On Generalizing Static Node Embedding to Dynamic Settings

Di Jin
University of Michigan, Ann Arbor
dijin@umich.edu

Sungchul Kim
Adobe Research
sukim@adobe.com

Ryan A. Rossi
Adobe Research
rrossi@adobe.com

Danai Koutra
University of Michigan, Ann Arbor
dkoutra@umich.edu

ABSTRACT

Temporal graph embedding has been widely studied thanks to its superiority in tasks such as prediction and recommendation. Despite the advances in algorithms and novel frameworks such as deep learning, there has been relatively little work on systematically studying the properties of temporal network models and their cornerstones, the graph time-series representations that are used in these approaches. This paper aims to fill this gap by introducing a general framework that extends an arbitrary existing static embedding approach to handle dynamic tasks, and conducting a systematic study of *seven* base static embedding methods and *six* temporal network models. Our framework generalizes static node embeddings derived from the time-series representation of stream data to the dynamic setting by modeling the temporal dependencies with classic models such as the reachability graph. While previous works on dynamic modeling and embedding have focused on representing a stream of timestamped edges using a time-series of graphs based on a specific time-scale (*e.g.*, 1 month), we introduce the notion of an ϵ-graph time-series that uses a fixed number of edges for each graph, and show its superiority in practical settings over the standard solution. From the 42 methods that our framework subsumes, we find that leveraging the new ϵ-graph time-series representation and capturing temporal dependencies with the proposed reachability or summary graph tend to perform well. Furthermore, the new dynamic embedding methods based on our framework perform comparably and on average better than the state-of-the-art embedding methods designed specifically for temporal graphs in link prediction tasks.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Information systems** → **Temporal data**; **Data stream mining**; *Network data models*; • **Networks** → **Network dynamics**; • **Computing methodologies** → **Learning latent representations**; • **Theory of computation** → *Dynamic graph algorithms.*

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498428

KEYWORDS

representation learning; dynamic networks; graph time-series

ACM Reference Format:
Di Jin, Sungchul Kim, Ryan A. Rossi, and Danai Koutra. 2022. On Generalizing Static Node Embedding to Dynamic Settings. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3488560.3498428

1 INTRODUCTION

Real-world networks that record the interaction between entities have grown rapidly, for example, the Internet [4], various online social networks (*e.g.*, Facebook, Snapchat), citation networks [14], and more. Specifically, when nodes and edges continuously change over time with addition, deletion (e.g., a phone call, an email, or physical proximity between two entities), we have a particular type of evolving network structure. Learning an appropriate network representation (embedding) that accurately captures the temporal dynamics and temporal structural properties of these entities is important for many downstream time-series forecasting/prediction tasks such as recommendation and entity resolution. Most recent research efforts devoted in the field follow the same pipeline: given a time-series of graphs, $\mathcal{G} = \{G_1, \cdots, G_T\}$, modeling the individual graph structures (within-snapshot property) along with the temporal dependencies (across-snapshot relation), and deriving node embeddings that incorporate both perspectives. While these works show advantage from various perspectives, the promising performance comes at the cost of time and model complexity, such as introducing additional transition variables to reflect the temporal dependency between snapshots [7], or latent weights on edges between snapshots [19, 25].

In this work, we propose a general framework that simplifies the process above and can generalize *any* static embedding method to a more powerful and predictive dynamic embedding method without introducing transitional variables. The framework consists of three components: (**C1**) a graph time-series representation, (**C2**) a temporal network model that appropriately models and weighs the temporal dependencies in the graph time-series, and (**C3**) a base embedding method to learn a time-series of embeddings along with a fusion mechanism to derive the final temporal node embeddings. The framework is highly expressive as any unique combination of **C1**-**C3** gives rise to a new dynamic embedding method.

While previous works on dynamic modeling and embedding have focused on representing the stream of timestamped edges [18] using a time-series of graphs based on a specific time-scale τ (*e.g.*,

τ = 1 hour, or 1 month) [6, 7, 14, 25, 26, 31], we instead propose the notion of an ϵ-graph time-series that uses a fixed number of edges for each graph in the time-series. Theoretically, by fixing the number of edges to be ϵ in each graph, we ensure that every graph in the sequence has an equal probability of giving rise to the same exact distribution of higher-order graphlets and other structural patterns[1], and therefore, the new ϵ-graph time-series forces the models to avoid capturing simple trivial differences due to edge counts, and instead, allow the models to capture actual *structural changes* to the graphs over time. We also introduce a number of general temporal models. The first one is based on the notion of a temporal reachability graph (TRG). TRG is derived by transforming a dynamic graph into a static graph where an edge from u to w indicates a temporal walk. The second model is called a weighted temporal summary graph (TSG). Notably, a weighted temporal summary graph captures the temporal recurrence and recency of links by appropriately weighting links with respect to a function f that assigns larger weights to links that are more recent and recurrent whereas links that occur in the more distant past are assigned lower weights. All temporal models can leverage either the new ϵ-graph or τ-graph time-series representation.

This paper aims to provide a systematic exploration of the most useful graph time-series representations and temporal network models (used to incorporate the temporal dependencies into base embedding methods) in downstream temporal prediction tasks. To the best of our knowledge, this is the first work of this kind. Our primary findings are: (1) node embeddings derived from the ϵ-graphs outperforms the τ-graph time-series in the predictive task with higher stableness, and (2) by composing the static node embedding approaches with classic temporal models such as TRG or TSG, our proposed framework performs comparably or even better than recent dynamic embedding approaches with less complexity. Based on these findings, we hope that this work will benefit future research on developing and evaluating more advanced dynamic embedding methods, as well as practitioners who deploy temporal graph embedding in various applications due to its simplicity and effectiveness. Our main contributions are as follows:

- **General Framework.** We describe a general framework for leveraging graph stream data and classic temporal network models for prediction-based applications that can generalize any static graph embedding method.
- **Powerful Graph Time-series Representation.** We introduce the notion of ϵ-graph time-series and show its superiority over the conventional way of discretizing the edge stream based on the application time-scale (e.g., hour, day).
- **Systematic Study.** We applied our framework to systematically study 42 dynamic node embeddings by combining time-series representations, temporal network models, and static methods. Strikingly, our empirical analysis on 8 real-world networks shows that our framework achieves comparable or better predictive performance than existing state-of-the-art, but more complex, *dynamic* node embedding methods.

[1]This is in contrast to graphs with different amounts of edges. E.g., given two arbitrary graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ where $|E_1| \ll |E_2|$, then the counts of all $k \in \{3, 4, \ldots\}$-node network motifs (graphlets) in G_2 are almost surely larger than G_1.

Table 1: Qualitative comparison of embedding methods on temporal graphs with respect to the graph time-series representation used (application time-scale, or fixed number of edges), the temporal model (weighing snapshots), and the embedding fusion used (if any).

	Time-scale (τ)	#Edges (ϵ)	Weighting	Fusion
DANE [16], TIMERS [30]	✓	✗	✗	✗
DynGem [7], Dyngraph2vec [6]	✓	✗	✗	✓
tNodeEmbed [26], EvolveGCN [19]	✓	✗	✓	✓
DySAT [25], DyHATR [29]	✓	✗	✗	✓
our framework	✓	✓	✓	✓

2 RELATED WORK

Snapshot-based approaches. Most temporal embedding methods break down the graph into snapshots based on the application time-scale (1 month, etc.). One direction is based on the most recent snapshot, for instance, DANE [16] embeds both nodes and the associated attributes in the graph by minimizing the loss of reconstruction of the snapshot at a given times point k, and updates the embeddings for snapshot at $k+1$ based on the change of graph structure and node attributes. DynGEM [7] adopts the auto-encoder to generate nonlinear embeddings from the snapshot at k while addressing stability. TIMERS [30] models the relative changes in adjacency matrices between snapshots and leverages incremental SVD to derive embeddings. Another direction is to track back a certain number of snapshots from the time point k by deriving node embeddings from each individual tracked snapshot and then merging them through specific operation. Dyngraph2vec [6] adopts l snapshots to predict edges at $k+1$. It uses various deep architectures (*i.e.*, auto-encoder, RNN) to derive latent features by minimizing loss of reconstruction error: $||f(\mathbf{A}_{k-l+1}, \cdots, \mathbf{A}_k) - \mathbf{A}_{k+1}||_F^2$. tNodeEmbed [26] is an end-to-end framework based on node embeddings derived from individual snapshots using static methods. The embeddings are merged by minimizing the loss of specific tasks (*i.e.*, link prediction and node classification) through LSTM. DySAT [25] and DyHATR [29] use self-attention to compute node embeddings by jointly employing graph structural property and temporal dynamics. EvolveGCN [19] uses GCN to generate node embeddings for the past snapshots, and learns the hidden parameters for the next using GRU/LSTM. Unlike the above methods that jointly explore the graph structural changes over fixed timespans, our proposed ϵ-graph time series does not require the specification of time-scales.

