Virtual Nodes Go Temporal

Proceedings Track Submission

Anonymous Author(s)
Anonymous Affiliation
Anonymous Email

Abstract

Learning representations of temporally evolving graphs, also known as Continuous-Time Dynamic Graphs (CTDGs), has gained considerable attention due to their ability to model a wide range of real-world phenomena. Recent efforts extend the well-established message-passing paradigm and Graph Neural Network (GNN) models, originally designed for static graphs, to account for the temporal dimension of dynamic graphs. Although these methods have shown promising results, they often inherit limitations from their static counterparts, particularly regarding the capture of long-range interactions. In static settings, adding Virtual Nodes (VNs) has proven effective in overcoming locality constraints and boosting performance. In this work, we conduct a theoretical analysis of the impact of VNs in CTDG-based models. Specifically, we introduce the concept of information flow, which examines how information propagates through a graph following an event. From this perspective, we highlight inherent limitations of existing CTDG-based approaches and demonstrate how adding VNs can address these constraints. Building on these insights, we propose k-TVNs, a framework that incorporates a set of fully connected VNs, each representing a distinct community within the graph. Through both theoretical investigation and empirical validation, we show that incorporating VNs substantially improves the performance of CTDG models.

1 Introduction

Graph Neural Networks (GNNs) [1–3] have become the standard approach for learning robust graph representations that power a wide range of downstream tasks. From biomedical applications like protein function prediction [4, 5] to recommendation systems [6], GNNs have consistently shown remarkable effectiveness in capturing complex relational patterns. Following their success in static settings, these models have been extended to dynamic graphs [7], which generalize static graphs by evolving over time through events such as edge additions or deletions (friendship formation or removal in a social network) or node insertions or removals (users joining or leaving the network).

According to Kazemi et al. [8], dynamic graphs are generally categorized into: (i) Discrete-Time Dynamic Graphs (DTDGs), which aggregate events into fixed time intervals and (ii) Continuous-Time Dynamic Graphs (CTDGs), which accommodate temporally irregular events and provide a more flexible and widely adopted representation that subsumes the discrete case [9]. Unlike static graphs, these dynamic variants must account for evolving neighborhood structures and capture temporal dependencies. Consequently, specialized CTDG models have been developed to address these challenges. By integrating the temporal dimension, these models produce evolving node representations according to the continually shifting graph topology whenever an event occurs.

The family of CTDG models typically extends the message-passing framework [10] to accommodate temporal evolving graph topology. However, similar to static graph settings, these models often suffer from over-squashing and over-smoothing [11], which constrain their ability to propagate information effectively when new events occur. Consequently, nodes distant from an event location may fail to update their representations appropriately, posing challenges when capturing long-range dependencies, which is often a requirement in real-world dynamic graphs, where successive events can arise in

Submitted to the Fourth Learning on Graphs Conference (LoG 2025, Proceedings Track). Do not distribute.

different communities. One approach to mitigate this limitation is to enhance graph connectivity, either by rewiring (introducing connections between different sub-graphs) or by employing Graph Transformers [12], which leverage attention mechanisms to weight message-passing across all node pairs. Although Graph Transformers have shown promising results, their computational and memory overhead often make them infeasible for real-world dynamic settings, where updates must be performed after each event. In this work, we explore Virtual Nodes (VNs) as a more practical solution for propagating information while preserving local node structure. Specifically, we propose k-Temporal VNs (k-TVNs), which partitions the graph into k communities, each represented by a Virtual Node (VN). These VNs are fully connected with one another, thereby enabling global information flow across the entire graph in addition to the local connectivity within each community.

Although VNs have been studied in static graph settings [13, 14], particularly in relation to graph expressivity and over-smoothing, to the best of our knowledge, this is the first work to explore their impact in dynamic graph settings. Our goal is to explore both the theoretical and empirical benefits of incorporating VNs. To begin, we introduce the concept of "Information-Flow," which quantifies how events propagate through a graph and relates to the expressivity of a CTDG function. Building on this notion, we conduct a theoretical examination of CTDG expressivity and show how adding VNs can enhance information propagation. Motivated by these insights, we introduce a clustering-based approach for constructing VNs, facilitating the capture of community-level information. Finally, we validate our theoretical findings via empirical evaluation on real-world benchmark datasets, demonstrating that integrating VNs into CTDG functions yields performance gains compared to existing benchmark methods. Our main contributions can be summarized as follows:

- We formalize the concept of *information flow* to quantify how events propagate in a CTDG, establishing a clear connection to the expressivity of dynamic graphs.
- Through theoretical analysis, we demonstrate that incorporating VNs into a CTDG function improves its capacity for information propagation, thereby capturing long-range dependencies.
- We introduce a clustering-based strategy for constructing VNs, termed *k*-TVNs, and demonstrate empirically that our approach surpasses existing baselines in real-world benchmark datasets in both information propagation and downstream task performance.

2 Related Work

Numerous approaches have recently been proposed for learning representations of dynamic graphs. Broadly, existing methods can be classified into three categories:

Non-MP methods primarily rely on temporal embedding updates without explicit graph-based message passing. JODIE [15] and DeepCoevolve [16] employ mutually recursive recurrent neural networks to model node evolution, with JODIE introducing a *temporal projection* step to address inactivity. Temporal random walk methods [17] impose temporal constraints on sampled sequences before processing them with embedding models like Node2Vec [18]. Recent extensions enhance sampling strategies by incorporating graph topology [19], yet these methods remain constrained by their lack of explicit graph-level message passing, reducing their effectiveness in scenarios where topology-driven propagation is crucial, such as stochastic or multi-community settings.

MP-based methods facilitate information propagation through explicit neighborhood aggregation. DyRep [20] and DyGNN [21] incorporate recurrent memory components to update node representations based on temporal interactions; however, their dependence on one-hop neighborhoods can result in over-smoothing. Temporal Graph Attention (TGAT) [22] and Temporal Graph Networks (TGN) [23] extend this paradigm with attention-based aggregation and functional time encoding. More recent works explore spectral approaches, such as FreeDyG [24], which applies frequency-domain transformations for temporal dependency modeling. Finally, CTAN [25] refines information propagation by formulating event-driven diffusion as a dynamical system, whereas Temporal Graph Rewiring [26] improves long-range interaction efficiency by leveraging expander graph propagation.

Hybrid methods aim to strike a balance between computational efficiency and expressivity by integrating multiple modeling paradigms. GraphMixer [27] streamlines the MP framework by employing MLP-based encoders and neighborhood pooling, thereby lowering computational costs while preserving performance. SimpleDyG [28] conceptualizes dynamic graphs as sequences and uses a Transformer model to capture temporal patterns efficiently. PRES [29] enhances scalability by incorporating a prediction-correction mechanism to address temporal discontinuities.

3 Preliminaries

A *static graph* is defined as $G = (V, E)$, where V is the set of vertices and E is the edges. Let $n = |V|$ (rep. $m = |E|$) denote the number of vertices (resp. edges). A graph is often represented by its adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$[1], where the (i, j)-th entry denotes the weight of the edge between the i-th and j-th nodes, or 0 if no edge exists. On the other hand, a *Continuous-time dynamic graph (CTDG)*, evolves over time as nodes and edges are added or removed. In this perspective, the graph at time t can be represented as $G_t = (V_t, E_t)$, where V_t (resp. E_t) is the set of nodes (resp. edges) at time t. In accordance with the literature, we treat the node set V as constant over time and we focus on events consisting of adding edges. This assumption aligns with real-world scenarios where edge existence is often irreversible (e.g., a purchase event remains permanently in the system).

