

# Efficient-Dyn: Dynamic Graph Representation Learning via Event-based Temporal Sparse Attention Network

Yan Pang<sup>a</sup>, Chao Liu<sup>a,\*</sup>

<sup>a</sup>*Department of Electrical Engineering, University of Colorado Denver, CO*

## Abstract

Static graph neural networks have been widely used in modeling and representation learning of graph structure data. However, many real-world problems, such as social networks, financial transactions, recommendation systems, etc., are dynamic, that is, nodes and edges are added or deleted over time. Therefore, in recent years, dynamic graph neural networks have received more and more attention from researchers. In this work, we propose a novel dynamic graph neural network, Efficient-Dyn. It adaptively encodes temporal information into a sequence of patches with an equal amount of temporal-topological structure. Therefore, while avoiding the use of snapshots to 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 both 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-arts. Link prediction experiments are conducted on both continuous and discrete graph datasets. Through comparing with several state-of-the-art graph embedding baselines, the experimental results demonstrate that Efficient-Dyn has a faster inference speed while having competitive performance.

**Keywords:** dynamic graph neural network, representation learning, link prediction

## 1. Introduction

Recently, static graph neural networks (SGNNs) have seen a notable surge of interest with the encouraging technique for learning complicated systems of relations or interactions [1]. Unfortunately, abundant cases indicate that SGNN is progressively difficult to deal with the time-varied graph-based structure [2]. These kinds of graphs are challenging because their links and nodes may emerge and disappear along with moments. Since the dynamic graph neural networks (DGNN) append a new temporal dimension to accumulate the variation of embedding or representations, they become prevalent and powerful tools to employ in diverse fields, such as social media [3], bio-informatics [4], knowledge bases [5], brain neuroscience [6], protein-protein interaction networks [7], recommendation system [8], etc.

In order 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 [9, 10, 11, 12, 13]. Current researches [14, 15, 16, 17, 18, 19, 20] refine the raw dynamic representations to two main branches, dynamic continuous and discrete graphs. For the former graphs, the raw representations are projected to a single 2D temporal graph, storing most information in graph evolution. However, some temporal information is lost in this

process of projection [15] and the corresponding dynamic continuous networks are considerably complex [13]. In terms of the latter, discrete graphs, the structural representations are sampled to graph snapshots at regular time intervals, such as one day, over time. Although the developing networks are easier than the continuous ones, the temporal information is lost much more [12]. More details about time encoding approaches are discussed in Appendix 7.1. We hope to find a general-purpose manner 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 for avoiding information loss.

Once the refined temporal graphs are ready, the DGNNs extract and analyze patterns for graph learning along temporal dimension. It is crucial to have an efficient and powerful network under specific tasks in this step. Some researches [16, 19, 20] 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 [14, 15] adaptively designate divergent and interpretable attention to past embedding over time, the performance on long time duration is increased than RNN-based DGNNs. However, because the standard transformer [21] contains fully-connected

\*Corresponding author

Email address: chao.liu@ucdenver.edu (Chao Liu)

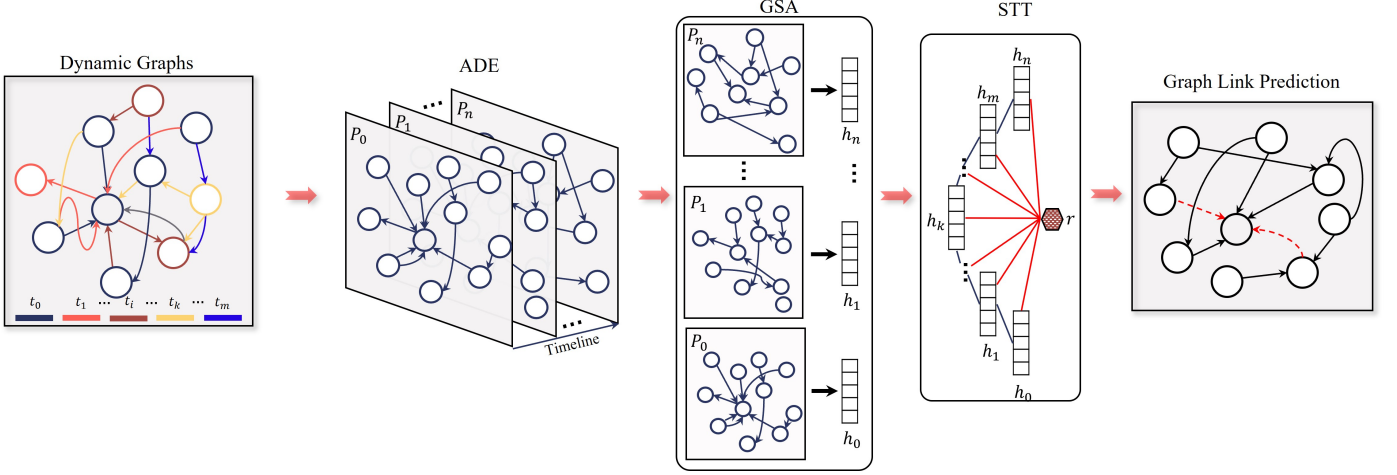


Figure 1: The overall architecture of Efficient-Dyn includes three main parts: Adaptive Data Encoding (ADE), Graph Structural Attention (GSA) and Sparse Temporal Transformer (STT).

attention conjunction  $N^2$ , where  $N$  is the number of temporal patches, it causes the heavy computation on a time-dependent sequence [22]. 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 in Efficient-Dyn called Sparse Temporal Transformer (STT) is proposed. Our lightweight STT simplifies the fully-connected attention conjunction and reduces the connections of patches to  $2N$ , which leads to far lower costs and acceptable performance than current standard transformer-based DGNNs.