Sequential-interaction-based approaches. There is another line of work that studies the sequential interaction between nodes in the graph. CTDNE [18] is the first approach to learn embeddings directly from the stream of timestamped edges at the finest temporal granularity. In that work, they proposed the notion of temporal walks and used it for embeddings [18]. More recently, node2bits [10] expanded on this idea by incorporating features in the temporal walks and hashing them. Alternatively, some other work has modeled the node-specific temporal dynamics as the point process where the probability of interaction is represented through different intensity functions. HTNE [33] proposes to model the node evolution through the Hawkes process. JODIE[13] models the sequential interaction in bipartite graphs to predict the change of embedding trajectory over time instead of interaction probability. CTDNE, HTNE and JODIE are designed to handle continuously sequential data, which is not the scope of this paper.

Table 2: Network statistics and properties

Data	$\|V\|$	$\|E\|$	Type	Timespan
enron	151	50,572	Unipartite	38 months
bitcoin	3,783	24,186	Unipartite	63 months
wiki-elec	7118	107,071	Unipartite	47 months
stackoverflow	24,818	506,550	Tripartite	79 months
fb-forum	899	33,720	Unipartite	24 weeks
reallity-call	6,809	52,050	Unipartite	16 weeks
wiki-edit	8,227	157,474	Bipartite	32 days
contacts-dublin	10,972	415,912	Unipartite	69 days

3 DATA

In this study we adopt a variety of real-world temporal networks from SNAP [15] and NR [21]. The complete data descriptions are given in Section A of the appendix. We summarize the graph statistics and temporal timespans in Table 2, and analyze the sequential graph statistics of three graphs over time. As the timespans vary from 32 days to 79 months, we adopt the time-scale following Table 2 to get the sequential graph time-series. We visualize 2 graph statistics, the number of edges $|V|$ and the average degree on 3 datasets with different time-scales in Figure 1, which are `contacts-dublin` (day), `wiki-elec` (month), and `fb-forum` (week). In the figure, we also visualize the same graph statistics using a different time-series representation by fixing the number of edges in each snapshot to $\frac{|E|}{T}$, where T denotes the timespan following the corresponding time-scale. For example, for `wiki-elec`, this number is $\frac{107\,701}{47}$ in each snapshot. From Figure 1, we compare the temporal patterns of the two time-series and we observe that following the fixed edge count in each snapshot gives a more stable temporal pattern based on both graph statistics. We discuss this new graph time-series in detail in Section 5.1. Besides, in this work, we focus on exploring the impacts of graph structures and temporal dependency between snapshots to the predictive tasks, thus we do not leverage node features such as geographic location or content.

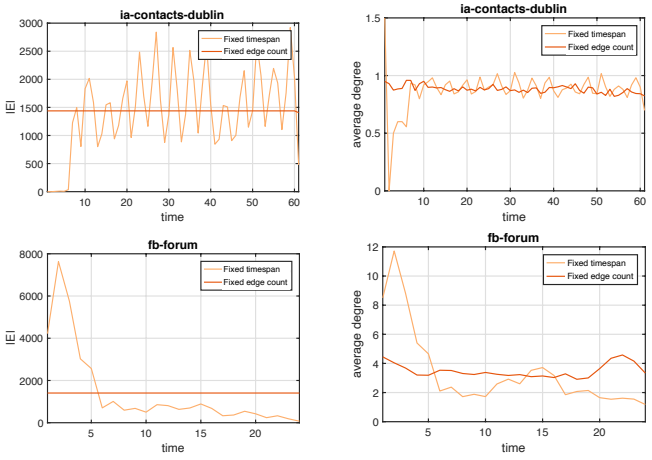


Figure 1: Graph properties (#edge and average degree) over two time-series representations (fixed timespans vs. fixed edge count). Fixing the edge number gives more stable temporal patterns while fixing the timespans shows higher fluctuation.

Table 3: Summary of notation

Symbol	Definition
$\mathcal{G} = \{G_k\}$	a graph time-series with snapshots indexed by k.
$G_k = (V_k, E_k)$	a directed and weighted temporal network from $\mathcal{G}$ with $\|V_k\|$ nodes and $\|E_k\|$ temporal edges
$\mathbf{A}_k$	adjacency matrix for graph G_k in $\mathcal{G}$.
$G_R = (V, E_R)$	the weighted temporal reachability graph
N_i^R	the set of nodes that are temporally reachable from node i
τ/ϵ	window size representing the timespan / number of edges
α	the decay factor in the temporal summary graph model
f	arbitrary base embedding method
$\mathbf{Z}$	$\|V\| \times d$ embedding matrix

4 PRELIMINARIES

We summarize symbols and notations used in this work in Table 3. Some important notions are given as follows.

Definition 1 (Temporal Graph). *Let V be a set of vertices, and $E \subseteq V \times V \times \mathbb{R}^+$ be the set of temporal edges between vertices in V. Each edge (u, v, t) has a unique time $t \in \mathbb{R}^+$.*

When edges represent contacts between two entities *e.g.*, a phone call or physical proximity, then a temporal walk represents a feasible route for a piece of information that obeys time.

Definition 2 (Temporal Walks). *A temporal walk from u to w in $G = (V, E)$ is a sequence of edges $e_1, \ldots, e_k$ such that $e_1 = (u, u_1, t_1), \ldots, e_k = (u_{k-1}, w, t_k)$ where $t_j < t_{j+1}$ for all $j = 1$ to k. We say that u is temporally connected to w if such a walk exists.*

This definition echoes the standard definition of a path, but adds the additional constraint that paths must respect time, i.e., follow the directionality of time. Temporal walks are inherently asymmetric because of the directionality of time. The notion of temporal walks has been recently used in embedding methods [18].

5 FRAMEWORK

The framework in this paper provides a fundamental basis for studying different temporal network representations and for generalizing existing static embedding methods to dynamic settings. As shown in Figure 2, given the continuous stream of timestamped edges, we first derive the time-series of graphs (Section 5.1). Then, we use temporal network models to incorporate the temporal dependencies of the graph-based time-series (Section 5.2). Lastly, our framework generalizes existing embedding methods and enables the new dynamic variants of these methods to learn more accurate and appropriate time-dependent embeddings. (Section 5.3).

5.1 Graph Time-Series Representations

We formally introduce two approaches for deriving a time-series of graphs from the stream of timestamped edges. For clarity, we use k to index the snapshots in the time-series in this section to avoid mixing with the timestamp t associated with an edge e.

5.1.1 τ-graph time-series. The τ-graph time-series representation is the foundation of many graph aggregation approaches [27] and is used by the vast majority of previous work [6, 9].

Definition 3 (τ-graph time-series). *Given a temporal network G=(V, E) representing a continuous edge stream with time-stamped*

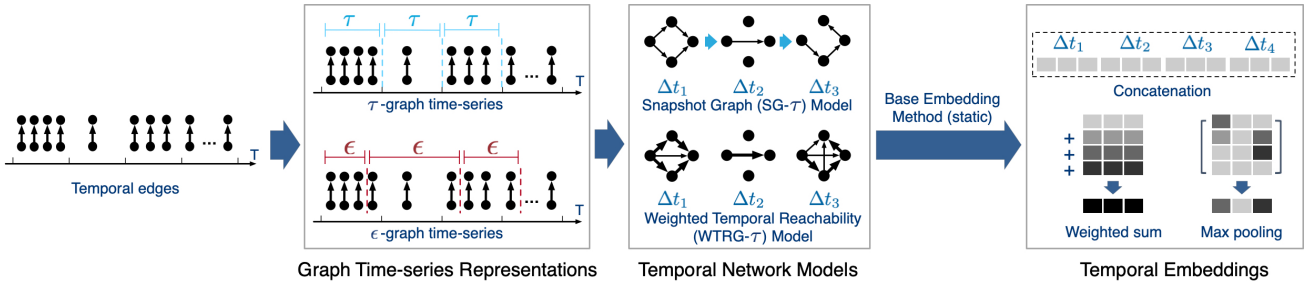


Figure 2: Framework Overview. First, it derives a graph time-series representation from the stream of timestamped edges using either an application-specific time-scale τ (*e.g.*, 1 day) or a fixed number of edges ϵ for each snapshot (Sec. 5.1). Next, given the $\{\tau, \epsilon\}$-graph time-series representation, it incorporates the temporal dependencies and weights with a temporal network model (Sec. 5.2). Finally, it adopts an arbitrary base method to learn a time-series of embeddings and then obtains the final temporal embeddings via a temporal fusion mechanism (Sec. 5.3).

edges E, we define a graph time-series $\mathcal{G}^\tau = \{G_k\}$ such that each graph G_k contains edges within a specific time-scale. Let t_0 denote the timestamp of the first edge in the temporal network (stream of timestamped edges) and τ is the application time-scale (e.g., *1 month), then*

$$E_k = \{(i, j, t) \in E \mid t_0 + k\tau > t \geq t_0 + (k-1)\tau\} \tag{1}$$

5.1.2 ϵ-graph time-series. While most work uses the previous approach for deriving the graph time-series, we introduce a new alternative based on the idea of using a fixed number of edges, that is, a time-series of graphs $\mathcal{G}^\epsilon = \{G_k\}$ such that each G_k consists of ϵ edges (Definition 4), *i.e.*$|E_k| = \epsilon, \forall k$. More formally,

Definition 4 (ϵ-graph time-series). *Given a temporal network $G = (V, E)$ representing a continuous edge stream E with timestamped edges and let ϵ denote a fixed number of temporal edges in the stream (ordered by time), we define a graph time-series $\mathcal{G}^\epsilon = \{G_k\}$ such that $|E_k| = \epsilon$, for all $k = 1, 2, \ldots$. E_k is given as follows:*

$$E_k = \bigcup_{i=(k-1)\epsilon+1}^{k\epsilon} e_i = \{e_{(k-1)\epsilon+1}, \ldots, e_{k\epsilon}\} \tag{2}$$

Note in both cases $E_1 \cup \cdots E_k \cup \cdots = E$. As observed in Fig. 1, while the conventionally-used τ representation follows human intuition to aggregate temporal edge streams within a specific time-scale such as 1 day or 1 hour, it can significantly deviate with large spikes even between consecutive graphs in the series. As a result, the modeled graph temporal dynamics can be impacted by such frequency change between sequential snapshots and fail to accurately capture the temporal graph structural evolution. For example, nodes that form a star structure in the past tend to form similar structures in the future, and the connection between individual subgraphs do not always follow the predefined time scales.