As in static graphs, the primary goal of temporal graph representation learning is to compute node embeddings that capture relevant information for various downstream tasks, such as predicting future events (link prediction) or classifying users (node classification). To incorporate temporal dynamics, recent adaptations of the message-passing framework have been proposed. For our theoretical analysis, we focus on the general TGN framework [23], which provides a unified perspective and encompasses various other message-passing models. Within this model, each node is characterized by two main components: (i) *memory*, which stores its historical information and tracks its evolution over time, and (ii) *current representation*, which is updated via the message-passing framework. When an event $\mathcal{E} = (u, v, e^t_{u,v})$ occurs at time t between nodes u and v, where $e^t_{u,v}$ represents the event's features, the memory state of node u is updated as follows:

$$s_u(t) = \textsc{MemUpd}([s_u(t^-), s_v(t^-), t - t^-, e^t_{u,v}]),$$

with t^- denoting the most recent time before t when the node's state was updated, and MEMUPD is an update function. This function can be implemented using a recurrent model such as Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), or a simpler MLP. Simultaneously, a message-passing framework aggregates information from a node's neighborhood to propagate updates based on the graph's topology. Specifically, for each node u, its temporal neighborhood $\mathcal{N}(u, t)$ at time t is defined to include all nodes that interacted with u at time t' within a specified time window, in addition to the corresponding event features, and can be formulated as: $\mathcal{N}(u, t) = \{(v, e^{t'}_{u,v}, t') \mid \exists (u, v, t') \in G_t\}$. A node's representation at the l-th message-passing layer is given by the following update:

$$\tilde{h}^{(\ell)}_u(t) = \textsc{Agg}^{(\ell)}(\{\!\{(h^{(\ell-1)}_v(t), t - t', e) \mid (v, e, t') \in \mathcal{N}(u, t)\}\!\}); h^{(\ell)}_u(t) = \textsc{Update}^{(\ell)}(h^{(\ell-1)}_u(t), \tilde{h}^{(\ell)}_u(t)),$$

where AGG is a permutation-invariant function that aggregates information within a node's neighborhood. This aggregation can be based on node degrees, as in Graph Convolutional Networks [1] or Graph Attention techniques [2], producing an aggregated vector that is subsequently passed to the UPDATE function. This latter function then computes the resulting updated representation for node v. In addition to modeling graph topology through message passing, a *time projection* mechanism is incorporated to capture the temporal dynamics of the evolving graph. For our theoretical analysis, we employ an attention-based approach similar to the one used in TGN [23], which can be formulated as:

$$t_v = (1 + \Delta t_v \cdot W_t) \circ s_v(t),$$

with W_t is a learnable weight matrix and Δt_v represents the time elapsed since the last interaction involving node v, allowing to capture temporal dependencies and dynamically adjust representations.

Theoretical Assumptions. Our study is based on the TGN model, assuming that the activation functions are 1-Lipschitz continuous – an assumption satisfied by commonly used functions such as ReLU, LeakyReLU, and Tanh [30]. Additionally, we focus on the effect of a single event update, in contrast to the setting where multiple events are processed in batches.

4 Information Flow in Continuous-Time Dynamic Graphs

In this section, we examine the theoretical impact of introducing a VN on a temporal graph function. We begin by formally defining the notion of expressivity in the context of CTDG functions, and subsequently leverage this framework to investigate how VNs can enhance a graph function's expressivity. While we focus on CTDGs, the insights extend to the Discrete-Time case due to the reconstruction relationship between these two categories (Proposition 1 in [9]). Throughout this paper $\|\cdot\|$ denotes the Euclidean (resp., spectral) norm for vectors (resp., matrices).

[1]We occasionally describe adjacency matrices at discrete times for clarity, but our model is event-driven CTDG. The snapshot-like description is only a shorthand for events before and after.

4.1 Expressivity through Information Flow

A widely used approach for analyzing the expressivity of graph-based functions is the Weisfeiler-Lehman (WL) test which was extended to dynamic graphs through the temporal-WL framework [9]. This approach primarily evaluates if a graph function can distinguish isomorphic graphs, a key requirement for graph-related downstream tasks. While valuable, this perspective has limitations, particularly when applied to graphs where nodes have features, that is common in the majority of dynamic graph datasets. In contrast, the current study approaches the expressivity subject differently by examining how information propagates through the graph when a new event occurs. Ideally, after an event, all node representations should be updated to reflect the modified structure. Otherwise, in an edge prediction task for instance, the event occurrence has not provided any additional information and any new perspective toward predicting the next link. Consequently, an ideal temporal graph function should update every node's embedding in response to a new edge, ensuring the graph's state is captured and the underlying distribution effectively modeled for downstream tasks.

Let $f : \mathcal{G} \to \mathcal{Y}$ be a graph-based function where $\mathcal{G}$ denotes the state of the graph over time and $\mathcal{Y}$ an output space. To quantify how much a node's embedding changes due to an event, we consider a distance function $d_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ within the output manifold $\mathcal{Y}$. For a node $u \in V$, we define the *information flow* as:

$$\mathcal{I}_u[f] = \mathbb{E}_{u \sim \mathcal{D}_V}[d_{\mathcal{Y}}(f_u(G^t), f_u(G^{t+1}))], \tag{1}$$

where G^t is graph at time t, $f_u(G^t)$ is the embedding of node u evaluated at that timestamp, and $\mathcal{D}_V$ is the graph node distribution. $\mathcal{I}_u[f]$ measures how much a node's embedding changes, based on a chosen distance metric, when the graph undergoes an event. This quantity, which is designed to focus on the local, node-level behavior, intuitively captures the extent of information propagation in the graph following an event. A larger change value indicates greater information spread, improving accuracy in downstream tasks. In this direction, we introduce the following definition:

Definition 1. *The graph-based function $f : \mathcal{G} \to \mathcal{Y}$, in respect to a node $u \in V$, is said to be (u, σ)–flowing if and only if: $\mathcal{I}_u[f] \leq \sigma$.*

Definition 1 formally defines how the information is expected to propagate following an event. Specifically, the parameter σ defines the allowable margin for embedding changes; a larger σ corresponds to greater "expected" variations and a possibility for increase in information flow at time $t+1$. While this approach provides a theoretical upper bound on this propagation, it does not directly guarantee the actual magnitude of updates in every case. Rather, it establishes an expected broader propagation, ensuring that the model is structurally capable of gathering distant updates when needed.

We note that the value of σ depends on the chosen distance metric $d_{\mathcal{Y}}$ within the output manifold. In our analysis, we focus on Norm-2 ($\|\cdot\|$), which we consider induces a suitable distance. Nonetheless, this choice does not restrict our theoretical conclusions, as norm equivalence holds in terms of information flow as theoretically proven in Lemma 2 (Appendix A). Consequently, different distance metrics can be used based on specific applications while yielding similar insights into a model's capacity for information propagation after an event. For example, in applications prioritizing outlier detection, Norm-1 may be more advantageous.

4.2 On the Expressive Power of Virtual Nodes

Following the previously introduced information flow concept for CTDGs in Section 4, we aim to analyze the effect of introducing VNs on a temporal graph function. Specifically, we focus on the TGN framework, which we consider an appropriate choice for encompassing the general message-passing CTDG function. Concretely, we consider a function f following the TGN framework, and we aim to analyze its expressivity in line with what has been introduced in Definition 1.

