DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks

Aravind Sankar[*], Yanhong Wu[†], Liang Gou[†], Wei Zhang[†], Hao Yang[†]
[*]University of Illinois at Urbana-Champaign, IL, USA
[†]Visa Research, Palo Alto, CA, USA
[*]asankar3@illinois.edu [†]{yanwu, ligou, wzhan, haoyang}@visa.com

ABSTRACT

Learning node representations in graphs is important for many applications such as link prediction, node classification, and community detection. Existing graph representation learning methods primarily target static graphs while many real-world graphs evolve over time. Complex time-varying graph structures make it challenging to learn informative node representations over time.

We present Dynamic Self-Attention Network (DySAT), a novel neural architecture that learns node representations to capture dynamic graph structural evolution. Specifically, DySAT computes node representations through joint self-attention along the two dimensions of *structural neighborhood* and *temporal dynamics.* Compared with state-of-the-art recurrent methods modeling graph evolution, dynamic self-attention is efficient, while achieving consistently superior performance. We conduct link prediction experiments on two graph types: communication networks and bipartite rating networks. Experimental results demonstrate significant performance gains for DySAT over several state-of-the-art graph embedding baselines, in both single and multi-step link prediction tasks. Furthermore, our ablation study validates the effectiveness of jointly modeling structural and temporal self-attention.

CCS CONCEPTS

• **Information systems → Social networks**; • **Computing methodologies → Neural networks**;

KEYWORDS

Dynamic Graphs, Self-Attention, Representation Learning

ACM Reference Format:
Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang. 2020. DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20), February 3–7, 2020, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3336191.3371845

1 INTRODUCTION

Learning latent representations (or embeddings) of nodes in graphs is a fundamental problem due to its prevalence in varied domains including social media [22], bioinformatics [8], and knowledge bases [36]. The objective is to learn low-dimensional vectors that capture the structural properties of a node and its neighborhoods. Such embeddings can benefit a plethora of applications, including node classification, link prediction, and recommendation [8, 15]. Existing graph representation learning work primarily target static graphs, which contain a fixed set of nodes and edges. However, many real-world graphs are *dynamic* where graph structures constantly evolve over time. They are usually represented as a sequence of graph snapshots at different time steps [16]. Examples include co-authorship networks where authors may periodically switch collaboration behaviors and email communication networks whose interaction structures may change dramatically due to sudden events.

Learning dynamic node representations is challenging due to the complex time-varying graph structures: nodes can emerge and leave, links can appear and disappear, and communities can merge and split. This requires the learned node representations to not only preserve *structural proximity* but also jointly capture their *temporal evolution.* In addition, multiple latent *facets* affect graph evolution, *e.g.*, in a co-authorship network, authors of different research communities or at different career stages, may expand their collaboration circles at varying rates. Latent facets such as research community or career stage, serve as unique perspectives to model temporal graph evolution. Thus, modeling *multi-faceted* variations in dynamic graph representation learning is crucial towards accurately predicting node properties and future links.

Existing dynamic graph representation learning methods mainly fall into categories: temporal regularizers that enforce smoothness of node representations from adjacent snapshots [39, 40], and recurrent neural networks [6, 9] that summarize historical snapshots via hidden states. Smoothing methods while effective in high-sparsity settings, may fail when nodes exhibit significantly distinct evolutionary behaviors. Conversely RNNs, while expressive, require large amounts of training data to outperform even static methods and scale poorly with an increase in the number of time steps.

Attention mechanisms have recently achieved great success in many sequential learning tasks [2, 37]. The underlying principle is to learn a function that aggregates a variable-sized input while focusing on the parts relevant to a given context. When a single sequence is used as both the input and context, it is called *self-attention.* Though attention mechanisms were initially designed to facilitate Recurrent Neural Networks (RNNs) to capture long-range dependencies, Vaswani et al. [33] demonstrate the efficacy of a pure

[*]Work done while at Visa Research

WSDM '20, February 3–7, 2020, Houston, TX, USA

ACM ISBN 978-1-4503-6822-3/20/02…$15.00
https://doi.org/10.1145/3336191.3371845

self-attentional network in achieving state-of-the-art performance in machine translation, while being significantly more efficient.

As dynamic graphs usually have periodical patterns such as recurrent links or communities, self-attentions can draw context from all past graph snapshots to adaptively assign interpretable weights for previous time steps. In this paper, we present a novel neural architecture named Dynamic Self-Attention Network (DySAT), to learn latent node representations on dynamic graphs. DySAT generates a dynamic node representation by joint self-attention along two dimensions: *structural neighborhoods* and *temporal dynamics*. Structural attention extracts features from local node neighborhoods in each snapshot through self-attentional aggregation, while temporal attention captures graph evolution over multiple time steps by flexibly weighting historical representations. To model *multi-faceted* variations in graph structures, we learn multiple attention heads in both structural and temporal attention layers that enable joint attention over different latent subspaces.

We conduct experiments on single-step and multi-step link prediction, using four benchmarks of different sizes including two communication networks [14, 21] and two rating networks [11]. Our results indicate significant gains (4.8% macro-AUC on average) for DySAT over several state-of-the-art baselines and consistent performance over time steps. We demonstrate the benefits of joint structural and temporal attention through an ablation study and visualize temporal attention weights to illustrate the capability of DySAT in adapting to datasets with varying evolutionary trends. We summarize the key contributions of this paper below:

- We propose a novel neural architecture named Dynamic Self-Attention Network that utilizes joint structural and temporal self-attention for dynamic graph representation learning.
- We propose a modular architectural design of DySAT with stacked structural and temporal self-attentional layers, which enables efficient computations compared with RNN-based solutions.
- We conduct a wide spectrum of experiments that demonstrate consistently superior performance for DySAT over state-of-the-art methods in single and multi-step link prediction tasks.

2 RELATED WORK

Our work is related to representation learning on static graphs, dynamic graphs, and recent advances in self-attentional architectures.

Static graph embeddings. Early unsupervised graph representation learning work exploit spectral graph properties to perform dimensionality reduction [30]. To improve scalability, skip-gram was utilized to learn node embeddings that maximize the likelihood of co-occurrence in fixed-length random walks [8, 22, 23, 26]. Recently, several graph neural network architectures have achieved tremendous success, among which many methods are designed for supervised or semi-supervised learning tasks [3, 13, 19, 25, 26, 34]. Hamilton et al. [10], Veličković et al. [35] extend graph convolutional methods through trainable neighborhood aggregation functions, to propose a general framework applicable to unsupervised graph representation learning. However, these methods are not designed to model temporal evolution in dynamic graphs.

Dynamic graph embeddings. Dynamic graphs are defined in two common ways: *snapshot sequence* [16], which is a collection of evolving graph snapshots at multiple discrete time steps; and *timestamped graph* [31], which is a single graph with continuous-valued timestamped links. Though snapshot-based methods may be applied to a timestamped graph by suitably creating snapshots, the converse is infeasible due to lack of fine-grained timestamps. In this paper, we target representation learning over graph snapshots.