On the contrary, the proposed ϵ-graph time-series controls for the number of edges over time, embedding methods can thus more accurately model and capture the actual change in the structural properties and subgraph patterns over time, as opposed to just the frequency of edges that is captured by the τ-graph time-series representation used in previous work. That is to say, the ϵ-graph time-series representation preserves the sequential order of timestamped edges *without* suffering from the structural instability of the graph due to the sometimes-drastic difference in edge counts from one time to the next. If a graph time-series representation is unable to capture the simplest 1st-order subgraph structures (edges), then by definition it cannot capture higher-order subgraph structures that are built on such lower-order ones. Hence, the proposed ϵ-graph time-series representation effectively models the *structural changes* between graphs whereas the τ-graph time-series captures changes in *edge frequencies* for a fixed application-specific time-scale such as 1 day or 1 hour.

5.2 Temporal Network Models

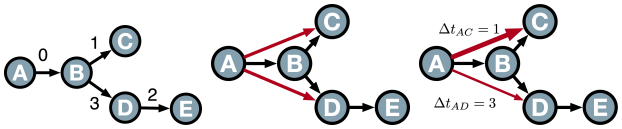
Now we introduce temporal network models that incorporate the temporal dependencies into the graph time-series representations to learn more effective time-dependent embeddings.

5.2.1 Snapshot Graph (SG) Model. This model simply leverages the $\{\tau, \epsilon\}$-graph time-series representation directly without encoding any additional temporal information into the representation. Hence, the temporal information (edge timestamps) associated with the edges in any graph $G_k \in \mathcal{G}$ is effectively ignored/discarded. For example, e_1 and e_2 are considered to occur simultaneously if they fall into the same snapshot, even though e_2 comes later than e_1 in the actual time-series. Therefore, this model incorporates the temporal dependencies at the level of the graph, *i.e.*, we only know that edges in G_{k-1} occurred before G_k.

5.2.2 Temporal Summary Graph (TSG) Model. The temporal summary graph model incorporates the temporal dependencies by deriving a weighted summary graph from the graph-based time series $\mathcal{G}$ [17, 24] where the more recent edges are assigned larger weights than those in the distant past. More formally, let $\{\mathbf{A}_1, \cdots, \mathbf{A}_T\}$ be a time-series of adjacency matrices of the graph time-series constructed using either Definition 3 or Definition 4. Furthermore, let $\mathbf{A}_k(i, j)$ denote the (i, j) entry of $\mathbf{A}_k$. We define the general *weighted temporal summary graph* (TSG) model as

$$\mathbf{S} = \sum_{k=1}^{T} f(\mathbf{A}_k, \alpha) \tag{3}$$

where f is a decay function for temporally weighting the edges (nonzeros), α is the decay factor ranging in $(0, 1)$, T is the total number of graphs in the time-series, and $\mathbf{S}$ is the weighted temporal summary graph. In this work, we define f as an exponential decay function [24], then we obtain $\mathbf{S} = \sum_{k=1}^{T}(1-\alpha)^{T-k}\mathbf{A}_k$, and the weight for an edge (i, j) = $\sum_{k=1}^{T}(1-\alpha)^{T-k}\mathbf{A}_k(i, j)$. Alternatively, instead of using all available graphs in the initial time-series, we can use only the L most recent graphs. For example, suppose $\mathcal{G}^\epsilon$ =



(a) A temporal graph (b) TRG (c) Weighed TRG

Figure 3: A toy temporal graph (a) and its temporal reachability modeling TRG (b) and WTRG (c). (b) An edge in the vanilla TRG represents a temporally-valid walk. The red edges represent the length-2 walks $\{A, B, C\}$ and $\{A, B, D\}$ in the original graph. (c) WTRG extends TRG by assigning weights to indicate the temporal closeness. *e.g.*, $\{A, B, C\}$ weights higher than $\{A, B, D\}$ as C is temporally closer to A than D ($\Delta t_{AC} < \Delta t_{AD}$), which reflects stronger temporal continuity.

$\{G_k\}_{k=1}^{T} = \{G_1, \ldots, G_T\}$ is an ϵ-graph time-series with T graphs. Instead of using all T graphs, we can leverage only the most recent L graphs, hence,

$$\mathcal{G}^{\epsilon} = \{G_k\}_{k=T-L+1}^{T} = \{G_{T-L+1}, \ldots, G_T\} \tag{4}$$

The idea of leveraging only the most recent graphs in the time-series was first explored in [24] and can be applied to any of the proposed temporal models in this section.

5.2.3 Temporal Reachability Graph (TRG) Model. The temporal reachability graph (TRG) is a graph derived from the timestamped edge stream where a link is added between two nodes if they are temporally connected. More formally, an edge (u, v) in the TRG model indicates the existence of a temporal walk from u to v in the original graph. The formal definition is given as follows.

Definition 5 (Temporal Reachability Graph). *Given an interval $\mathcal{I} \in \mathbb{R}^{+}$, the temporal reachability graph $G_R = (V, E_R)$ is defined as a directed graph where the edge $(u, v) \in E_R$ denotes the existence of a temporal walk leaving u and arriving v within that interval. We denote the number of edges in $\mathcal{I}$ as ω (which could be defined based on $\{\tau, \epsilon\}$-graph time-series).*

A TRG is a static unweighted graph where each edge indicates a temporally-valid walk reaching from the source to the destination. However, it does not capture the strength of reachability. For example in Fig. 3a, the walk $\{A, B, C\}$ takes two timestamps while $\{A, B, D\}$ takes four. Intuitively D is harder to reach than C from node A due to less temporal continuity. Vanilla TRG fails to capture such property since all the edges are equally important (shown in 3b). This would potentially affect the proximity-based embedding methods as they are based on the closeness of nodes in the graph. To overcome this drawback, we propose an extension of TRG called Weighted TRG (WTRG) that encapsulates the strength of reachability in the graph weights. We define the strength of reachability between a pair of nodes (i, j) as a function of both the number of temporally-valid paths and the timestamp difference. The weighting function is given as follows.

$$g_{i,j} = \sum_{w \in \mathcal{W}} e^{-(\Delta t_{i,j} | w)} \tag{5}$$

where w is a specific temporally-valid walk from i to j, and $\Delta t_{i,j}$ denotes the temporal delay reaching from i to j along that walk. We depict the process of deriving WTRG in Algorithm 1. The cornerstone of the algorithm is the temporally-reachable neighborhood N_i^R that records nodes that are reached by i and the latest timestamps associated with temporal paths. We formally define N_i^R as:

Algorithm 1 Weighted Temporal Reachability Graph

1: **procedure** TemporalReach($G = (V, E)$)
2: Set $E_R = \emptyset$, sort E_T in reverse time order
3: **while** next edge $(i, j, t) \in E$ **do**
4: **for** $(k, t_k) \in N_j^R$ **do**
5: $E_R \leftarrow E_R \cup \{(i, k)\}$
6: $g_{i,k} = g_{i,k} + e^{-(t_k - t)}$
7: $N_i^R \leftarrow N_i^R \cup \{(k, t_k)\}$
8: $E_R \leftarrow E_R \cup \{(i, j)\}$
9: $g_{i,j} = g_{i,j} + 1$ ▷ $\Delta t_{i,j} = 0$ as i, j are adjacent
10: $N_i^R \leftarrow N_i^R \cup \{(j, t)\}$
11: **end while**
12: **return** $G_R = (V, E_R, g)$

Definition 6 (Temporally reachable neighborhood). *Given a node i, its temporally reachable neighborhood N_i^R is defined as the set of tuples $\{(j, t_j)\}$ where j is the node reachable from i following a temporally-valid walk and t_j is the timestamp of the edge reaching j in that walk.*

Given an input temporal edge (i, j, t), Algorithm 1 loops through reachable neighbors in N_i^R to add edges in E_R and updates the weights based on Eq. (5) (line 5-8). It also adds (i, j) to the WTRG as well as the immediate weight (line 9-11). Overall, the computational complexity of the algorithm is $O(|E| \max d(N^R))$, where $\max d(N^R)$ is the maximum degree of a node in WTRG. While the derived WTRG can be dense with huge amounts of reachable neighbors, we show that this number is bounded by ω, which is the size of the interval associated with the WTRG (Section C of the supplementary material). Accordingly, the computational complexity of the algorithm is denoted as $O(|E|\omega)$. We follow Algorithm 2 to combine the embeddings over the graph time-series.