Theorem 1 (GCN-based aggregation)**.** *Let $f : \mathcal{G} \to \mathcal{Y}$ be a CTDG-based function based on L GCN-layers. After an event between nodes i and v, for any node u not involved in the event, we have:*

- *If $L < \min(d(u,i), d(u,v))$, then f is (u,σ)-flowing with $\sigma = \hat{w}_u \|W_t\| \prod_{l=1}^{L} \|W^{(l)}\|$.*
- *If $L \geq \min(d(u,i), d(u,v))$, then f is (u,σ)-flowing with*

$$\sigma = \prod_{l=1}^{L} \|W^{(l)}\| \left[\hat{w}_u \|W_t\| + \hat{w}_{u,i} \Delta_{t,t+1}(s_i)\right],$$

with W_t the linear temporal projection, $\hat{w}_u$ is the sum of temporal normalized walks of length $(L-1)$ starting from u and $\hat{w}_{u,i}$ is the normalized shortest path between u and i and $\Delta_{t,t+1}(s_i)$ denoting the difference in memory state for node i, as introduced by the event.

Theorem 1 examines how an event propagates through the graph and impacts the node embeddings. As expected from the message-passing mechanism of TGN, an event primarily affects nodes within the L-hop neighborhood of those directly involved. Nodes outside this neighborhood update only via temporal projection, which does not effectively capture the structural changes. Consequently, if subsequent events occur beyond this neighborhood, the node embeddings fail to incorporate them, limiting the model's effectiveness. A straightforward approach to address this limitation is to increase the number of layers to extend the update reach, however, this can lead to over-smoothing [31], a well-known issue in GNNs. Moreover, Theorem 1 shows that σ depends on the shortest path between the considered node and those involved in the event. While this dependence highlights a limitation in classical message-passing-based CTDG functions for long-range interactions, it also presents an opportunity: modifying the graph topology can influence information flow by strategically altering shortest paths. Similar conclusions are seen in the case of attention-based aggregation (Theorem 2). The proofs of the theorems are provided in Appendix B and Appendix C.

Theorem 2 (Attention-based aggregation)**.** *Let's consider a CTDG-based function $f : \mathcal{G} \to \mathcal{Y}$ based on L attention-based layers. After an event between node i and another node, the following properties hold for any node u not involved in the event, we have the following:*

- *If $L < \min(d(u,i), d(u,v))$, then f is (u,σ)-flowing with $\sigma = deg(u)\big[\|W_t\| + B\|W_t\|^2\big]$.*
- *If $L \geq \min(d(u,i), d(u,v))$, we have (u,σ)-flowing with*

$$\sigma = deg(u)\big[\|W_t\| + B\|W_t\|^2\big] + \Delta_{t,t+1}(s_i),$$

  where W_t denotes the linear temporal projection, $deg(u)$ denotes the degree of node u; and B is an upper-bound of latent representation space.

An effective strategy for improving information propagation in temporal graphs is to establish connections beyond immediate neighborhoods, enabling efficient diffusion across the entire graph. In static graphs, a well-studied approach is the introduction of a VN [14], which acts as a central hub, connecting multiple nodes to facilitate long-range message passing. While this technique has shown promise in static settings, applying it directly to dynamic graphs poses challenges due to their large scale and evolving nature. A single VN would need to process a vast amount of information, leading to both computational inefficiencies and an information bottleneck. To overcome this limitation, we propose a distributed VN framework, where multiple fully connected VNs represent distinct communities or clusters within the graph. Rather than relying on a single aggregation point, these VNs independently collect information from their associated "child" nodes and propagate it across the network via inter-VN communication.

We theorize that this hierarchical structure ensures stable information flow within the message-passing framework and enhances the model's ability to capture long-range dependencies in dynamic graphs.

$$\begin{pmatrix} \mathbf{A} & \begin{matrix} 1 & 0 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & & & \ddots & 1 \\ 0 & 0 & \dots & 0 & 1 \end{matrix} \\ \begin{matrix} 1 & 1 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & & & \ddots & 0 \\ 0 & 0 & \dots & 1 & 1 \end{matrix} & \mathbf{1} \end{pmatrix}$$

Problem Setup. We formally define the proposed multi-VN architecture. Consider a temporal graph $G^t \in \mathcal{G}$ at time t, with its adjacency matrix denoted as $A^t \in \mathbb{R}^{n \times n}$. To structure the graph, we partition its nodes into k distinct sets, denoted as $\mathcal{V} = \{\mathcal{V}_0, \dots, \mathcal{V}_k\}$. We introduce a *permutation* or *assignment function* $\Pi : \{1, \dots, n\} \to \{1, \dots, k\}$ that reorders the adjacency matrix based on the assigned node clusters. With the inclusion of VNs, the modified adjacency matrix A_{VN} can be formulated as follows:

By introducing VNs that aggregate local spatial information from each cluster and disseminate it globally, we argue that this approach enhances the underlying graph function's ability to capture both local and global dynamics while ensuring effective information flow. While prior work [13] has explored the role of VNs in static graphs, primarily in relation to under-reaching and over-smoothing, our focus is on their utility in facilitating information propagation following an event. To this end, we analyze their impact through our proposed information flow principle, which we consider more suited to the dynamic graph setting of interest.

Theorem 3 (VN Addition)**.** *Consider a CTDG-based function $f : \mathcal{G} \to \mathcal{Y}$ based on L GCN-like layers. Let g be a variant of f by adding k VNs, where each VN represents m nodes at max. After an event between nodes i and v, for any node u not involved in the event, we have:*

- *$\forall u \in \mathcal{V}$, if f is (u,σ)-flowing then g is (u,σ')-flowing with:*

$$\sigma' = \sigma + \frac{\Delta_{t,t+1}(s_i) + \Delta_{t,t+1}(s_v)}{k \times m}.$$

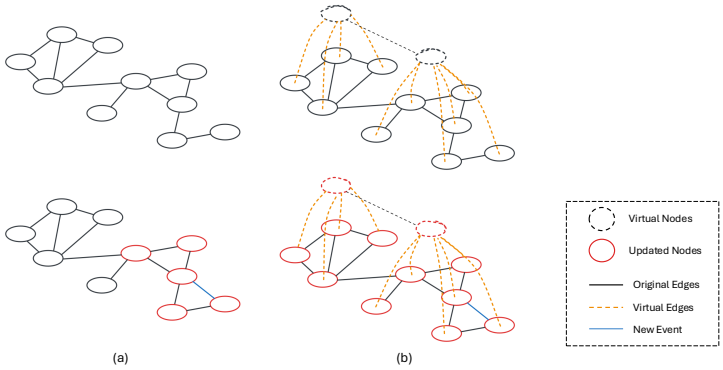

Figure 1: Illustration of the updated nodes after an event occurrence in the case of (a) the standard TGN with 1 Message-passing layer and (b) the TGN augmented with VNs. When incorporating VNs, the information is propagated to all the nodes and not only within the L-hop of the event.

Theorem 3 demonstrates how introducing a set of VNs enhances information flow across the graph. Unlike the original model (Theorem 1, 2), which relies on traditional message-passing where updates are restricted to nodes within a limited neighborhood, VNs introduce a structural mechanism that facilitates more global information flow. While the theoretical bound on σ' does not guarantee uniform updates across all nodes, it establishes a necessary condition for broader propagation. In practice, this increased bound provides greater flexibility for downstream tasks, particularly those requiring long-range dependencies, where events in one region of the graph may influence outcomes elsewhere. By bridging local communities through fully connected VNs, global updates become feasible without increasing the number of message-passing layers, thereby mitigating the risk of over-smoothing, as previously discussed. The proof of the theorem is provided in Appendix D.