Existing dynamic graph embedding techniques fall into two broad categories. *Temporal smoothness* methods ensure embedding stability across consecutive time-steps [27, 40]. Zhou et al. [39] additionally use the concept of triadic closure as guidance in social networks. Goyal et al. [7] explore graph-autoencoders to incrementally update node embeddings through initialization from the previous step. However, these methods cannot capture long-range variations in graph structure, and are inadequate when nodes exhibit vastly differing evolutionary behaviors. *Recurrent methods* [6, 9] capture temporal dynamics by maintaining hidden states to summarize historical snapshots, and achieve state-of-the-art results on dynamic link prediction. However, recurrent methods scale poorly with the increase in number of time-steps and cannot model multi-faceted graph evolution without encountering prohibitive costs. In contrast, DySAT captures the most relevant historical contexts through efficient self-attentions, to learn dynamic node representations.

A related problem is representation learning in streaming graphs where the objective is to improve efficiency over repeatedly retraining static models, by proposing incremental temporal updates [4, 17, 18, 38]. In contrast, our goal is to improve *representation quality* by exploiting the temporal evolution of graph structure.

In the scenario of *timestamped graphs*, representative methods include temporal random walks [20], and neural extensions of temporal point-processes [31, 32, 42]. As discussed earlier, this scenario is orthogonal to our setting and further, such techniques cannot be applied to graph snapshots that lack fine-grained timestamps.

Self-attention mechanisms. Recent advances in many NLP tasks have demonstrated the superiority of *self-attention* both in efficiency and performance [28, 29, 33]. A related approach is Graph Attention Network (GAT) [34], which employs neighborhood attention for node classification on static graphs. We propose joint self-attention on structural and temporal domains to learn dynamic node representations that capture its evolving neighborhoods.

3 PROBLEM DEFINITION

We formally define the problem of dynamic graph representation learning. A dynamic graph is defined as a series of observed static graph snapshots, $\mathbb{G} = \{\mathcal{G}^1, \ldots, \mathcal{G}^T\}$ where T is the number of time steps. Each snapshot $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}^t)$ is a weighted undirected graph with a shared node set $\mathcal{V}$, a link set $\mathcal{E}^t$, and a weighted adjacency matrix $\boldsymbol{A}^t$ at time step t. Unlike some previous methods that assume links can only be added in dynamic graphs, we also support removal of links over time. Dynamic graph representation learning aims to learn latent representations $\boldsymbol{e}_v^t \in \mathbb{R}^d$ for each node $v \in \mathcal{V}$ at time steps $t = \{1, 2, \ldots, T\}$, such that $\boldsymbol{e}_v^t$ preserves both the local graph structures centered at v and its temporal evolutionary behaviors such as link connection and removal up to time step t.

4 DYNAMIC SELF-ATTENTION NETWORK

In this section, we first present the major components or building blocks of DySAT. As depicted in Figure 1, DySAT has three modules from its top to bottom: *structural* attention block; *temporal* attention block; and *graph context* prediction. Our key innovation lies in decoupling graph evolution into two *modular* blocks, which enables highly efficient computations of dynamic node representations. The *structural* and *temporal self-attention* layers together model graph evolution, and can realize graph neural networks of arbitrary complexity through layer stacking. Finally, we present our proposed neural architecture DySAT, built upon these modules.

4.1 Structural Self-Attention

The input of this layer is a graph snapshot $\mathcal{G} \in \mathbb{G}$ and a set of input node representations $\{\boldsymbol{x}_v \in \mathbb{R}^D, \forall v \in \mathcal{V}\}$ where D is the input embedding dimension. The input to the initial layer is set as one-hot encoded vectors for each node. The output is a new set of node representations $\{\boldsymbol{z}_v \in \mathbb{R}^F, \forall v \in \mathcal{V}\}$ with F dimensions, that capture the local structural properties in snapshot $\mathcal{G}$.

Specifically, the *structural* self-attention layer attends over the immediate neighbors of a node v (in snapshot $\mathcal{G}$), by computing attention weights as a function of their input node embeddings. The operation of the structural attention layer is defined as:

$$\boldsymbol{z}_v = \sigma\Big(\sum_{u \in \mathcal{N}_v} \alpha_{uv} \boldsymbol{W}^s \boldsymbol{x}_u\Big), \quad \alpha_{uv} = \frac{\exp(e_{uv})}{\sum\limits_{w \in \mathcal{N}_v} \exp(e_{wv})} \tag{1}$$

$$e_{uv} = \sigma\Big(A_{uv} \cdot \boldsymbol{a}^T [\boldsymbol{W}^s \boldsymbol{x}_u || \boldsymbol{W}^s \boldsymbol{x}_v]\Big) \quad \forall (u,v) \in \mathcal{E}$$

where $\mathcal{N}_v = \{u \in \mathcal{V} : (u,v) \in \mathcal{E}\}$ is the set of immediate neighbors of node v in snapshot $\mathcal{G}$; $\boldsymbol{W}^s \in \mathbb{R}^{F \times D}$ is a shared weight transformation applied to each node in the graph; $\boldsymbol{a} \in \mathbb{R}^{2D}$ is a weight vector parameterizing the attention function implemented as feed-forward layer; $||$ is the concatenation operation and $\sigma(\cdot)$ is a non-linear activation function. Note that A_{uv} is the weight of link (u,v) in the current snapshot $\mathcal{G}$. The set of learned coefficients α_{uv}, obtained by a softmax over the neighbors of each node in $\mathcal{V}$, indicate the contribution of node u to node v at the current snapshot. We use a LeakyRELU non-linearity to compute attention weights, followed by an exponential linear unit (ELU) activation for the output representations. We employ sparse matrices to efficiently implement *masked* self-attention over neighbors, since α_{uv} is zero for all non-links in $\mathcal{G}$. Thus, a structural attention layer applied on a snapshot $\mathcal{G}$ outputs node embeddings, through a self-attentional aggregation of neighboring node embeddings, which can be viewed as a single message passing round among immediate neighbors.

4.2 Temporal Self-Attention

To further capture temporal evolutionary patterns in a dynamic graph, we design a temporal self-attention layer. The input of this layer is a sequence of representations for a particular node v at different time steps. The input representations are assumed to sufficiently capture local structural information at each time step, which enables a modular separation of structural and temporal modeling. Specifically, for each node v, we define the input as $\{\boldsymbol{x}_v^1, \boldsymbol{x}_v^2, \ldots, \boldsymbol{x}_v^T\}, \boldsymbol{x}_v^t \in \mathbb{R}^{D'}$ where T is the total number of time steps and D' is the dimensionality of the input representations. The layer output is a new representation sequence for v at each time step, *i.e.*, $\boldsymbol{z}_v = \{\boldsymbol{z}_v^1, \boldsymbol{z}_v^2, \ldots, \boldsymbol{z}_v^T\}, \boldsymbol{z}_v^t \in \mathbb{R}^{F'}$ with dimensionality F'. We denote the input and output representations of v, packed together across time, by $\boldsymbol{X}_v \in \mathbb{R}^{T \times D'}$ and $\boldsymbol{Z}_v \in \mathbb{R}^{T \times F'}$ respectively.

