Supra-Laplacian Encoding for Transformer on Dynamic Graphs

Yannis Karmim
Conservatoire National des Arts et Métiers
CEDRIC, EA 4629
F 75003, Paris, France
yannis.karmim@cnam.fr

Marc Lafon
Conservatoire National des Arts et Métiers
CEDRIC, EA 4629
F 75003, Paris, France
marc.lafon@lecnam.net

Raphaël Fournier S'niehotta
Conservatoire National des Arts et Métiers
CEDRIC, EA 4629
F 75003, Paris, France
fournier@cnam.fr

Nicolas Thome
Sorbonne Université
CNRS, ISIR
F-75005 Paris, France
nicolas.thome@isir.upmc.fr

Abstract

Fully connected Graph Transformers (GT) have rapidly become prominent in the static graph community as an alternative to Message-Passing models, which suffer from a lack of expressivity, oversquashing, and under-reaching. However, in a dynamic context, by interconnecting all nodes at multiple snapshots with self-attention,GT loose both structural and temporal information. In this work, we introduce **S**upra-**LA**placian encoding for spatio-temporal **T**ransform**E**rs (SLATE), a new spatio-temporal encoding to leverage the GT architecture while keeping spatio-temporal information. Specifically, we transform Discrete Time Dynamic Graphs into multi-layer graphs and take advantage of the spectral properties of their associated supra-Laplacian matrix. Our second contribution explicitly model nodes' pairwise relationships with a cross-attention mechanism, providing an accurate edge representation for dynamic link prediction. SLATE outperforms numerous state-of-the-art methods based on Message-Passing Graph Neural Networks combined with recurrent models (*e.g.* , LSTM), and Dynamic Graph Transformers, on 9 datasets. Code is open-source and available at this link https://github.com/ykrmm/SLATE.

1 Introduction

Dynamic graphs are crucial for modeling interactions between entities in various fields, from social sciences to computational biology [58, 16, 19, 20]. Link prediction on dynamic graphs is an all-important task, with diverse applications, such as predicting user actions in recommender systems, forecasting financial transactions, or identifying potential academic collaborations. Dynamic graphs can be modeled as a time series of static graphs captured at regular intervals (Discrete Time Dynamic Graphs, DTDG) [41, 54].

Standard approaches for learning representations on DTDGs combine Message-Passing GNNs (MP-GNNs) with temporal RNN-based models [59, 39, 34]. In static contexts, Graph Transformers (GT) [11, 51, 26] offer a compelling alternative to MP-GNNs that faced several limitations [53, 42]. Indeed, their fully-connected attention mechanism captures long-range dependencies, resolving issues such as oversquashing [1]. GTs directly connect nodes, using the graph structure as a soft bias through positional encoding [37]. Incorporating Laplacian-based encodings in GTs provably enhances their expressiveness compared to MP-GNNs [26, 11].

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

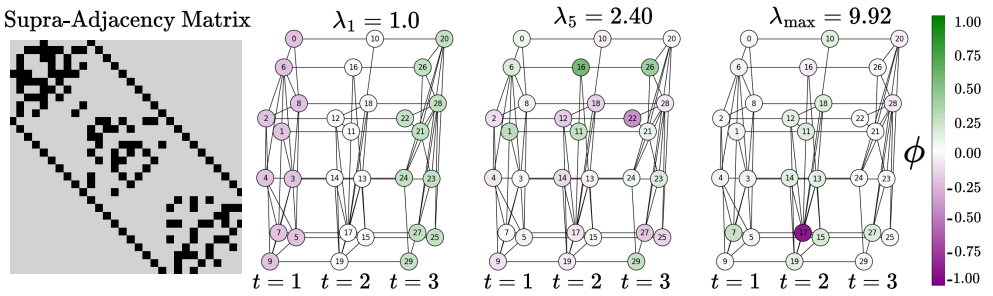

Figure 1: SLATE is a fully connected transformer for dynamic link prediction, which innovatively performs a joint spatial and temporal encoding of the dynamic graph. SLATE models a DTDG as a multi-layer graph with temporal dependencies between a node and its past. Building the supra-adjacency matrix of a randomly-generated toy dynamic graph with 3 snapshots (*left*) and analysing the spectrum of its associated supra-Laplacian (*right*) provide fundamental spatio-temporal information. The projections on eigenvectors associated with smaller eigenvalues (λ_1) capture global graph dynamics: node colors are different for each time step. Larger eigenvalues (*e.g.* $\lambda_{\max}$), capture more localized spatio-temporal information (see Appendix A.1).

Exploiting GTs on dynamic graphs would require a spatio-temporal encoding that effectively retains both structural and temporal information. The recent works that have extended GTs to dynamic graphs capture spatio-temporal dependencies between nodes by using *partial* attention mechanisms [31, 56, 47, 18]. Moreover, these methods also employ encodings which *independently* embed the graph structure and the temporal dimension. Given that the expressiveness of GTs depends on an accurate spatio-temporal encoding, designing one that interweaves time and position information could greatly enhance their potential and performance.

The vast majority of neural-based methods for dynamic link prediction rely on node representation learning [34, 55, 38, 59, 39]. Recent works enrich node embeddings with pairwise information for a given node-pair using co-occurrence neighbors matching [60, 48] or cross-attention on historical sub-graphs [47]. However these methods neglect the global information of the graph by sampling different spatio-temporal substructures around targeted nodes.

Pioneering work in the complex network community has studied temporal graphs with multi-layers models and supra-adjacency matrices [44, 25]. The spectral analysis of such matrices can provide valuable structural and temporal information [8, 36]. However, how to adapt this formalism for learning dynamic graphs with transformer architectures remains a widely open question.

In this work, we introduce **S**upra-**LA**placian encoding for spatio-temporal **T**ransform**E**rs (SLATE), a new unified spatio-temporal encoding which allows to fully exploit the potential of the GT architecture for the task of dynamic link prediction. As illustrated on Figure 1, adapting supra-Laplacian matrices to dynamic graph can provide rich spatio-temporal information for positional encoding. SLATE is based on the following two main contributions:

- We bridge the gap between multi-layer networks and Discrete Time Dynamic Graphs (DTDGs) by adapting the spectral properties of supra-Laplacian matrices for transformers on dynamic graphs. By carefully transforming the supra-Laplacian matrices for DTDGs, we derive a connected multi-layer graph that captures various levels of spatio-temporal information. We introduce a fully-connected spatio-temporal transformer that leverages this unified supra-Laplacian encoding.

- The proposed transfomer captures dependencies between nodes across multiple time steps, creating dynamic representations.

  To enhance link prediction, we introduce a lightweight edge representation module using *cross-attention* only between the temporal representations of node pairs, precisely capturing their evolving interactions. This results in a unique edge embedding, significantly streamlining the prediction process and boosting both efficiency and accuracy.

We conduct an extensive experimental validation of our method across 11 real and synthetic discrete-time dynamic graph datasets. SLATE outperforms state-of-the-art results by a large margin. We also validate the importance of our supra-Laplacian unified spatio-temporal encoding and the edge module for optimal performances. Finally, SLATE remains efficient since it uses a single-layer transformer, and we show impressive results on larger graph datasets, indicating good scalability, and limited time-memory overhead.

2 Related work

Dynamic Graph Neural Networks on DTDGs. The standard approach to learn on DTDGs [41, 54] involves using two separate spatial and temporal models. The spatial model is responsible for encoding the structure of the current graph snapshot, while the temporal model updates the dynamic either of the graph representations [39, 40, 59, 30, 28] or the graph model parameters [34, 15]. Recently, ROLAND [59] introduced a generic framework to use any static graph model for spatial encoding coupled with a recurrent-based (LSTM [17], RNN, GRU) or attention-based temporal model. These above methods mainly use a MP-GNN as spatial model [23, 46, 58]. However, MP-GNNs are known to present critical limitations: they struggle to distinguish simple structures like triangles or cycles [33, 4], and fail to capture long-range dependencies due to oversquashing [1, 42]. To overcome these limitations, some works have adopted a fully-connected GT as spatial model, benefiting from its global attention mechanism [6, 50, 61]. In [39], the local structure is preserved by computing the attention on direct neighbors. In contrast to these works, SLATE uses a unique spatio-temporal graph transformer model, greatly simplifying the learning process.

Graph Transformer. In static contexts, Graph Transformers have been shown to provide a compelling alternative to MP-GNNs [11]. GTs [51, 37, 22, 57] enable direct connections between all nodes, using the graph's structure as a soft inductive bias, thus resolving the oversquashing issue. The expressiveness of GTs heavily depends on positional or structural encoding [11, 32, 12, 3]. In [11], the authors use the eigenvectors associated with the k-lowest eigenvalues of the Laplacian matrix, which allows GTs to distinguish structures that MP-GNNs are unable to differentiate. Following the success of Laplacian positional encoding on static graphs, SLATE uses the eigenvectors of the supra-Laplacian of a multi-layer graph representation of DTDGs as spatio-temporal encoding.

Dynamic Graph Transformers. To avoid separately modelling structural and temporal information as dynamic Graph Neural Networks usually do on DTDGs, recent papers have adopted a unified model based on spatio-temporal attention [31, 18]. This novel approach make those models close to transformer-based methods classically employed to learn on Continuous Time Dynamic Graphs (CTDG) [52, 47, 49]. Among them, some preserve the local structure by computing attention only on direct neighbors [52, 39], while others sample local spatio-temporal structures around nodes [47, 31, 56, 18] and perform fully-connected attention. However, their spatio-temporal encoding is still built by concatenating a spatial and a temporal encoding that are computed independently. The spatial encoding is either based on a graph-based distance [47, 18] or on a diffusion-based measure [31]. The temporal encoding is usually sinus-based [52, 49, 2] as in the original transformer paper [45]. Another drawback of these methods [31, 47, 18, 49] is that they use only sub-graphs to represent the local structure around a given node. Therefore, their representations of the nodes are computed on different graphs and thus fail to capture global and long-range interactions. Contrary to those approaches, our SLATE model uses the same graph to compute node representations in a fully-connected GT between all nodes within temporal windows. It features a unified spatio-temporal encoding based on the supra-Laplacian matrix.

Dynamic Link Prediction methods. For dynamic link prediction, many methods are based only on *node* representations and use MLPs or cosine similarity to predict the existence of a link [39, 40, 38]. Recent approaches complement node representations by incorporating pairwise information. Techniques like co-occurrence neighbors matching [60, 48] or cross-attention on historical sub-graphs [47] are employed. However, these methods often overlook the global graph structure by focusing on sampled spatio-temporal substructures. For instance, CAW-N [48] uses anonymous random walks around a pair of nodes and matches their neighborhoods, while DyGformer [60] applies transformers to one-hop neighborhoods and calculates co-occurrences. These localized approaches fail to capture the broader graph context. TCL [47] is the closest to SLATE, using cross-attention between spatio-

temporal representations of node pairs. TCL samples historical sub-graphs using BFS and employs contrastive learning for node representation. However, it still relies on sub-graph sampling, missing the full extent of the global graph information. In contrast, SLATE leverages the entire graph's spectral properties through the supra-Laplacian, incorporating the global structure directly into the spatio-temporal encoding. This holistic approach allows SLATE to provide a richer understanding of dynamic interactions, leading to superior link prediction performance.

3 The SLATE Method

In this section, we describe our fully-connected dynamic graph transformer model, SLATE, for link prediction. The core idea in section 3.1 is to adapt the supra-Laplacian matrix computations for dynamic graph transformer (DGTs), and to introduce our new spatio-temporal encoding based on its spectral analysis. In section 3.2, we detail our full-attention transformer to capture the spatio-temporal dependencies between nodes at different time steps. Finally, we detail our edge representation module for dynamic link prediction in section 3.3. Figure 2 illustrates the overall SLATE framework.