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

In order to evaluate our proposed model, we conduct experiments on two continuous datasets [20] under inductive link prediction tasks. The experiments demonstrate that the proposed dynamic network significantly outperforms state-of-the-art networks on the continuous graph datasets. In addition, we also designed an abbreviated version of Efficient-Dyn, which consists of only GSA and STT. This abbreviated version is then utilized to learn the representation on the discrete datasets under both inductive and transductive link prediction tasks. Experiments show that the abbreviated version is still faster, more efficient, and has higher accuracy on four discrete dynamic graph datasets [23, 24, 25].

The contributions in this paper are summarized as follows:

- An innovative general-purpose DGNN, Efficient-Dyn, is proposed to trade off the accuracy and efficiency on the dynamic representations under link prediction tasks.
- We recommend a new approach, Adaptive Data Encoding, to preprocess the raw dynamic graph representations. The ADE can alleviate the information loss in the pro-

cess, and the corresponding developing networks are more effective in the representation learning tasks

- We propose a lightweight temporal self-attentional module called Sparse Temporal Transformer. The STT-based Efficient-Dyn can substantially reduce the computation by comparing RNN-based and standard transformer-based solutions on both continuous dynamic graph datasets.
- The abbreviated version of Efficient-Dyn, comprised only GSA and STT, can also be utilized on the discrete dynamic graph datasets. The experiments consistently demonstrate superior performance for Efficient-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 datasets and networks. Section 3 identifies critical terminology and fundamental concepts used in this paper. Section 4 describes the detailed mathematical process of Efficient-Dyn. The experiment design and results are discussed in Section 5. Finally, Section 6 presents our conclusion.

## 2. Related Work

### 2.1. Graph Representations

In general, graph representations can be categorized into two distinct levels: static and dynamic. The former includes only structural information, while the latter includes another critical parameter: time. The dynamic continuous representation contains node interactions [26], and timestamped edges [12], where instantaneous events are recorded into the raw graph representations, such as creation and removal of nodes and edges.

For representation learning, the raw representations should be preprocessed to dynamic graphs first. Current researches mainly focus on two dynamic graphs: the continuous and discrete graphs [9, 10, 12, 13]. 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 [11]. However, some temporal information is lost in the project processing. TGAT [15] adds temporal constraints to each interaction on the graph. However, the network is complicated because it has to extract temporal information at each moment, which is the general issue of the dynamic continuous graphs.

For the dynamic discrete graphs, the researches [9, 10, 11] group graph embedding with a certain temporal granularity over time. The discrete graphs include discrete equal time intervals, which can be represented with multiple snapshots along temporal dimension [27]. Because the temporal information is sampled at the discrete moments, such as one day/month, to several graph snapshots, the discrete representation is less complicated than a continuous representation [13]. However, this kind of graph tracking manner causes more information loss in the processing. 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.

In order to find a general-purpose time encoding approach and trade-off accuracy and efficiency, we propose the event-based ADE module 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 the time dimension on the static ones, the RNN-based DGNNs [18, 28] 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 [14]. In order to solve this issue, the transformer-based DGNNs [15, 14] are introduced to deal with the temporal information along the time dimension. TGAT [15] utilize a temporal graph attention layer to aggregate temporal-topological features on the continuous graph datasets. DySAT [14] generates a dynamic representation by joint self-attention of both structural and temporal information on discrete graph datasets. However, the common problem is that their computation is enormous on a long temporal sequence due to fully-connected attention conjunction of the standard transformer. Dynamic graph networks should achieve the desired trade-off between accuracy and efficiency under the graph representation learning tasks. In order to achieve this target, our proposed Efficient-Dyn which contains a lightweight module, STT. Instead of the fully-connected attention conjunction, 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 achieves a good or better performance 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 in this paper.

