURE 4 (Left) GSA: The local structural patterns are extracted and added with the position embedding on each time-dependent patch in the GSA module. (Middle) The lightweight STT: The global temporal patterns are investigated over time. Each patch is only connected to its two adjacent patches and the relay one. The updated relay patch is utilized to predict the predicted patch's link. (Right) Patch conveys information to each other due to the fully connected attention conjunction of the standard transformer, which causes the heavy computation. GSA, graph structural attention; PE, position embedding; STT, Sparse Temporal Transformer. [Color figure can be viewed at wileyonlinelibrary.com]

the temporal patterns and receive good accuracy on both continuous and discrete dynamic representations. However, due to the fully connected attention conjunction shown in Figure 4 Right, the computation of these transformer-based DGNNs on a long-time sequence is expensive. To reduce the computation, we design a lightweight module, STT, instead of the ST to deal with the global temporal patterns of time-dependent patches.

Figure 4 Middle illustrates the architecture of the STT module, which consists of N time-dependent patches from GSA and a single relay patch. Each temporal patch is connected with two adjacent patches and the relay one on the STT. The relay patch's functionality is congregating and distributing the representation among all time-dependent patches. Thus, the STT module can learn global representations with the relay patch. By comparing the fully connected attention conjunction, the advantage of our connection is that the computation to extract the temporal information is cut down by reducing the interaction times of patches.

The input of STT is a sequence of representations for a center node u at all temporal patches. The embedding of the relay patch is initialized as the average of all time-dependent patches at the beginning. The context of $c_i(t)$ of the patch i is updated by aggregating the representations from its two neighbor patch $i - 1$ and $i + 1$, the relay $r(t - 1)$, the state of itself at last moment $z_i(t - 1)$, and the embedding p_i .

$$c_i(t) = [z_{i-1}(t - 1); z_i(t - 1); z_{i+1}(t - 1); p_i; r(t - 1)]. \quad (5)$$

The temporal self-attention function of the current state $z_i(t)$ of patch i is defined as in Equation (6).

$$z_i(t) = \phi_1(\beta_i(t) \cdot c_i(t) W_i), \quad (6)$$

$$\beta_i(t) = \text{softmax} \left(\frac{z_i(t - 1) W_q \cdot (c_i(t) W_k)^T}{\sqrt{d}} \right), \quad (7)$$

where $\beta_i(t)$ is the self-attention coefficients of temporal patches, W_q , W_k , W_i are learnable parameters, and d is the feature dimension of z_i . A layer normalization operation⁴⁵ is added after the transaction among all patches.

Meanwhile, current state of relay patch $r(t)$ gather all representations from temporal patches $Z(t)$, and the state of itself at last moment $r(t - 1)$ in Equation (8).

$$r(t) = \phi_2(\lambda(t) \cdot [r(t-1); Z(t)] W'_v), \tag{8}$$

$$\lambda(t) = \text{softmax} \left(\frac{r(t-1) W'_q \cdot ([r(t-1); Z(t)] W'_k)^T}{\sqrt{d'}} \right), \tag{9}$$

where $\lambda(t)$ is the self-attention coefficients of relay patch, W'_q, W'_k, W'_i are learnable parameters, and d' is the feature dimension of the relay. Both ϕ_1 and ϕ_2 are nonlinear activation function. Similarly, the layer normalization operation is also added after the transaction on relay patches.

4.4 | Graph link prediction

Graph link prediction is one of the core graph tasks whose purpose is to forecast the connection among nodes based on node representations. The training processing analyzes the representations of $T - 1$ temporal patches in the inductive task. The network makes the link prediction to the unseen nodes on the final predicted graph (PG). In this procedure, we utilize the deep walk¹ approach to sample some positive (connected links) and negative (unrelated links) on PG. A binary cross-entropy loss emboldens positive cases to have similar representations while suppressing the negative ones in Equation (10).

$$L = \sum_{u \in V} \left(\sum_{v \in N_{\text{walk}}(u)} -\log(\varphi(\langle e_v, e_u \rangle)) - \omega_n \cdot \sum_{v' \in P_{\text{walk}}(u)} \log(\varphi(1 - \langle e_{v'}, e_u \rangle)) \right), \tag{10}$$

where φ is the nonlinear activation function, $\langle \cdot \rangle$ is the inner-product, ω_n is a constant fine-tuned hyperparameter. The $N_{\text{walk}}(u)$ is the sampled positive cases in fixed-length random deep walks on PG, while the $P_{\text{walk}}(u)$ is the sampled negative cases.