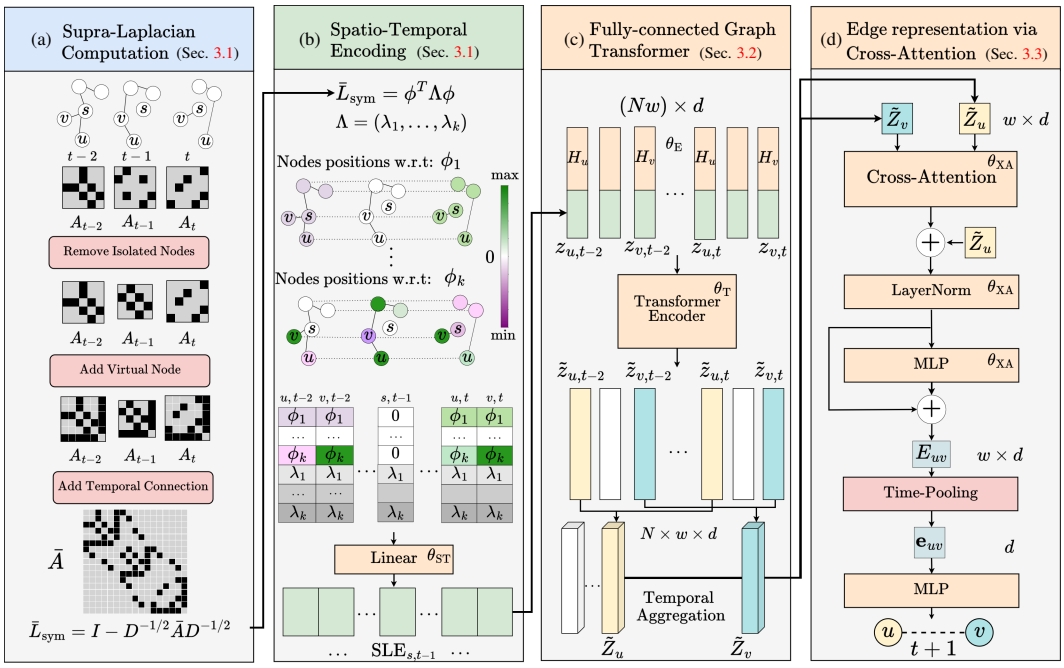

Figure 2: The **SLATE model** for link prediction with dynamic graph transformers (DGTs). To recover the lost spatio-temporal structure in DGTs, we adapt the supra-Laplacian matrix computation to DGTs by making the input graph provably connected (a), and use its spectral analysis to introduce a specific encoding for DGTs (b). (c) Applies a fully connected spatio-temporal transformer between all nodes at multiple time-step. Finally, we design in (d) an edge representations module dedicated to link prediction using cross-attention on multiple temporal representations of the nodes.

Notations. Let us consider a DTDG $\mathcal{G}$ as an undirected graph with a fixed number of N nodes across snapshots, represented by the set of adjacency matrices $\mathcal{A} = \{A_1, ..., A_T\}$. Its supra-graph, the multi-layer network $\bar{G} = (\bar{V}, \bar{E})$, is associated to a supra-adjacency matrix $\bar{A}$, obtained by stacking A_i diagonally (see Eq. (7) in Appendix A.1). Then, the supra-Laplacian matrix $\bar{L}$ is defined as $\bar{L} = I - \bar{D}^{-1/2}\bar{A}\bar{D}^{-1/2}$, where I is the identity matrix and $\bar{D}$ is the degree matrix of $\bar{G}$. Let $\mathbf{x_u} \in \mathbb{R}^F$ be the feature vector associated with the node u (which remains fixed among all snapshots). Finally, let consider the random variable $y \in \{0, 1\}$ such that $y = 1$ if nodes u and v are connected and $y = 0$ otherwise.

3.1 Supra-Laplacian as Spatio-Temporal Encoding

In this section, we cast Discrete Time Dynamic Graphs (DTDGs) as multi-layer networks, and use the spectral analysis of their supra-graph and generate a powerful spatio-temporal encoding for our fully-connected transformer.

DTDG as multi-layer graphs. If a graph is connected, its spectral analysis provides a rich information of the global graph dynamics, as shown in Figure 1. The main challenge in casting DTDG as multi-layer graphs relates to its disconnectivity, which induces as many zero eigenvalues as connected components. DTDG have in practice a high proportion of isolated nodes per snapshot (see Figure 3 in experiments), making the spectral analysis on the raw disconnected graph useless. Indeed, it mainly indicates positions relative to isolated nodes, losing valuable information on global dynamics and local spatio-temporal structures. We experimentally validate that it is mandatory to compute the supra-Laplacian matrix on a connected graph to recover a meaningful spatio-temporal structure.

Supra-Laplacian computation. To overcome this issue and make the supra-graph connected, we follow three steps: (1) remove isolated nodes in each adjacency matrix, (2) introduce a virtual node in each snapshot to connect clusters, and (3) add a temporal self-connection between a node and its past if it existed in the previous timestep. We avoid temporal dependencies between virtual nodes to prevent artificial connections. These 3 transformation steps make the resulting supra-graph provably connected. This process is illustrated in Figure 2a, and we give the detailed algorithm in Appendix A.3.

Spatio-temporal encoding. With a connected $\bar{\mathcal{G}}$, the second smallest eigenvalue λ_1 of the supra-Laplacian $\bar{L}$ is guaranteed to be non-negative (see proof in Appendix A.3), and its associated Fiedler vector ϕ_1 reveals the dynamics of $\bar{\mathcal{G}}$ (Figure 1). In practice, similar to many static GT models [26, 37, 12], we retrieve the first k eigenvectors of the spectrum of $\bar{L}$, with k being a hyper-parameter. The spectrum can be computed in $O(k^2N')$ and have a memory complexity of $O(kN')$ where N' is the size of $\bar{A}$, and we follow the literature to normalize the eigenvectors and resolve sign ambiguities [26]. The supra-Laplacian spatio-temporal encoding vector of the node u at time t is:

$$\text{SLE}_{u,t} = \left\{ \begin{array}{ll} g_{\theta_{\text{ST}}}(\bar{L}_{u,t} \cdot [\phi_1, \phi_2, ..., \phi_k] \oplus \text{diag}(\Lambda)) & \text{if } u_t \text{ is not isolated} \\ g_{\theta_{\text{ST}}}(\mathbf{0}_k \oplus \text{diag}(\Lambda)) & \text{otherwise} \end{array} \right\} \tag{1}$$

where $\oplus$ denotes the concatenation operator. $\bar{L}_{u,t} \cdot [\phi_1, \phi_2, ..., \phi_k] = [\phi_1^{u,t}, \phi_2^{u,t}, ..., \phi_k^{u,t}]$ contains the projections of the node u at time t in the eigenspace spanned by the k first eigenvectors of $\bar{L}$, $\text{diag}(\Lambda)$ contains the eigenvalues of $\bar{L}$ (which are the same for all nodes) and $g_{\theta_{\text{ST}}}$ is a linear layer allowing to finely adapt the supra-graph spectrum features to the underlying link prediction task. Note that because we did not include isolated nodes in the computation of the supra-Laplacian, we replace the eigenvector projections by a null vector $\mathbf{0}^k$ for these nodes. All the steps involved in constructing our spatio-temporal encoding are illustrated in Figure 2b.

3.2 Fully-connected spatio-temporal transformer

In this section, we describe the architecture of our fully-connected spatio-temporal transformer, f_{θ_T}, to construct node representations that captures long-range dependencies between the nodes at each time step. We illustrate our fully-connected GT in Figure 2c. We employ a single transformer block, such that our architecture remains lightweight. This is in line with recent findings showing that a single encoder layer with multi-head attention is sufficient for high performance, even for dynamic graphs [51].

The input representation of the node u_t is the concatenation of the node embeddings (which remains the same for each snapshot) and our supra-Laplacian spatio-temporal encoding:

$$\mathbf{z}_{u,t} = g_{\theta_{\text{E}}}(\mathbf{x_u}) \oplus \text{SLE}_{u,t} \tag{2}$$

where $g_{\theta_{\text{E}}}$ is a linear projection layer and $\oplus$ denotes the concatenation operator. Then we stack all the representations of *each nodes at each time step* within a time window of size w to obtain the input sequence, $Z \in \mathbb{R}^{(Nw)\times d}$, of the GT.

The fully-connected spatio-temporal transformer, f_{θ_T}, then produces a unique representation $\tilde{Z} \in \mathbb{R}^{(Nw)\times d}$ for each node at each time-step :

$$\tilde{Z} = f_{\theta_T}(Z). \tag{3}$$

Surprisingly, considering all temporal snapshots did not yield better results in our experiments (see Figure 4 in section 4.2).

Unlike previous DGT methods that sample substructures around each nodes [31, 56, 47], SLATE leverages the full structure of the DTDG within the time window. This approach ensures that no nodes are arbitrarily discarded in the representation learning process, as we use the same information source Z for all nodes.

3.3 Edge Representation with Cross-Attention

In this section, we present our innovative edge representation module `Edge`. It is designed for efficient dynamic link prediction and leverage the node representations learned by our fully-connected spatio-temporal GT. We illustrated our module in Figure 2d. This module is composed of a cross-attention model, $f_{\theta_{\text{XA}}}$, that captures pairwise information between the historical representation of two targeted nodes followed by a classifier to determine the presence of a link.

For a link prediction at time $t+1$ on a given node pair (u, v), we aggregate all temporal representations of u and v resulting in two sequences $\tilde{Z}_{u,t} = [\tilde{\mathbf{z}}_{u,t-w}, \ldots, \tilde{\mathbf{z}}_{u,t}]$ and $\tilde{Z}_{v,t} = [\tilde{\mathbf{z}}_{v,t-w}, \ldots, \tilde{\mathbf{z}}_{v,t}]$. We use these multiple embeddings to build a pairwise representation that captures dynamic relationships over time. Then, the cross-attention module $f_{\theta_{\text{XA}}}$ produces a pairwise representation of the sequence $E_{u,v} \in \mathbb{R}^{w\times d}$:

$$E_{u,v} = f_{\theta_{\text{XA}}}(\tilde{Z}_{u,t}, \tilde{Z}_{v,t}). \tag{4}$$

We obtain the final edge representation $\mathbf{e_{u,v}} \in \mathbb{R}^d$ by applying an average time-pooling operator and we compute the probability that the nodes u_{t+1} and v_{t+1} are connected with:

$$p(y = 1|\ \mathbf{e_{u,v}}) = \sigma(\text{MLP}(\mathbf{e_{u,v}})). \tag{5}$$

SLATE differs from methods that enrich node and edge representations with pairwise information by sampling substructures around each node [47, 60, 48, 18]. Instead, we first compute node representations based on the same dynamic graph information contained in Z. Then, we capture fine-grained dynamics specific to each link (u, v) through a cross-attention mechanism.

Our training resort to the standard Binary Cross-Entropy loss function. In practice, for a node u, we sample a negative pair v_{neg} and a positive pair v_{pos}:

$$\mathcal{L}_\theta = \text{BCE}(p(y = 1|\mathbf{e}_{\mathbf{u},\mathbf{v}_{\text{pos}}})) + \text{BCE}(p(y = 0|\mathbf{e}_{\mathbf{u},\mathbf{v}_{\text{neg}}})). \tag{6}$$

In this context, $\theta = \{\theta_{\text{XA}}, \theta_{\text{T}}, \theta_{\text{ST}}, \theta_{\text{E}}\}$ represents all the parameters within the edge representation module θ_{XA}, the fully-connected transformer θ_T, the spatio-temporal linear layer θ_{ST} and the node embedding parameters θ_E as illustrated in Figure 2.