5 Virtual Nodes meet Graph Clustering

The theoretical insights provided in Section 4 underscore the advantages of integrating VNs into a TGN model, which also holds for any CTDG-based function. VNs serve as intermediaries that facilitate long-range message passing and mitigate the challenges posed by limited local neighborhood aggregation. Building upon this concept, we propose a **multi-virtual-node framework**, where instead of a single VN connected to all nodes, we introduce multiple VNs, each responsible for a specific subset of the graph. This framework, denoted as k Temporal VNs (k-TVNs) ensures a more structured approach of propagating information across different regions of the graph, thus enhancing the expressivity of the underlying function.

As illustrated in Figure 1, at any considered time t, given a graph topology $A^{(t)}$ and a number of communities k, we define an assignment function $\Pi : \{1, \dots, n\} \to \{1, \dots, k\}$ that allocates each node to its corresponding VNs. The choice of this assignment function plays a crucial role in determining how effectively information is propagated. A naive strategy would be to randomly assign nodes to clusters; however, this could result in inefficient message passing, where redundant information circulates within communities without effectively reaching distant parts of the graph, which is our main aim. This can lead to unnecessary computational overhead and a loss of expressivity in node representations. To overcome this limitation, we propose leveraging graph clustering techniques to assign nodes to VNs in a topology-aware manner. Specifically, nodes that exhibit strong *structural connectivity* (e.g., those within the same densely connected community) are grouped under the same VN. This clustering-based assignment ensures that intra-community information is efficiently aggregated before being propagated to other clusters. This within-then-beyond community structure balances local expressivity with global connectivity.

As outlined in Algorithm 1, our method dynamically determines node assignments based on the graph's evolving topology at each time step t. Once the assignment is established, the *virtual connections* are generated, allowing nodes within each cluster to exchange information through their respective VN. This process facilitates *localized aggregation*, ensuring that each node benefits from

rich intra-cluster information before broader *inter-cluster propagation* extends insights across the entire graph. By structuring information flow in this hierarchical manner, our approach enhances long-range dependency modeling while avoiding the pitfalls of redundant aggregation, ultimately improving the robustness and efficiency of message passing in dynamic graphs.

Algorithm 1 k-Temporal Virtual Nodes (Main Algorithm)

Require: Adjacency matrix A and corresponding node features X at t, graph-based function f, number of clusters k, clustering function g

1: **for** $t \in \mathcal{T}$ **do**
2: Compute the assignment function $\Pi(.) = g(A^{(t)}, k)$.
3: Build the augmented adjacency matrix A_{VN}.
4: Compute the prediction $\hat{Y} = f(A_{VN}, X)$
5: Compute the Loss and update the model.
6: **return** Trained model f

Defining the appropriate clustering algorithm for this task is subject to two main constraints. The first constraint pertains to its *effectiveness*, as the algorithm must accurately identify and group the underlying communities within the graph. The second constraint is related to its computational *efficiency*. Given the dynamic nature of the graph, the clustering process must be executed multiple times, as cluster assignments must be periodically updated to reflect structural changes. Although this reassignment operation is performed in a batching-like manner, it must remain computationally feasible to prevent excessive overhead.

Lemma 1. *Let G be an undirected graph with adjacency matrix $A \in \mathbb{R}^{n\times n}$. Consider its rank-$k$ truncated singular value decomposition $A \approx U_k\Sigma_k U_k^\top$, with $\Sigma_k \in \mathbb{R}^{k\times k}$ a diagonal matrix containing k dominant singular values of A and $U_k \in \mathbb{R}^{n\times k}$ a semi-orthogonal matrix containing corresponding left singular vectors. Then, for any pair of nodes $u, v \in \mathcal{V}$, we have:*

$$\|A_{u,:} - A_{v,:}\| \approx \|((U_k)_{u,:} - (U_k)_{v,:})\Sigma_k\|,$$

where the error is bounded:

$$\big|\|A_{u,:} - A_{v,:}\| - \|((U_k)_{u,:} - (U_k)_{v,:})\Sigma_k\|\big| \leq 2\sigma_{k+1},$$

with σ_{k+1} being the $(k+1)$-th smallest singular value of A.

Lemma 1 implies that distances in the original adjacency space A are well approximated by distances in the lower-dimensional singular vector space U_k, scaled by the singular values. Thus, if U_k has a well-defined cluster structure (i.e., the rows of U_k can be clustered into well separated k clusters), the adjacency matrix inherits this property, reflecting the clustering tendencies of the graph's nodes. Moreover, when the graph has strongly defined communities, the rank k of A tends to be low, as most structural information is concentrated in a small number of dominant singular values. Leveraging those insights, we can directly use the adjacency matrix to construct communities. Specifically, building on algorithms previously introduced in the literature [32, 33], we introduce a variant of k-means that explicitly integrates graph topology by operating directly on the adjacency matrix.

The main idea, which is summarized in Algorithm 2 (Appendix F) consists of an iterative procedure of this adapted k-means method. At each iteration, we assign each node (represented by its adjacency-matrix row) to the nearest cluster center in Euclidean distance, then update each center as the mean of its assigned rows. Since each row in the adjacency matrix encodes a node's connectivity pattern, row-wise comparisons naturally capture both the number and identity of shared neighbors. These shared neighborhoods, in turn, drive cluster formation, effectively capturing the graph's underlying communities. By dynamically updating cluster centers based on row-wise distances, the algorithm groups nodes with similar adjacency patterns, i.e., those with strongly overlapping neighbor sets. As a result, the clustering yields a meaningful partition of the graph, where each cluster center represents a prototypical connectivity pattern of its node subset.

On the complexity of the approach. In its current formulation, the algorithm process all n rows and performs distance comparisons to K_0 centers in each iteration, resulting in an $\mathcal{O}(K_0 n^2)$ complexity. However, we note that for the dynamic graphs, and given their large scale, a batch-based procedure is usually adopted in which only a sub-graph (subset of nodes and their corresponding rows) is

Table 1: MRR performance of a TGN and k-TVNs on the TGB Benchmark with baseline results imported directly from the leaderboard. '–' refers to OOM [36].

Model	tgbl-wiki		tgbl-review		tgbl-coin		tgbl-comment		tgbl-flight	
	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test
DyRep [20]	7.2	5.0	21.6	22.0	51.2	45.2	29.1	28.9	57.3	55.6
EdgeBank$_{\text{tw}}$ [37]	60.0	57.1	2.4	2.5	49.2	58.0	12.4	14.9	36.3	38.7
EdgeBank$_{\infty}$ [37]	52.7	49.5	2.3	2.3	31.5	35.9	10.9	12.9	16.6	16.7
DyGFormer [38]	81.6	79.8	21.9	22.4	73.0	75.2	61.3	67.0	–	–
GraphMixer [27]	11.3	11.8	42.8	52.1	–	–	–	–	–	–
TGAT [22]	13.1	14.1	32.4	35.5	–	–	–	–	–	–
TNCN [39]	74.1	71.8	32.5	37.7	74.0	76.2	64.3	69.7	83.1	82.0
NAT [40]	77.3	74.9	30.2	34.1	–	–	–	–	–	–
CAWN [41]	74.3	71.1	20.0	19.3	–	–	–	–	–	–
TCL [42]	19.8	20.7	19.9	19.3	–	–	–	–	–	–
TGN [23]	43.5	39.6	31.3	34.9	60.7	58.6	35.6	37.9	73.1	70.5
k-TVNs (Ours)	61.3	57.2	33.5	37.3	67.1	67.9	42.8	44.3	75.8	73.6
Relative Improvement (%)	40.9	44.4	7.0	6.9	10.5	15.8	20.2	16.9	3.7	4.3

processed at each iteration. Consequently, in the context of a CTDG-based function, the scalability of the algorithm is ensured depending on the chosen batch size.