**DEFINITION 1. Dynamic Graph Neural Network.** *In general, a dynamic graph,  $DG = (V, E; T)$ , contains two main aspects: structural messages  $(V, E)$ , and temporal information  $T$ , where  $V$  and  $E$  are the sets of dynamic vertices and edges. Unlike static graph neural networks, nodes and edges emerge and disappear along the temporal dimension.*

**DEFINITION 2. Graph Link Prediction.** *Given a graph  $G = (V, E)$  with adjacency matrix  $A$ , the task is to estimate the link between nodes  $u$  and  $m$  by analyzing the aggregated information on both nodes.*

**DEFINITION 3. Inductive Learning.** *Given  $DG = (V, E; T)$ , the DGNN can only investigate the graph information from the beginning to time  $T - 1$ . The analyzed representations are utilized to predict the future links at time  $T$ . This task is prevalent and crucial since it makes predictions on unseen nodes and links in the future.*

**DEFINITION 4. Transductive Learning.** *Given  $DG = (V, E; T)$ , the DGNN can observe all nodes from the beginning to the end,  $T$ . The model learns representation and implements the task on each snapshot or moment.*

## 4. Methods

As shown in Figure 1, a general-purpose Efficient-Dyn includes three components connected serially. In order 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 graph learning, ADE adequately encodes the temporal information to 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

A continuous dynamic graph can be regarded as a particular static graph with an additional temporal dimension. Remarkably, all events are stored to the continuous representation along time, such as link addition, link deletion, node addition, node deletion, et al [11]. Recent researches [16, 29, 30] project the topological graph structures and node features from the continuous representation to the final 2D temporal representations for future representation learning. In order to record the evolution of graph structure over time, TGAT [15] introduce temporal constraints on neighborhood aggregation methods. However, the continuous representation of these networks is far complicated. Also, the data processing of these networks is low-efficiency because they pay much attention to the inoperative information for the target node. We hope to find a

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 Node $u$ 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

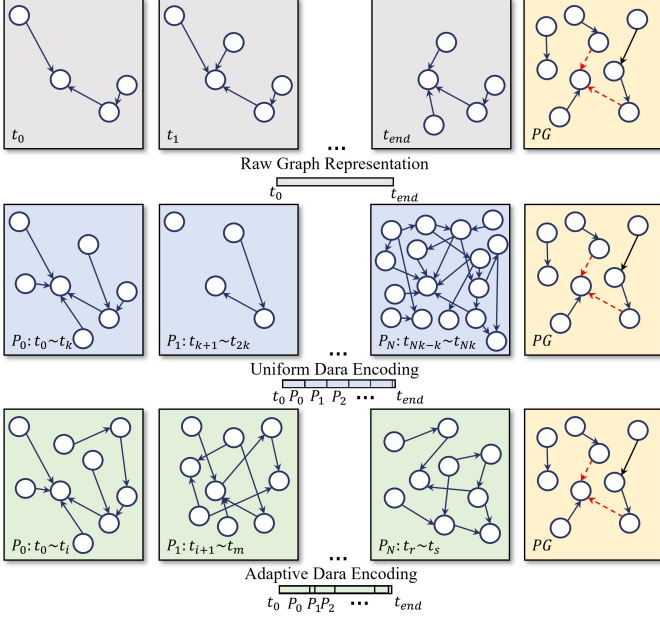


Figure 2: The comparison of three different encoding approaches Top row: raw graph representation; Second row: uniform data Encoding approach is to encode the time-dependent representation by the same time interval uniformly; Bottom row: Adaptive data Encoding approach adaptively encodes the whole raw representation to a sequence of event-based patches. Because each patch contains an equal amount of temporal-topological patterns, the processing is high-efficient in parallel computing. PG indicates the predicted graph.

general-purpose approach to simplify the continuous representation along the temporal dimension and speed up the computation for time-dependent structural patterns. Thus, in this work, an ADE module is proposed to adaptively encode temporal information into a set of event-based temporal patches with an equal amount of temporal-topological structure.

Figure 2 exposes the comparison of three encoding approaches: raw graph representation without encoding, uniform data encoding (UDE), and ADE. Based on the raw time-dependent representations with superabundant details, the UDE separates the graph patterns into several patches by the same temporal intervals, such as one day, one week, one month, et al. However, because the events have happened irregularly along the time dimension, 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 patches with fewer events. Thus, it is inefficient to pro-

cess the patches with different amounts of structural patterns in parallel computing.

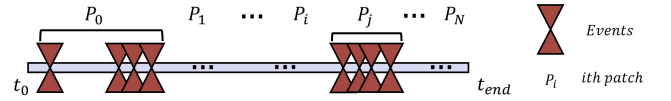


Figure 3: The adaptive data encoding module adaptively encodes temporal information into a sequence of patches by events along the temporal dimension. Each temporal patch contains an equal amount of temporal-topological patterns with the others.

As shown in Figure 3, the raw data are divided into  $N$  event-based patches along a temporal dimension by ADE. The amount of structural patterns of patch  $P_i$  is equal to the one of patch  $P_j$ . In this procedure, it is crucial to balance the number of patches,  $N$ , and the quality of structural embedding of each patch along the time dimension. If  $N$  is too large, it is also inefficient to analyze the graph structure due to minor embedding of each patch. A cost function of ADE is utilized to adaptively determine the number and the embedding of patches in Equation 1 and 2.

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

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

Where  $E$  is the embedding of the raw graph representations, and  $\tau$  is a constant parameter. The  $R$  is the number of total events, and  $N = \frac{R}{\Delta}$  is the number of patches.

In Equation 1, the first item is to minimize the difference between the quality of final projected representations of all patches. Ideally, when  $N = 1$ , the raw graph representations indicate the difference is the least, which is equal around to 0. The second item is encouraged to increase a large value of  $N$ . By minimizing the cost function of  $L_A$ , an optimum balance between  $N$  and  $E_i$  can finally be obtained.