5.3 Temporal Embeddings

5.3.1 Base embedding methods. Given the graph time-series representation and temporal model (Section 5.1-5.2), the proposed approach can leverage any existing static embedding method to derive time-dependent node embeddings that capture the temporal dependencies between nodes as well as the temporal structural (role-based) or proximity-based properties [12, 23, 32]. We use the proposed framework to generalize a wide variety of static

Algorithm 2 General Framework for Temporal Embeddings

Input: ϵ or τ for deriving the graph time-series representation, base embedding method f (*e.g.*, GraphWave, role2vec)

1: Construct a graph time-series $\mathcal{G} = \{G_1, G_2, \ldots, G_T\}$ using a graph time-series representation $\{\tau, \epsilon\}$ from Section 5.1.
2: Initialize $\mathbf{Z}_0$
3: **for each** $G_k \in \mathcal{G}$ **do** ▷ for $k = 1, 2, \ldots$
4: Use Alg. 1 to derive the temporal reachability graph for G_k
5: Compute node embedding matrix $\mathbf{Z}_k$ using the base embedding method f with the temporal reachability graph from Alg. 1
6: Concatenate or aggregate (using sum, mean, etc.) the embedding matrix, *e.g.*, $\bar{\mathbf{Z}}_k = (1 - \theta)\bar{\mathbf{Z}}_{k-1} + \theta \mathbf{Z}_k$ where $\bar{\mathbf{Z}}_k$ is the temporally weighted embedding using the above exponential weighting kernel $\mathbb{K}(\cdot)$ and $0 \leq \theta \leq 1$ is a hyperparameter controlling the importance of past information relative to more recent (Section 5.3.2).
return $\bar{\mathbf{Z}}_k$ ▷ temporally weighted *or* concatenated

base embedding methods that include both community-based and role-based structural node embedding methods [23]. Namely, they are: **(1)** LINE [28], **(2)** Node2vec [8], **(3)** Graph2Gaussian [2], **(4)** struc2vec [20], **(5)** Role2vec [1], **(6)** Graphwave [5], and **(7)** MultiLENS [11]. Among these static methods, approaches (1-3) are community / proximity-based and (4-6) are role-based. (7) is a hybrid that is based on structural similarity of node-central subgraphs.

5.3.2 Temporal fusion. Given the time-series of node embeddings $\{\mathbf{Z}_k\}_{k=1}^{T}$, we explore two temporal fusion techniques.

Concatenation: Given a time-series of embeddings, one simple approach to obtain a final embedding is to concatenate the embeddings as follows: $\mathbf{Z} = \left[\mathbf{Z}_1 \cdots \mathbf{Z}_T\right]$. We could further weight the embeddings based on temporal recency, *i.e.*, under-weighting node embeddings that occur in the distant past since they are not as important as the more recent ones for prediction.

Temporally weighting: This technique aggregates (*e.g.*, sum, mean) the embedding matrix, *e.g.*, $\bar{\mathbf{Z}}_k = (1-\theta)\bar{\mathbf{Z}}_{k-1} + \theta\mathbf{Z}_k$ where $\bar{\mathbf{Z}}_k$ is the temporally weighted embedding using the above exponential weighting kernel $\mathbb{K}(\cdot)$. $0 \leq \theta \leq 1$ is a hyperparameter controlling the importance of past information relative to more recent.

6 EXPERIMENTS

In this section, we systematically investigate the effectiveness of the proposed framework by exploring the following research questions: **(Q1)** How well does the widely-used τ-graph time-series representation perform compared with the proposed ϵ-graph time-series? **(Q2)** How effective is the proposed WTRG model comparing with the vanilla TRG model? What temporal models are most useful for incorporating temporal dependencies into static embedding methods? **(Q3)** Are the dynamic embeddings generalized via our framework useful for temporal prediction, and how do they compare to the state-of-the-art dynamic methods?

6.1 Experimental Setup

6.1.1 Data. We learn node embeddings from the graph time-series starting from roughly $\frac{1}{3}$ of the timespans. For example, for the `bitcoin` dataset, we train the classifier based on node embeddings derived from month 20 to month 25 out of 63 months, inclusive. This ensures that there are sufficient edges for training. For all datasets, we perform training on the first 6 graphs and predict links on the 7th graph. Depending on the time-scale shown in Table 2, they represent 6 months (`enron, bitcoin, wiki-elec and overflow`), weeks (`fb-forum` and `reality-call`), or days (`wiki-edit` and `contact-dublin`). We create evaluation examples from the links in the 7th graph and an equal number of randomly sampled pairs of unconnected nodes as negative samples [25].

6.1.2 Model configuration and variants. We consider the task of link prediction over time and systematically compare the performance of different temporal network models and representations. Given a set of timestamped edges up to timestamp T, *i.e.*, $\mathcal{G} = \{G_1, \cdots G_T\}$, the temporal link prediction task aims to predict the future links that will form in G_{T+1}. We first follow the conventional setup to construct the τ-graph time-series $\mathcal{G}^{\tau} = \{G_1, \cdots G_T\}$ for model training and G_{T+1} for testing, where each snapshot $G_k (k \in \{1, 2, \cdots, T\})$ represents edges that occur within a consistent time scale shown in Table 2. Then we construct the ϵ-graph time-series representation $\mathcal{G}^{\epsilon}$. For fair comparison, we set $\epsilon = |E_{T+1}|$ to ensure the trained models based on both ϵ- and τ-based temporal networks are used to predict links in the same hold-out test set G_{T+1}. Thus, graphs in the ϵ-graph time-series $\mathcal{G}^{\epsilon} = \{G_1, \ldots, G_T\}$ and G_{T+1} are also consistent with respect to the ϵ representation, where $|E_1| = |E_2| = \cdots = |E_{T+1}|$.

For each $\{\epsilon, \tau\}$-graph time-series representation, we select a temporal network model from $\{SG, TSG, WTRG\}$ and a base embedding method using the framework. Therefore, we have totally 6 dynamic variants: $\{SG\text{-}\epsilon, TSG\text{-}\epsilon, WTRG\text{-}\epsilon, SG\text{-}\tau, TSG\text{-}\tau, WTRG\text{-}\tau\}$. To train the classifier, we applying these dynamic variants to derive node embeddings and feed them to the logistic regression model for prediction with regularization strength 1.0 and stopping criteria 10^{-4}. Following [3], we concatenate the node embeddings $\mathbf{z}_i$ and $\mathbf{z}_j$ to obtain an edge embedding $\mathbf{z}_{ij} = \left[\mathbf{z}_i \ \mathbf{z}_j\right]$. For temporal fusion, we use the temporally weighting technique from Section 5.3.2 with $\theta = 0.8$ for dimensional consistency. The TSG decay parameter α is set to 0.8 for computational fairness. We run all experiments 3 times and report the mean and std. For reproducibility, we provide the detailed configuration of both the base and dynamic graph embedding methods in Section B of the supplementary material. The source code and the complete experimental results are given at https://github.com/DerekDiJin/static_temporal_embedding.

6.2 WTRG vs. TRG

We first study the effectiveness of the WTRG model over the vanilla TRG model. As WTRG incorporates the strength of reachability in edge weights, we consider embedding methods that handle weighted graphs, namely, they are node2vec, struc2vec and MultiLENS. We run all methods on two datasets using both TRG and WTRG with τ-graph time series. The results are given in Table 4.

Our first observation is that structure-based embedding methods tend to outperform node2vec, the proximity-based method. Additionally, we observe that WTRG improves most embedding methods in link prediction, except for node2vec on `wiki-elec` dataset. One possible reason is that random walkers in WTRG are more likely to visit nodes that are close in time, and thus limiting the derived embeddings to incorporate distant neighborhood information. We put the in-depth study of WTRG in the future work. For embedding methods that are based on structural information, WTRG outperforms TRG by 0.8% in AUC ,1.3% in ACC, and 1.4% in F1 score on average. As we observe that the WTRG model tends to outperform the vanilla TRG model, we use WTRG in the rest of the experiments.