The key objective of the temporal self-attentional layer is to capture the temporal variations in graph structure over multiple time steps. The input representation of node v at time-step t, $\boldsymbol{x}_v^t$, encodes the current local structure around v (equation 1) We use $\boldsymbol{x}_v^t$ as the query to attend over its historical representations ($< t$), tracing the evolution of the local neighborhood around v. In contrast to structural attention which operates on the representations of neighboring nodes, temporal attention depends entirely on the temporal history of each node, thus facilitating efficient parallelism across nodes. The decoupling of local neighborhood and temporal history of each node into independent layers, is one of the key factors contributing to the efficiency of our model.

To compute the output representation of node v at t, we use the scaled dot-product form of attention [33] where the queries, keys, and values are set as the input node representations. The queries, keys, and values are first transformed to a different space through linear projection matrices $\boldsymbol{W}_q \in \mathbb{R}^{D' \times F'}$, $\boldsymbol{W}_k \in \mathbb{R}^{D' \times F'}$ and $\boldsymbol{W}_v \in \mathbb{R}^{D' \times F'}$ respectively. We allow each time-step t to attend over all time-steps up to and including t, to preserve the auto-regressive property. The temporal self-attention function is defined as:

$$\boldsymbol{Z}_v = \boldsymbol{\beta_v}(\boldsymbol{X}_v \boldsymbol{W}_v), \qquad \beta_v^{ij} = \frac{\exp(e_v^{ij})}{\sum\limits_{k=1}^{T} \exp(e_v^{ik})},$$

$$e_v^{ij} = \Big(\frac{((\boldsymbol{X}_v \boldsymbol{W}_q)(\boldsymbol{X}_v \boldsymbol{W}_k)^T)_{ij}}{\sqrt{F'}} + M_{ij}\Big) \tag{2}$$

where $\boldsymbol{\beta_v} \in \mathbb{R}^{T \times T}$ is the attention weight matrix obtained by the multiplicative attention function and $\boldsymbol{M} \in \mathbb{R}^{T \times T}$ is a mask matrix with each entry $M_{ij} \in \{-\infty, 0\}$ to enforce the auto-regressive property. To encode the temporal order, we define $\boldsymbol{M}$ as:

$$M_{ij} = \begin{cases} 0, & i \leq j \\ -\infty, & \text{otherwise} \end{cases}$$

When $M_{ij} = -\infty$, the softmax results in a zero attention weight, *i.e.*, $\beta_v^{ij} = 0$, which switches off the attention from time-step i to j.

4.3 Multi-Faceted Graph Evolution

By stacking structural and temporal self-attentional layers, our approach can sufficiently capture a single type or *facet* of graph evolution. However, real-world dynamic graphs typically evolve along multiple latent facets, *e.g.*, evolution of users' movie-watching preferences across different *genres* (such as sci-fi, comedy, etc.) exhibit significantly distinct temporal trends. Thus, we endow our model with expressivity to capture dynamic graph evolution from different latent perspectives through multi-head attentions [33].

Multi-head attention, which creates multiple independent instances of the attentional function named *attention heads* that operate on different portions of the input embedding, is widely utilized to improve the diversity of attention mechanisms. This facilitates joint attention over different subspaces through projections along multiple latent facets. Multi-head attention also improves model capacity and stability by avoiding prediction bias. We use multiple attention heads in both structural and temporal layers:

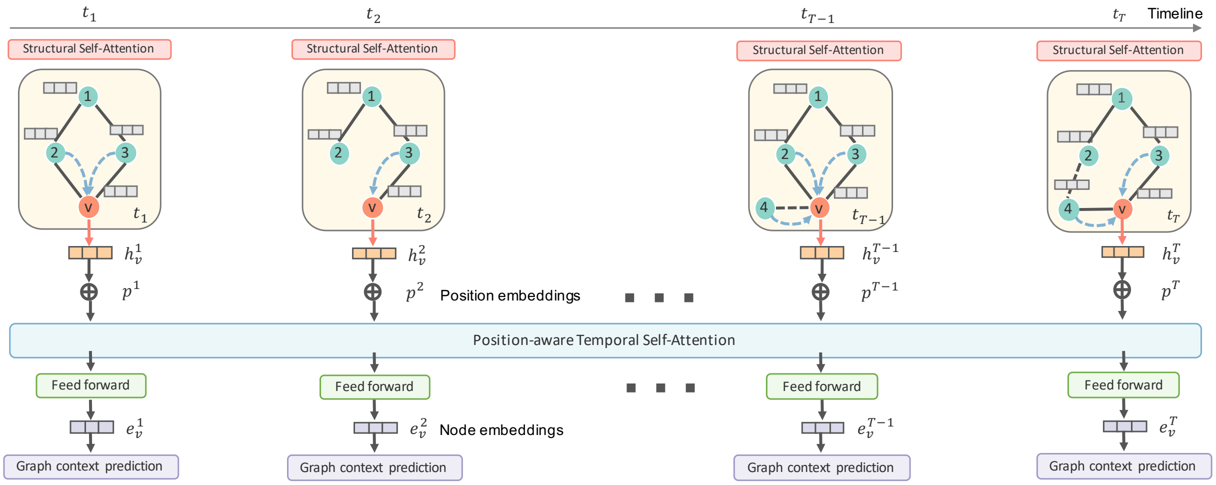

Figure 1: Neural architecture of DySAT: we employ structural attention layers followed by temporal attention layers. Dashed black lines indicate new links and dashed blue arrows refer to neighbor-based structural-attention.

***Structural multi-head self-attention*:** multiple attention heads are computed in each structural attention layer (one per facet), followed by concatenation to compute output representations.

$$\boldsymbol{h}_v = \text{Concat}(z_v^1, z_v^2, \ldots, z_v^{H_S}) \quad \forall v \in V$$

where H_S is the number of attention heads, and $\boldsymbol{h}_v \in \mathbb{R}^F$ is the output of node v after multi-head attention. Note that while structural attention is applied independently at each snapshot, the parameters of structural attention heads are shared across different snapshots.

***Temporal multi-head self-attention*:** similar to the above setting, multiple temporal attention heads (or facets) are computed over historical time steps, to compute final node representations.

$$\boldsymbol{H}_v = \text{Concat}(Z_v^1, Z_v^2, \ldots, Z_v^{H_T}) \quad \forall v \in V$$

where H_T is the number of temporal attention heads, and $\boldsymbol{H}_v \in \mathbb{R}^{T \times F'}$ is the output of temporal multi-head attentions.