In Equation 2, the function  $\phi$  is the Pearson Correlation Coefficient [31] to measure the linear correlation of embedding between each two patches. The  $\epsilon$  is a tiny constant parameter to ensure that each patch contains an equal amount of temporal-topological patterns.

#### 4.2. Graph Structural Attention

Since the raw graph representations are encoded to the optimal  $N$  time-dependent patches, the next step is to extract the lo-

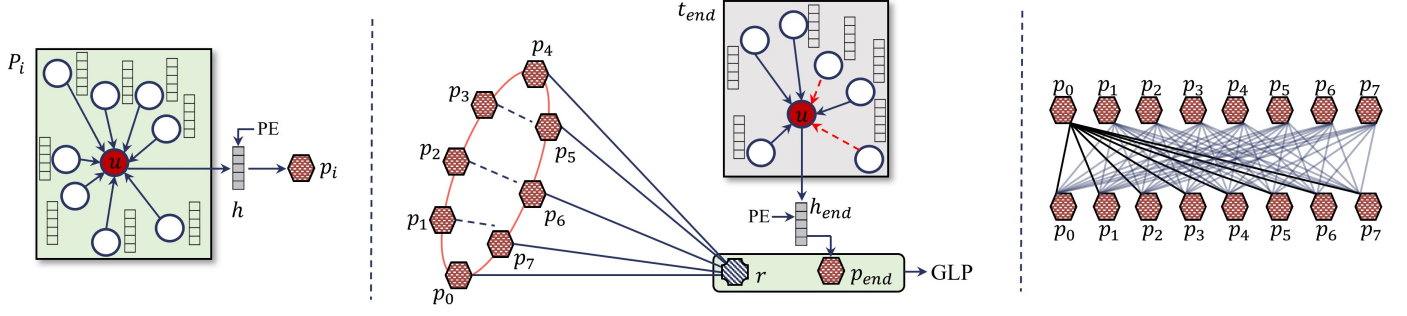


Figure 4: Left: GSA: The local structural patterns are extracted and added with the position embedding on each time-dependent patch in the GSA module. Middle: the lightweight STT: The global temporal patterns are investigated over time. Each patch is only connected to its two adjacent patches and the relay one. The updated relay patch is utilized to make the link prediction in the predicted patch. Right: Patch conveys information to each other due to the fully-connected attention conjunction of the standard transformer, which causes heavy computation.

cal structural patterns on each encoded patch. The input of GSA is a set of node representations of the current patch. Inspired by GAT [32], a node self-attention matrix,  $\alpha_{um}$ , is learned to determine the relevance between neighbors and the center node,  $u$ , on the time-dependent patch, as shown in Equation (3) and (4).

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

$$\alpha_{um} = \text{softmax}(e_{um}), \quad (4)$$

Where  $e_{um}$  indicates the relevance of neighbor node  $m$  to the center node  $u$ . The  $\sigma$  is the exponential linear unit (ELU) activation function [33]. The  $A$  is the adjacency matrix of the patch to indicate the linking relation of current patch, The  $\gamma$  indicates the self-attention mechanism, and  $W_p$  is the weight matrix of patch  $p$ . The  $x_u$  and  $x_m$  are the node representation of the center node  $u$  and neighbor node  $m$ .

Once the attention coefficients are received, the activation function  $\psi$  is applied to get the non-linear node representation of the current patch, as shown in Equation (5).

$$h_u = \psi \left( \sum_{m \in N_u} \alpha_{um} \cdot W_p X_m \right) \quad (5)$$

Where  $\psi$  is the Gaussian Error Linear Unit (GELU) [34] for the final output representations,  $h_u \in \mathbb{R}^d$  is the patch embedding, and  $d$  is the updated feature dimension.

Since the updated structural patterns of each patch are extracted, the next step 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 final output of GSA,  $p_i$ , contains both local structural patterns and temporal position information of the current patch.

#### 4.3. Sparse Temporal Transformer

Since the GSA module extracts the structural patterns of time-dependent patches, the next step is to gather the global evolution of structural patterns along the time dimension. Current transformer-based DGNNs [15, 14] utilize the standard transformer to extract the temporal patterns and receive good accuracy on both continuous and discrete dynamic representations.

However, due to the fully-connected attention conjunction as shown in Figure 4 Right, the computation is expensive of these transformer-based DGNNs on a long time sequence. In order to reduce the computation, we design a lightweight module, STT, instead of the standard transformer 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 functionality of the relay patch is to congregate and distribute 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  $p_i$  in Equation 6.

$$c_i(t) = [z_{i-1}(t-1); z_i(t-1); z_{i+1}(t-1); p_i; r(t-1)] \quad (6)$$

The temporal self-attention function of the current state  $z_i(t)$  of patch  $i$  is defined as in Equation 7.

$$z_i(t) = \phi_1(\beta_i(t) \cdot c_i(t) W_i) \quad (7)$$

$$\beta_i(t) = \text{softmax} \left( \frac{z_i(t-1) W_q \cdot (c_i(t) W_k)^T}{\sqrt{d}} \right) \quad (8)$$