3.4 SLATE Scalability

The theoretical complexity of attention computation is $O(N^2)$ per snapshot, scaling to $O((NT)^2)$ when considering all T snapshots. However, as shown in our experiments (Figure 4) and consistent with recent works [21], a large temporal context is often unnecessary. By using a time window w with $w \ll T$ (similar to other DGT architectures [31, 56]), we reduce complexity to $O((Nw)^2)$. For predictions at time $t + 1$, we focus only on snapshots from G_{t-w} to G_t. Ablation studies confirm that smaller time windows deliver excellent results across various real-world datasets. We further leverage FLASH Attention [9] to optimize memory usage and computation. Additionally, we incorporate Performer [5], which approximates the softmax computation of the attention matrix, reducing the complexity to $O(Nw)$. This enables us to scale efficiently to larger graphs, as shown in Table 5, while maintaining high performance (see Table 15) with manageable computational resources.

4 Experiments

We conduct extensive experiments to validate SLATE for link prediction on discrete dynamic graphs, including state-of-the-art comparisons in section 4.1. In section 4.2, we highlight the benefits of our two main contributions, the importance of connecting our supra-graph, and the ability of SLATE to scale to larger datasets with reasonable time and memory consumption compared to MP-GNNs.

Implementation details. We use one transformer Encoder Layer [45]. For larger datasets, we employ Flash Attention [9] for improved time and memory efficiency. Further details regarding model parameters and their selections are provided in Table 8. We fix the token dimension at $d = 128$ and the time window at $w = 3$ for all our experiments. We use an SGD optimizer for all of our experiments. Further details on hyper-parameters search, including the number of eigenvectors for our spatio-temporal encoding, are in Appendix D.

4.1 Comparison to state-of-the-art

Since both the continuous and discrete communities evaluate on similar data, we compare SLATE to state-of-the-art DTDG (Table 1) and CTDG (Table 2) models. Best results are in bold, second best are underlined. More detailed results and analyses are presented in Appendix E.1.

Baselines and evaluation protocol. To compare the benefits of fully connected spatio-temporal attention with a standard approach using transformers, we designed the ROLAND-GT model based on the ROLAND framework [59]. This model follows the stacked-GNN approach [41], equipped with the encoder f_{θ_T} described in section 3 including static Laplacian positional encoding [11], and a LSTM [17] updating the node embeddings.

We adhere to the standardized evaluation protocols for continuous models [60] and discrete models [55]. Our evaluation follows these protocols, including metrics, data splitting, and the datasets provided. Results in Table 1 and Table 2 are from the original papers, except those marked with †. We report the average results and standard deviations from five runs to assess robustness. Additional results, including hard negative sampling evaluation, are in Appendix E.2.

Datasets. In Table 6 Appendix C, we provide detailed statistics for the datasets used in our experiments. An in-depth description of the datasets is given in Appendix C. We evaluate on DTDGs datasets provided by [60] and [55], we add a synthetic dataset SBM based on stochastic block model [29], to evaluate on denser DTDG.

Table 1: Comparison to DTDG models on discrete data. ROC-AUC

Method	HepPh	AS733	Enron	Colab	SBM†	**Avg.**
GCN† [23]	74.52 ± 0.80	96.65 ± 0.05	91.31 ± 0.45	88.28 ± 0.58	<u>95.96</u> ± 0.32	89.34 ± 0.44
GIN† [53]	71.47 ± 0.56	93.53 ± 0.55	91.16 ± 1.17	85.38 ± 0.61	88.86 ± 0.46	86.08 ± 0.67
EvolveGCN [34]	76.82 ± 1.46	92.47 ± 0.04	90.12 ± 0.69	83.88 ± 0.53	94.21 ± 0.66	87.50 ± 0.68
GRUGCN [40]	82.86 ± 0.53	94.96 ± 0.35	92.47 ± 0.36	84.60 ± 0.92	92.55 ± 0.41	89.48 ± 0.51
DySat [39]	81.02 ± 0.25	95.06 ± 0.21	93.06 ± 0.97	87.25 ± 1.70	91.92 ± 0.39	89.67 ± 0.70
VGRNN [15]	77.65 ± 0.99	95.17 ± 0.62	93.10 ± 0.57	85.95 ± 0.49	93.88 ± 0.07	89.15 ± 0.55
HTGN [55]	91.13 ± 0.14	**98.75** ± 0.03	<u>94.17</u> ± 0.17	<u>89.26</u> ± 0.17	94.80 ± 0.23	<u>93.62</u> ± 0.15
ROLAND-GT† [59]	81.40 ± 0.45	94.75 ± 0.87	90.20 ± 1.12	82.95 ± 0.45	94.88 ± 0.31	88.83 ± 0.64
SLATE	**93.21** ± 0.37	<u>97.46</u> ± 0.45	**96.39** ± 0.17	**90.84** ± 0.41	**97.69** ± 0.21	**95.12** ± 0.32

Comparison to discrete models, on DTDG. Table 1 We showcases the performance of SLATE against various discrete models on DTDG datasets, highlighting its superior performance across multiple metrics and datasets. SLATE outperforms all state of the art models on the HepPh, Enron, and Colab datasets, demonstrating superior dynamic link prediction capabilities. Notably, it surpasses HTGN by +2.1 points in AUC on HepPh and +1.1 points in AP on Enron. Moreover, SLATE shows a remarkable improvement of +7.6 points in AUC over EvolveGCN on Colab. It also performs competitively on the AS733 dataset, with scores that are closely second to HTGN, demonstrating its robustness across different types of dynamic graphs. What also emerges and validates our method from this comparison is the average gain of +6.29 points by our fully connected spatio-temporal attention model over the separate spatial attention model and temporal model approach, as used in ROLAND-GT. We also demonstrate significant gains against sparse attention models like DySat, with

an increase of +6.45. This study, conducted on the protocol from [55], emphasizes SLATE capability in handling discrete-time dynamic graph data, offering significant improvements over existing models.

Table 2: Comparison to CTDG models on discrete data using [60] protocol (AUC).

Method	CanParl	USLegis	Flights	Trade	UNVote	Contact	**Avg.**
JODIE [27]	78.21 ± 0.23	82.85 ± 1.07	96.21 ± 1.42	69.62 ± 0.44	68.53 ± 0.95	96.66 ± 0.89	82.01 ± 0.83
DyREP [43]	73.35 ± 3.67	82.28 ± 0.32	95.95 ± 0.62	67.44 ± 0.83	67.18 ± 1.04	96.48 ± 0.14	80.45 ± 1.10
TGAT [52]	75.69 ± 0.78	75.84 ± 1.99	94.13 ± 0.17	64.01 ± 0.12	52.83 ± 1.12	96.95 ± 0.08	76.58 ± 0.71
TGN [38]	76.99 ± 1.80	83.34 ± 0.43	98.22 ± 0.13	69.10 ± 1.67	69.71 ± 2.65	97.54 ± 0.35	82.48 ± 1.17
CAWN [48]	75.70 ± 3.27	77.16 ± 0.39	98.45 ± 0.01	68.54 ± 0.18	53.09 ± 0.22	89.99 ± 0.34	77.16 ± 0.74
EdgeBank [35]	64.14 ± 0.00	62.57 ± 0.00	90.23 ± 0.00	66.75 ± 0.00	62.97 ± 0.00	94.34 ± 0.00	73.50 ± 0.00
TCL [47]	72.46 ± 3.23	76.27 ± 0.63	91.21 ± 0.02	64.72 ± 0.05	51.88 ± 0.36	94.15 ± 0.09	75.11 ± 0.73
GraphMixer [7]	83.17 ± 0.53	76.96 ± 0.79	91.13 ± 0.01	65.52 ± 0.51	52.46 ± 0.27	93.94 ± 0.02	77.20 ± 0.36
DyGformer [60]	**97.76** ± 0.41	77.90 ± 0.58	98.93 ± 0.01	70.20 ± 1.44	57.12 ± 0.62	**98.53** ± 0.01	83.41 ± 0.51
SLATE	92.37 ± 0.51	**95.80** ± 0.11	**99.07** ± 0.41	**96.73** ± 0.29	**99.94** ± 0.05	98.12 ± 0.37	**96.88** ± 0.26

Comparison to continuous models, on DTDG. Table 2 In dynamic link prediction, SLATE outperforms models focused on node (TGN, DyRep, TGAT), edge (CAWN), and combined node-pairwise information (DyGFormer,TCL). Notably, it surpasses TCL by over 21 points in average, showcasing the benefits of our temporal cross attention strategies. SLATE's advantage stems from its global attention mechanism, unlike the sparse attention used by TGAT, TGN, and TCL. By employing fully-connected spatio-temporal attention, SLATE directly leverages temporal dimensions through its `Edge` module. This strategic approach allows SLATE to excel, as demonstrated by its consistent top performance and further evidenced in Appendix with hard negative sampling results (see Table 17 and Table 16 in Appendix E.1). We demonstrate average results that are superior by 13 points compared to the most recent model on DTDG, DyGFormer [60].

4.2 Model Analysis

Impact of different SLATE component. Table 3 presents the AUC results of different configurations of SLATE on four datasets. This evaluation demonstrates the impact of our proposed spatio-temporal encoding and the `Edge` module on dynamic link prediction performance.

Encoding	`Edge` Module	Enron	CanParl	USLegis	UNtrade
LapPE[11] + sinus-based [45]	✗	89.18 ± 0.33	82.98 ± 0.71	85.22 ± 0.24	90.24 ± 1.05
SLATE	✗	90.57 ± 0.27	89.45 ± 0.38	93.30 ± 0.29	94.01 ± 0.73
LapPE [11] + sinus-based [45]	✓	90.75 ± 0.08	90.23 ± 0.41	87.50 ± 0.50	90.56 ± 0.69
SLATE	✓	**96.39** ± 0.18	**92.37** ± 0.51	**95.80** ± 0.11	**96.73** ± 0.29

Table 3: Validation of different SLATE component. Results in AUC over 4 datasets.

First, we show the naive spatio-temporal encoding approach using the first k Laplacian eigenvectors associated with the k lowest values [11] (Appendix A.4), combined with sinusoidal unparametrized temporal encoding [45] (Appendix A.5), without the `Edge` module. The Laplacian is computed sequentially on the w snapshots, then concatenated with the temporal encoding indicating the position of the snapshot, with $k = 12$ for both SLATE and the naive encoding. The AUC scores across all datasets are significantly lower, highlighting the limitations of this naive encoding method in capturing complex spatio-temporal dependencies.

Replacing the baseline encoding with our proposed SLATE encoding, still without the `Edge` module, results in significant improvements: +6.47 points on CanParl, +8.08 points on USLegis, and +3.77 points on UNtrade. These improvements demonstrate the effectiveness of our spatio-temporal encoding. Adding the `Edge` module to the naive encoding baseline yields further improvements: +7.25 points on CanParl and +1.57 points on Enron. However, it still falls short compared to the enhancements provided by the SLATE encoding.

Finally, the complete model, SLATE with the `Edge` module, achieves the highest AUC scores across all datasets: +9.39 points on CanParl and +10.58 points on USLegis. These substantial gains confirm that integrating our unified spatio-temporal encoding and the `Edge` module effectively captures intricate dynamics between nodes over time, resulting in superior performance.

Dataset	SLATE w/o trsf	**SLATE**
Colab	85.03 ± 0.72	**90.84** ± 0.41
USLegis	63.35 ± 1.24	**95.80** ± 0.11
UNVote	78.30 ± 2.05	**99.94** ± 0.05
AS733	81.50 ± 1.35	**97.46** ± 0.45

Table 4: Importance of connectivity transformations steps to connect the supra-adjacency matrix. AUC performance in dynamic link prediction.