On the convergence of the method. Ensuring that our clustering procedure converges to a high-quality solution is crucial for preserving the representativeness of the augmented (virtual) nodes. The algorithm employed here is a variant of Lloyd's k-means [32], applied to the rows of the adjacency matrix (i.e., each node's neighborhood). During each iteration, the total within-cluster sum of squared distances either strictly decreases or remains unchanged. Since there are finitely many ways to partition n nodes into K_0 clusters, the algorithm necessarily terminates in a finite number of steps and settles at a (possibly local) optimum of the clustering objective. We do not provide further theoretical guarantees on its convergence behavior; however, prior work offers detailed insights on such guarantees [34] and on the role of initialization [35]. In practice, we observe that small number of iterations (e.g., 10) consistently delivers satisfactory performance for the downstream tasks.

We provide a complete analysis of a convergence and time complexity in Appendix F.2.

6 Empirical Validation

This section evaluates the practical impact of our theoretical findings on real-world datasets and the improved downstream performance. We specifically examine their effect on link prediction within temporal graph representation learning. We note that our objective is not to establish new state-of-the-art results but to demonstrate how integrating VNs as an augmentation mechanism enhances model performance. Specifically, following our theoretical analysis, we focus on TGN. We run the k-TVNs at each batch and aggregate information within each community using a weighted average based on node degrees. Full details on these choices are provided in Appendix G.2.

Experimental Setup. For our comparison, we have aimed to illustrate different families of methodologies as presented in Section 2. From a dataset perspective, we consider both a set of real-world benchmark datasets, extracted from the classification TGB benchmark Huang et al. [36] and synthetically generated dataset in which we control the graph's topology to showcase the necessity for long-range interactions. We set the number of clusters to 2 (an empirical analysis of this parameter is provided in Appendix F), and we run the clustering at each batch. Our code and implementation are provided in the supplementary materials. Details on the datasets, additional implementation regarding both model and VNs hyper-parameters are provided in Appendix G.

6.1 Experimental Results

Real-World datasets. We start by considering real-world benchmark datasets, where we consider the different temporal graphs constituting the TGB benchmark [36]. Table 1 presents the average test Mean Reciprocal Rank (MRR) scores. The results clearly demonstrate that augmenting the

Model	Long-Range Graph	
	PascalVOC 10	PascalVOC 30
JODIE	0.67 ± 0.03	0.65 ± 0.07
DyRep	0.69 ± 0.02	0.70 ± 0.02
TGAT	0.77 ± 0.02	0.76 ± 0.03
CTAN	**0.80 ± 0.01**	**0.78 ± 0.01**
TGN	0.78 ± 0.10	0.71 ± 0.04
k-TVNs	**0.80 ± 0.08**	0.76 ± 0.01

Table 2: Average test AUC (± denotes standard deviation) of the different methods on long-range PascalVOC 10 and 30 datasets. Best performance per dataset in **bold**.

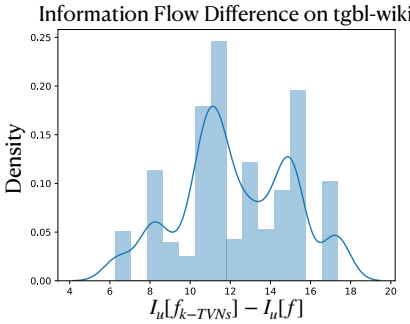

Figure 2: Difference in terms of information flow between a standard TGN model f and a TGN augmented with the k-TVNs framework $f_{k\text{-TVNs}}$ on `tgbl-wiki`.

TGN model with our proposed k-TVNs consistently improves performance. Notably, we observe a significant increase in MRR across all datasets. In particular, the performance improvement reaches up to 40% in one case. These findings highlight the effectiveness of k-TVNs in enhancing information propagation and improving predictive accuracy in temporal graph learning.

Long-Range Temporal Graphs. We additionally evaluate our approach on datasets that require long-range reasoning for effective performance. Specifically, we consider a temporal adaptation of the **PascalVOC-SP** graph [43], following the evaluation protocol proposed by Gravina et al. [25]. Table 2 presents the average and standard deviation of test AUC on this dataset. Consistent with our findings on real-world datasets, our proposed k-TVNs demonstrate the ability to improve model performance. In this scenario, where long-range dependencies play a crucial role and ensuring efficient information flow is essential, the addition of VNs becomes even more significant.

Empirical Validation of information flow. The primary motivation behind introducing VNs and our proposed k-TVNs method is to enhance information propagation within the graph following an event occurrence. In this perspective, we empirically validate the impact of VNs on node embeddings by comparing a standard TGN model, denoted as f, with its augmented version incorporating k-TVNs, denoted as $f_{\text{k-TVNs}}$ through the quantity considered in Equation 1. We provide an average of $\mathcal{I}_u$ for each model providing therefore a valid unbiased estimator. Figure 2 reports the results for TGBL-Wiki where we observe a an increased difference, indicating larger changes to node embeddings after an event. This empirical pattern is consistent with the theoretical insights derived in Theorem 3.

Clustering Approaches. As outlined in Section 5 and Algorithm 1, various clustering strategies can be employed to assign nodes to VNs. To assess their impact, we evaluate our used k-Means-Based Clustering against a number of alternatives. First, we consider Random assignment, where nodes are assigned to VNs randomly. We afterwards consider Louvain Algorithm where nodes are divided into clusters by maximizing intra-cluster connectivity while minimizing inter-cluster links [44]. Table 3 provides the resulting MRR for each method for the tgbl-wiki, where we see that our proposed Algorithm 2 provides the best performance.

	`tgbl-wiki`	
	Val	Test
Random	56.2	51.9
Louvain	56.3	52.3
k-TVNs	**61.3**	**57.2**

Table 3: Effect of the clustering on downstream performance on `tgbl-wiki`.

7 Conclusion

In this work, we investigate the limitations of message-passing-based frameworks for Continuous-Time Dynamic Graphs. Theoretically, we show that these models often fail to propagate information effectively when an event occurs, reducing their accuracy in modeling and predicting subsequent events. To address this issue, we propose augmenting temporal graphs with Virtual Nodes (VNs), a concept well-established for static graphs, to enhance connectivity without leading to over-smoothing. Specifically, we demonstrate that including these nodes improves the model's overall information flow. Building on this theoretical insight, we introduce k-TVNs, a method that uses a clustering algorithm to identify key communities that should be connected. Experimental evaluations on both real-world and long-range datasets empirically validate the effectiveness of k-TVNs.