Regarding the transductive task, the positive and negative cases are sampled at each temporal patch. Thus, the final loss function is to calculate the sum of all costs on each patch in Equation (11).

$$L = \sum_{t=1}^{T_N} \sum_{u \in V} \left(\sum_{v \in N_{\text{walk}}^t(u)} -\log(\varphi(\langle e_v^t, e_u^t \rangle)) - \omega_n \cdot \sum_{v' \in P_{\text{walk}}^t(u)} \log(\varphi(1 - \langle e_{v'}^t, e_u^t \rangle)) \right). \tag{11}$$

5 | EXPERIMENTS

5.1 | Data sets

We experimentally validate Sparse-Dyn on six real-world dynamic graph data sets: two dynamic continuous and four dynamic discrete data sets. Table 2 summarizes the statistics of the details of these six data sets.

Reddit and Wikipedia: These two dynamic continuous graph data sets³³ describe the active users and their editions on Reddit and Wikipedia in 1 month. The dynamic labels represent the state of the user on their editions. Reddit contains 10,984 nodes and 672,447 links, while Wikipedia contains 9227 nodes and 157,474 links.

Enron and UCI: These two dynamic discrete graph data sets describe the network communications. Enron includes 143 nodes (employees) and 2347 links (email interactions), while UCI includes 1809 nodes (users) and 16,822 links (messages).^{46,47}

Yelp and ML-10M: These two dynamic discrete graph data sets describe the bipartite networks from Yelp and MovieLens.⁴⁸ Yelp has 6509 nodes (users and businesses) and 95,361 links (relationship), while ML-10M has 20,537 nodes (users with the tags) and 43,760 links (interactions).

5.2 | Experimental setup

At the beginning of training, the Glorot and Bengio⁴⁹ are to initialize the learnable parameters W of each layer to avoid the gradient from exploding or vanishing suddenly. The GELU yields the final output nonlinear representations at the end of GSA. The ELU is utilized as the activation function for both temporal patches and relay in STT. A binary cross-entropy loss sends the probability distribution over predicted link prediction in both inductive and transductive tasks. The numbers of adaptive temporal patches on Reddit and Wikipedia are 16 and 12. In addition, the dropout approach⁵⁰ is introduced to avoid overfitting during the training process, with the dropout rate in a range of 0.3–0.7, which depends on the data set and tasks.

TABLE 2 Statistics of the dynamic graph data sets

Continuous	Nodes	Links	Time duration (s)
Reddit	10,984	672,447	2,678,390
Wikipedia	9227	157,474	2,678,373
Discrete	Nodes	Links	Time steps
Enron	143	2347	16
UCI	1809	16,822	13
Yelp	6509	95,361	12
ML-10M	20,537	43,760	13

5.3 | Multihead attention mechanism

The multihead mechanism is also used to stabilize the learning process under the link prediction task. At the end of the GSA and STT modules, the multihead attention mechanism is adopted separately.

The structural multihead attention for GSA: Since the multihead attention mechanism is adopted at GSA, the final representation h_u in Section 4.2 is concatenated with the output from each single head in Equation (12).

$$h_u = \bowtie(h_u^1, h_u^2, \dots, h_u^k), \tag{12}$$

where \bowtie is the concatenate operation, and k is the number of multiple heads. The graph structural attention heads share parameters across temporal patches.

The temporal multihead attention for STT: Similar with the above setting, the representation of each patch z_i and relay $r(t)$ in Section 4.3 are concatenated with the output from each single head in Equations (13) and (14).

$$z_i(t) = \bowtie(z_i^1(t), z_i^2(t), \dots, z_i^k(t)), \tag{13}$$

$$r(t) = \bowtie(r^1(t), r^2(t), \dots, r^k(t)). \tag{14}$$

To evaluate the contribution of the multihead attention mechanism, we set a series of experiments for Efficient_Dyn with different head numbers independently in the range 1, 2, 4, 8, and 10 on two continuous and two discrete dynamic graph data sets. As shown in Figure 5, it can be observed that the accuracy of multihead networks is better than the single-head networks on all four data sets. In addition, the accuracy does not keep increasing with a more significant number of heads. The performance of the multihead network stabilizes with eight attention heads for both modules.