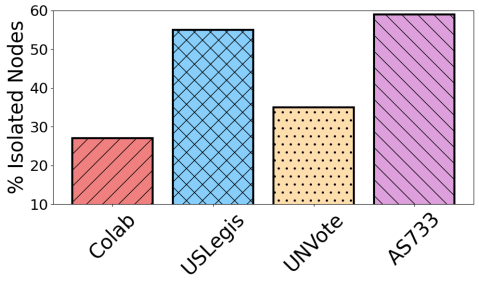

Figure 3: Average percentage of isolated nodes per snapshot on real world dynamic graphs data.

Critical role of supra-adjacency transformation. Here, we demonstrate the importance of the transformation steps of the supra-adjacency matrix, as detailed in section 3.1, by removing isolated nodes, adding virtual nodes, and incorporating temporal connections (Figure 2). Table 4 presents the performance of SLATE with and without transformation (trsf) on four datasets. Without these critical transformations, there is a systematic drop in performance, particularly pronounced in datasets with a high number of isolated nodes, as shown in Figure 3 (27% in Colab, 53% in USLegis, 35% in UNVote, and 59% in AS733). These results clearly highlight the significant improvements brought by our proposed transformations. More detailed experiments regarding each transformation, particularly on the importance of removing isolated nodes and adding a virtual node, are presented in Tables 12 and 13.

Models	Mem.	t / ep.	Nb params.
EvolveGCN	46Go	1828s	1.8 M
DySAT	42Go	1077s	1.8 M
VGRNN	21Go	931s	**0.4 M**
ROLAND-GT w/o Flash	OOM	-	1.9 M
ROLAND-GT	44Go	1152s	1.9 M
SLATE w/o Flash	OOM	-	2.1 M
SLATE	48Go	1354s	2.1 M
SLATE-Performer	**17Go**	**697s**	2.1 M

Table 5: An analysis of model efficiency comparing the memory usage (Mem.), training time per epoch (t/ep.) and the number of parameters (Nb params) on Flights dataset

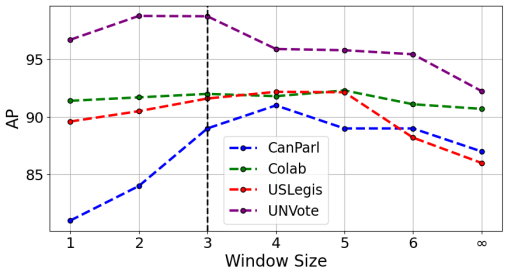

Figure 4: Model performance based on the window size, $w = \infty$ corresponds to considering all snapshots. Results in average precision (AP).

Impact of the time-window size. We demonstrate in Figure 4 the impact of the time window size on the performance of the SLATE model. A window size of 1 is equivalent to applying a global attention transformer to the latest snapshot before prediction, and an infinite window size is equivalent to considering all the snapshots for global attention. This figure highlights the importance of temporal context for accurate predictions within dynamic graphs. We observe that, in most cases, too much temporal context can introduce noise into the predictions. The USLegis, UNVote and CanParl datasets are political graphs spanning decades (72 years for UNVote), making it unnecessary to look too far back. For all of our main results in Table 2 and Table 1 we fix for simplicity $w = 3$. However, our ablations have identified $w = 4$ as an optimal balance, capturing sufficient temporal context without introducing noise into the transformer encoder and ensuring scalability for our model. Therefore, SLATE performances could further be improved by more systematic cross-validation of its hyper-parameters, *e.g.* w.

Model efficiency. The classic attention mechanism, with a complexity of $O(N^2)$, can be memory-consuming when applied across all nodes at different time steps. However, using Flash-Attention [9] and a light transformer architecture with just one encoder layer, we successfully scaled to the Flights dataset, containing 13,000 nodes and a window size of $w = 3$. By using the Performer encoder [5], which approximates attention computation with linear complexity, memory usage is reduced to 17GB. Our analysis shows that our model empirically matches the memory consumption of various

DTDG architectures while maintaining comparable computation times (Table 5). Furthermore, it is not over-parameterized relative to existing methods. We trained on an NVIDIA-Quadro RTX A6000 with 49 GB of total memory.

4.3 Qualitative results

We present qualitative results in Figure 5 comparing the graph and its spectrum before and after applying the proposed transformation in SLATE. The projection is made on the eigenvector associated with the first non-zero eigenvalue. Before transformation, the DTDG contains isolated nodes (7, 23 and 26) and two distinct clusters in the snapshot at $t = 3$. In this case, the projection is purely spatial, as there are no temporal connections, and some projections also occur on isolated nodes due to the presence of distinct connected components. After the proposed transformation into a connected multi-layer graph, the projection captures richer spatio-temporal properties of the dynamic graph. By connecting the clusters with a virtual node and adding temporal edges, our approach removes the influence of isolated nodes and enables the construction of an informative spatio-temporal encoding that better reflects the dynamic nature of the graph.

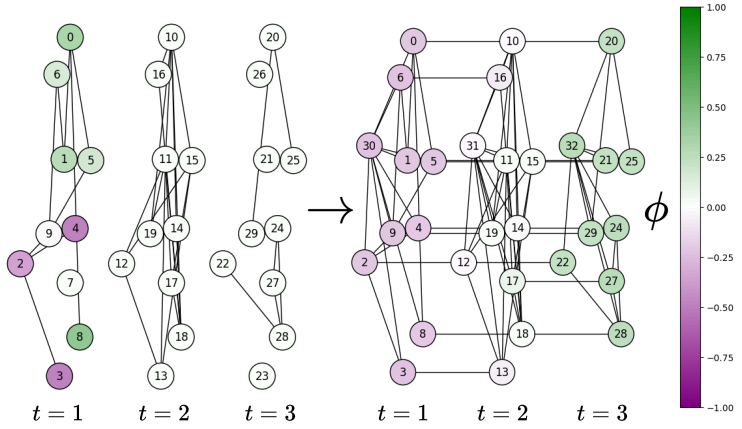

Figure 5: Projection of the eigenvector associated with the first non-zero eigenvalue on a toy DTDG before and after transformation. On the *left*, the DTDG is unprocessed, showing only spatial projections due to the lack of temporal connections. On the *right*, after applying the SLATE transformation, the graph captures rich spatio-temporal properties, allowing for a more informative spatio-temporal encoding

5 Conclusion

We have presented the SLATE method, an innovative spatio-temporal encoding for transformers on dynamic graphs, based on supra-Laplacian analysis. Considering discrete-time dynamic graphs as multi-layer networks, we devise an extremely efficient unified spatio-temporal encoding thanks to the spectral properties of the supra-adjacency matrix. We integrate this encoding into a fully-connected transformer. By modeling pairwise relationships in a new edge representation module, we show how it enhances link prediction on dynamic graphs. SLATE performs better than previous state-of-the-art approaches on various standard benchmark datasets, setting new state-of-the-art results for discrete link prediction.

Despite its strong performances, SLATE currently operates in a transductive setting and cannot generalize to unseen nodes. We aim to explore combinations with MP-GNNs to leverage the strengths of local feature aggregation and global contextual information. On the other hand, SLATE scales reasonably well to graphs up to a certain size but, as is often the case with transformers, future work is required to scale to very large graphs.

6 Acknowledgement

We acknowledge the financial support provided by PEPR Sharp (ANR-23-PEIA-0008, ANR, FRANCE 2030). We would also like to thank the LITIS laboratory in Rouen and especially Leshanshui Yang, who helped us better position our method. We also thank Elias Ramzi for his feedback on the paper and assistance in writing the abstract.

References

[1] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[2] Soumyanil Banerjee, Ming Dong, and Weisong Shi. Spatial-Temporal Synchronous Graph Transformer network (STSGT) for COVID-19 forecasting. *Smart Health*, 26, October 2022.

[3] Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks. In *International Conference on Machine Learning*, pages 748–758. PMLR, 2021.

[4] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In *Advances in neural information processing systems*, volume 33, pages 10383–10395, 2020.

[5] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2022.

[6] Peng Chu, Jiang Wang, Quanzeng You, Haibin Ling, and Zicheng Liu. Transmot: Spatial-temporal graph transformer for multiple object tracking. In *Proceedings of the IEEE/CVF Winter Conference on applications of computer vision*, pages 4870–4880, 2023.

[7] Weilin Cong, Si Zhang Meta, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou Meta, Hanghang Tong, and Mehrdad Mahdavi. GraphMixer: Do We Really Need Complicated Model Architectures For Temporal Networks? *arXiv preprint arXiv:2302.11636*, 2023.

[8] Emanuele Cozzo, Guilherme Ferraz de Arruda, Francisco A Rodrigues, and Yamir Moreno. Multilayer networks: metrics and spectral properties. *Interconnected networks*, pages 17–35, 2016.

[9] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359, 2022.

[10] Xiaowen Dong, Pascal Frossard, Pierre Vandergheynst, and Nikolai Nefedov. Clustering on multi-layer graphs via subspace analysis on Grassmann manifolds. *IEEE Transactions on Signal Processing*, 62(4):905–918, 2 2014.

[11] Vijay Prakash Dwivedi and Xavier Bresson. A Generalization of Transformer Networks to Graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[12] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[13] Paul Erdős and Alfréd Rényi. On random graphs I. *Publ. math. debrecen*, 6(290-297):18, 1959.

[14] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[15] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. Variational graph recurrent neural networks. *Advances in neural information processing systems*, 32, 2019.

[16] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[18] Shengxiang Hu, Guobing Zou, Shiyi Lin, Liangrui Wu, Chenyang Zhou, Bofeng Zhang, and Yixin Chen. Recurrent Transformer for Dynamic Graph Representation Learning with Edge Temporal States. *arXiv preprint arXiv:2304.10079v1*, 4 2023.

[19] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A.A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature 2021 596:7873*, 596(7873):583–589, 7 2021.

[20] Oumar Kaba and Siamak Ravanbakhsh. Equivariant networks for crystal structures. *Advances in Neural Information Processing Systems*, 35:4150–4164, 2022.

[21] Yannis Karmim, Leshanshui Yang, Raphaël Fournier S'Niehotta, Clément Chatelain, Sébastien Adam, and Nicolas Thome. Temporal receptive field in dynamic graph learning: A comprehensive analysis, 2024.

[22] Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *Advances in Neural Information Processing Systems*, 35:14582–14595, 2022.

[23] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907*, 9 2016.

[24] Thomas N. Kipf and Max Welling. Variational Graph Auto-Encoders. *arXiv preprint arXiv:1611.07308*, 11 2016.

[25] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. Multilayer networks. *Journal of complex networks*, 2(3):203–271, 2014.

[26] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking Graph Transformers with Spectral Attention. *Advances in Neural Information Processing Systems*, 26:21618–21629, 6 2021.

[27] Srijan Kumar, Xikun Zhang, and Jure Leskovec. JODIE: Predicting dynamic embedding trajectory in temporal interaction networks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1269–1278, 8 2019.

[28] Ai-Te Kuo, Haiquan Chen, Yu-Hsuan Kuo, and Wei-Shinn Ku. Dynamic Graph Representation Learning for Depression Screening with Transformer. *arXiv preprint arXiv:2305.06447v1*, 5 2023.

[29] Clement Lee and Darren J. Wilkinson. A review of stochastic block models and extensions for graph clustering. *Applied Network Science 2019 4:1*, 4(1):1–50, 12 2019.

[30] Jia Li, Jiao Su, Zhichao Han, Pengyun Wang, Lujia Pan, Hong Cheng, and Jianfeng Zhang. Predicting Path Failure In Time-Evolving Graphs. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1279–1289, 5 2019.