Table 4: Performance of WTRG over TRG on τ-graph time series

		`bitcoin`		`wiki-elec`	
Method	Metric	TRG	WTRG	TRG	WTRG
node2vec	AUC	0.9214	**0.9239**	**0.7348**	0.7344
	ACC	0.8294	**0.8412**	**0.6171**	0.6144
	F1	0.8285	**0.8408**	**0.5909**	0.5889
struc2vec	AUC	0.9274	**0.9301**	0.7840	**0.7933**
	ACC	0.7959	**0.8109**	0.6583	**0.6703**
	F1	0.7925	**0.8081**	0.6388	**0.6534**
MultiLENS	AUC	0.9226	**0.9389**	0.8106	**0.8143**
	ACC	0.8656	**0.8793**	0.7438	**0.7539**
	F1	0.8655	**0.8792**	0.7385	**0.7493**

Table 5: Mean rank (and std.) of the temporal network models across all base embedding methods and graphs based on AUC, ACC and F1, lower is better. The top-3 temporal network models are based on the new ϵ-graph time-series representation (fixed #edges)

Temporal Model	Mean Rank (MR) AUC	ACC	F1
WTRG-ϵ	**2.30 ± 2.16**	2.73 ± 1.95	2.66 ± 1.90
TSG-ϵ	2.43 ± 1.84	**2.61 ± 2.19**	2.70 ± 2.24
SG-ϵ	2.57 ± 1.64	2.66 ± 1.86	**2.59 ± 1.88**
WTRG-τ	2.80 ± 1.96	2.80 ± 1.98	2.86 ± 1.99
SG-τ	2.95 ± 1.91	2.84 ± 1.87	2.82 ± 1.83
TSG-τ	3.63 ± 1.75	3.46 ± 1.87	3.45 ± 1.80
SG	4.32 ± 1.88	3.89 ± 1.91	3.93 ± 1.94

6.3 Fixed #edges (ϵ) vs. time-scale (τ)

In this section, we investigate the effectiveness of different graph time-series representations (**Q1**). Due to the large number of experimental results, we define 2 evaluation metrics for this experiment. These newly proposed measurements are for readers to have a clear overview of the comparison results across all components in the proposed framework across all the datasets. We first evaluate the general performance of each temporal model through the mean ranking (and std) across all datasets and embedding methods in terms of the AUC, ACC and F1 score. Let $\mathbf{y}_{jk} \in \mathbb{R}^{|\mathcal{M}|}$ denote the vector of AUC (or ACC, F1) scores of the temporal models $\mathcal{M}$ for an embedding method $f_j \in \mathcal{F}$ and graph dataset k. Further, let $\pi(\mathbf{y}_{jk}, M_i)$ denote the rank of the temporal model $M_i \in \mathcal{M}$ for a given embedding method f_j and graph dataset $d_k \in \mathcal{D}$. The mean rank is computed as

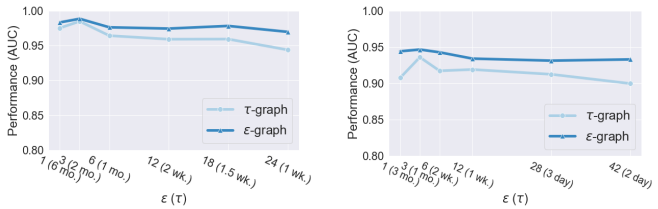
$$\text{MR}_i = \frac{1}{|\mathcal{D}||\mathcal{F}|} \sum_{d_k \in \mathcal{D}} \sum_{f_j \in \mathcal{F}} \pi(\mathbf{y}_{jk}, M_i) \tag{6}$$

Therefore, smaller values of MR indicate better model performance. We report the results in Table 5. In addition to the general performance, we also provide an intuitive ranking based on the number of times each model performs the best following [22]. This metric s_i reflects the occurrence of temporal model M_i to be optimal:

$$s_i = \sum_{d_k \in \mathcal{D}} \sum_{f_j \in \mathcal{F}} \mathbb{I}\{\pi(\mathbf{y}_{jk}, M_i) = 1\} \tag{7}$$

where $\mathbb{I}\{\pi(\mathbf{y}_{jk}, M_i) = 1\}$ returns 1 if $\pi(\mathbf{y}_{jk}, M_i) = 1$ and 0 otherwise. $\mathbb{I}\{\pi(\mathbf{y}_{jk}, M_i) = 1\}$ indicates that the temporal model M_i performs best for the given graph dataset d_k and base embedding method f_j. Thus, s_i denotes the total score of model M_i based on the number of times temporal model M_i appeared first in the ranking across all base embedding methods and graph datasets.

Performance. Based on the results shown in Table 5, our first observation is that the top-3 temporal models are those that use the proposed ϵ-graph time-series representation. These models perform comparably well in terms of AUC, ACC and F1 and are in general better than τ-graph time-series representation used in previous work. This finding indicates the general effectiveness of ϵ-graph time-series in representing the temporal network. We also compute an overall score by summing over each s_i for all evaluation criterion (bottom row in Table 6). We observe that the top models are ϵ-based, which demonstrate the effectiveness of ϵ-graph time-series in capturing the graph *structural changes* over edge *frequency changes* in prediction tasks.



(a) Performance on `bitcoin` (b) Performance on `fb-forum`

Figure 4: Sensitivity analysis. Link prediction performance on τ- and ϵ-graph. The τ-graphs are created based on different timescales, the ϵ-graphs are created via equal division. mo.: month. wk.: week.

Sensitivity Analysis. We also conduct a parameter sensitivity analysis on two datasets, `bitcoin` and `fb-forum`, to evaluate the impact of different values of τ and ϵ on the overall performance. For `bitcoin`, we train our model on temporal data spanning 6 months and test on the 7th month. For`fb-forum`, we train on 12 weeks and test on the 13rd week. We create the τ-graph time series using different scales (e.g., months, weeks, days), and generate the same number of ϵ-graphs with equal number of edges. Based on the result shown in Figure 4, we observe that our model running on the ϵ-graph time series consistently outperforms the τ-graph time series, while being more robust. This also indicates that in practice, the ϵ-graph time series can be used as an alternative to create temporal graph snapshots for various mining tasks, especially in preliminary graph analysis when the optimal timescale is undetermined.

Result 1. Overall, the proposed ϵ-graph time-series representation based on a fixed number of edges outperforms the time-scale τ-graph time-series across different scales, while being more robust.

6.4 Temporal Model Comparison

To answer **Q2**, we follow the formulation in Section 6.3 to quantitatively evaluate and rank the temporal models according to their effectiveness in prediction. We show the complete performance of temporal network models that perform the best following Equation (7) with respect to individual datasets in Table 6 to supplement the mean ranking in Table 5.

Notably, the TSG-ϵ model has the highest # of first ranks across all datasets, especially on datasets with short timespans (`wiki-edit` and `contacts`). It also has the highest # of first ranking, in terms of ACC and F1, which indicates that this model is generally promising but at the same time less stable than the other ϵ-based models. We also confirm this finding in Table 5 as it shows relatively higher variance of ranking. WTRG-ϵ performs the second best and is a close second to TSG-ϵ on datasets with long timespans. This is potentially due to how they model the temporal recency: TSG models the past information with exponential decay (3), while WTRG models it with the absolute temporal difference (5). Thus, TSG could still capture the temporal evolution within a relatively short period of time. Nevertheless, both TSG and WTRG perform well on all datasets even though spikes and fluctuation are observed such as `fb-forum` (Figure 1). There is not a single temporal model that prevails across all datasets. On the other hand, the WTRG model tends to perform well regardless of the timescales in graph representation, while the TSG model tends to perform well on graphs with short timespans. Besides, the temporal models that are combined with the proposed

Table 6: Temporal model performance across the temporal graphs. Each (i, j) is the # of times temporal model $M_j \in \mathcal{M}$ in graph G_i performed best comparing to the other models across all base embedding methods $f \in \mathcal{F}$ and evaluation criterion. We bold the temporal model that performs best overall for each graph.

	TSG-ϵ	WTRG-ϵ	SG-ϵ	SG-τ	WTRG-τ	SG	TSG-τ
bitcoin	**6**	**6**	4	5	0	0	0
stackoverflow	1	4	3	3	**9**	0	1
enron	4	1	1	3	**8**	4	0
wiki-elec	2	6	**7**	6	0	0	0
fb-forum	**10**	**10**	0	1	0	0	0
wiki-edit	**7**	3	2	3	2	2	2
reality-call	1	0	2	4	**6**	4	4
contacts-dublin	**9**	2	8	1	1	0	0
overall score	40	32	27	26	26	10	7

ϵ-graph time-series representation tend to outperform their other τ-counterparts, which is consistent with our previous findings from Section 6.3.

Result 2. Out of all models, WTRG-ϵ and TSG-ϵ tend to perform the best. Empirically, WTRG-ϵ is more stable overall (Table 5) and TSG-ϵ performs well on datasets with short timespans (Table 6).

6.5 Dynamic Embeddings: Variants vs. SOTA

To answer **Q3**, we use our framework to derive new dynamic embedding methods (by selecting the representation, temporal model, base method, etc.), and compare their performance to the state-of-the-art dynamic embedding methods on all 8 datasets. One would presumably expect that the state-of-the-art dynamic methods will outperform the dynamic embedding methods generalized by our framework, as are typically more complex and have been designed specifically for learning such dynamic node embeddings. We use 9 recent dynamic approaches during 2017 $\sim$ 2020 as baselines, including CTDNE [18], node2bits [10], DANE [16], DynGem [7] TIMERS [30], DynAE/DynAERNN [6], DySAT [25], DyHATR [29], and EvolveGCN[19].

Figure 5 shows the mean AUC for each method where the average is taken over all graphs investigated. As representative dynamic embedding methods from the proposed framework, we use 4 dynamic embedding variants of struc2vec (s2v-TSG-ϵ/τ, s2v-WTRG-ϵ/τ) and 4 variants of MultiLENS (ML-TSG-ϵ/τ, ML-WTRG-ϵ/τ). Strikingly, we observe that the dynamic embedding methods from the framework perform comparably or even better than the state-of-the-art methods that are designed particularly for temporal graphs and time-series prediction. In particular, ML-TSG-ϵ performs best with a mean gain of 12.34% followed by s2v-TSG-ϵ with a gain in AUC of 10.97%. Also, we note that our proposed framework is computationally efficient. Taking MultiLENS as an base embedding method, our proposed framework has the computational complexity $O(|E||\mathcal{G}|)$ where $|\mathcal{G}|$ is the number of graphs in the time-series, and the number of parameters to learn is $O(|V||\mathcal{G}|)$. This is also validated empirically throughout our experiment.