5.4 | Inductive learning on continuous data sets

These experiments compare different approaches with two continuous data sets under the inductive link prediction task. In these experiments, we compare Sparse-Dyn networks with

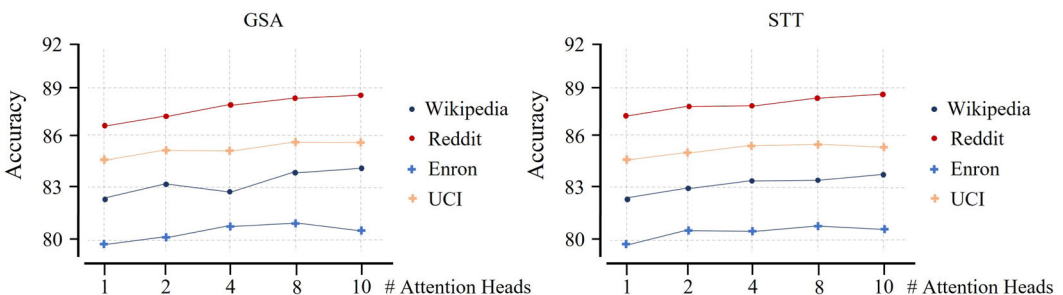


FIGURE 5 Accuracy of Efficient_Dyn with different multiheads on four data sets in the inductive link prediction tasks [Color figure can be viewed at wileyonlinelibrary.com]

another four approaches as the baselines: GAT-T,⁴² GraphSAGE-LSTM,⁵¹ Const-TGAT,²⁷ and TGAT. GAT-T concatenates the time encoding to the graph structural features when gathering the temporal information. GraphSAGE-LSTM considers Long Short-Term Memory (LSTM) to aggregate the temporal information over time. TGAT utilizes a temporal attention coefficient matrix to aggregate temporal representations. Const-TGAT pays the same temporal attention to collecting the temporal patterns.

Table 3 shows the accuracy of these approaches under the link prediction task on two dynamic continuous data sets. It can be observed that the performances of transformer-based networks are better than RNN-based ones. With the temporal attention, the accuracy of TGAT can exceed 2.4% and 1.68% than the ones of Const-TGAT on Reddit and Wikipedia.

TGAT consists of two main components: functional time encoding and an ST. We modify the architecture of TGAT with STT instead of the ST and name it as Sparse-Dyn*. To check the performance of ADE and STT separately on dynamic continuous representations, the performance of TGAT, Sparse-Dyn*(functional time encoding + STT), and Sparse-Dyn (ADE + STT) is analyzed in Table 3. Compared with TGAT and Sparse-Dyn*, the latter's accuracy achieves 92.83% on Reddit and 87.46% on Wikipedia, which surpasses 2.15% and 2.04% than the former. Meanwhile, the inference time of Sparse-Dyn* is less than TGAT, demonstrating that STT is more effective by comparing the fully connected connection of the ST. Due to ADE, the inference time of Sparse-Dyn is further reduced by comparing it with the time of Sparse-Dyn*. The Sparse-Dyn's accuracy is less than TGAT by 2.03% and 1.73% on Reddit and Wikipedia data sets because the functional time encoding component utilizes more details with temporal constraints of graph representations. However, our inference speed is only around 0.6 times that of TGAT on both continuous data sets, which is more competitive in practical usage. These experiments demonstrate the contribution of Sparse-Dyn consisting of both EDA and STT on dynamic continuous representations under the link prediction task.

5.5 | Inductive learning on discrete data sets

The previous experiments demonstrate the power of Sparse-Dyn on dynamic continuous data sets. The Sparse-Dyn can also be utilized on discrete graph data sets. Since the discrete representations have several graph snapshots along temporal dimension, we compare our Sparse-Dyn\$, which only consists of GSA and STT, with another seven baselines: node2vec,⁴ GraphSAGE, GAT, Dynamic Triad,²⁹ DynGEM,⁵² DynAERNN,³¹ and DySAT. The node2vec handles the second-order random walk sampling to grasp node representations. Dynamic Triad combines triadic closure to preserve both structural information and evolution patterns.

TABLE 3 Link prediction on dynamic continuous data sets

Data sets	GAT-T	GraphSAGE-L	Const-TGAT	TGAT	Sparse-Dyn*	Sparse-Dyn	TGAT	Sparse-Dyn*	Spars-e-Dyn
Reddit	90.24	89.43	88.28	90.68	92.83	88.65	30.164s	23.187s	18.956s
Wikipedia	84.76	82.43	83.60	85.28	87.36	83.55	15.377s	11.637s	9.623s

Note: Left, inductive learning task results (accuracy%); Right, inference time (s). The Sparse-Dyn* combines the functional time encoding component from TGAT and our STT component.