[31] Yixin Liu, Shirui Pan, Yu Guang Wang, Fei Xiong, Liang Wang, Qingfeng Chen, and Vincent CS Lee. Anomaly Detection in Dynamic Graphs via Transformer. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12081–12094, 2021.

[32] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. GraphiT: Encoding Graph Structure in Transformers. *arXiv preprint arXiv:2106.05667v1*, 6 2021.

[33] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[34] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5363–5370, 2020.

[35] Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. Towards better evaluation for dynamic link prediction. *Advances in Neural Information Processing Systems*, 35:32928–32941, 2022.

[36] Filippo Radicchi and Alex Arenas. Abrupt transition in the structural formation of interconnected networks. *Nature Physics*, 9(11):717–720, 7 2013.

[37] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[38] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. TGN: Temporal Graph Networks for Deep Learning on Dynamic Graphs. *arXiv preprint arXiv:2006.10637*, 6 2020.

[39] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In James Caverlee, Xia (Ben) Hu, Mounia Lalmas, and Wei Wang, editors, *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*, pages 519–527. ACM, 2020.

[40] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Gconv: Structured sequence modeling with graph convolutional recurrent networks. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25*, pages 362–373. Springer, 2018.

[41] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey. *IEEE Access*, 9:79143–79168, 2021.

[42] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[43] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. DyRep: Representation Learning over Dynamic Graphs. *ICLR 2019*, 3 2018.

[44] Eugenio Valdano, Luca Ferreri, Chiara Poletto, and Vittoria Colizza. Analytical computation of the epidemic threshold on temporal networks. *Physical Review X*, 5(2), 2015.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv preprint arXiv:1706.03762*, 6 2017.

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[47] Lu Wang, Xiaofu Chang, Shuang Li, Yunfei Chu, Hui Li, Wei Zhang, Xiaofeng He, Le Song, Jingren Zhou, and Hongxia Yang. TCL: Transformer-based Dynamic Graph Modelling via Contrastive Learning. *arXiv preprint arXiv:2105.07944*, 2021.

[48] Yanbang Wang, Yen Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *ICLR 2021 - 9th International Conference on Learning Representations*, 1 2021.

[49] Zehong Wang, Qi Li, Donghua Yu, and Xiaolong Han. Temporal graph transformer for dynamic network. In *International Conference on Artificial Neural Networks*, pages 694–705. Springer, 2022.

[50] Siqi Wei, Bin Wu, Aoxue Xiang, Yangfu Zhu, and Chenguang Song. DGTR: Dynamic graph transformer for rumor detection. *Frontiers in Research Metrics and Analytics*, 7:1055348, 1 2022.

[51] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Simplifying and empowering transformers for large-graph representations. *Advances in Neural Information Processing Systems*, 36, 2024.

[52] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. TGAT: Inductive Representation Learning on Temporal Graphs. *ICLR 20*, 2 2020.

[53] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[54] Leshanshui Yang, Clement Chatelain, and Sebastien Adam. Dynamic Graph Representation Learning With Neural Networks: A Survey. *IEEE Access*, 12:43460–43484, 2024.

[55] Menglin Yang, Min Zhou, Marcus Kalander, Zengfeng Huang, and Irwin King. Discrete-time Temporal Network Embedding via Implicit Hierarchical Learning in Hyperbolic Space. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 17:1975–1985, 7 2021.

[56] Yu Yang, Hongzhi Yin, Jiannong Cao, Tong Chen, Quoc Viet Hung Nguyen, Xiaofang Zhou, and Lei Chen. Time-aware Dynamic Graph Embedding for Asynchronous Structural Evolution. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):9656–9670, 7 2022.

[57] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie Yan Liu. Graphormer: Do Transformers Really Perform Bad for Graph Representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

[58] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[59] Jiaxuan You, Tianyu Du, and Jure Leskovec. ROLAND: Graph Learning Framework for Dynamic Graphs. *KDD*, 8 2022.

[60] Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems*, 36:67686–67700, 2023.

[61] Ge Zheng, Wei Koong Chai, Jiankang Zhang, and Vasilis Katos. VDGCNeT: A novel network-wide Virtual Dynamic Graph Convolution Neural network and Transformer-based traffic prediction model. *Knowledge-Based Systems*, 275:110676, 9 2023.

A Supra-Laplacian and other positional encoding

A.1 Spectral Theory on multi-layer networks

To leverage the benefits of fully-connected spatio-temporal attention across all nodes at multiple timestamps, we encode the spatio-temporal structure by considering a DTDG as a multi-layer graph. For a simple DTDG $\mathcal{G} = \{G_1, G_2, G_3\}$ with a fixed number of nodes, we define the square symmetric supra-adjacency matrix $\bar{A} \in \mathbb{R}^{N \times N}$ as follows:

$$\bar{A} = \begin{pmatrix} A_1 & I & 0 \\ I & A_2 & I \\ 0 & I & A_3 \end{pmatrix} \tag{7}$$

Then, we can utilize the rich spectral properties associated with its supra-Laplacian $\bar{L} = \bar{D} - \bar{A}$. Several studies have analyzed the spectrum of those multi-layer graphs [8, 10, 36]. Especially, [36] demonstrated that ϕ_1, the Fiedler vector, associated with the second smallest eigenvalue λ_1, known as the algebraic connectivity or Fiedler value, highlights structural changes between each layer. For a DTDG, this provides valuable information about the graph's dynamics over time. We verified this property experimentally by generating a DTDG containing 3 snapshots of a random Erdős-Rényi graph [13] with 10 nodes each and connecting them temporally according to Eq. (7) (see illustration on Figure 1). We then project all nodes of the DTDG onto different vectors associated with eigenvalues λ_i, with $\lambda_0 \leq \lambda_1 \leq ... \leq \lambda_{\text{max}}$. We observe that projecting onto ϕ_1 provides dynamic information, while projecting onto ϕ_i associated with larger eigenvalues λ_i reveals increasingly localized structures. These properties strongly motivate the use of spectral analysis of a multi-layer graph derived from a DTDG to achieve a unified spatio-temporal encoding.

A.2 Supra-graph construction

Algorithm 1: Computation of supra-laplacian spectrum

Input: $\mathcal{G}$, w,k, $t+1$
Output: ϕ, Λ
`adjacencies` $\leftarrow$ [ ]
for i *from* $max(0, t-w)$ *to* t **do**
  $A_i \leftarrow$ `GetAdjacency`(G_i)
  $A_i \leftarrow$ `RemoveIsolated`(A_i)
  $A_i \leftarrow$ `AddVirtualNode`(A_i)
  `adjacencies.Append`(A_i)
$\bar{\mathcal{A}} \leftarrow$ `BlockDiag`(`adjacencies`)// Eq. (7)
$\bar{\mathcal{A}} \leftarrow$ `AddTempConnection`$(\bar{\mathcal{A}})$
// Add temporal self-connection only if nodes aren't isolated
$\bar{L} = I - D^{-1/2}\bar{A}D^{-1/2}$
$\phi^T \Lambda \phi =$ `GetKFirstEigVectors`$(\bar{L}, k)$ // $O(k^2 N)$

In practice, when isolated nodes are removed, we obtain a mask of size N. This mask helps us identify which nodes are isolated at each time step and determines whether their positional encoding will be $\mathbf{0}^k$ or the projection on the basis of the k eigenvectors. The mask also guides us in adding temporal connections between a node and its past, as isolated nodes do not have temporal connections. In summary, the matrix $\bar{A}$ has a different size from $N \times W$ because we remove isolated nodes and add virtual nodes. The masks help us map the actual indices in $\mathcal{G}$ to the rows in $\bar{A}$.

In Figure 6, we illustrate the process of transforming a random DTDG into a connected multi-layer network. On the left, we see three independent snapshots, with several isolated nodes (6, 23, and 26) and multiple clusters in the snapshot at $t = 3$. The proposed transformation in SLATE ensures that the resulting multi-layer graph becomes fully connected by adding temporal connections, removing isolated nodes, and introducing a virtual node to bridge the different clusters within each snapshot.

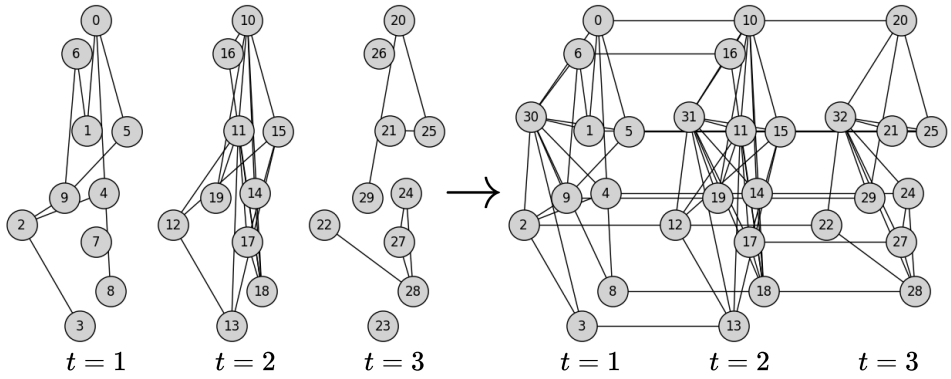

Figure 6: Transformation of a random DTDG into a connected multi-layer network. The *left* side shows independent snapshots with isolated nodes and disconnected clusters. The proposed transformation (*right*) ensures connectivity by introducing temporal edges, removing isolated nodes, and adding a virtual node to connect the clusters within each snapshot.

A.3 SupraLaplacian Positional Encoding

Proof of the positivity of λ_1 when the graph is connected [14]

Theorem 1: The second smallest eigenvalue, λ_1 (the Fiedler value), is strictly positive if and only if the graph is connected.

Proof 1: Assume that the graph is not connected. This implies that it can be divided into at least two disjoint connected components without any edges connecting them. For such a graph, it is possible to construct a vector whose entries correspond to these connected components such that the product $\phi_i^T \bar{L} \phi_i = 0$, where ϕ_i is an eigenvector. This demonstrates that $\lambda_1 = 0$. On the contrary, if $\lambda_1 > 0$, the only vector that satisfies $\phi_i^T \bar{L} \phi_i = 0$ under normal conditions (non-zero ϕ_i) is the constant vector, indicating that the graph cannot be divided without cutting edges, thus it is connected.

A.4 Laplacian Positional Encoding

$$L_t = I - D_t^{-1/2} A_t D_t^{-1/2} = \phi_t^T \Lambda \phi_t \tag{8}$$

$$\text{LapPE}_i^t = (\phi_{i,1}^t, \phi_{i,2}^t, ..., \phi_{i,d_{\text{pos}}}^t) \tag{9}$$

L_t represents the Laplacian matrix of the graph G_t. It is obtained by decomposing the graph as the product of eigenvectors ϕ_t and eigenvalues Λ_t. The Laplacian positional encoding defined in Eq. (9) provides a unique positional representation of the node $u_{i,t}$ with respect to the k eigenvectors of G_t.

A.5 Unparameterized temporal encoding

$$\text{timePE}(t, k) = \begin{cases} \sin\left(\frac{t}{10000^{(2k/d_{\text{time}})}}\right) & \text{if } k \text{ is even} \\ \cos\left(\frac{t}{10000^{((2k+1)/d_{\text{time}})}}\right) & \text{if } k \text{ is odd} \end{cases} \tag{10}$$

In Eq. (10), t refers to the t-th snapshot of our DTDG $\mathcal{G}$, and k is the dimension in our temporal encoding vector of size d_{time}. This temporal is from [45]. To build the ROLAND-GT separate spatio-temporal encoding we concatene the positional encoding LapPE (Eq. (9)) and the time encoding (Eq. (10)).