Result 3. The dynamic embeddings derived from our framework leveraging conventional static embedding methods (Section 5) perform better than state-of-the-art dynamic embedding methods.

Notably, in this experiment we do not aim to show substantial improvement of our framework over all the dynamic approaches,

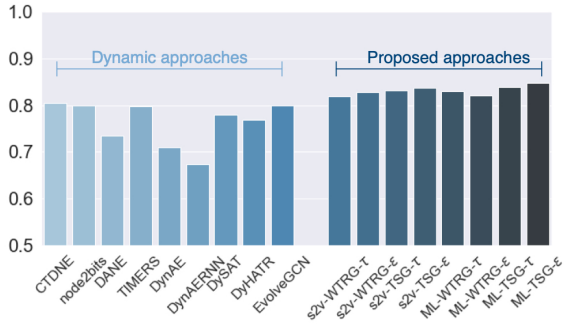


Figure 5: Predictive results of the dynamic embedding methods and our framework. Our proposed framework approximates well to approaches specifically designed for temporal graphs with comparable or even better performance (ML = MultiLENS, s2v = struc2vec).

especially those based on deep learning, since node features are not used. Instead, these results show that our proposed framework could capture the graph structures and temporal dependency at least as good as those recent dynamic approaches with less complexity (*i.e.*, no transitional or latent variables), Unlike methods that are based on complex models as "black boxes", the components of our proposed framework are easy-to-understand, which further motivates its usage for practitioners in predictive applications.

7 CONCLUSION

Despite the recent increasing interest in temporal networks in the field of representation learning, there has been relatively little work that systematically studies the properties of temporal network models and their cornerstones, the graph time-series representations. This works attempts to fill this gap by proposing a general yet powerful framework. We introduce the notion of ϵ-graph time-series to address data imbalance that arises with the traditional way of deriving a graph time-series based on a—sometimes arbitrary—timescale (*e.g.*, 1 day or 1 week). We find that the ϵ-graph time-series is beneficial to most existing embedding methods in temporal link prediction. Furthermore, our proposed framework gives rise to new dynamic embedding methods by combining these graph time-series representations, temporal models, and base static embedding methods. We find that although there is no single temporal model (or embedding method) that could prevail on any dataset, our proposed WTRG model and TSG model along with the ϵ time-series tend to perform the best across all datasets studied. We further show that these dynamic embedding approaches from our framework outperform recent, powerful dynamic embedding methods. Our future work is to incorporate node-wise features to further improve the model performance.

ACKNOWLEDGMENTS

This material is partially based upon work supported by the National Science Foundation under CAREER Grant No. IIS 1845491 and Army Young Investigator Award No. W911NF1810397, an Adobe Digital Experience research faculty award, and Amazon, Google, and Facebook faculty awards. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of funding parties.

REFERENCES

[1] N. K. Ahmed, R. A. Rossi, R. Zhou, J. B. Lee, X. Kong, T. L. Willke, and H. Eldardiry. Learning role-based graph embeddings. In *IJCAI StarAI*, 2018.

[2] A. Bojchevski and S. Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *ICLR*, 2018.

[3] S. Cao, W. Lu, and Q. Xu. Grarep: Learning graph representations with global structural information. In *CIKM*, pages 891–900. ACM, 2015.

[4] K. G. Coffman and A. M. Odlyzko. Growth of the internet. In *Optical fiber telecommunications IV-B*, pages 17–56. Elsevier, 2002.

[5] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec. Learning structural node embeddings via diffusion wavelets. In *KDD*, pages 1320–1329. ACM, 2018.

[6] P. Goyal, S. R. Chhetri, and A. Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, page 104816, 2019.

[7] P. Goyal, N. Kamra, X. He, and Y. Liu. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*, 2018.

[8] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864, 2016.

[9] R. Hisano. Semi-supervised graph embedding approach to dynamic link prediction. In *International Workshop on Complex Networks*, pages 109–121. Springer, 2018.

[10] D. Jin, M. Heimann, R. Rossi, and D. Koutra. node2bits: Compact time-and attribute-aware node representations for user stitching. In *ECML PKDD*, page 22, 2019.

[11] D. Jin, R. A. Rossi, E. Koh, S. Kim, A. Rao, and D. Koutra. Latent network summarization: Bridging network embedding and summarization. In *KDD*. ACM, 2019.

[12] J. Jin, M. Heimann, D. Jin, and D. Koutra. Toward understanding and evaluating structural node embeddings. *ACM Trans. Knowl. Discov. Data*, 16(3), nov 2021.

[13] S. Kumar, X. Zhang, and J. Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *KDD*, pages 1269–1278. ACM, 2019.

[14] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *KDD*, 2005.

[15] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[16] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu. Attributed network embedding for learning in a dynamic environment. In *CIKM*, pages 387–396. ACM, 2017.

[17] Y. Liu, T. Safavi, A. Dighe, and D. Koutra. Graph summarization methods and applications: A survey. *ACM Computing Surveys (CSUR)*, 51(3):1–34, 2018.

[18] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim. Continuous-time dynamic network embeddings. In *WWW BigNet*, 2018.

[19] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *AAAI*, volume 34, pages 5363–5370, 2020.

[20] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo. Struc2vec: Learning node representations from structural identity. In *KDD*, 2017.

[21] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[22] R. A. Rossi, N. K. Ahmed, H. Eldardiry, and R. Zhou. Similarity-based multi-label learning. In *IJCNN*, pages 1–8, 2018.

[23] R. A. Rossi, D. Jin, S. Kim, N. K. Ahmed, D. Koutra, and J. B. Lee. On proximity and structural role-based embeddings in networks: Misconceptions, techniques, and applications. *ACM TKDD*, 14(5):1–37, 2020.

[24] R. A. Rossi and J. Neville. Time-evolving relational classification and ensemble methods. In *PAKDD*, volume 7301, pages 1–13. Springer, 2012.

[25] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *WSDM*, pages 519–527, 2020.

[26] U. Singer, I. Guy, and K. Radinsky. Node embedding over temporal graphs. In *IJCAI*, pages 4605–4612, 7 2019.

[27] S. Soundarajan, A. Tamersoy, E. B. Khalil, T. Eliassi-Rad, D. H. Chau, B. Gallagher, and K. A. Roundy. Generating graph snapshots from streaming edge data. In *WWW Companion Volume*, pages 109–110. ACM, 2016.

[28] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. LINE: Large-scale Information Network Embedding. In *WWW*, pages 1067–1077, 2015.

[29] H. Xue, L. Yang, W. Jiang, Y. Wei, Y. Hu, and Y. Lin. Modeling dynamic heterogeneous network forlink prediction using hierarchical attentionwith temporal rnn. In *ECML PKDD*, 2020.

[30] Z. Zhang, P. Cui, J. Pei, X. Wang, and W. Zhu. Timers: Error-bounded svd restart on dynamic networks. In *AAAI*, 2018.

[31] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang. Dynamic network embedding by modeling triadic closure process. In *AAAI*, 2018.

[32] J. Zhu, X. Lu, M. Heimann, and D. Koutra. Node proximity is all you need: Unified structural and positional node and graph embedding. In *SDM*, pages 163–171. SIAM, 2021.

[33] Y. Zuo, G. Liu, H. Lin, J. Guo, X. Hu, and J. Wu. Embedding temporal network via neighborhood formation. In *KDD*, pages 2857–2866, 2018.

Appendix

A DATA DESCRIPTION

The detailed description of the graph datasets used in this work is given as follows.

- `enron`[2] records email exchanges between employees of Enron from May, 1999 to June, 2002.
- `bitcoin`[3] is a who-trusts-whom network of people who trade using bitcoins from Nov, 2010 to Feb., 2017. We study the user connectivity by dropping the edge signs.
- `wiki-elec`[2] contains the voting history based on the Wikipedia page edit history from Mar., 2004 to Jan., 2008.
- `stackoverflow`[3] is a temporal network consisting of three types of interactions on the stack exchange website MathOverflow: a user answers questions, a user comments on questions, and a user comments on answers.
- `wiki-edit`[4] is a public bipartite dataset containing one month of edits made by users in the Wikipedia page.
- `fb-forum`[2] is the Facebook-like Forum network that records users' activity in the forum.
- `contacts-dublin`[2] is a human contact network where nodes represent humans and edges between them represent proximity (i.e., contacts in the physical world).
- `reality-call`[2] is a subgraph of the reality mining study. Nodes are participants and edges are phone calls.

These graph datasets have been widely used in a variety of existing works as well.

B BASE AND DYNAMIC EMBEDDING METHOD CONFIGURATION

We configured all the base methods to achieve the best performance according to the respective papers. For all the static methods based on random walks (*i.e.*, node2vec, struc2vec), we perform 20 walks with the maximum walk length $L = 20$. For node2vec, we perform grid search over $p, q \in \{0.25, 0.50, 1, 2, 4\}$ as mentioned in [8] and report the best performance. For LINE and Multi-Lens, we incorporate 2nd-order proximity in the graph. For role2vec, we leverage the node degree as the feature for roles. For Graphwave, we perform the method to automatically select the scaling parameter with exact heat kernel matrix calculation. We set the final embedding with dimension $K = 128$ for evaluation, and leverages the weighted summation fusion approach so that the embedding dimensions of individual graphs are fixed to be the same.