Abbreviations: Sparse-Dyn, sparse dynamic; STT, Sparse Temporal Transformer; TGAT, temporal graph attention.

TABLE 4 Link prediction on dynamic discrete graph data sets

Inductive	node2vec	GraphSAGE	GAT	DynamicTriad	DynGEM	DynAERNN	DySAT	Sparse-Dyn§
Enron	75.86	74.67	69.25	68.77	62.85	59.63	78.52	81.36
UCI	74.76	79.41	73.78	71.67	79.82	81.91	83.72	85.47
Yelp	65.17	58.81	65.91	62.83	66.84	73.46	69.23	72.59
ML-10M	84.89	89.14	84.51	84.32	83.51	88.19	92.54	94.28
Transductive	node2vec	GraphSAGE	GAT	DynamicTriad	DynGEM	DynAERNN	DySAT	Sparse-Dyn§
Enron	83.05	81.88	75.97	78.98	69.72	72.01	86.60	87.94
UCI	80.49	82.89	81.86	80.28	79.82	83.52	85.81	87.53
Yelp	65.34	58.56	65.37	62.69	65.94	68.91	69.87	72.01
ML-10M	87.52	89.92	86.75	88.43	85.96	89.47	96.38	97.52

Note: Top, Inductive learning task results (accuracy%); Bottom, Transductive learning task results (accuracy%). The Sparse-Dyn§ only consists of GSA and STT modules. Abbreviations: GAT, graph attention; GSA, graph structural attention; STT, Sparse Temporal Transformer.

TABLE 5 Inference time (ms) on discrete graph data sets in the inductive learning task

Data sets	DySAT	Sparse-Dyn§
Enron	2.961	0.997
UCI	13.953	4.965
Yelp	1360.31	509.62
ML-10M	10,678.36	3746.55

DynGEM utilizes a deep autoencoder to generate nonlinear embeddings of snapshots. DynGEM constructs both dense and recurrent layers to investigate the temporal graph evolution. DySAT extracts node representations via fully connected self-attention on both graph structural and temporal patterns.

Table 4 summarizes the results of these eight approaches on four dynamic discrete graph data sets. We find that the accuracy of DySAT exceeds the other state-of-the-art approaches, except Sparse-Dyn§, under the link prediction task, which benefits from the fully connected attention conjunction architecture of transform by extracting temporal patterns over time. This phenomenon demonstrates that the transformer-based DGNN outperforms the traditional graph learning approaches, including RNN-based DGNNs. By comparing DySAT and Sparse-Dyn§, we found the accuracy of Sparse-Dyn§ is 81.36%, 85.47%, 72.59%, and 94.28% on Enron, UCI, Yelp, and ML-10M data sets, which are better than DySAT. Meanwhile, the inference time of Sparse-Dyn§ is much less than the time of DySAT on all four dynamic discrete data sets, as shown in Table 5, demonstrating that STT is also competitive and effective on dynamic discrete graph representations.

5.6 | Transductive learning on discrete data sets

Besides previous experiments, we also evaluate our proposed network on dynamic discrete data sets under the transductive link prediction task. From Table 4 bottom, we observe the transformer-based DGNNs (DySAT and Sparse-Dyn§) have better performances than RNN-based ones on all four discrete data sets, which also proves the self-attention architecture is powerful for transductive graph learning. As a result, the accuracy of Sparse-Dyn§ achieves 87.94%, 87.53%, 72.01%, and 87.52% on Enron, UCI, Yelp, and ML-10M separately, which also demonstrates the improvements delivered by our innovative architecture of Sparse-Dyn.

6 | CONCLUSION

This paper proposes a Sparse-Dyn graph neural network, Sparse-Dyn, that trade-offs the accuracy and efficiency under inductive and transductive link prediction tasks. Sparse-Dyn consists of three main components: ADE, GSA, and STT. The ADE module adaptively encodes temporal information into a sequence of patches with an equal amount of temporal-topological structure, which reduces the information loss in the projection processing due to adaptive generation with a more delicate time granularity. Also, it

simplifies the evolved network in future representation learning. The GSA module learns the local structural representations on each encoded patch along the temporal dimension. The lightweight STT is utilized to extract global temporal patterns over time. Benefiting from the information delivery on the simplified architecture, the STT-based Sparse-Dyn can substantially reduce the computation by comparing RNN-based and ST-based solutions on both continuous dynamic graph data sets. The Sparse-Dyn is evaluated on two dynamic continuous and four dynamic discrete graph data sets. The results illustrate that Sparse-Dyn is competitive and efficient in inference speed and performance.