References

[1] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*, 2017. 1, 3

[2] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *ICLR*, 2018. 3

[3] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *7th International Conference on Learning Representations*, 2019. 1

[4] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8):595–608, 2016. 1

[5] Aymen Qabel, Sofiane Ennadir, Giannis Nikolentzos, Johannes F. Lutzeyer, Michail Chatzianastasis, Henrik Boström, and Michalis Vazirgiannis. Structure-aware antibiotic resistance classification using graph neural networks. In *NeurIPS 2022 AI for Science: Progress and Promises*, 2022. URL https://openreview.net/forum?id=_BjtIlib8N9. 1

[6] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based Recommendation with Graph Neural Networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 346–353, 2019. 1

[7] Alessio Gravina and Davide Bacciu. Deep learning for dynamic graphs: models and benchmarks. *IEEE Transactions on Neural Networks and Learning Systems*, 2024. 1

[8] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020. 1

[9] Amauri Souza, Diego Mesquita, Samuel Kaski, and Vikas Garg. Provably expressive temporal graph networks. *Advances in neural information processing systems*, 35:32257–32269, 2022. 1, 3, 4

[10] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017. 1

[11] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=i80OPhOCVH2. 1

[12] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021. 2

[13] EunJeong Hwang, Veronika Thost, Shib Sankar Dasgupta, and Tengfei Ma. An analysis of virtual nodes in graph neural networks for link prediction (extended abstract). In *The First Learning on Graphs Conference*, 2022. URL https://openreview.net/forum?id=dI6KBKNRp7. 2, 5

[14] Joshua Southern, Francesco Di Giovanni, Michael Bronstein, and Johannes F Lutzeyer. Understanding virtual nodes: Oversmoothing, oversquashing, and node heterogeneity. *arXiv preprint arXiv:2405.13526*, 2024. 2, 5

[15] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19. ACM, July 2019. doi: 10.1145/3292500.3330895. URL http://dx.doi.org/10.1145/3292500.3330895. 2

[16] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. Deep coevolutionary network: Embedding user and item features for recommendation. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. 2

[17] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion proceedings of the the web conference 2018*, pages 969–976, 2018. 2

[18] Aditya Grover and Jure Leskovec. Node2Vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016. 2

[19] Ming Jin, Yuan-Fang Li, and Shirui Pan. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. *Advances in Neural Information Processing Systems*, 35:19874–19886, 2022. 2

[20] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyePrhR5KX. 2, 8

[21] Yao Ma, Ziyi Guo, Zhaocun Ren, Jiliang Tang, and Dawei Yin. Streaming graph neural networks. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 719–728, 2020. 2

[22] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJeW1yHYwH. 2, 8

[23] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. In *ICML 2020 Workshop on Graph Representation Learning*, 2020. 2, 3, 8

[24] Yuxing Tian, Yiyan Qi, and Fan Guo. Freedyg: Frequency enhanced continuous-time dynamic graph model for link prediction. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=82Mc5ilInM. 2

[25] Alessio Gravina, Giulio Lovisotto, Claudio Gallicchio, Davide Bacciu, and Claas Grohnfeldt. Long range propagation on continuous-time dynamic graphs. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 16206–16225. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/gravina24a.html. 2, 9, 21, 22

[26] Katarina Petrović, Shenyang Huang, Farimah Poursafaei, and Petar Veličković. Temporal graph rewiring with expander graphs. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*, 2024. URL https://openreview.net/forum?id=LN9fe4CWIH. 2

[27] Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ayPPc0SyLv1. 2, 8

[28] Yuxia Wu, Yuan Fang, and Lizi Liao. On the feasibility of simple transformer for dynamic graph modeling. In *Proceedings of the ACM on Web Conference 2024*, pages 870–880, 2024. 2

[29] Junwei Su, Difan Zou, and Chuan Wu. Pres: Toward scalable memory-based dynamic graph neural networks. *arXiv preprint arXiv:2402.04284*, 2024. 2

[30] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/d54e99a6c03704e95e6965532dec148b-Paper.pdf. 3

[31] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023. 5

[32] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28 (2):129–137, 1982. 7, 8

[33] Sébastien Bubeck, Marina Meilă, and Ulrike von Luxburg. How the initialization affects the stability of the k-means algorithm. *ESAIM: Probability and Statistics*, 16:436–452, 2012. 7

[34] Shokri Z. Selim and M. A. Ismail. K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(1):81–87, 1984. doi: 10.1109/TPAMI.1984.4767478. 8

[35] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 9780898716245. 8

[36] Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. Temporal graph benchmark for machine learning on temporal graphs. *Advances in Neural Information Processing Systems*, 36, 2024. 8

[37] Farimah Poursafaei, Andy Huang, Kellin Pelrine, and Reihaneh Rabbany. Towards better evaluation for dynamic link prediction. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=1GVpwr2Tfdg. 8

[38] Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xHNzWHbklj. 8

[39] Xiaohui Zhang, Yanbo Wang, Xiyuan Wang, and Muhan Zhang. Efficient neural common neighbor for temporal graph link prediction, 2024. URL https://arxiv.org/abs/2406.07926. 8

[40] Yuhong Luo and Pan Li. Neighborhood-aware scalable temporal network representation learning. In *Learning on Graphs Conference*, pages 1–1. PMLR, 2022. 8

[41] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=KYPz4YsCPj. 8

[42] 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. 8

[43] Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022. 9

[44] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008. 9

[45] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 22

Appendix: Virtual Nodes Go Temporal

A On the Equivalence of Norms

As explained in Section 4, we rather focus on the ℓ_2-norm in our analysis. In this section, we aim to prove that such choice doesn't limit the capacity of our analysis, and that similar insights can be seen when considering other norms.

We note, as explained previously, that the value of σ depends on the chosen distance metric $d_{\mathcal{Y}}$ within the output manifold.

Lemma 2. *Let $f : \mathcal{G} \to \mathcal{Y}$ be a CTDG graph-function, with D being the node embedding dimension. If f is (u, σ)-flowing in respect to Norm-2 then f is $(u, D^{\frac{2p}{2-p}}\sigma)$-flowing in respect to any Norm-p with $p \geq 1$.*

The previous Lemma shows that choosing Norm-2 to induce the output manifold distance does not restrict our theoretical conclusions, as norm equivalence holds in terms of information flow. Consequently, different distance metrics can be used based on specific applications while yielding similar insights into a model's capacity for information propagation after an event. In the rest of this section we provide the proof of the Lemma.

Lemma. *Let $f : \mathcal{G} \to \mathcal{Y}$ be a CTDG graph-function. If f is (u, σ)-flowing in respect to Norm-2 then f is $(u, D^{\frac{2p}{2-p}}\sigma)$-flowing in respect to any Norm-p with $p \geq 1$.*

Proof. Let's consider that $\mathcal{Y} \subseteq \mathbb{R}^D$ with D being the dimension of our graph node's embedding space. Let $x = (x_1, \ldots, x_D) \in \mathcal{Y}$. We recall that the set of norms p within $\mathbb{R}^D$ are written as:

$$\forall p > 0, \quad \|x\|_p = \left(\sum_{i=1}^{D} |x_i|^p \right)^{1/p}$$

Let $f : \mathcal{G} \to \mathcal{Y}$ be a CTDG graph-function. Let's consider that f is (u, σ)-flowing in respect to Norm-2:

$$\mathcal{I}_u[f] = \mathbb{E}_{\mathcal{D}}[\|f_u(G^{t+1}) - f_u(G^t)\|] \leq \sigma$$

Building on *Holder's inequality*, we have the following:

$$q > p \geq 1 \Rightarrow D^{\frac{1}{q}-\frac{1}{p}} \|x\|_p \leq \|x\|_q \leq D^{\frac{1}{q}} \|x\|_p,$$