A.6 GCN Positional Encoding

We add in our comparison in Figure 7, the GCN positional encoding against our SLATE Model. This encoding is derived from a 2-layer GCN as designed by Kipf and Welling [23]. This method aggregates the local neighborhood information around a target node with message passing. We use the node embedding as positional encoding to enhance the transformer's awareness of the local structural context. This approach aims to integrate structural insights into the transformer model. It is inspired by the prevalent hybrid architectures combining MP-GNNs and transformers in static Graph Transformers [37]. It reflects an evolving trend in graph neural network research, where the strengths of both MP-GNNs in capturing local graph structures and transformers in modeling complex data dependencies are leveraged to enhance model performance on graph-based tasks. However, in our experiments, we found that SLATE significantly outperformed the GCN-based positional encoding.

B Baselines

Discrete Time Dynamic Graphs Link Prediction models We describe the DTDG models from [55]:

- **GIN and GAT** [24] : We use static models [53, 46] to showcase the necessity of dynamic model for efficient learning on dynamic graph. GAT is a sparse attention-based model, and GIN is a MP-GNN model design to have a the maximal expressivity of $1 - WL$.
- **GRUGCN** [40] : GRUGCN is one of the first discrete dynamic graph GNN models. They introduced the now standard approach which combines a GNN to process the snapshot, and updating embeddings using a temporal model, in their case a GRU.
- **EvolveGCN** [34]: EvolveGCN is an innovative approach adapting the Graph Convolutional Network (GCN) model for dynamically evolving graphs without relying on node embeddings, effectively capturing the dynamic nature of graph sequences through an RNN to update GCN parameters.
- **DySat** [39] : DySAT use self-attention mechanisms to learn node representations in dynamic graphs. It applies self-attention both structurally and temporally with separate module for time and space, the space module is similar to GAT [46], and the temporal model is a 1-D transformer.
- **VGRNN** [15] : VGRNN introduce node embedding techniques for dynamic graphs, focusing on variational graph recurrent neural networks to capture temporal dynamics. They employ latent variables for node representation, with SI-VGRNN advancing the model through semi-implicit variational inference for better flexibility. The method is suited for sparse graphs.
- **HTGN** [55]: They introduce a novel approach for embedding temporal networks through a hyperbolic temporal graph network (HTGN), effectively utilizing hyperbolic space to capture complex, evolving relationships and hierarchical structures in temporal networks.
- **ROLAND** [59]: ROLAND is a generic framework for graph representation learning on DTDG. They allow to efficiently implement any static graph models combine with a RNN-based temporal module.

Continuous Time Dynamic Graphs Link Prediction models We report the description of the CTDG baselines provided in [60].

- **JODIE** [27]: Tailored for user-item interaction dynamics within bipartite networks, JODIE utilizes dual recurrent neural networks to refresh user and item states, introducing a projection technique to predict future state trajectories.
- **DyRep** [43]: Introduces a recurrent mechanism for real-time node state updates, complemented by a temporal attention module to assimilate evolving structural insights of dynamic graphs effectively.
- **TGAT** [52]: Enhances node representations through the aggregation of temporal-topological neighbor features, leveraging a local self-attention mechanism and time encoding to discern temporal dynamics.

- **TGN** [38]: TGN dynamically updates node memories during interactions using a sophisticated mechanism comprising a message function, aggregator, and updater, thereby crafting temporal node representations through an embedding module.
- **CAWN** [48]: Initiates by extracting causal anonymous walks per node to delve into network dynamics and identity correlations, followed by encoding these walks with recurrent neural networks to synthesize comprehensive node representations.
- **EdgeBank** [35]: Adopts a non-parametric, memory-centric strategy for dynamic link prediction, maintaining a repository of interactions for memory updates and utilizing retention-based prediction to distinguish between positive and negative links.
- **TCL** [47] : TCL applies contrastive learning to dynamic graph, using a transformer-based architecture to capture temporal and topological information. It introduces a dual-stream encoder for processing temporal neighborhoods and employs attention mechanisms for semantic inter-dependencies, optimizing through mutual information maximization.
- **DyGFormer** [60]: DyGFormer introduces a novel transformer-based architecture for dynamic graph learning, focusing on first-hop interactions to derive node representations. It employs a neighbor co-occurrence encoding scheme to capture node correlations and a patching technique for efficient processing of long temporal sequences. This approach ensures model effectiveness in capturing temporal dependencies and node correlations.

C Datasets

C.1 Datasets description

Table 6: Dataset statistics used in our experiments, with a horizontal bar separating datasets from [60] and datasets from [55].

Datasets	Domains	Nodes	Links	Snapshots
CanParl	Politics	734	74,478	14
USLegis	Politics	225	60,396	12
Flights	Transports	13,169	1,927,145	122
Trade	Economics	255	507,497	32
UNVote	Politics	201	1,035,742	72
Contact	Proximity	692	2,426,279	8064
HepPh	Citations	15,330	976,097	36
AS733	Router	6,628	13,512	30
Enron	Mail	184	790	11
Colab	Citations	315	943	10
SBM	Synthetic	1000	4,870,863	50

- **CanParl**: Can. Parl. is a network that tracks how Canadian Members of Parliament (MPs) interacted between 2006 and 2019. Each dot represents an MP, and a line connects them if they both said "yes" to a bill. The line's thickness shows how often one MP supported another with "yes" votes in a year.
- **UsLegis**: USLegis is a Senate co-sponsorship network that records how lawmakers in the US Senate interact socially. The strength of each connection indicates how many times two senators have jointly supported a bill during a specific congressional session
- **Flights**: Flights is a dynamic flight network that illustrates the changes in air traffic throughout the COVID-19 pandemic. In this network, airports are represented as nodes, and the actual flights are represented as links. The weight of each link signifies the number of daily flights between two airports.
- **Trade**: UNTrade covers the trade in food and agriculture products between 181 nations over a span of more than 30 years. The weight assigned to each link within this dataset reflects the cumulative sum of normalized import or export values for agricultural goods exchanged between two specific countries.

- **UNVote**: UNVote documents roll-call votes conducted in the United Nations General Assembly. Whenever two nations cast a "yes" vote for an item, the link's weight connecting them is incremented by one.
- **Contact**: Contact dataset provides insights into the evolving physical proximity among approximately 700 university students over the course of a month. Each student is uniquely identified, and links between them indicate their close proximity. The weight assigned to each link reveals the degree of physical proximity between the students
- **Enron**: Enron consists of emails exchanged among 184 Enron employees. Nodes represent employees, and edges indicate email interactions between them. The dataset includes 10 snapshots and does not provide node or edge-specific information
- **Colab**: Colab represents an academic cooperation network, capturing the collaborative efforts of 315 researchers from 2000 to 2009. In this network, each node corresponds to an author, and an edge signifies a co-authorship relationship.
- **HepPH**: HepPh is a citation network focused on high-energy physics phenomenology, sourced from the e-print arXiv website. Within this dataset, each node represents a research paper, while edges symbolize one paper citing another. The dataset encompasses papers published between January 1993 and April 2003, spanning a total of 124 months.
- **AS733**: AS733 represents an Internet router network dataset, compiled from the University of Oregon Route Views Project. This dataset consists of 733 instances, covering the time period from November 8, 1997, to January 2, 2000, with intervals of 785 days between data points.
- **SBM**: SBM is a synthetic dynamic datasets generated with Stochastic Block Models methods. It contains 1000 nodes and 50 snapshots. We added this datasets, because unlike most of real world datasets, SBM is not a sparse graph.

C.2 Datasets split

For the datasets from [60], we follow the same graph splitting strategy, which means 70% of the snapshots for training, 15% for validation, and 15% for testing. We use the same number of snapshots as in HTGN [55], the value varies for each dataset (Table 7).

Datasets	HepPh	AS733	Enron	Colab
l (number snapshots in *test*)	6	10	3	3

Table 7: l represents the number of snapshots in the test dataset. The DTDG is split temporally, following [55]

D Implementation details and parameters search

For each of our experiments, we used a fixed embedding size of $d = 128$, a time window $w = 3$, and a single layer of transformer Encoder. Additionally, for the calculation of our positional encoding vectors, we consider that the graph is always undirected. In Table 8, we provide the remaining hyperparameters that we adjusted based on the datasets. We selected these datasets by choosing the hyperparameters that yielded the best validation performance in AP. `k` is the number of linearly independent eigenvectors we retrieve, it's important to note that d does not increase when `dim_pe` grow because $d' = d - \texttt{k}$. `nhead_xa` is the number of head inside the Edge Representation module define in section 3.3. `nhead_encoder` is the number of head inside SLATE section 3, `dim_ffn` is the dimension of the feed forward networks in SLATE and `norm_first` is a condition in SLATE to weither or not applying a layer norm before the full attention.

Parameters	Search Range
k	[4,6,10,12,14]
nhead_xa	[1,2,4,8]
nhead_encoder	[1,2,4,8]
dim_ffn	[128,512,1024]
norm_first	[True,False]
learning_rate	[0.1,0.01,0.001,0.0001]
weight_decay	[0,5e-7]

Table 8: Hyperparameter search range.

E Experiments: Additionnal results

E.1 AP results for DTDG models

We present additional results with Average Precision metrics to evaluate the dynamic link prediction capibility of models. **SLATE** outperforms all other DTDG models across various datasets, achieving the highest average precision (AP) scores. Specifically, SLATE surpasses the best-performing model, HTGN, with significant improvements: +1.22 on HepPh, +1.09 on Enron, and +3.33 on SBM. This highlights the effectiveness of our approach in dynamic link prediction tasks.

Table 9: Comparison to DTDG models on discrete data. AP

Method	HepPh	AS733	Enron	Colab	SBM	Avg
GCN	73.67 ± 1.05	97.11 ± 0.01	91.00 ± 0.73	90.17 ± 0.25	94.57 ± 0.30	89.30 ± 0.47
GIN	70.55 ± 0.84	93.43 ± 0.47	89.47 ± 1.52	87.82 ± 0.52	85.64 ± 0.11	85.38 ± 0.69
EvolveGCN	81.18 ± 0.89	95.28 ± 0.01	92.71 ± 0.34	87.53 ± 0.22	92.34 ± 0.17	89.81 ± 0.33
GRUGCN	85.87 ± 0.23	96.64 ± 0.22	93.38 ± 0.24	87.87 ± 0.58	91.73 ± 0.46	91.09 ± 0.35
DySat	84.47 ± 0.23	96.72 ± 0.12	93.06 ± 1.05	90.40 ± 1.47	90.73 ± 0.42	91.07 ± 0.66
VGRNN	80.95 ± 0.94	96.69 ± 0.31	93.29 ± 0.69	87.77 ± 0.79	90.53 ± 0.14	89.85 ± 0.57
HTGN	89.52 ± 0.28	**98.41** ± 0.03	94.31 ± 0.26	91.91 ± 0.07	94.71 ± 0.13	93.77 ± 0.15
ROLAND-GT	82.75 ± 0.31	93.66 ± 0.14	89.86 ± 0.29	85.03 ± 1.96	93.62 ± 0.28	88.98 ± 0.59
SLATE	**90.74** ± 0.51	98.16 ± 0.36	**95.40** ± 0.29	**92.15** ± 0.28	**98.04** ± 0.29	**94.90** ± 0.34

E.2 Comparison state of the art: Hard Negative Sampling

We present a extensive set of results for our method in comparison to CTDG models in the task of dynamic link prediction on discrete-time dynamic graphs in Table 17 and Table 16. Here, we emphasize the effectiveness of our model when employing hard historical negative sampling. Historical negative sampling technique (hist) was introduced in [35] to enhance the evaluation of a model's dynamic capability by selecting negatives that occurred in previous time-steps but are not present at the current time for prediction. Inductive negative sampling evaluating the capability of models to predict new links that never occured before. Our results demonstrate that our model excels at distinguishing hard negative edges compared to other CTDG models, as evidenced by improved performance in both AP and AUC metrics. SLATE also consistently outperforms other models using the indudctive (ind) sampling method across multiple datasets, showcasing its superior capability in capturing dynamic graph interactions. Notably, SLATE achieves significant improvements on datasets such as USLegis and Trade, demonstrating its robustness and effectiveness in dynamic link prediction tasks.