DATA AVAILABILITY STATEMENT

No. Research data are not shared.

ORCID

Yan Pang  <http://orcid.org/0000-0002-6483-8326>

Zhen Wang  <http://orcid.org/0000-0002-8637-8375>

Mengyu Wang  <http://orcid.org/0000-0003-0707-0071>

Ji Zhang  <http://orcid.org/0000-0001-7167-6970>

Chao Liu  <http://orcid.org/0000-0002-8703-4232>

REFERENCES

1. Perozzi B, Al-Rfou R, Skiena S. Deepwalk: online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; 2014: 701-710.
2. Tang L, Ma W, Grobler M, Meng W, Wang Y, Wen S. Faces are protected as privacy: an automatic tagging framework against unpermitted photo sharing in social media. *IEEE Access*. 2019;7: 75556-75567.
3. Zhu T, Li J, Hu X, Xiong P, Zhou W. The dynamic privacy-preserving mechanisms for online dynamic social networks. *IEEE Trans Knowl Data Eng*. 2022;34(6):2962-2974.
4. Grover A, Leskovec J. node2vec: scalable feature learning for networks. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; 2016:855-864.
5. Wang Z, Zhang J, Feng J, Chen Z. Knowledge graph embedding by translating on hyperplanes. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol 28; 2014.
6. Goering S, Klein E. Fostering neuroethics integration with neuroscience in the BRAIN initiative: comments on the NIH neuroethics roadmap. *AJOB Neurosci*. 2020;11(3):184-188.
7. Fout AM. *Protein Interface Prediction Using Graph Convolutional Networks*. Ph.D. Thesis. Colorado State University; 2017.
8. Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J. Graph convolutional neural networks for web-scale recommender systems. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*; 2018:974-983.
9. Jiang L, Cheng Y, Yang L, Li J, Yan H, Wang X. A trust-based collaborative filtering algorithm for E-commerce recommendation system. *J Ambient Intell Hum Comput*. 2019;10(8):3023-3034.
10. Chen C, Huang T. Camdar-adv: generating adversarial patches on 3D object. *Int J Intell Syst*. 2021;36(3): 1441-1453.
11. Yan H, Chen M, Hu L, Jia C. Secure video retrieval using image query on an untrusted cloud. *Appl Soft Comput*. 2020;97:106782.
12. Shi Z, Chang C, Chen H, Du X, Zhang H. PR-NET: progressively-refined neural network for image manipulation localization. *Int J Intell Syst*. 2022;37(5):3166-3188.
13. Ai S, Koe ASV, Huang T. Adversarial perturbation in remote sensing image recognition. *Appl Soft Comput*. 2021;105:107252.