For the state-of-the-art dynamic embedding methods, we follow the configuration given by the original paper or code repository provided. Specifically, for CTDNE, we set #walks= 10, the walking length $L = 20$. For node2bits, we perform short-term temporal random-walk with scope to be 3. The the #walks and the walking length are set to be the same as CTDNE. For DANE, we leverage both the offline computation model to derive node embeddings based on the first 6 graphs, and the online model to derive node embeddings for the 6th graph based on the first 5. We set the intermediate embedding dimensions to be 100 for both models and report the best performance. For TIMERS, we set the tolerance threshold value that is used to restart the optimal SVD calculation to be 0.17 as provided in the provided code repository. For DyAE/DyAERNN, we leverage the 2-layer auto-encoder/decoder with 400 and 200 units, respectively. We set the regularization hyperparameter to be 10^{-6}, bounding ratio for number of units in consecutive layers to be 0.3 as suggested in the paper, and perform grid search in the range of ±10% of the default value. In the learning stage, the sgd learning rate is set to be 10^{-6} with minibatch size to be 100. Lastly, for DySAT, we leverage the base model and perform grid search on the default hyperparameters in the range of ±10% of the default values.

[2]http://networkrepository.com
[3]https://snap.stanford.edu/data/
[4]https://github.com/srijankr/jodie

C COMPUTATIONAL COMPLEXITY OF WTRG

In this section we detail the computational complexity in deriving WTRG by showing the following property.

Property 1. *The number of edges in G_R is bounded by the number of temporally-valid walks in G.*

Based on Def. 5, an edge $(u, v) \in E_R$ indicates a temporally-valid walk reaching from u to v in G. However, this edge could correspond to multiple unique temporal walks with different intermediate nodes and associated timestamps, therefore, $|E_R|$ is no more than the number of temporally-valid walks in G.

Let N_i^R denote the temporally reachable nodes of i, $\Delta(G_R) = \max\{d(N_1^R), \ldots, d(N_n^R)\}$ is the maximum degree of a node in G_R, and ω is the window size. Then

$$|N_i^R| \leq \Delta(G_R) \leq \omega \tag{8}$$

According to Def. 5, a TRG is comprised by edges within the interval with size ω. These edges comprise upto ω different temporal walks originating from a specific node i. Therefore, based on Property 1, the number of edges originating from node i is bounded by the number of temporally-valid walks, which is ω. Therefore, as shown in Section 5.2, the number of edges in the derived WTRG graph is $O(|E|\omega)$.

D COMPLETE EXPERIMENTAL RESULTS

In this section, we show the complete experimental results using both the dynamic approaches generated from our proposed framework, and the state-of-the-art approaches specifically designed to handle dynamic graphs. In total, we perform 3 runs of the experiments on 8 dynamic graphs and report the average AUC, ACC and F1 metrics with the standard deviation. The dynamic methods generated from our proposed framework include 7 base static embedding methods coupled with 6 temporal models as well as the static form, *i.e.*, not using the temporal information. We also include 7 state-of-the-art dynamic embedding approaches. We leverage Equation (6) and Equation (7) to aggregate the extensive results for interpretation. Due to the space limit, we show the complete results on the first 3 datasets used in this work in Table 7, and refer the interested readers to our code repository indicated in Section 6.1 for the complete results on other datasets.

Table 7: Complete experimental results on the first 3 datasets. The values are represented using percentage %.