4.4 DySAT Architecture

In this section, we present our neural architecture DySAT for dynamic graph representation learning, that uses the defined *structural* and *temporal* self-attention layers as fundamental modules. The input is a collection of T graph snapshots, and the outputs are node representations at each time step. As illustrated in Figure 1, DySAT consists of a *structural* block followed by a *temporal* block, where each block contains multiple stacked layers of the corresponding layer type. The *structural* block extracts features from higher-order local neighborhoods of each node through a self-attentional aggregation and stacking, to compute intermediate node representations for each snapshot. This sequence of node representations then feeds as input to the *temporal* block, which attends over multiple historical time steps, capturing temporal variations in the graph structure. The outputs of temporal block comprise the set of final dynamic node representations, which are optimized to preserve local graph context in each time step. In the rest of this section, we describe each component of DySAT in detail.

Structural attention block. This module is composed of multiple stacked structural self-attention layers to extract features from nodes at different distances. We apply each layer on each graph snapshot with shared parameters, as illustrated in Figure 1, to capture the local structure around a node at each time step.

Note that the embeddings input to a layer can potentially vary across different snapshots. We denote the node representations output by the structural attention block, as $\{\boldsymbol{h}_v^1, \boldsymbol{h}_v^2, \ldots, \boldsymbol{h}_v^T\}, \boldsymbol{h}_v^t \in \mathbb{R}^F$, which feed as input to the *temporal* attention block.

Temporal attention block. First, we capture the ordering information in the temporal attention module by using *position* embeddings [5], $\{\boldsymbol{p}^1, \ldots, \boldsymbol{p}^T\}, \boldsymbol{p}^t \in \mathbb{R}^F$, which embed the absolute temporal position of each snapshot. The position embeddings are then combined with the output of the structural attention block to obtain a sequence of input representations: $\{\boldsymbol{h}_v^1 + \boldsymbol{p}^1, \boldsymbol{h}_v^2 + \boldsymbol{p}^2, \ldots, \boldsymbol{h}_v^T + \boldsymbol{p}^T\}$ for node v across multiple time steps. This block also follows a similar structure with multiple stacked temporal self-attention layers. The final layer outputs pass into a position-wise *feed-forward* layer to give the final node representations $\{\boldsymbol{e}_v^1, \boldsymbol{e}_v^2, \ldots, \boldsymbol{e}_v^T\}$ $\forall v \in V$.

Graph context prediction. To enable the learned representations to capture structural evolution, our objective function preserves the local structure around a node across multiple time steps. We use the dynamic representation of a node v at time step t, $\boldsymbol{e}_v^t$ to preserve local proximity around v at t. In particular, we use a binary cross-entropy loss at each time step to encourage nodes co-occurring in fixed-length random walks, to have similar representations.

$$L = \sum_{t=1}^{T} \sum_{v \in \mathcal{V}} \Big(\sum_{u \in \mathcal{N}_{walk}^t(v)} -\log\left(\sigma(<\boldsymbol{e}_u^t, \boldsymbol{e}_v^t>)\right) - w_n \cdot \sum_{u' \in P_n^t(v)} \log\left(1 - \sigma(<\boldsymbol{e}_{u'}^t, \boldsymbol{e}_v^t>)\right)\Big) \quad (3)$$

where σ is the sigmoid function, $< . >$ denotes the inner product operation, $\mathcal{N}_{walk}^t(v)$ is the set of nodes that co-occur with v on fixed-length random walks at snapshot t, P_n^t is a negative sampling distribution for snapshot $\mathcal{G}^t$ (commonly defined as a function of degree distribution), and w_n, the negative sampling ratio, is a tunable hyper-parameter to balance the positive and negative samples.

	Communication		Rating	
Attribute	**Enron**	**UCI**	**Yelp**	**ML-10M**
# of Nodes	143	1,809	6,569	20,537
# of Links	2,347	16,822	95,361	43,760
# of Time steps	12	13	12	13

Table 1: Summary statistics of four datasets. Link counts include the number of snapshots containing each link, *e.g.*, a link that occurs in three snapshots, is counted thrice.

5 EXPERIMENTS

To analyze dynamic node embedding quality from different perspectives, we propose three research questions to guide our experiments:

(**RQ$_1$**) How does DySAT perform in comparison to state-of-the-art *static* and *dynamic* graph representation learning methods on single-step and multi-step link prediction tasks?

(**RQ$_2$**) How does DySAT compare with existing work under link prediction focused on *new links* and *unseen nodes*?

(**RQ$_3$**) Are the proposed structural and temporal self-attentional layers in DySAT *effective*, *efficient*, and *interpretable*?

5.1 Datasets

We experiment on four dynamic graphs including two communication and rating networks of variable sizes similar to [6, 7] (Table 1).

Communication networks. We consider two publicly available communication network datasets: Enron [14] and UCI [21]. In Enron, the communication links denote email interactions between core employees, while the links in UCI represent messages sent between peer users on an online social network platform.

Rating networks. We examine two bipartite networks from Yelp[1] and MovieLens [11]. In Yelp, the dynamic graph comprises links between users and businesses, derived from the observed ratings over time. ML-10M consists of a user-tag interactions where the links connect users with the tags they applied on certain movies.

5.2 Baselines

First, we present comparisons against several static graph embedding methods to analyze the gains of using temporal information for link prediction. To ensure a fair comparison, we provide access to the entire history of snapshots by constructing an aggregated graph up to time t, with link weights proportional to the cumulative weight till t agnostic to link occurrence times.

- **node2vec** [8]: A static embedding method that employs second-order random walk sampling to learn node representations.
- **GraphSAGE** [10]: An inductive node representation learning framework. We evaluate different aggregators including GCN, meanpool, maxpool, and LSTM, to report the best in each dataset.
- **GraphSAGE + GAT** [10, 34]: A variant of GraphSAGE with Graph Attentional layer [34] as the aggregation function.
- **GCN-AE** [41]: GCN trained as an autoencoder towards link prediction along the suggested lines of Zitnik et al. [41].
- **GAT-AE** [34, 41]: Similar GAT autoencoder for link prediction.

Second, we evaluate DySAT against the most recent dynamic graph embedding baselines that operate on discrete snapshots:

- **DynamicTriad** [39]: A dynamic graph embedding technique that combines triadic closure with temporal smoothness.
- **DynGEM** [7]: A deep neural embedding method that incrementally learn graph autoencoders of varying layer sizes.
- **DynAERNN** [6]: A deep neural network composed of dense and recurrent layers to capture temporal graph evolution.