E.3 Model Analysis: Additional results

Figure 7 provides a comparison between SLATE spatio-temporal encoding and separate spatial and temporal encodings, including the Laplacian [12] (Lap Eq. (8)) and GCN (Appendix A.6) encodings. For calculating the spatial encoding, we selected two common strategies; the first involves using, as we do, the first k eigenvectors of the Laplacian [12], but only for the current snapshot. Empirically,

Table 10: Impact of Edge module on Dynamic Link Prediction task. ROC-AUC.

Datasets	SLATE w/o Edge	**SLATE**
CanParl	89.45 ± 0.38	**92.37** ± 0.51
USLegis	93.30 ± 0.29	**95.80** ± 0.11
Flights	**99.04** ± 0.61	**99.07** ± 0.41
Trade	94.01 ± 0.73	**96.73** ± 0.29
UNVote	93.56 ± 0.68	**99.94** ± 0.05
Contact	97.41 ± 0.10	**98.12** ± 0.37
HepPh	90.44 ± 1.07	**93.21** ± 0.37
AS733	96.84 ± 0.26	**97.46** ± 0.45
Enron	90.57 ± 0.27	**96.39** ± 0.18
COLAB	86.34 ± 0.34	**90.84** ± 0.41

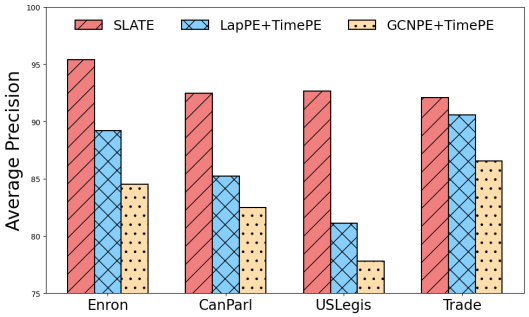

Figure 7: Comparison of SLATE encoding against separate structural/positional encoding and time encoding.

we found that the GCN encoding did not yield satisfying results, in contrast to the hybrid architecture strategies widely used for static Graph Transformers [37].

We show in Table 10 SLATE, with its cross-attention mechanism for edge representation, significantly enhances the predictive accuracy of the SLATE w/o Edge model. We show improvement across various datasets, further emphasizing the importance of modeling temporal interactions explicitly, we gain for example +6.4 points on UNVote, +2.6 points on USLegis and +5.8 points on Enron. SLATE's ability to capture intricate dynamics between two nodes across time dimensions results in substantial performance gains, making it a valuable addition to our model architecture.

Impact of the time-pooling function. In Table 11, we present the performance of the time-pooling function used in section 3.3, across the US Legis, UN Vote, and Trade datasets, with the time window set to $w = 3$. Using $k = 3$ corresponds to averaging over all snapshots within the window, whereas $k = 1$ focuses exclusively on the last element of E_{uv}. The results indicate that averaging (mean pooling) consistently outperforms max pooling, irrespective of the k value. For our primary analysis, we therefore adopt $k = 3$.

Table 11: Comparison of SLATE for several time pooling methods (random sampling), on USLegis, UNVote and Trade.

Pool	USLegis		UNvote		Trade	
	AUC	AP	AUC	AP	AUC	AP
Max	93.03	88.68	87.92	87.72	93.37	93.32
Avg. $k = 3$	94.50	89.67	**99.72**	**99.75**	96.71	96.88
Avg. $k = 2$	**95.35**	**92.17**	99.69	99.67	96.76	96.93
Avg. $k = 1$	94.67	91.28	99.59	99.44	**96.78**	**96.97**

Detailed analysis of the DTDG-to-multi-layer transformation in SLATE We provide a closer examination of the performance of SLATE under various transformations applied to the DTDG during its conversion into a multi-layer graph. The Table 12 demonstrates the negative effect of retaining isolated nodes, which leads to a significant drop in performance on both the Colab and USLegis datasets. By removing these nodes and focusing on the spectrum associated with the first non-zero eigenvalue, SLATE achieves a substantial performance improvement.

The Table 13 highlights the importance of introducing a virtual node (VN) that connects clusters within each snapshot. Without the VN, the model underperforms, as shown in the results for the Enron dataset. This confirms that each transformation step, from removing isolated nodes to adding temporal connections and VNs, plays a critical role in enhancing the quality of the spatio-temporal encoding.

AUC : Impact of time window on multiple models The analysis in Table 14 demonstrates that the impact of the time window on model performance is consistent across different types of models, including our transformer-based approach and two MP-GNNs (EGCN and DySAT). Interestingly, we observe that a relatively short time window produces optimal results for all models on the UNVote

Models	Colab	USLegis
SLATE with isolated nodes	86.73	66.57
SLATE	**90.84**	**95.80**

Table 12: Performance impact in AUC of keeping isolated nodes on Colab and USLegis datasets. Removing isolated nodes and focusing on the first non-zero eigenvalue spectrum leads to a significant performance boost.

Models	AP	AUC
SLATE w/o VN	93.74	95.18
SLATE	**95.40**	**96.39**

Table 13: Effect of introducing a virtual node (VN) on the Enron dataset. The addition of the VN improves SLATE's performance in terms of both AP and AUC.

dataset, which spans 72 snapshots. Specifically, both EGCN and DySAT achieve their highest AUC with $W = 4$, while SLATE achieves peak performance at $W = 2$. This indicates that capturing spatio-temporal dynamics does not necessarily require long temporal windows, and in fact, shorter windows can often lead to better performance by focusing on more immediate temporal interactions.

Model	Nb param.	$W = 1$	$W = 2$	$W = 3$	$W = 4$	$W = 5$	$W = \infty$
EGCN	1.8M	86.96	86.48	86.74	**87.66**	85.26	86.74
DySAT	1.8M	83.93	81.90	86.15	**88.71**	80.08	77.04
SLATE	2.1M	96.68	**99.73**	98.74	95.90	95.79	92.24

Table 14: Effect of time window size on AUC for different models. Shorter windows provide optimal results across all models.

E.4 More qualitative analysis

We conduct a fine-grained analysis of the impact of not processing the DTDG correctly. Figure 8 demonstrates that without temporal connections, the result is purely spatial projections with no spatio-temporal information, as the three snapshots remain independent. Figure 9 illustrates the effect of retaining isolated nodes while adding temporal connections. Keeping these nodes leads to multiple disconnected components in the graph, where many projections focus solely on the isolated nodes, neglecting the core structure of the DTDG. This issue is further intensified by the fact that we consider only the k eigenvectors associated with the first non-zero eigenvalue, limiting the ability to capture the full spatio-temporal dynamics.

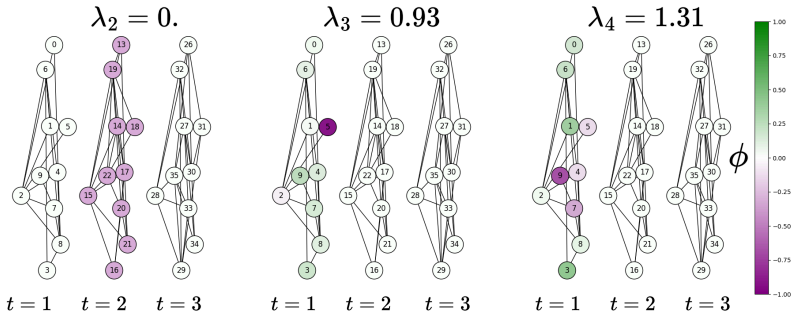

Figure 8: Effect of missing temporal connections in a DTDG. Without temporal edges, the figure illustrates that the projections are purely spatial, and the three snapshots remain independent, with no spatio-temporal interaction captured.

E.5 Scalability

In Table 5, we demonstrate that Performer [5] significantly reduces memory consumption and speeds up training time per epoch. Moreover, as shown in Table 15, using Performer an efficient

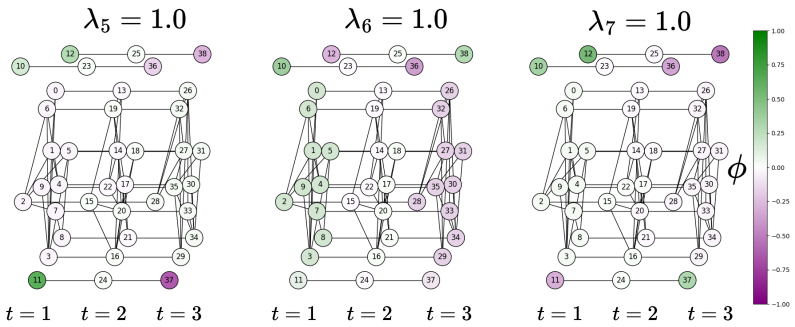

Figure 9: Effect of retaining isolated nodes in a DTDG with added temporal connections. The figure shows that keeping isolated nodes results in multiple disconnected components, where many projections focus on these nodes, obscuring the overall spatio-temporal structure of the graph.

approximation of the attention matrix with linear complexity—does not significantly degrade the results compared to the standard Transformer encoder. Performer is a highly advantageous solution for scaling to larger graphs while maintaining the benefits of dynamic graph transformers. Its linear complexity allows it to handle larger datasets efficiently, without sacrificing performance.

Models	AS733	USLegis	UNtrade
SLATE-Transformer	97.46 ± 0.45	95.80 ± 0.11	96.73 ± 0.29
SLATE-Performer	95.39 ± 0.61	95.14 ± 0.84	96.21 ± 0.77

Table 15: AUC performance comparison between SLATE using a standard Transformer encoder ([45]) and a Performer encoder ([5])