Base Method	Temporal Model	bitcoin AUC	bitcoin ACC	bitcoin F1	stackoverflow AUC	stackoverflow ACC	stackoverflow F1	enron AUC	enron ACC	enron F1
Node2vec	**Static**	89.05 ± 0.44	76.74 ± 0.70	75.89 ± 0.74	95.97 ± 0.02	84.34 ± 0.16	84.09 ± 0.17	76.11 ± 0.75	70.19 ± 0.88	69.96 ± 0.93
	SG-τ	93.80 ± 0.62	86.63 ± 0.20	86.58 ± 0.20	96.11 ± 0.05	88.00 ± 0.14	87.94 ± 0.15	80.44 ± 0.85	72.03 ± 0.42	72.02 ± 0.43
	WTRG-τ	95.15 ± 0.45	88.71 ± 0.61	88.65 ± 0.63	96.45 ± 0.05	89.67 ± 0.34	89.65 ± 0.35	79.78 ± 0.48	72.63 ± 0.67	72.62 ± 0.68
	SG-ϵ	95.08 ± 0.63	88.76 ± 0.67	88.74 ± 0.67	96.11 ± 0.05	88.13 ± 0.29	88.08 ± 0.29	78.37 ± 0.36	69.02 ± 0.72	68.86 ± 0.75
	WTRG-ϵ	95.89 ± 0.19	89.82 ± 0.47	89.79 ± 0.48	96.51 ± 0.02	89.41 ± 0.09	89.37 ± 0.09	78.37 ± 1.28	70.93 ± 0.30	70.91 ± 0.30
	TSG-τ	89.37 ± 0.40	76.79 ± 1.02	75.97 ± 1.12	96.08 ± 0.04	84.65 ± 0.25	84.42 ± 0.26	76.43 ± 1.50	70.35 ± 1.74	70.17 ± 1.83
	TSG-ϵ	90.46 ± 0.05	77.15 ± 0.09	76.37 ± 0.09	96.18 ± 0.02	84.69 ± 0.17	84.45 ± 0.19	77.83 ± 0.39	70.95 ± 0.32	70.86 ± 0.34
LINE	**Static**	87.58 ± 0.00	75.86 ± 0.00	75.11 ± 0.00	97.08 ± 0.00	91.41 ± 0.00	91.39 ± 0.00	76.19 ± 0.00	69.85 ± 0.00	69.70 ± 0.00
	SG-τ	89.75 ± 0.00	82.48 ± 0.00	82.46 ± 0.00	96.79 ± 0.00	91.87 ± 0.00	91.87 ± 0.00	81.84 ± 0.00	74.21 ± 0.00	74.06 ± 0.00
	WTRG-τ	89.49 ± 0.00	82.24 ± 0.00	82.24 ± 0.00	96.95 ± 0.00	91.92 ± 0.00	91.92 ± 0.00	81.89 ± 0.00	73.63 ± 0.00	73.49 ± 0.00
	SG-ϵ	89.18 ± 0.00	83.49 ± 0.00	83.48 ± 0.00	96.96 ± 0.00	91.85 ± 0.00	91.85 ± 0.00	80.38 ± 0.00	72.45 ± 0.00	72.36 ± 0.00
	WTRG-ϵ	91.68 ± 0.00	84.81 ± 0.00	84.81 ± 0.00	96.79 ± 0.00	91.48 ± 0.00	91.48 ± 0.00	82.71 ± 0.00	73.23 ± 0.00	73.18 ± 0.00
	TSG-τ	87.59 ± 0.00	76.48 ± 0.00	75.86 ± 0.00	97.25 ± 0.00	91.44 ± 0.00	91.42 ± 0.00	76.21 ± 0.00	68.05 ± 0.00	67.74 ± 0.00
	TSG-ϵ	89.28 ± 0.00	76.71 ± 0.00	75.99 ± 0.00	97.23 ± 0.00	91.26 ± 0.00	91.23 ± 0.00	75.22 ± 0.00	67.28 ± 0.00	67.06 ± 0.00
Struc2vec	**Static**	91.56 ± 0.16	81.70 ± 0.28	81.40 ± 0.33	96.99 ± 0.02	84.82 ± 0.17	84.57 ± 0.17	80.33 ± 0.44	73.31 ± 0.89	73.20 ± 0.94
	SG-τ	95.19 ± 0.59	87.83 ± 0.68	87.78 ± 0.68	96.92 ± 0.05	88.38 ± 0.29	88.30 ± 0.30	81.95 ± 0.86	72.10 ± 1.74	71.90 ± 1.83
	WTRG-τ	92.10 ± 0.97	84.22 ± 1.36	84.19 ± 1.35	96.66 ± 0.10	88.12 ± 1.00	88.04 ± 1.03	81.66 ± 1.31	72.61 ± 1.33	72.41 ± 1.36
	SG-ϵ	96.37 ± 0.68	89.85 ± 0.25	89.83 ± 0.25	97.01 ± 0.04	88.39 ± 0.11	88.30 ± 0.11	82.62 ± 0.56	73.42 ± 0.69	73.27 ± 0.71
	WTRG-ϵ	92.42 ± 0.33	83.78 ± 0.65	83.74 ± 0.65	96.67 ± 0.14	87.50 ± 0.59	87.40 ± 0.60	81.44 ± 0.79	72.98 ± 0.92	72.89 ± 0.90
	TSG-τ	91.61 ± 0.16	81.93 ± 0.13	81.65 ± 0.15	96.95 ± 0.03	84.92 ± 0.11	84.67 ± 0.11	80.76 ± 0.51	72.55 ± 0.49	72.44 ± 0.49
	TSG-ϵ	92.17 ± 0.07	81.85 ± 0.75	81.46 ± 0.79	97.16 ± 0.03	84.67 ± 0.15	84.38 ± 0.16	81.77 ± 0.18	74.88 ± 1.69	74.81 ± 1.72
Role2vec	**Static**	85.02 ± 0.04	76.27 ± 0.24	75.49 ± 0.24	92.10 ± 0.15	84.00 ± 0.05	83.77 ± 0.05	72.12 ± 1.31	66.77 ± 0.80	66.70 ± 0.82
	SG-τ	94.90 ± 0.84	87.77 ± 0.63	87.73 ± 0.63	95.45 ± 0.11	85.26 ± 0.14	85.15 ± 0.16	74.09 ± 1.09	66.59 ± 0.71	66.48 ± 0.72
	WTRG-τ	93.07 ± 1.11	85.64 ± 1.12	85.63 ± 1.12	95.51 ± 0.05	88.27 ± 0.71	88.24 ± 0.72	78.85 ± 1.17	71.38 ± 1.82	71.32 ± 1.85
	SG-ϵ	93.33 ± 0.66	85.70 ± 0.96	85.68 ± 0.97	95.34 ± 0.20	83.94 ± 0.06	83.67 ± 0.07	76.11 ± 1.15	67.76 ± 1.71	67.52 ± 1.75
	WTRG-ϵ	93.59 ± 0.33	86.81 ± 1.13	86.78 ± 1.12	95.90 ± 0.08	86.89 ± 0.85	86.79 ± 0.87	76.41 ± 2.22	68.34 ± 2.62	68.28 ± 2.67
	TSG-τ	85.50 ± 0.26	75.96 ± 0.43	75.16 ± 0.44	92.36 ± 0.05	84.00 ± 0.03	83.76 ± 0.03	71.45 ± 2.11	66.29 ± 1.73	66.15 ± 1.82
	TSG-ϵ	86.03 ± 0.20	75.78 ± 0.88	75.01 ± 0.91	93.13 ± 0.10	83.80 ± 0.06	83.52 ± 0.07	75.79 ± 1.59	69.71 ± 1.81	69.67 ± 1.84
Graphwave	**Static**	91.73 ± 0.00	77.96 ± 0.00	77.25 ± 0.00	96.89 ± 0.00	84.49 ± 0.00	84.26 ± 0.00	77.55 ± 0.00	71.88 ± 0.00	71.84 ± 0.00
	SG-τ	99.13 ± 0.00	91.90 ± 0.00	91.85 ± 0.00	96.84 ± 0.00	84.49 ± 0.00	84.26 ± 0.00	74.25 ± 0.00	65.48 ± 0.00	65.11 ± 0.00
	WTRG-τ	99.13 ± 0.00	91.36 ± 0.00	91.29 ± 0.00	96.89 ± 0.00	84.51 ± 0.00	84.28 ± 0.00	75.52 ± 0.00	67.69 ± 0.00	67.52 ± 0.00
	SG-ϵ	99.48 ± 0.00	91.12 ± 0.00	91.05 ± 0.00	97.07 ± 0.00	84.25 ± 0.00	83.97 ± 0.00	71.94 ± 0.00	62.35 ± 0.00	61.81 ± 0.00
	WTRG-ϵ	99.42 ± 0.00	89.88 ± 0.00	89.77 ± 0.00	97.07 ± 0.00	84.25 ± 0.00	83.97 ± 0.00	74.38 ± 0.00	65.08 ± 0.00	64.83 ± 0.00
	TSG-τ	91.98 ± 0.00	77.26 ± 0.00	76.46 ± 0.00	96.96 ± 0.00	84.49 ± 0.00	84.26 ± 0.00	76.77 ± 0.00	70.50 ± 0.00	70.45 ± 0.00
	TSG-ϵ	91.96 ± 0.00	76.56 ± 0.00	75.76 ± 0.00	96.99 ± 0.00	84.25 ± 0.00	83.97 ± 0.00	78.24 ± 0.00	71.35 ± 0.00	71.31 ± 0.00
G2G	**Static**	85.16 ± 0.87	76.37 ± 0.91	75.97 ± 1.02	93.15 ± 0.10	83.80 ± 0.11	83.66 ± 0.12	75.67 ± 0.15	71.01 ± 0.38	70.92 ± 0.39
	SG-τ	72.64 ± 1.56	67.63 ± 1.51	67.61 ± 1.52	92.94 ± 0.18	86.95 ± 0.28	86.94 ± 0.28	76.99 ± 0.73	70.28 ± 0.98	70.05 ± 1.08
	WTRG-τ	80.32 ± 0.92	72.04 ± 1.15	72.04 ± 1.15	95.64 ± 0.30	90.30 ± 0.17	90.29 ± 0.17	79.12 ± 0.43	70.87 ± 1.14	70.82 ± 1.21
	SG-ϵ	71.66 ± 3.08	67.00 ± 2.46	66.96 ± 2.47	92.32 ± 0.51	86.58 ± 0.41	86.58 ± 0.41	76.75 ± 0.35	70.15 ± 0.72	69.94 ± 0.84
	WTRG-ϵ	77.97 ± 1.37	71.08 ± 2.18	71.07 ± 2.18	95.37 ± 0.24	90.67 ± 0.38	90.65 ± 0.39	77.85 ± 0.48	70.73 ± 0.80	70.64 ± 0.81
	TSG-τ	85.59 ± 0.91	76.20 ± 0.66	75.75 ± 0.66	93.27 ± 0.32	83.91 ± 0.54	83.77 ± 0.57	74.97 ± 0.44	70.26 ± 0.61	70.16 ± 0.63
	TSG-ϵ	87.20 ± 0.87	77.98 ± 0.72	77.63 ± 0.76	93.83 ± 0.10	84.03 ± 0.19	83.86 ± 0.19	75.65 ± 0.12	69.56 ± 0.57	69.47 ± 0.63
MultiLENS	**Static**	91.28 ± 0.00	81.85 ± 0.00	81.63 ± 0.00	97.12 ± 0.00	90.24 ± 0.00	90.19 ± 0.00	80.12 ± 0.00	71.11 ± 0.00	70.83 ± 0.00
	SG-τ	82.43 ± 0.00	75.47 ± 0.00	75.06 ± 0.00	96.95 ± 0.00	92.98 ± 0.00	92.98 ± 0.00	81.12 ± 0.00	73.80 ± 0.00	73.68 ± 0.00
	WTRG-τ	84.48 ± 0.00	77.80 ± 0.00	77.55 ± 0.00	96.97 ± 0.00	92.16 ± 0.00	92.15 ± 0.00	80.28 ± 0.00	74.25 ± 0.00	74.10 ± 0.00
	SG-ϵ	89.56 ± 0.00	80.22 ± 0.00	79.99 ± 0.00	97.24 ± 0.00	92.84 ± 0.00	92.84 ± 0.00	81.58 ± 0.00	74.12 ± 0.00	74.00 ± 0.00
	WTRG-ϵ	87.33 ± 0.00	79.13 ± 0.00	78.94 ± 0.00	96.79 ± 0.00	92.56 ± 0.00	92.55 ± 0.00	81.46 ± 0.00	72.58 ± 0.00	72.37 ± 0.00
	TSG-τ	91.28 ± 0.00	81.85 ± 0.00	81.63 ± 0.00	97.12 ± 0.00	90.24 ± 0.00	90.19 ± 0.00	80.12 ± 0.00	71.11 ± 0.00	70.83 ± 0.00
	TSG-ϵ	92.50 ± 0.00	82.01 ± 0.00	81.67 ± 0.00	97.18 ± 0.00	89.84 ± 0.00	89.78 ± 0.00	81.59 ± 0.00	72.70 ± 0.00	72.43 ± 0.00
CTDNE		92.70 ± 0.12	86.29 ± 0.14	86.24 ± 0.14	97.43 ± 0.01	91.81 ± 0.22	91.79 ± 0.22	67.84 ± 2.64	63.74 ± 1.32	63.72 ± 1.34
Node2bits		88.97 ± 0.16	80.69 ± 0.48	80.43 ± 0.51	97.01 ± 0.03	90.85 ± 0.15	90.82 ± 0.15	83.95 ± 0.19	76.25 ± 0.35	76.25 ± 0.35
DANE		73.84 ± 0.00	67.29 ± 0.00	64.92 ± 0.00	75.31 ± 0.00	74.13 ± 0.00	73.22 ± 0.00	86.02 ± 0.00	76.21 ± 0.00	76.17 ± 0.00
TIMERS	-	63.50 ± 0.00	61.68 ± 0.00	58.83 ± 0.00	96.30 ± 0.00	89.00 ± 0.00	88.93 ± 0.00	87.72 ± 0.00	79.78 ± 0.00	79.77 ± 0.00
DynAE		61.41 ± 0.21	58.57 ± 0.00	55.36 ± 0.00	73.23 ± 0.94	71.57 ± 0.26	70.82 ± 0.31	76.90 ± 1.23	70.65 ± 1.64	70.62 ± 1.66
DynAERNN		57.30 ± 3.93	58.36 ± 0.46	54.83 ± 0.61	72.48 ± 0.92	71.20 ± 1.32	70.34 ± 1.63	59.10 ± 0.31	57.64 ± 1.16	57.55 ± 1.11
DySAT		61.08 ± 0.12	58.10 ± 0.68	58.09 ± 0.68	92.97 ± 1.14	84.06 ± 0.97	84.15 ± 1.51	86.75 ± 0.01	79.83 ± 0.03	79.82 ± 0.03