Hence, for $q = 2$:

$$\begin{aligned} q > p \geq 1 &\Rightarrow D^{\frac{1}{2}-\frac{1}{p}} \|x\|_p \leq \|x\|_2 \\ &\Rightarrow D^{\frac{p-2}{2p}} \|x\|_p \leq \|x\|_2 \\ &\Rightarrow \|x\|_p \leq D^{\frac{2p}{2-p}} \|x\|_2 \end{aligned}$$

Consequently, we have the following:

$$\begin{aligned} \mathbb{E}_{\mathcal{D}}[\|f_u(G^{t+1}) - f_u(G^t)\|_p] &\leq \mathbb{E}_{\mathcal{D}}[D^{\frac{2p}{2-p}} \|f_u(G^{t+1}) - f_u(G^t)\|] \\ &\leq D^{\frac{2p}{2-p}} \sigma \end{aligned}$$

Therefore, f is $(u, D^{\frac{2p}{2-p}}\sigma)$-flowing in respect to Norm-p with $p \geq 1$

□

B Proof of Theorem 1

Theorem (GCN-based aggregation)**.** *Let's consider a CTDG-based function $f : \mathcal{G} \to \mathcal{Y}$ based on L GCN-like layers. After an event between nodes i and v, for any node u not involved in the event, we have the following:*

- *if $L < \min(d(u,i), d(u,v))$, then f is (u,σ)-flowing with:*

$$\sigma = \hat{w}_u \|W_t\| \prod_{l=1}^{L} \|W^{(l)}\|$$

- *if $L \geq \min(d(u,i), d(u,v))$, then f is (u,σ)-flowing with:*

$$\sigma = \prod_{l=1}^{L} \|W^{(l)}\| \left[\hat{w}_u \|W_t\| + \hat{w}_{u,i} \Delta_{t,t+1}(s_i)\right],$$

where $\hat{w}_u$ is the sum of temporal normalized walks of length $(L-1)$ starting from u; the term $\hat{w}_{u,i}$ denotes the normalized shortest path between u and i; the terms W_t and $W^{(l)}$ are weight matrices involved in the propagation, with W_t capturing temporal updates; and $\Delta_{t,t+1}(s_i)$ is the difference in memory state for node i, as introduced by the event.

Proof. Let $f : \mathcal{G} \to \mathcal{Y}$ be a CTDG graph-function, based on the framework described in Section 3. We consider the aggregation to follow a GCN-like aggregation, which can be formulated at layer ℓ for a node as:

$$h_u^{(\ell)} = \sigma^{(\ell)} \left(\sum_{v \in \mathcal{N}(u) \bigcup \{u\}} \frac{W^{(\ell)} h_v^{(\ell-1)}}{\sqrt{(1+\deg(u))(1+\deg(v))}} \right) \tag{2}$$

with $W^{(\ell)} \in \mathbb{R}^{e_{\ell-1} \times e_\ell}$ being the learnable weight matrix and e_ℓ being the embedding dimension of layer ℓ and $\sigma^{(\ell)}$ is the activation function of ℓ-th layer.

For nodes not involved in the event and for which $L < \min(d(u,i), d(u,v))$. We consider the nodes that are not part of the event and that are distant from the nodes involved in the event. Distant in this case is defined by the number of propagation layers used in the message-passing. For these nodes, the update is only dependent on the temporal projection, since the nodes are not updated based on the event.

In this proof, we consider that we are dealing with a GCN-like propagation:

$$\begin{aligned}
&\|f_u(G^{t+1}) - f_u(G^t)\| = \|h_{u,t+1}^{(l)} - h_{u,t}^{(l)}\| \\
&= \|\sigma^{(l)} \left(\sum_{v \in \mathcal{N}(u) \bigcup \{u\}} \frac{W^{(\ell)} h_{v,t+1}^{(\ell-1)}}{\sqrt{(1+\deg(u))(1+\deg(v))}} \right) - \\
&\qquad \sigma^{(l)} \left(\sum_{v \in \mathcal{N}(u) \bigcup \{u\}} \frac{W^{(\ell)} h_{v,t}^{(\ell-1)}}{\sqrt{(1+\deg(u))(1+\deg(v))}} \right) \| \\
&\leq \| \sum_{v \in \mathcal{N}(u) \bigcup \{u\}} \frac{W^{(\ell)} h_{v,t+1}^{(\ell-1)}}{\sqrt{(1+\deg(u))(1+\deg(v))}} - \\
&\qquad \sum_{v \in \mathcal{N}(u) \bigcup \{u\}} \frac{W^{(\ell)} h_{v,t}^{(\ell-1)}}{\sqrt{(1+\deg(u))(1+\deg(v))}} \| \\
&= \|W^{(\ell)}\| \sum_{v \in \mathcal{N}(u) \bigcup \{u\}} \frac{\|h_{v,t+1}^{(\ell-1)} - h_{v,t}^{(\ell-1)}\|}{\sqrt{(1+\deg(u))(1+\deg(v))}}
\end{aligned}$$

Using the same process in an iterative way, we get the following:

$$\begin{aligned}
&\|f_u(G^{t+1}) - f_u(G^t)\| = \|h^{(l)}_{u,t+1} - h^{(l)}_{u,t}\| \\
&\leq \prod_{l=1}^{L} \|W^{(l)}\| \sum_{z \in \mathcal{N}(y) \bigcup \{y\}} \frac{\|W_t\|}{\sqrt{(1+\deg(\mathrm{u}))} \dots \sqrt{(1+\deg(\mathrm{z}))}} \\
&\leq \hat{w}_u \|W_t\| \prod_{l=1}^{L} \|W^{(l)}\|
\end{aligned}$$

with $\hat{w}_u$ being the sum of temporal normalized walks of length $(L-1)$ starting from node u.

For nodes not involved in the event and for which $L \geq \min(d(u,i), d(u,v))$. We now consider the nodes that are not included in the event but are within a reachable distance of the event. Specifically, we consider the nodes that are $L \geq d(u,i)$. Similar to the previous part, we have the following inequality:

$$\begin{aligned}
&\|f_u(G^{t+1}) - f_u(G^t)\| = \|h^{(l)}_{u,t+1} - h^{(l)}_{u,t}\| \\
&\leq \prod_{l=1}^{L} \|W^{(l)}\| \sum_{z \in \mathcal{N}(y) \bigcup \{y\}} \frac{\|h^{(0)}_{v,t+1} - h^{(0)}_{v,t}\|}{\sqrt{(1+\deg(\mathrm{u}))} \dots \sqrt{(1+\deg(\mathrm{z}))}}
\end{aligned}$$

Since we have that $L \geq d(u,i)$, the consider node u have some information from the node related to the event i, in this perspective, we can write:

$$\begin{aligned}
&\|f_u(G^{t+1}) - f_u(G^t)\| \\
&\leq \prod_{l=1}^{L} \|W^{(l)}\| \sum_{z \in \mathcal{N}(y) \bigcup \{y\}} \frac{\|h^{(0)}_{v,t+1} - h^{(0)}_{v,t}\|}{\sqrt{(1+\deg(\mathrm{u}))} \dots \sqrt{(1+\deg(\mathrm{z}))}} \\
&\leq \prod_{l=1}^{L} \|W^{(l)}\| \Bigg[\sum_{z \in \mathcal{N}(y) \bigcup \{y\} \setminus \{i\}} \frac{\|h^{(0)}_{v,t+1} - h^{(0)}_{v,t}\|}{\sqrt{(1+\deg(\mathrm{u}))} \dots \sqrt{(1+\deg(\mathrm{z}))}} \\
&+ \frac{\|h^{(0)}_{i,t+1} - h^{(0)}_{i,t}\|}{\sqrt{(1+\deg(\mathrm{u}))}(1+\deg(\mathrm{j})) \dots (1+\deg(\mathrm{y}))\sqrt{(1+\deg(\mathrm{i}))}} \Bigg]
\end{aligned}$$

The first term is related to the temporal projection update and the second is related to the event occurrence. Specifically, after the event happening, node i's representation is updated and passed along with the message-passing. Building on this, we have:

$$\begin{aligned}
&\|f(G^{t+1}) - f(G^t)\| \\
&\leq \prod_{l=1}^{L} \|W^{(l)}\| \big[\hat{w}_u \|W_t\| + \hat{w}_{u,i} \underset{t,t+1}{\Delta}(s_i)\big]
\end{aligned}$$

where like previously explained $\hat{w}_u$ being the sum of temporal normalized walks of length $(L-1)$ starting from node u and $\hat{w}_{u,i}$ is the normalized shortest path between u and i.