We use a held-out validation set (20% links) to tune model hyperparameters. We randomly sample 25% examples for training, use the remaining 75% for testing, and report results averaged across 10 randomized runs, along with standard deviation. We train a downstream classifier using Hadamard Operator ($\boldsymbol{e}_u^t \odot \boldsymbol{e}_v^t$) to compute a feature vector for a pair of nodes, as recommended by [8] unless specified otherwise. Additional details on parameters and tuning can be found in our technical report [24]. We implement DySAT in Tensorflow [1] and use mini-batch gradient descent with Adam [12] for training. Our implementation is publicly available[2].

5.3 Link Prediction Comparison (RQ 1)

In this section, we conduct experiments on single-step and multi-step link prediction (or forecasting). Using the node embeddings trained on graph snapshots up to time step t, single-step link prediction predicts the connections between nodes at time step $t + 1$. We choose this task since it has been widely used [6, 7] in evaluating the quality of dynamic node representations to predict the temporal evolution of graph structures. In contrast, multi-step link prediction predicts links at multiple time steps starting from $t + 1$.

5.3.1 *Experimental Setup*.

Each model is trained on the input snapshots $\{\mathcal{G}^1, \dots, \mathcal{G}^t\}$, to obtain the latest node representations. For single-step link prediction, the latest embeddings $\{\boldsymbol{e}_v^t, \forall v \in \mathcal{V}\}$, are used to predict the links at $\mathcal{G}^{t+1}$, *i.e.*, classifying each example (node pair) into links and non-links. To evaluate link prediction, we use a downstream logistic regression classifier with evaluation examples from the links in $\mathcal{G}^{t+1}$ and an equal number of randomly sampled pairs of unconnected nodes (non-links) [8, 39].

For multi-step link prediction, the latest embeddings $\{\boldsymbol{e}_v^t, \forall v \in \mathcal{V}\}$ are used to predict links at multiple future time steps $\{t + 1, \dots, t + \Delta\}$. In each dataset, we choose the last $\Delta = 6$ snapshots for evaluation. Evaluation examples are created for each future time step $t + x$ $(1 \leq x \leq \Delta)$, by similarly sampling the links in $\mathcal{G}_{t+x}$ and an equal number of randomly sampled non-links. Here, the links formed by nodes that newly appear in the future evaluation snapshots are excluded since most existing methods do not support updates for new nodes. We use the Area Under the ROC Curve (AUC) [8] metric to evaluate link prediction performance.

5.3.2 *Single-Step Link Prediction*.

We evaluate the models at each time step t by training separate models up to snapshot t and evaluate at $(t+1)$ for each $t = 1, \dots, T$. We use micro and macro averaged AUC as evaluation metrics. Micro-AUC is calculated across the link instances from all the time steps while Macro-AUC is computed by averaging the AUC at each time step. Our results (Table 2) indicate that DySAT achieves consistent gains of 4–5% macro-AUC, in comparison to the best baseline across all datasets. Considering that the performance gain reported in other graph representation learning papers [34, 35] is usually around 2%, this improvement

[1] https://www.yelp.com/dataset/challenge

[2] https://github.com/aravindsankar28/DySAT

Method	Enron		UCI		Yelp		ML-10M	
	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC
node2vec	83.72 ± 0.7	83.05 ± 1.2	79.99 ± 0.4	80.49 ± 0.6	67.86 ± 0.2	65.34 ± 0.2	87.74 ± 0.2	87.52 ± 0.3
G-SAGE	82.48* ± 0.6	81.88* ± 0.5	79.15* ± 0.4	82.89* ± 0.2	60.95† ± 0.1	58.56† ± 0.2	86.19‡ ± 0.3	89.92‡ ± 0.1
G-SAGE + GAT	72.52 ± 0.4	73.34 ± 0.6	74.03 ± 0.4	79.83 ± 0.2	66.15 ± 0.1	65.09 ± 0.2	83.97 ± 0.3	84.93 ± 0.1
GCN-AE	81.55 ± 1.5	81.71 ± 1.5	80.53 ± 0.3	83.50 ± 0.5	66.71 ± 0.2	65.82 ± 0.2	85.49 ± 0.1	85.74 ± 0.1
GAT-AE	75.71 ± 1.1	75.97 ± 1.4	79.98 ± 0.2	81.86 ± 0.3	65.92 ± 0.1	65.37 ± 0.1	87.01 ± 0.2	86.75 ± 0.2
DynamicTriad	80.26 ± 0.8	78.98 ± 0.9	77.59 ± 0.6	80.28 ± 0.5	63.53 ± 0.3	62.69 ± 0.3	88.71 ± 0.2	88.43 ± 0.1
DynGEM	67.83 ± 0.6	69.72 ± 1.3	77.49 ± 0.3	79.82 ± 0.5	66.02 ± 0.2	65.94 ± 0.2	73.69 ± 1.2	85.96 ± 0.3
DynAERNN	72.02 ± 0.7	72.01 ± 0.7	79.95 ± 0.4	83.52 ± 0.4	69.54 ± 0.2	68.91 ± 0.2	87.73 ± 0.2	89.47 ± 0.1
DySAT	**85.71** ± 0.3	**86.60** ± 0.2	**81.03** ± 0.2	**85.81** ± 0.1	**70.15** ± 0.1	**69.87** ± 0.1	**90.82** ± 0.3	**93.68** ± 0.1

Table 2: Single-step link prediction results (micro and macro averaged AUC with std. deviation). We show GraphSAGE (denoted by G-SAGE) results with the best performing aggregators (∗, †, and ‡ represent GCN, LSTM, and max-pooling respectively). DySAT achieves significant performance gains of 4.8% macro-AUC on average across all datasets.

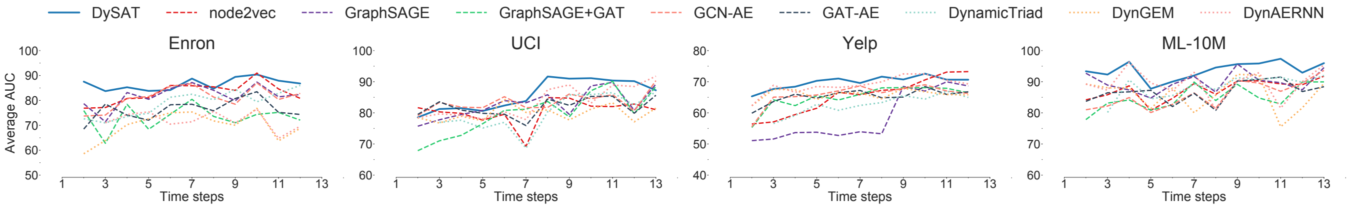

Figure 2: Performance comparison of DySAT with different graph representation learning methods on single-step link prediction: solid line denotes DySAT; dashed and dotted lines denote static and dynamic graph embedding baselines respectively.