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 operation [35] 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 9.

$$r(t) = \phi_2(\lambda(t) \cdot [r(t-1); Z(t)] W'_v) \quad (9)$$

$$\lambda(t) = \text{softmax} \left( \frac{r(t-1) W'_q \cdot ([r(t-1); Z(t)] W'_k)^T}{\sqrt{d'}} \right) \quad (10)$$

Where  $\lambda(t)$  is the self-attention coefficients of relay patch,  $W'_q, W'_k, W'_v$  are learnable parameters, and  $d'$  is the feature dimension of the relay. Both  $\phi_1$  and  $\phi_2$  are non-linear 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. In the inductive task, the representation of  $T-1$  temporal patches are analyzed in the training processing. The network makes the link prediction to the unseen nodes on the final predicted graph (PG). In this procedure, we utilize the deep walk [3] approach to sample some positive (connected links) and negative (unrelated links) on PG. A binary cross-entropy loss is to embolden positive cases to have similar representations while suppressing the negative ones in Equation 11.

$$L = \sum_{u \in V} \left( \sum_{v \in N_{walk}(u)} -\log(\varphi(\langle e_v, e_u \rangle)) - \omega_n \cdot \sum_{v' \in P_{walk}(u)} \log(\varphi(1 - \langle e_{v'}, e_u \rangle)) \right) \quad (11)$$

Where  $\varphi$  is the non-linear activation function,  $\langle \cdot \rangle$  is the inner-product,  $\omega_n$  is a constant fine-tuned hyper-parameter. The  $N_{walk}(u)$  is the sampled positive cases in fixed-length random deep walks on PG, while the  $P_{walk}(u)$  is the sampled negative cases.

In terms of 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 12.

$$L = \sum_{t=1}^{T_N} \sum_{u \in V} \left( \sum_{v \in N_{walk}^t(u)} -\log(\varphi(\langle e_v^t, e_u^t \rangle)) - \omega_n \cdot \sum_{v' \in P_{walk}^t(u)} \log(\varphi(1 - \langle e_{v'}^t, e_u^t \rangle)) \right) \quad (12)$$

## 5. Experiments

### 5.1. Datasets

We experimentally validate the general-purpose Efficient-Dyn on six real-world dynamic graph datasets: two continuous and four discrete datasets. Table 2 and Table 3 summarizes the statistics of the details of these six datasets.

**Reddit and Wikipedia** These two dynamic continuous graph datasets describe the active users and their editions on Reddit and Wikipedia in one month. The dynamic labels represent the state of the user on their editions. Reddit contains 10984 nodes

Table 2: Statistics of the continuous graph datasets

Datasets	Nodes	Links	Time Duration(s)
Reddit	10984	672447	2678390
Wikipedia	9227	157474	2678373

Table 3: Statistics of the discrete graph datasets

Datasets	Nodes	Links	Time Steps
Enron	143	2347	16
UCI	1809	16822	13
Yelp	6509	95361	12
ML-10M	20537	43760	13

and 672447 links, while Wikipedia contains 9227 nodes and 157474 links.

**Enron and UCI** These two dynamic discrete graph datasets describe the network communications. Enron includes 143 nodes (employees) and 2347 links (email interactions), while UCI includes 1809 nodes (users) and 16822 links (messages).

**Yelp and ML-10M** These two dynamic discrete graph datasets describe the bipartite networks from Yelp and MovieLens. The Yelp has 6509 nodes (users and businesses) and 95361 links (relationship), while ML-10M has 20537 nodes (users with the tags) and 43760 links (interactions).

### 5.2. Inductive learning on continuous datasets

These experiments compare different approaches on two continuous datasets under the inductive link prediction task. In these experiments, we compare our general-purpose networks with another four approaches as the baselines: GAT-T [32], GraphSAGE-LSTM [36], Const-TGAT [15], 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 collect the temporal patterns. In order to compare the performance of STT and standard transformer to collect the temporal information, we modify the architecture of TGAT with STT instead of the standard transformer and name it as Efficient-Dyn\*.

Table 4 shows the accuracy of these approaches under the link prediction task on two dynamic continuous datasets. 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. By comparing with TGAT and Efficient-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 Efficient-Dyn\* is less than TGAT in Table 7, which demonstrates STT is more effective by comparing the fully-connected connection of the standard transformer. The Efficient-Dyn's accuracy is less than TGAT by 2.03% and 1.73% on Reddit and Wikipedia datasets because TGAT utilizes more details with temporal constraints of graph representations. However, our inference speed is only around 0.6 times that of



Table 4: Inductive learning task results for link prediction on continuous datasets. Efficient-Dyn\* combines the functional time encoding components from TGAT and our STT components.

Datasets	GAT-T	GraphSAGE-L	Const-TGAT	TGAT	Efficient-Dyn*	Efficient-Dyn
Reddit	90.24	89.43	88.28	90.68	92.83	88.65
Wikipedia	84.76	82.43	83.60	85.28	87.36	83.55

Table 5: Inductive learning task results for link prediction on discrete graph datasets. Efficient-Dyn§ only consists of GSA and STT modules.