14. Wang X, Li J, Yan H. An improved anti-quantum MST3 public key encryption scheme for remote sensing images. *Enterp Inf Syst.* 2021;15(4):530-544.
15. Wang X, Li J, Li J, Yan H. Multilevel similarity model for high-resolution remote sensing image registration. *Inf Sci.* 2019;505:294-305.
16. Yan H, Hu L, Xiang X, Liu Z, Yuan X. PPCL: privacy-preserving collaborative learning for mitigating indirect information leakage. *Inf Sci.* 2021;548:423-437.
17. Li J, Ye H, Li T, et al. Efficient and secure outsourcing of differentially private data publishing with multiple evaluators. *IEEE Trans Dependable Secure Comput.* 2022;19(01):67-76.
18. Maras MH. Internet of things: security and privacy implications. *Int Data Privacy Law.* 2015;5(2):99.
19. Tianqing Z, Zhou W, Ye D, Cheng Z, Li J. Resource allocation in IoT edge computing via concurrent federated reinforcement learning. *IEEE Internet Things J.* 2021;9(2):1414-1426.
20. Liu Wx, Cai J, Chen QC, Wang Y. DRL-R: deep reinforcement learning approach for intelligent routing in software-defined data-center networks. *J Network Comput Appl.* 2021;177:102865.
21. Cui P, Wang X, Pei J, Zhu W. A survey on network embedding. *IEEE Trans Knowl Data Eng.* 2018;31(5):833-852.
22. Cai H, Zheng VW, Chang KCC. A comprehensive survey of graph embedding: problems, techniques, and applications. *IEEE Trans Knowl Data Eng.* 2018;30(9):1616-1637.
23. Kazemi SM, Goel R, Jain K, et al. Representation learning for dynamic graphs: a survey. *J Mach Learn Res.* 2020;21(70):1-73.
24. Barros CD, Mendonça MR, Vieira AB, Ziviani A. A survey on embedding dynamic graphs. *ACM Comput Surveys (CSUR).* 2021;55(1):1-37.
25. Skarding J, Gabrys B, Musial K. Foundations and modeling of dynamic networks using dynamic graph neural networks: a survey. *IEEE Access.* 2021;9:79143-79168.
26. Sankar A, Wu Y, Gou L, Zhang W, Yang H. DySAT: deep neural representation learning on dynamic graphs via self-attention networks. In: *Proceedings of the 13th International Conference on Web Search and Data Mining*; 2020:519-527.
27. Xu D, Ruan C, Korpeoglu E, Kumar S, Achan K. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962.* 2020.
28. Ma Y, Guo Z, Ren Z, Tang J, Yin D. Streaming graph neural networks. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*; 2020:719-728.
29. Zhou L, Yang Y, Ren X, Wu F, Zhuang Y. Dynamic network embedding by modeling triadic closure process. In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol 32; 2018.
30. Jiang N, Jie W, Li J, Liu X, Jin D. GATrust: a multiaspect graph attention network model for trust assessment in OSNs. *IEEE Trans Knowl Data Eng.* 2022;34(6):2962-2974.
31. Goyal P, Chhetri SR, Canedo A. dyngraph2vec: capturing network dynamics using dynamic graph representation learning. *Knowl-Based Syst.* 2020;187:104816.
32. Chen J, Xu X, Wu Y, Zheng H. GC-LSTM: graph convolution embedded LSTM for dynamic link prediction. *arXiv preprint arXiv:1812.04206.* 2018.
33. Kumar S, Zhang X, Leskovec J. Predicting dynamic embedding trajectory in temporal interaction networks. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*; 2019:1269-1278.
34. Chen X, Zhang F, Zhou F, Bonsangue M. Multi-scale graph capsule with influence attention for information cascades prediction. *Int J Intell Syst.* 2022;37(3):2584-2611.
35. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. *Adv Neural Inf Process Syst.* 2017:5998-6008.
36. Guo Q, Qiu X, Liu P, Shao Y, Xue X, Zhang Z. Star-transformer. *arXiv preprint arXiv:1902.09113.* 2019.
37. Latapy M, Viard T, Magnien C. Stream graphs and link streams for the modeling of interactions over time. *Soc Network Anal Min.* 2018;8(1):1-29.
38. Taheri A, Gimpel K, Berger-Wolf T. Learning to represent the evolution of dynamic graphs with recurrent models. In: *Companion Proceedings of the 2019 World Wide Web Conference*; 2019:301-307.
39. Hajiramezanali E, Hasanzadeh A, Duffield N, Narayanan KR, Zhou M, Qian X. Variational graph recurrent neural networks. *arXiv preprint arXiv:1908.09710.* 2019.

40. Han Z, Jiang J, Wang Y, Ma Y, Tresp V. The graph hawkes network for reasoning on temporal knowledge graphs. In: *Learning with Temporal Point Processes Workshop at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019) NeurIPS 2019*; 2019.
41. Benesty J, Chen J, Huang Y, Cohen I. Pearson correlation coefficient. In: *Noise Reduction in Speech Processing*. Springer; 2009:1-4.
42. Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. Graph attention networks. In: *International Conference on Learning Representations*. 2018.
43. Trottier L, Giguere P, Chaib-Draa B. Parametric exponential linear unit for deep convolutional neural networks. In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE; 2017:207-214.
44. Hendrycks D, Gimpel K. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*. 2016.
45. Krueger D, Maharaj T, Kramár J, et al. Zoneout: regularizing RNNs by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*. 2016.
46. Klimt B, Yang Y. The Enron corpus: a new dataset for email classification research. In: *European Conference on Machine Learning*. Springer; 2004:217-226.
47. Panzarasa P, Opsahl T, Carley KM. Patterns and dynamics of users' behavior and interaction: network analysis of an online community. *J Am Soc Inf Sci Technol*. 2009;60(5):911-932.
48. Harper FM, Konstan JA. The MovieLens data sets: history and context. *ACM Trans Interactive Intell Syst (TIIS)*. 2015;5(4):1-19.
49. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings*; 2010:249-256.
50. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res*. 2014;15(1):1929-1958.
51. Hamilton WL, Ying R, Leskovec J. Inductive representation learning on large graphs. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*; 2017:1025-1035.
52. Goyal P, Kamra N, He X, Liu Y. DynGEM: deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*. 2018.