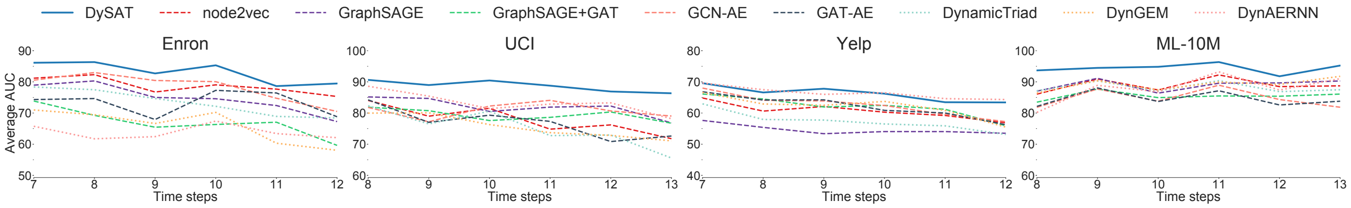

Figure 3: Performance comparison of DySAT with different models on multi-step link prediction for the next 6 time steps. DySAT outperforms competing baselines by a large margin, and maintains consistently high performance over time.

is significant. DynAERNN typically comes second-best, validating the effectiveness of RNN-based methods for temporal modeling. Comparative analysis of different methods yields several insights.

First, GraphSAGE often achieves comparable performance to dynamic embedding methods across different datasets, despite being agnostic to temporal information. One possible explanation is that GraphSAGE uses trainable neighborhood-aggregators while existing dynamic embedding methods either employ skip-gram or adjacency reconstruction methods for structural proximity. We conjecture that joint structural and temporal modeling with expressive aggregators (*e.g.*, multi-head attention), is responsible for the consistently superior performance of DySAT on link prediction. Second, node2vec maintains a consistent performance despite being agnostic to temporal information, which indicates further improvements to DySAT on applying second-order sampling techniques.

Further, we compare the AUC at each time step (Figure 2). We find the performance of DySAT to be relatively more stable than other methods. This contrast is pronounced in the communication networks (Enron and UCI), where we observe drastic drops in performance of static embedding methods at certain time steps.

5.3.3 ***Multi-Step Link Prediction.*** In this section, we evaluate various models on multi-step link prediction over six evaluation snapshots. From Figure 3, we observe a slight decay in performance over time for all the models, which is expected. DySAT achieves significant gains over the baselines and maintains a stable performance over time, while static methods often exhibit large variations due to their lack of temporal context. This demonstrates the capability of the temporal attention module to capture the most relevant historical context, for predictions at each time step in the future.

5.4 Impact of New Graph Elements (RQ 2)

Dynamic graphs usually contain unseen nodes and links over the course of temporal evolution. To investigate if DySAT can learn effective representations for graph elements that do not appear in previous snapshots, we analyze the sensitivity of different methods for (single-step) link prediction on: *new links* and *unseen nodes.*

5.4.1 ***Link Prediction on New Links.*** We evaluate link prediction only on the *new* links at each time step. A link is considered *new* at time step t for evaluation if it is present in $\mathcal{G}_{t+1}$ and has not

Method	Enron		UCI		Yelp		ML-10M	
	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC
node2vec	76.92 ± 1.2	75.86 ± 0.5	73.67 ± 0.3	74.76 ± 0.8	67.36 ± 0.2	65.17 ± 0.2	85.22 ± 0.2	84.89 ± 0.1
G-SAGE	73.92^* ± 0.7	74.67^* ± 0.6	76.69^* ± 0.3	79.41^* ± 0.1	$62.25^\dagger$ ± 0.2	$58.81^\dagger$ ± 0.3	$85.23^\ddagger$ ± 0.3	$89.14^\ddagger$ ± 0.2
G-SAGE + GAT	67.02 ± 0.8	68.32 ± 0.7	73.18 ± 0.4	76.79 ± 0.2	66.53 ± 0.2	65.45 ± 0.1	80.84 ± 0.3	82.53 ± 0.1
GCN-AE	74.46 ± 1.1	74.02 ± 1.6	74.76 ± 0.1	76.75 ± 0.6	66.18 ± 0.2	65.77 ± 0.3	82.45 ± 0.3	82.48 ± 0.2
GAT-AE	69.75 ± 2.2	69.25 ± 1.9	72.52 ± 0.4	73.78 ± 0.7	66.07 ± 0.1	65.91 ± 0.2	84.98 ± 0.2	84.51 ± 0.3
DynamicTriad	69.59 ± 1.2	68.77 ± 1.7	67.97 ± 0.7	71.67 ± 0.9	63.76 ± 0.2	62.83 ± 0.3	84.72 ± 0.2	84.32 ± 0.2
DynGEM	60.73 ± 1.1	62.85 ± 1.9	77.49 ± 0.3	79.82 ± 0.5	66.42 ± 0.2	66.84 ± 0.2	73.77 ± 0.7	83.51 ± 0.3
DynAERNN	59.05 ± 2.7	59.63 ± 2.7	77.72 ± 0.5	81.91 ± 0.6	**74.33** ± 0.2	**73.46** ± 0.2	87.42 ± 0.2	88.19 ± 0.2
DySAT	**78.87** ± 0.6	**78.58** ± 0.6	**79.24** ± 0.3	**83.66** ± 0.2	69.46 ± 0.1	69.14 ± 0.1	**89.29** ± 0.2	**92.65** ± 0.1

Table 3: Single-step link prediction results restricted to *new* links (micro and macro averaged AUC with std. deviation). All methods achieve lower AUC scores in comparison to predicting all links; DySAT outperforms baselines on most datasets.

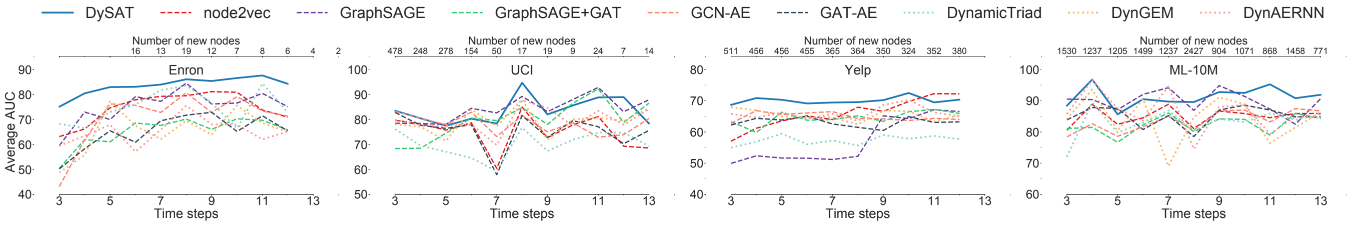

Figure 4: Performance comparison of DySAT with different graph embedding models on single-step link prediction restricted to previously unseen *new* nodes at each time step. Despite the lack of historical context for an unseen node, DySAT outperforms competing baselines due to its ability to factor the temporal evolution of its neighbors to compute its representation.

previously appeared in $\mathcal{G}_t$. Thus, the evaluation examples comprise *new* links at $\mathcal{G}_{t+1}$ and an equal number of random non-links.