Datasets	node2vec	GraphSAGE	GAT	DynamicTriad	DynGEM	DynAERNN	DySAT	Efficient-Dyn§
Enron	75.86	74.67	69.25	68.77	62.85	59.63	78.52	<b>81.36</b>
UCI	74.76	79.41	73.78	71.67	79.82	81.91	83.72	<b>85.47</b>
Yelp	65.17	58.81	65.91	62.83	66.84	<b>73.46</b>	69.23	72.59
ML-10M	84.89	89.14	84.51	84.32	83.51	88.19	92.54	<b>94.28</b>

TGAT on both continuous datasets, which is more competitive in practical usage. These experiments demonstrate the contribution of Efficient-Dyn consisting of both EDA and STT on dynamic continuous representations under the link prediction task.

### 5.3. Inductive learning on discrete datasets

The previous experiments demonstrate the power of Efficient-Dyn on dynamic continuous datasets. Our proposed general-purpose network can also be utilized on discrete graph datasets. Since the discrete representations have several graph snapshots along temporal dimension, we compare our Efficient-Dyn§, which only consists of GSA and STT, with another seven baselines: node2vec [4], GraphSAGE, GAT, Dynamic Triad [17], DynGEM [37], DynAERNN [18] and DySAT. The node2vec handles second-order random walk sampling to grasp node representations. Dynamic Triad combines triadic closure to preserve both structural information and evolution patterns. DynGEM utilizes a deep autoencoder to generate non-linear embeddings of snapshots. DynGEM constructs both dense and recurrent layers to investigate the temporal graph evolution. DySAT extract node representations via fully-connected self-attention on both graph structural and temporal patterns.

Table 5 summarizes the results of these eight approaches on four dynamic discrete graph datasets. We find that the accuracy of DySAT exceeds the other state-of-the-art approaches, except Efficient-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, includes RNN-based DGNNs. By comparing DySAT and Efficient-Dyn§, we found the accuracy of Efficient-Dyn§ is 81.36%, 85.47%, 72.59%, and 94.28% on Enron, UCI, Yelp, and ML-10M datasets, which are better than DySAT. Meanwhile, the inference time of Efficient-Dyn§ is less than DySAT, which demonstrates that Efficient-Dyn§ is also competitive and effective on dynamic discrete graph representations.

### 5.4. Transductive learning on discrete datasets

Besides previous experiments, we also evaluate our general-purpose network on dynamic discrete datasets under the transductive link prediction task. From Table 6, we observe the

transformer-based DGNNs (DySAT and Efficient-Dyn§) have better performances on four discrete datasets, which also prove the self-attention architecture is powerful for transductive graph learning. As a result, the accuracy of Efficient-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 Efficient-Dyn.

## 6. Conclusion

This paper proposes a novel general-purpose dynamic graph neural network, Efficient-Dyn, that trade-offs the accuracy and efficiency under both inductive and transductive link prediction tasks. Efficient-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 avoids the information loss in the projection processing due to adaptive generation with a more delicate time granularity. 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. Benefited from the information delivery on the simplified architecture, the computation is further reduced compared with current state-of-the-art approaches. The Efficient-Dyn is evaluated on two continuous and four discrete graph datasets. The result illustrates that our proposed network is competitive and efficient on both inference speed and performance.

**Declaration of competing interest** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

- [1] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, M. Bronstein, Temporal graph networks for deep learning on dynamic graphs, arXiv preprint arXiv:2006.10637.
- [2] D. Fu, J. He, Sdg: A simplified and dynamic graph neural network, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 2273–2277.
- [3] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

Table 6: Transductive learning task results for link prediction on discrete graph datasets.

Datasets	node2vec	GraphSAGE	GAT	DynamicTriad	DynGEM	DynAERNN	DySAT	Efficient-Dyn§
Enron	83.05	81.88	75.97	78.98	69.72	72.01	86.60	<b>87.94</b>
UCI	80.49	82.89	81.86	80.28	79.82	83.52	85.81	<b>87.53</b>
Yelp	65.34	58.56	65.37	62.69	65.94	68.91	69.87	<b>72.01</b>
ML-10M	87.52	89.92	86.75	88.43	85.96	89.47	96.38	<b>97.52</b>

Table 7: Inference time (ms) on continuous graph datasets in the inductive learning task

Datasets	TGAT	Efficient-Dyn*	Efficient-Dyn
Reddit	30164.54554	23187.62019	18956.50313
Wikipedia	15377.43926	11637.05243	9623.40568

Table 8: Inference time (ms) on discrete graph datasets in the inductive learning task

Datasets	DySAT	Efficient-Dyn§
Enron	2.961	0.997
UCI	13.953	4.965
Yelp	1360.31	509.62
ML-10M	10678.36	3746.55

[4] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

[5] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 28, 2014.

[6] S. Goering, E. Klein, Fostering neuroethics integration with neuroscience in the brain initiative: Comments on the nih neuroethics roadmap, *AJOB neuroscience* 11 (3) (2020) 184–188.

[7] A. M. Fout, Protein interface prediction using graph convolutional networks, Ph.D. thesis, Colorado State University (2017).

[8] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 974–983.