How to cite this article: Pang Y, Shan A, Wang Z, et al. Sparse-Dyn: Sparse dynamic graph multirepresentation learning via event-based sparse temporal attention network. *Int J Intell Syst*. 2022;37:8770-8789. doi:10.1002/int.22967

APPENDIX A

A.1 | Event-based data encoding

This section discusses the event-based data encoding approaches with raw dynamic representations. As shown in Figure A1A, the raw dynamic continuous representation is a sequence of particular static graphs along the time dimension, which stores all events, such as node emerging, node disappearing, link addition, link removing, and so forth. The crucial information is the recorded time-dependent events in the raw representations. Before the representation learning, the raw representations should be preprocessed. For the continuous graphs, the general approach is to project the raw representations to a single 2D temporal graph, as shown in Figure A1B. The primary issue is that some temporal information is lost on the single graph, including node or edge vanishing and multiedge situations. Also, the developing networks are complicated and contain heavy computation because they have to extract the temporal information at each moment on the dynamic continuous graph.

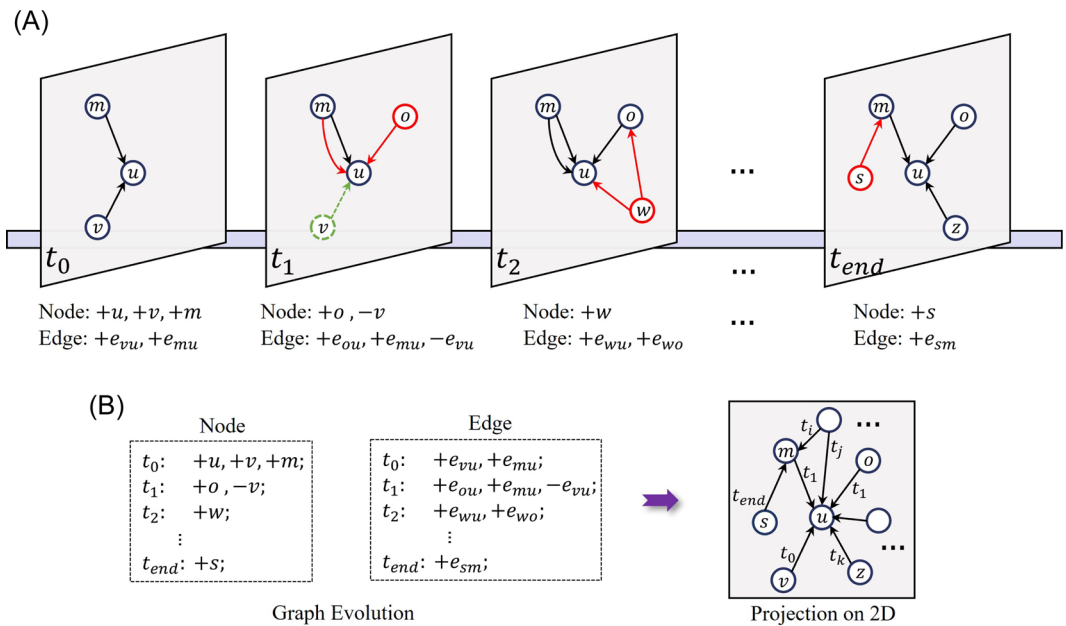


FIGURE A1 Visual illustration for projection from the raw dynamic continuous representations. (A) The generation process of a continuous temporal graph and its snapshots at each moment. The solid red line represents an addition, and the green dash represents deletion. (B) The temporal information and the final state of the projected temporal graph. Some temporal information for nodes and multiedges is lost. [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/ai.22967)]

To alleviate the above issues, a straight thought is to convert the raw continuous representations to several small temporal graphs with temporal intervals instead of the large single one. Unlike dynamic discrete graphs that sample the representations at each discrete interval, we project all events in each period to temporal patches. Each encoded graph patch holds the temporal information in the duration with the same time interval, such as 1 day, 1 week, 1 month, and so forth. However, it is impossible to guarantee that events are uniformly distributed along the time dimension. With the UDE approach, some patches contain superabundant details due to more events in the corresponding duration and vice versa.