Table 16: Comparison to CTDG models on discrete data. ROC-AUC

NSS	Method	CanParl	USLegis	Flights	Trade	UNVote	Contact
rns	JODIE	78.21 ± 0.23	82.85 ± 1.07	96.21 ± 1.42	69.62 ± 0.44	68.53 ± 0.95	96.66 ± 0.89
	DyREP	73.35 ± 3.67	82.28 ± 0.32	95.95 ± 0.62	67.44 ± 0.83	67.18 ± 1.04	96.48 ± 0.14
	TGAT	75.69 ± 0.78	75.84 ± 1.99	94.13 ± 0.17	64.01 ± 0.12	52.83 ± 1.12	96.95 ± 0.08
	TGN	76.99 ± 1.80	83.34 ± 0.43	98.22 ± 0.13	69.10 ± 1.67	69.71 ± 2.65	97.54 ± 0.35
	CAWN	75.70 ± 3.27	77.16 ± 0.39	98.45 ± 0.01	68.54 ± 0.18	53.09 ± 0.22	89.99 ± 0.34
	EdgeBank	64.14 ± 0.00	62.57 ± 0.00	90.23 ± 0.00	66.75 ± 0.00	62.97 ± 0.00	94.34 ± 0.00
	TCL	72.46 ± 3.23	76.27 ± 0.63	91.21 ± 0.02	64.72 ± 0.05	51.88 ± 0.36	94.15 ± 0.09
	GraphMixer	83.17 ± 0.53	76.96 ± 0.79	91.13 ± 0.01	65.52 ± 0.51	52.46 ± 0.27	93.94 ± 0.02
	DyGformer	**97.76 ± 0.41**	77.90 ± 0.58	98.93 ± 0.01	70.20 ± 1.44	57.12 ± 0.62	**98.53 ± 0.01**
	SLATE	92.37 ± 0.51	**95.80 ± 0.11**	**99.07 ± 0.41**	**96.73 ± 0.29**	**99.94 ± 0.05**	98.12 ± 0.37
hist	JODIE	62.44 ± 1.11	67.47 ± 6.40	68.97 ± 1.87	68.92 ± 1.40	76.84 ± 1.01	96.35 ± 0.92
	DyREP	70.16 ± 1.70	91.44 ± 1.18	69.43 ± 0.90	64.36 ± 1.40	74.72 ± 1.43	96.00 ± 0.23
	TGAT	70.86 ± 0.94	73.47 ± 5.25	72.20 ± 0.16	60.37 ± 0.68	53.95 ± 3.15	95.39 ± 0.43
	TGN	73.23 ± 3.08	83.53 ± 4.53	68.39 ± 0.95	63.93 ± 5.41	73.40 ± 5.20	93.76 ± 1.29
	CAWN	72.06 ± 3.94	78.62 ± 7.46	66.11 ± 0.71	63.09 ± 0.74	51.27 ± 0.33	83.06 ± 0.32
	EdgeBank	63.04 ± 0.00	67.41 ± 0.00	74.64 ± 0.00	86.61 ± 0.00	89.62 ± 0.00	92.17 ± 0.00
	TCL	69.95 ± 3.70	83.97 ± 3.71	70.57 ± 0.18	61.43 ± 1.04	52.29 ± 2.39	93.34 ± 0.19
	GraphMixer	79.03 ± 1.01	85.17 ± 0.70	70.37 ± 0.23	63.20 ± 1.54	52.61 ± 1.44	93.14 ± 0.34
	DyGformer	**97.61 ± 0.40**	90.77 ± 1.96	68.09 ± 0.43	73.86 ± 1.13	64.27 ± 1.78	**97.17 ± 0.05**
	SLATE	88.71 ± 0.43	**90.69 ± 0.50**	**76.83 ± 0.69**	**92.14 ± 0.38**	**98.62 ± 0.49**	94.29 ± 0.09
ind	JODIE	52.88 ± 0.80	59.05 ± 5.52	69.99 ± 3.10	66.82 ± 1.27	73.73 ± 1.61	94.47 ± 1.08
	DyREP	63.53 ± 0.65	89.44 ± 0.71	71.13 ± 1.55	65.60 ± 1.28	72.80 ± 2.16	94.23 ± 0.18
	TGAT	72.47 ± 1.18	71.62 ± 5.42	73.47 ± 0.18	66.13 ± 0.78	53.04 ± 2.58	94.10 ± 0.41
	TGN	69.57 ± 2.81	78.12 ± 4.46	71.63 ± 1.72	66.37 ± 5.39	72.69 ± 3.72	91.64 ± 1.72
	CAWN	72.93 ± 1.78	76.45 ± 7.02	69.70 ± 0.75	71.73 ± 0.74	52.75 ± 0.90	87.68 ± 0.24
	EdgeBank	61.41 ± 0.00	68.66 ± 0.00	**81.10** ± 0.00	74.20 ± 0.00	72.85 ± 0.00	85.87 ± 0.00
	TCL	69.47 ± 2.12	82.54 ± 3.91	72.54 ± 0.19	67.80 ± 1.21	52.02 ± 1.64	91.23 ± 0.19
	GraphMixer	70.52 ± 0.94	84.22 ± 0.91	72.21 ± 0.21	66.53 ± 1.22	51.89 ± 0.74	90.96 ± 0.27
	DyGformer	**96.70** ± 0.59	87.96 ± 1.80	69.53 ± 1.17	62.56 ± 1.51	53.37 ± 1.26	**95.01** ± 0.15
	SLATE	93.74 ± 0.08	**90.23** ± 0.29	76.98 ± 1.64	**91.45** ± 0.39	**92.78** ± 0.06	94.03 ± 0.43

Table 17: Comparison to CTDG models on discrete data. Average Precision .

NSS	Method	CanParl	USLegis	Flights	Trade	UNVote	Contact
rns	JODIE	69.26 ± 0.31	75.05 ± 1.52	95.60 ± 1.73	64.94 ± 0.31	63.91 ± 0.81	95.31 ± 1.33
	DyREP	66.54 ± 2.76	75.34 ± 0.39	95.29 ± 0.72	63.21 ± 0.93	62.81 ± 0.80	95.98 ± 0.15
	TGAT	70.73 ± 0.72	68.52 ± 3.16	94.03 ± 0.18	61.47 ± 0.18	52.21 ± 0.98	96.28 ± 0.09
	TGN	70.88 ± 2.34	75.99 ± 0.58	97.95 ± 0.14	65.03 ± 1.37	65.72 ± 2.17	96.89 ± 0.56
	CAWN	69.82 ± 2.34	70.58 ± 0.48	98.51 ± 0.01	65.39 ± 0.12	52.84 ± 0.10	90.26 ± 0.28
	EdgeBank	64.55 ± 0.00	58.39 ± 0.00	89.35 ± 0.00	60.41 ± 0.00	58.49 ± 0.00	92.58 ± 0.00
	TCL	68.67 ± 2.67	69.59 ± 0.48	91.23 ± 0.02	62.21 ± 0.03	51.90 ± 0.30	92.44 ± 0.12
	GraphMixer	77.04 ± 0.46	70.74 ± 1.02	90.99 ± 0.05	62.61 ± 0.27	52.11 ± 0.16	91.92 ± 0.03
	DyGformer	**97.36 ± 0.45**	71.11 ± 0.59	**98.91 ± 0.01**	66.46 ± 1.29	55.55 ± 0.42	**98.29 ± 0.01**
	SLATE	92.44 ± 0.25	**92.66 ± 0.41**	98.61 ± 0.44	**96.91 ± 0.23**	**99.91 ± 0.09**	97.68 ± 0.13
hist	JODIE	51.79 ± 0.63	51.71 ± 5.76	66.48 ± 2.59	61.39 ± 1.83	70.02 ± 0.81	95.31 ± 2.13
	DyREP	63.31 ± 1.23	**86.88 ± 2.25**	67.61 ± 0.99	59.19 ± 1.07	69.30 ± 1.12	96.39 ± 0.20
	TGAT	67.13 ± 0.84	62.14 ± 6.60	72.38 ± 0.18	55.74 ± 0.91	52.96 ± 2.14	96.05 ± 0.52
	TGN	68.42 ± 3.07	74.00 ± 7.57	66.70 ± 1.64	58.44 ± 5.51	69.37 ± 3.93	93.05 ± 2.35
	CAWN	66.53 ± 2.77	68.82 ± 8.23	64.72 ± 0.97	55.71 ± 0.38	51.26 ± 0.04	84.16 ± 0.49
	EdgeBank	63.84 ± 0.00	63.22 ± 0.00	70.53 ± 0.00	81.32 ± 0.00	84.89 ± 0.00	88.81 ± 0.00
	TCL	65.93 ± 3.00	80.53 ± 3.95	70.68 ± 0.24	55.90 ± 1.17	52.30 ± 2.35	93.86 ± 0.21
	GraphMixer	74.34 ± 0.87	81.65 ± 1.02	71.47 ± 0.26	57.05 ± 1.22	51.20 ± 1.60	93.36 ± 0.41
	DyGformer	**97.00 ± 0.31**	85.30 ± 3.88	66.59 ± 0.49	64.41 ± 1.40	60.84 ± 1.58	**97.57 ± 0.06**
	SLATE	84.38 ± 0.81	83.53 ± 1.64	**75.09 ± 1.17**	**84.05 ± 0.98**	**96.85 ± 0.27**	93.58 ± 0.16
ind	JODIE	48.42 ± 0.66	50.27 ± 5.13	69.07 ± 4.02	60.42 ± 1.48	67.79 ± 1.46	93.43 ± 1.78
	DyREP	58.61 ± 0.86	83.44 ± 1.16	70.57 ± 1.82	60.19 ± 1.24	67.53 ± 1.98	94.18 ± 0.10
	TGAT	68.82 ± 1.21	61.91 ± 5.82	75.48 ± 0.26	60.61 ± 1.24	52.89 ± 1.61	94.35 ± 0.48
	TGN	65.34 ± 2.87	67.57 ± 6.47	71.09 ± 2.72	61.04 ± 6.01	67.63 ± 2.67	90.18 ± 3.28
	CAWN	67.75 ± 1.00	65.81 ± 8.52	69.18 ± 1.52	62.54 ± 0.67	52.19 ± 0.34	89.31 ± 0.27
	EdgeBank	62.16 ± 0.00	64.74 ± 0.00	**81.08** ± 0.00	72.97 ± 0.00	66.30 ± 0.00	85.20 ± 0.00
	TCL	65.85 ± 1.75	78.15 ± 3.34	74.62 ± 0.18	61.06 ± 1.74	50.62 ± 0.82	91.35 ± 0.21
	GraphMixer	69.48 ± 0.63	79.63 ± 0.84	74.87 ± 0.21	60.15 ± 1.29	51.60 ± 0.73	90.87 ± 0.35
	DyGformer	**95.44 ± 0.57**	81.25 ± 3.62	70.92 ± 1.78	55.79 ± 1.02	51.91 ± 0.84	**94.75** ± 0.28
	SLATE	93.42 ± 0.75	**95.21** ± 0.51	79.03 ± 0.95	**92.87** ± 0.62	**93.74** ± 0.29	94.52 ± 0.86

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our paper focuses on a unified spatio-temporal encoding based on the spectrum of the supra-Laplacian, as developed in section 3.1. We also introduce a fully-connected architecture utilizing this spatio-temporal encoding section 3.2 for the task of link prediction section 3.3. Each of these claims is validated in Table 3, as well as the claim of better SLATE performance against state-of-the-art methods in Tables 1 and 2.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of SLATE in the conclusion section 5, where we list multiple negative points of our work and suggest possible improvements, particularly in terms of better scalability and evaluating SLATE on other graph or node-based tasks.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We state in our section 3.1 that a connected graph has its second eigenvalue strictly positive. We include this proof and its source in Appendix Appendix A.3.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in Appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide a comprehensive overview of our model in Figure 2. Detailed discussions of our architecture can be found in sections 3.2 and 3.3. Also, algorithm of our supra-Laplacian computation is in Appendix A.3. Implementation specifics are outlined in the implementation details section of section 4 and further elaborated in Appendix D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code of SLATE is provided at this link: https://github.com/ykrmm/SLATE. Our code is designed to be comprehensible, ensuring that all presented results are reproducible.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The selection of datasets, their splitting, and descriptions are presented in Appendix C. We use the same evaluation protocols as papers well-recognized by the community [60, 55]. Hyperparameter optimization is detailed in Appendix D, and the optimizer settings are described at the beginning of section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Following the protocols we are based on, all results in the paper, including those from ablation studies, are averaged over 5 runs with the standard deviation reported.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The analysis of the time-memory efficiency of our model is presented in Table 5, where we also compare it with other state-of-the-art models. We also detailed the number of parameters of SLATE.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research adheres strictly to the NeurIPS Code of Ethics. Our study does not involve sensitive data or unethical practices, and we have followed all relevant guidelines to ensure ethical compliance throughout our work.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our model is a discriminative model for link prediction on academic datasets, which do not contain any private or sensitive information.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We don't release new data or harmful generative models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

Justification: We properly cite all the datasets and baselines used in our paper. Each dataset and model is credited to its original creators, and we adhere to the specified licenses and terms of use. The evaluation protocols we employ are based on established standards from previous works, ensuring compliance with the original authors' conditions.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Our paper introduces new assets, including code implementations and datasets for DTDG. Detailed documentation is provided alongside these assets, following structured templates that include information about training and licensing.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: We do not conduct research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: We do not conduct research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.