[9] P. Cui, X. Wang, J. Pei, W. Zhu, A survey on network embedding, *IEEE Transactions on Knowledge and Data Engineering* 31 (5) (2018) 833–852.

[10] H. Cai, V. W. Zheng, K. C.-C. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, *IEEE Transactions on Knowledge and Data Engineering* 30 (9) (2018) 1616–1637.

[11] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, P. Poupart, Representation learning for dynamic graphs: A survey., *J. Mach. Learn. Res.* 21 (70) (2020) 1–73.

[12] C. D. Barros, M. R. Mendonça, A. B. Vieira, A. Ziviani, A survey on embedding dynamic graphs, *arXiv preprint arXiv:2101.01229*.

[13] J. Skardinga, B. Gabrys, K. Musial, Foundations and modelling of dynamic networks using dynamic graph neural networks: A survey, *IEEE Access*.

[14] A. Sankar, Y. Wu, L. Gou, W. Zhang, H. Yang, 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, pp. 519–527.

[15] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, K. Achan, Inductive representation learning on temporal graphs, *arXiv preprint arXiv:2002.07962*.

[16] Y. Ma, Z. Guo, Z. Ren, J. Tang, D. Yin, Streaming graph neural networks, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 719–728.

[17] L. Zhou, Y. Yang, X. Ren, F. Wu, Y. Zhuang, Dynamic network embedding by modeling triadic closure process, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.

[18] P. Goyal, S. R. Chhetri, A. Canedo, dyngraph2vec: Capturing network dynamics using dynamic graph representation learning, *Knowledge-Based Systems* 187 (2020) 104816.

[19] J. Chen, X. Xu, Y. Wu, H. Zheng, Gc-lstm: Graph convolution embedded lstm for dynamic link prediction, *arXiv preprint arXiv:1812.04206*.

[20] S. Kumar, X. Zhang, J. Leskovec, Predicting dynamic embedding trajectory in temporal interaction networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1269–1278.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[22] Q. Guo, X. Qiu, P. Liu, Y. Shao, X. Xue, Z. Zhang, Star-transformer, *arXiv preprint arXiv:1902.09113*.

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

[24] B. Klimt, Y. Yang, The enron corpus: A new dataset for email classification research, in: European Conference on Machine Learning, Springer, 2004, pp. 217–226.

[25] P. Panzarasa, T. Opsahl, K. M. Carley, 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.

[26] M. Latapy, T. Viard, C. Magnien, Stream graphs and link streams for the modeling of interactions over time, *Social Network Analysis and Mining* 8 (1) (2018) 1–29.

[27] A. Taheri, K. Gimpel, T. Berger-Wolf, Learning to represent the evolution of dynamic graphs with recurrent models, in: Companion Proceedings of The 2019 World Wide Web Conference, 2019, pp. 301–307.

[28] E. Hajiramezanali, A. Hasanzadeh, N. Duffield, K. R. Narayanan, M. Zhou, X. Qian, Variational graph recurrent neural networks, *arXiv preprint arXiv:1908.09710*.

[29] R. Trivedi, M. Farajtabar, P. Biswal, H. Zha, Dyrep: Learning representations over dynamic graphs, in: International conference on learning representations, 2019.

[30] Z. Han, J. Jiang, Y. Wang, Y. Ma, V. Tresp, 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.

[31] J. Benesty, J. Chen, Y. Huang, I. Cohen, Pearson correlation coefficient, in: Noise reduction in speech processing, Springer, 2009, pp. 1–4.

[32] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, *arXiv preprint arXiv:1710.10903*.

[33] L. Trottier, P. Giguere, B. Chaib-Draa, Parametric exponential linear unit for deep convolutional neural networks, in: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2017, pp. 207–214.

[34] D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus), *arXiv preprint arXiv:1606.08415*.

[35] D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, A. Courville, C. Pal, Zoneout: Regularizing rnns by randomly preserving hidden activations, *arXiv preprint arXiv:1606.01305*.

[36] W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 1025–1035.

[37] P. Goyal, N. Kamra, X. He, Y. Liu, Dyngem: Deep embedding method for dynamic graphs, *arXiv preprint arXiv:1805.11273*.

[38] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feed-forward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, *The journal of machine learning research* 15 (1) (2014) 1929–1958.



## 7. Appendix

### 7.1. Event-based Data Encoding

This section is to discuss the event-based data encoding approaches on raw dynamic representations. As shown in Figure 5.a, the raw dynamic continuous representation is a sequence of particular static graphs along time dimension, which stores all events, such as node emerging, node disappearing, link addition, link removing, et al. 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 5.b. The primary issue is that some temporal information is lost on the single graph, including node or edge vanishing and multi-edge 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.