As shown in Figure A2, We design some experiments to verify the above phenomenon. The top row of Figure A2 is the distribution of events at each temporal patch on Wikipedia with three uniform time intervals: 1 day, 5 days, and 1 week. It can be observed that the distribution of events is not homogeneous at all three patches. It is becoming increasingly apparent with a longer time interval. The standard variance of the patch with a 1-week interval is 9957, far outweighing the one with a 1-day interval. A similar phenomenon emerges on Reddit, as shown in the bottom row of Figure A2. The calculation is also inefficient on these inhomogeneous patches in the future representation learning in parallel computing.

Our proposed ADE approach adaptively encodes temporal information into a sequence of patches by events. This approach's advantage is avoiding using snapshots to cause information loss and achieving a finer time granularity, which is close to what continuous networks could provide. Also, the equal amount of temporal-topological structure of patches is more efficient in future representation learning.

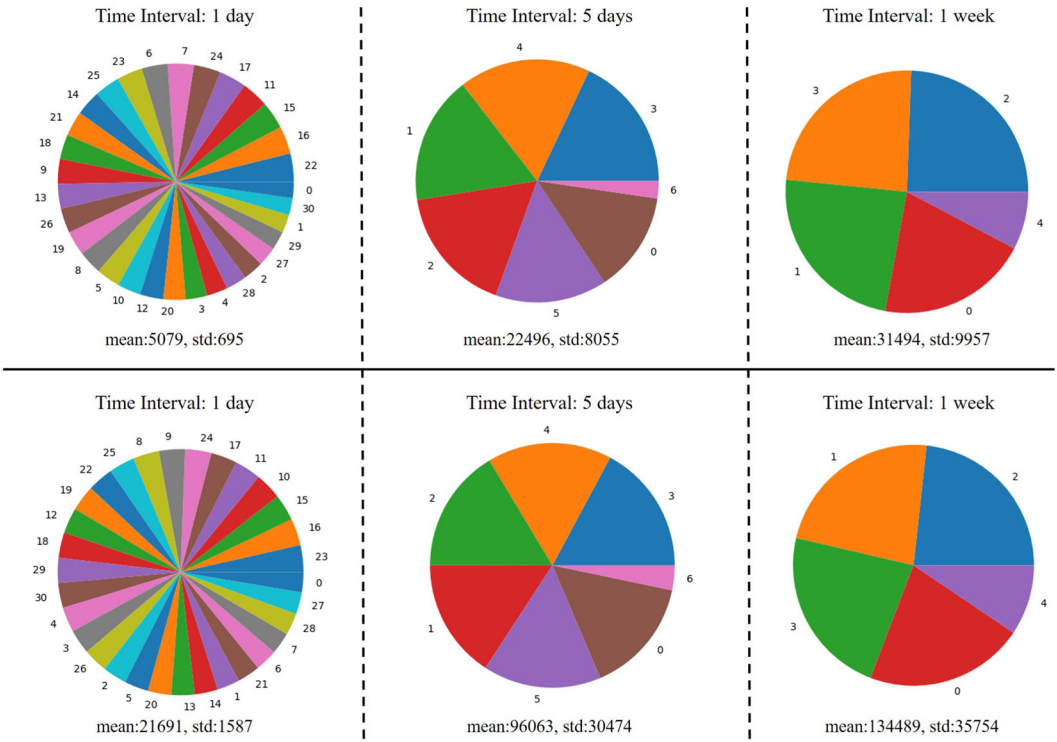


FIGURE A2 Events distribution is inhomogeneous with different uniform time intervals on Wikipedia and Reddit continuous data sets. Top, Wikipedia; bottom, Reddit. [Color figure can be viewed at wileyonlinelibrary.com]

We also observe that the distribution of events tends to be uniform on the temporal patch with a smaller time interval, as shown in Figure A2. The majority and minority of event numbers in the temporal patches with 1-day intervals are 6087 and 3499 on Wikipedia, while the numbers are 28,305 and 3651 for the one with 5-day intervals. Ideally, if we split the raw representations by each moment, the distribution of events will be almost homogeneous, and no information loss. However, the computation will be much heavier in future representation learning due to an enormous number of patches. Thus, it is crucial to balance the number and amount of temporal-topological structure of patches, which is discussed in Section 4.1.