Table 3 summarizes the micro and macro averaged AUC scores for different methods on the four datasets. From Table 3, we find that all methods achieve lower AUC scores than the original setup of using all links at $\mathcal{G}_{t+1}$, which is reasonable since accurate prediction of new links at $\mathcal{G}_{t+1}$ is more challenging than predicting all the links at $\mathcal{G}_{t+1}$. DySAT achieves consistent relative gains of 3–5% Macro-AUC over the best baselines on most datasets, while being inferior to DynAERNN on the yelp dataset. One possible reason is that RNN-based methods are able to better predict new links due to their limited history prioritization, especially in Yelp where most links at each step are new. Overall, DySAT achieves consistently superior performance over baselines on new link prediction.

5.4.2 ***Link Prediction on Unseen Nodes***. In this section, we analyze model sensitivity to previously unseen nodes that newly appear at time t. A node is considered *new* at time step t in $\mathcal{G}_t$, if it has not appeared (has no links) in any of the previous $t-1$ snapshots. In this experiment, the evaluation set at time step t only comprises the subset of links at $\mathcal{G}_{t+1}$ involving the new nodes in $\mathcal{G}_t$ and corresponding randomly sampled non-links. Figure 4 also depicts the number of new nodes appearing at each time step.

From Figure 4, DySAT consistently outperforms other baselines, demonstrating its ability to characterize previously unseen nodes. Although temporal attention focuses on the latest embedding e_v^t due to absence of history, the intermediate structural representation h_v^t receives back-propagation signals via temporal attention on neighboring *seen* nodes, which indirectly affects the final embedding e_v^t. We hypothesize that this indirect temporal signal is one of the reasons for DySAT to achieve improvements over baselines, albeit its inability to exploit temporal context for unseen nodes.

5.5 Self-Attention Analysis (RQ 3)

Since self-attention plays a critical role in the formulation of DySAT, we conduct exploratory analysis to study: *effectiveness*: how does joint structural and temporal attention impact the performance of DySAT?, *efficiency*: how efficient is self-attention in comparison to existing recurrent models?, and *interpretability*: are the learnt attention weights visually interpretable for humans?

5.5.1 ***Effectiveness of Self-Attention***. We evaluate the effectiveness of our self-attentional architecture DySAT in two parts:

- **Structural and temporal self-attention.** We conduct an ablation study by independently removing the structural and temporal attention blocks from DySAT to create simpler architectures. Note that removal of temporal attention block results in a model that differs from static methods since the embeddings are jointly optimized (Eqn. 3) to predict snapshot-specific neighborhoods, however without any explicit temporal evolution modeling. From the results (Table 4), we observe that in some datasets, direct structural attention achieves reasonable results, while the removal of structural attention consistently deteriorates performance. In contrast to Table 2 where GAT-AE (structural attention over time-agnostic neighborhoods) often performs inferior to GCN-AE, here structural attention applied over snapshot-specific neighborhoods, is shown to be effective. DySAT consistently outperforms the variants with 3% average gain in Macro-AUC, validating our choice of joint structural and temporal self-attention.

Method	Enron		UCI		Yelp		ML-10M	
	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC
Original	**85.71** ± 0.3	**86.60** ± 0.2	**81.03** ± 0.2	**85.81** ± 0.1	**70.15** ± 0.1	**69.87** ± 0.1	**90.82** ± 0.3	**93.68** ± 0.1
No Structural	85.21 ± 0.2	86.50 ± 0.3	74.48 ± 0.3	81.24 ± 0.2	70.11 ± 0.1	67.85 ± 0.1	88.56 ± 0.2	90.34 ± 0.2
No Temporal	84.50 ± 0.3	85.68 ± 0.4	76.61 ± 0.2	79.97 ± 0.3	68.34 ± 0.1	67.20 ± 0.3	89.61 ± 0.4	91.10 ± 0.2

Table 4: Ablation study on structural and temporal attention layers (micro and macro AUC with std. deviation). DySAT (joint structural and temporal attention) outperforms the ablation variants by a margin 3% Macro-AUC on average.

- **Multi-head attention.** To analyze the benefits of multifaceted modeling via attention heads, we vary the number of structural and temporal heads in DySAT independently in the range $\{1, 2, 4, 8, 16\}$, while keeping the layer sizes fixed for fairness. From Figure 5, we observe that DySAT benefits from multi-head attention on both structural and temporal attention layers. The performance stabilizes with 8 attention heads, which appears sufficient to capture graph evolution from multiple latent facets.

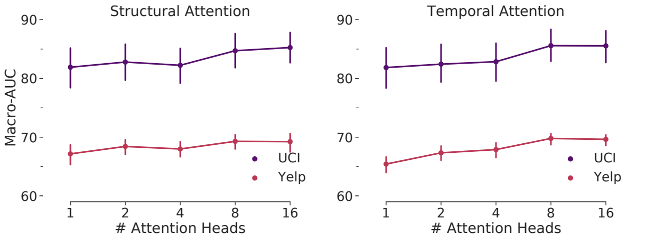

Figure 5: Sensitivity analysis on the number of structural and temporal attention heads in DySAT (UCI and Yelp). Model performance stabilizes with 8 attention heads.

5.5.2 *Scalability Analysis.*

In this section, we compare the scalability of DySAT against state-of-the-art dynamic method DynAERNN, which uses an RNN to model dynamic graph evolution. We choose this method since it is conceptually closest to our self-attentional architecture, and outperforms other dynamic embedding baselines. We compare training times by varying the temporal history *window*, *i.e.*, the number of previous snapshots used as history. Figure 6 depicts runtime per epoch on ML-10M, using a machine with Nvidia Tesla V100 GPU and 16 CPU cores.

DySAT achieves significantly lower training times than DynAERNN, *e.g.*, the runtime per epoch of DySAT is 88 seconds with a window size of 10 on ML-10M, in comparison to 723 seconds for DynAERNN. Self-attention is easily parallelized across both time as well as attention heads, while RNNs suffer due to the inherently sequential nature of back-propagation through time. Our results demonstrate the practical scalability advantage for pure self-attentional architectures over RNN-based methods.

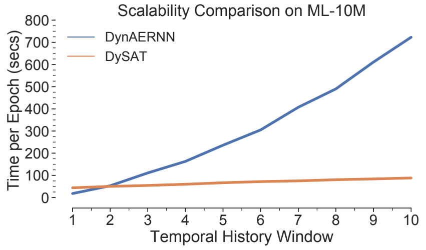

Figure 6: Scalability comparison of DySAT with DynAERNN. DySAT is faster by nearly an order of magnitude.