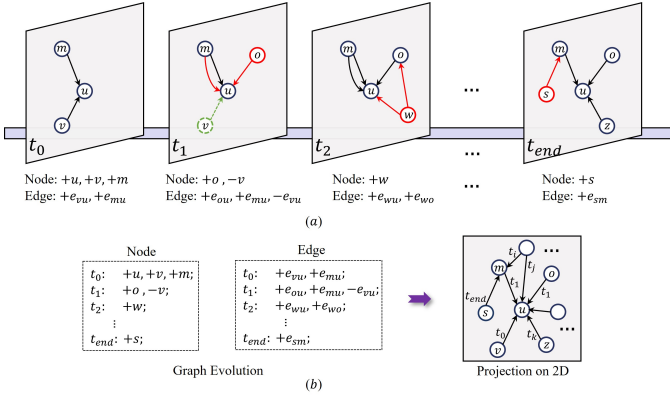


Figure 5: Visual illustration for projection from the raw dynamic continuous representations. a). The generation process of a continuous temporal graph and its snapshots at each moment. The solid red line represents an addition, and the dash green one represents deletion. b). The temporal information and the final state of the projected temporal graph. Some temporal information for nodes and multi-edges is lost.

In order 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 one day, one week, one month, et al. However, it is impossible to guarantee that events are uniformly distributed along the time dimension. With the uniform data encoding approach, some patches contain superabundant details due to more events in the corresponding duration and vice versa.

As shown in Figure 6, We design some experiments to verify the above phenomenon. The top row of Figure 6 is the distribution of events at each temporal patch on Wikipedia with three uniform time intervals: one day, five days, and one 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 one-week interval is 9957, far outweighing the one with a one-day interval. A similar phenomenon emerges on Reddit, as shown in the bottom row of Figure 6. The calculation is also inefficient on these inhomogeneous patches in the future representation learning in parallel computing.

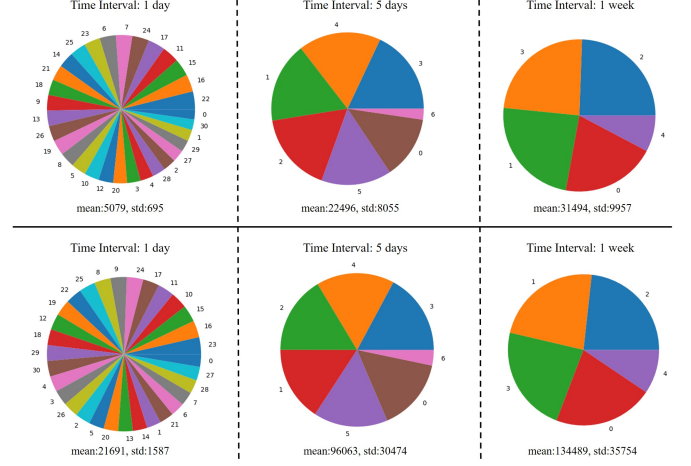


Figure 6: The events distribution is inhomogeneous with different uniform time intervals on Wikipedia and Reddit continuous datasets. Top: Wikipedia; Bottom: Reddit.

Our proposed adaptive data encoding approach adaptively encodes temporal information into a sequence of patches by events. The advantage of this approach is to avoid using snapshots to cause information loss and achieve 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.

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 6. The majority and minority of event numbers in the temporal patches with one-day intervals are 6087 and 3499 on Wikipedia, while the numbers are 28305 and 3651 of the one with five-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.

### 7.2. Multi-head Attention Mechanism

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

**The structural multi-head attention for GSA** Since the multi-head 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 13.

$$h_u = \bowtie (h_u^1, h_u^2, \dots, h_u^k) \quad (13)$$

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

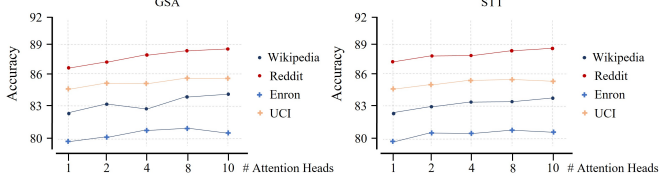


Figure 7: The accuracy of Efficient\_Dyn with different multi-heads on four datasets in the inductive link prediction tasks

**The temporal multi-head 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 Equation 14 and 15.

$$z_i(t) = \bowtie (z_i^1(t), z_i^2(t), \dots, z_i^k(t)) \quad (14)$$

$$r(t) = \bowtie (r^1(t), r^2(t), \dots, r^k(t)) \quad (15)$$

In order to evaluate the contribution of the multi-head attention mechanism, we set a series of experiments for Efficient\_Dyn with different head numbers independently in the range 1, 2, 4, 8, 10 on two continuous and two discrete dynamic graph datasets. As shown in Figure 7, it can be observed that the accuracy of multi-head networks is better than the single-head networks on all four datasets. In addition, the accuracy does not keep increasing with a larger number of heads. The performance of the multi-head network stabilizes with eight attentions heads for both modules.

### 7.3. Experimental Setup

At the beginning of training, the Xavier [38] is to initialize the learnable parameters  $W$  of each layer, which is to avoid the gradient from exploding or vanishing suddenly. The Gaussian Error Linear Unit (GELU) yields the final output non-linear representations at the end of GSA. The exponential linear unit (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 of Reddit and Wikipedia are 16 and 12, finally. In addition, the dropout approach [39] is introduced to avoid over-fitting during the training process, with the dropout rate in a range 0.3 to 0.7, which depends on the dataset and tasks.