5.5.3 *Temporal Attention Visualization.*

We visualize the distribution of temporal attention weights learned by DySAT on datasets with distinct evolutionary behaviors. We consider two diverse datasets UCI and Yelp: UCI is a communication network with periodically recurring user-user interactions, while Yelp is a rating network where new users and businesses get added over time. Figure 7 visualizes a heatmap of the mean temporal attention weights (averaged over all nodes) for the first 11 time steps.

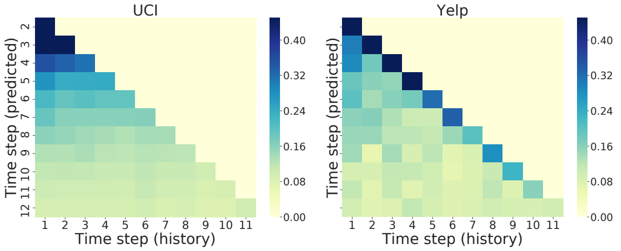

Figure 7: Heatmap of mean temporal attention weights on UCI and Yelp. Attention weights are uniformly distributed in UCI, while Yelp exhibits a bias towards recent snapshots.

In Figure 7, each row depicts the set of all attention weights over historical time steps $t_1 \ldots t_{k-1}$ for predicting links at time step k. We find the attention weights to be biased towards recent snapshots in Yelp, while being more uniformly distributed in UCI. This observation conforms to the nature of graph evolution since rating behaviors in Yelp tend to be bursty and correlated with events such as restaurant opening, discounted sales, etc., while inter-user communications in UCI typically span longer time intervals. Thus, DySAT is able to learn different attention weight distributions adaptive to the mechanism of real-world temporal graph evolution.

6 DISCUSSION

In DySAT, we stack temporal attention layers on top of structural attention layers. We choose this design as graph structures are not stable over time, which makes the converse option impractical. Another potential design choice is applying self-attention along the two dimensions of neighbors and time together following a strategy similar to DiSAN [28]. In practice, this would be computationally expensive due to variable number of neighbors per node across multiple snapshots. We leave exploring other architectural choices based on structural and temporal self-attentions as future work.

DySAT can be directly extended to incrementally learn embeddings in a streaming environment, enabling both computational and memory efficiency. Our initial experiments indicate promising results [24], and opens the door to exploration of self-attentional architectures for incremental graph representation learning.

7 CONCLUSION

In this paper, we introduce a novel self-attentional architecture named DySAT for dynamic graph representation learning. Specifically, DySAT computes dynamic node representations through joint self-attention over the *structural neighborhood* and *historical representations*, thus effectively capturing temporal evolution of graph structure. Our experimental results on four real-world datasets indicate significant performance gains for DySAT over several state-of-the-art static and dynamic graph embedding baselines. An interesting future direction is exploring continuous-time generalizations to incorporate more fine-grained temporal variations.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations (ICLR)*.

[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*. 3844–3852.

[4] Lun Du, Yun Wang, Guojie Song, Zhicong Lu, and Junshan Wang. 2018. Dynamic Network Embedding: An Extended Approach for Skip-gram based Network Embedding.. In *IJCAI*. 2086–2092.

[5] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1243–1252.

[6] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. 2018. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *arXiv preprint arXiv:1809.02657* (2018).

[7] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. 2017. DynGEM: Deep Embedding Method for Dynamic Graphs. In *IJCAI International Workshop on Representation Learning for Graphs (ReLiG)*.

[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[9] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. 2019. Variational graph recurrent neural networks. In *Advances in Neural Information Processing Systems*. 10700–10710.

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[11] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TIIS)* 5, 4 (2016), 19.

[12] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

[13] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference for Learning Representations (ICLR)*.

[14] Bryan Klimt and Yiming Yang. 2004. Introducing the Enron Corpus. In *CEAS 2004 - First Conference on Email and Anti-Spam, July 30-31, 2004, Mountain View, California, USA*.

[15] Adit Krishnan, Hari Cheruvu, Cheng Tao, and Hari Sundaram. 2019. A Modular Adversarial Approach to Social Recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 1753–1762.

[16] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2.

[17] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 387–396.

[18] Xi Liu, Ping-Chun Hsieh, Nick Duffield, Rui Chen, Muhe Xie, and Xidao Wen. 2018. Streaming Network Embedding through Local Actions. *arXiv preprint arXiv:1811.05932* (2018).

[19] Kanika Narang, Chaoqi Yang, Adit Krishnan, Junting Wang, Hari Sundaram, and Carolyn Sutter. 2019. An Induced Multi-Relational Framework for Answer Selection in Community Question Answer Platforms. *arXiv preprint arXiv:1911.06957* (2019).

[20] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In *3rd International Workshop on Learning Representations for Big Networks (WWW BigNet)*.

[21] Pietro Panzarasa, Tore Opsahl, and Kathleen M. Carley. 2009. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *JASIST* 60, 5 (2009), 911–932.

[22] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[23] Aravind Sankar, Adit Krishnan, Zongjian He, and Carl Yang. 2019. Rase: Relationship aware social embedding. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[24] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2018. Dynamic Graph Representation Learning via Self-Attention Networks. *arXiv preprint arXiv:1812.09430* (2018).

[25] Aravind Sankar, Xinyang Zhang, and Kevin Chen-Chuan Chang. 2017. Motif-based Convolutional Neural Network on Graphs. *CoRR* abs/1711.05697 (2017). arXiv:1711.05697

[26] Aravind Sankar, Xinyang Zhang, and Kevin Chen-Chuan Chang. 2019. Meta-GNN: Metagraph Neural Network for Semi-supervised learning in Attributed Heterogeneous Information Networks. In *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 137–144.

[27] Purnamrita Sarkar and Andrew W Moore. 2006. Dynamic social network analysis using latent space models. In *Advances in Neural Information Processing Systems*. 1145–1152.

[28] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[29] Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[30] Joshua B Tenenbaum, Vin De Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290, 5500 (2000), 2319–2323.

[31] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 3462–3471.

[32] Rakshit Trivedi, Mehrdad Farajtbar, Prasenjeet Biswal, and Hongyuan Zha. 2018. Representation Learning over Dynamic Graphs. *arXiv preprint arXiv:1803.04051* (2018).

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*. 6000–6010.

[34] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations (ICLR)*.

[35] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341* (2018).

[36] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*.

[37] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In *International Conference on Learning Representations (ICLR)*.

[38] Ziwei Zhang, Peng Cui, Jian Pei, Xiao Wang, and Wenwu Zhu. 2018. Timers: Error-bounded svd restart on dynamic networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[39] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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

[41] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. 2018. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics* 34, 13 (2018), 457–466.

[42] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding Temporal Network via Neighborhood Formation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2857–2866.