□

C Proof of Theorem 2

Theorem (Attention-based aggregation)**.** *Let's consider a CTDG-based function $f : \mathcal{G} \to \mathcal{Y}$ based on L attention-based layers. After an event between node i and another node, the following properties hold for any node u not involved in the event, we have the following:*

- *if $L < \min(d(u,i), d(u,v))$, then f is (u,σ)-flowing with:*

$$\sigma = deg(u)\left[\|W_t\| + B\|W_t\|^2\right];$$

- *if $L \geq \min(d(u,i), d(u,v))$, we have (u,σ)-flowing with:*

$$\sigma = deg(u)\left[\|W_t\| + B\|W_t\|^2\right] + \Delta_{t,t+1}(s_i),$$

  where $deg(u)$ denotes the degree of node u; and B is an upper-bound of latent representation space.

Proof. In this theorem, and similar to the previous section in which we discuss Theorem 1, in this theorem we consider the function $f : \mathcal{G} \to \mathcal{Y}$ to be based on attention-like aggregation, where the aggregation of the representation of the neighborhood is weighted based on attention and can be written as:

$$h_u^{(t+1)} = \sigma^{(\ell)}\left(\sum_{k \in \mathcal{N}(u) \bigcup \{u\}} \alpha_k^{(t)} h_k^{(t)}\right) \tag{3}$$

Let's consider a node u at time t, we have the following:

$$\begin{aligned}\|h_u^{(t+1)} - h_u^{(t)}\| &= \|\sum_{k \in \mathcal{N}_u(t+1)} \alpha_k^{(t+1)} h_k^{(t+1)} - \sum_{k \in \mathcal{N}_u(t)} \alpha_k^{(t)} h_k^{(t)}\| \\ &\leq \sum_{k \in \mathcal{N}_u(t+1)} \|\alpha_k^{(t+1)}[h_k^{(t+1)} - h_k^{(t)}] + h_k^{(t)}[\alpha_k^{(t+1)} - \alpha_k^{(t)}]\|\end{aligned}$$

The previous quantity is divided into a first part related to the attention mechanism and the second is related to the neighborhood update. We start by analyzing the attention part, we have:

$$\alpha_{ij}^{(t)} = \frac{\exp\left(e_{ij}^{(t)}\right)}{\sum_{k \in N(i)} \exp\left(e_{ik}^{(t)}\right)}, \tag{4}$$

with:

$$e_{ij}^{(t)} = \text{LeakyReLU}\left(w^\top [s_i^{(t)}, s_j^{(t)}]\right), \tag{5}$$

where $[a, b]$ denote the concatenation of two vectors a and b. We can therefore derive the following:

$$\begin{aligned}\|\alpha_k^{(t+1)} - \alpha_k^{(t)}\| &\leq \|w\|([s_i^{(t+1)}, s_i^{(t)}] - [s_u^{(t+1)}, s_u^{(t)}]) &(6)\\ &\leq \|w\|\delta(s_u)\delta(s_i) &(7)\end{aligned}$$

For nodes not involved in the event and for which $L < \min(d(u,i), d(u,v))$. Similar to the previous proof, we first consider the node that are not within reach of the events $L < \min(d(u,i), d(u,v))$. Specifically, we consider that the number of propagation $L < \min(d(u,i), d(u,v))$ with u being the node and i being the node included in the event and d. The difference in terms of representation is related only to the temporal projection aspect:

$$\|\alpha_k^{(t+1)} - \alpha_k^{(t)}\| \leq \|w\|\|W_t\|^2 {\Delta_t}^2$$

We additionally assume that the representation space is bounded; specifically for each node $u \in \mathcal{N}$ at any timestamp t, there exists a finite constant $B_u^{(t)} \geq 0$ such that:

$$\|h_u^{(t)}\| \leq B_u^{(t)}, \quad \text{where } B = \max_{u \in \mathcal{N},\, t} B_u^{(t)} \implies \|h_u^{(t)}\| \leq B \ \ \forall u, t$$

We consequently derive the corresponding upper-bound and derive that:

$$\sigma = \sum_{k \in \mathcal{N}(u)} \Delta(s_k) + B\|W_t\|^2 {\Delta_t}^2$$

For nodes not involved in the event and for which $L \geq \min(d(u,i), d(u,v))$. Let's now consider a node u that is not involved in the event and for which $L \geq \min(d(u,i), d(u,v))$. For the considered node u, one of the nodes i included in the event is within the neighborhood, then its memory will be updated as well. From the previously derived inequality, we can directly write:

$$\begin{aligned}
\|h_u^{(t+1)} - h_u^{(t)}\| &\leq \sum_{k \in \mathcal{N}(u)} \Delta(s_k) + B\|W_t\|^2 \\
&\leq \left[\sum_{k \in \mathcal{N}(u) \setminus \{i\}} \Delta(s_k) + B\|W_t\|^2 \right] + \Delta(s_i) + B\|W_t\|^2 \\
&\leq \sum_{k \in \mathcal{N}(u) \setminus \{i\}} \|W_t\| + B\|W_t\|^2 + \Delta(s_i) + B\|W_t\|^2 \\
&\leq deg(u)\big[\|W_t\| + B\|W_t\|^2\big] + \Delta(s_i)
\end{aligned}$$

We consequently derive that:

$$\sigma = deg(u)\big[\|W_t\| + B\|W_t\|^2\big] + \Delta(s_i)$$

□

D Proof of Theorem 3

Theorem. *Consider a CTDG-based function $f : (\mathcal{G}, \mathcal{X}) \to \mathcal{Y}$ based on L GCN-like layers. Let g be a variant of f generated by adding k Virtual Nodes (VNs), where each Virtual Node (VN) represents m nodes. After an event between nodes i and v, for any node u not involved in the event, we have the following:*

- *$\forall u \in \mathcal{V}$, if f is (u, σ)-flowing then g is (u, σ')-flowing with:*

$$\sigma' = \sigma + \frac{\Delta_{t,t+1}(s_i) + \Delta_{t,t+1}(s_v)}{k \times m},$$

Proof. Let $f : \mathcal{G} \to \mathcal{Y}$ be a CTDG graph-function, based on the framework described in Section 3. Let $u \in \mathcal{V}$ be a node within our considered graph and we consider that f is (u, σ)-flowing, therefore:

$$\mathcal{I}_u[f] = \mathbb{E}_{\mathcal{D}}[\|f_u(G^{t+1}) - f_u(G^t)\|] \leq \sigma$$

Let g be an augmented version of f based on k VNs, where each VN represents m nodes. Specifically, we consider $\mathcal{S}_{\mathcal{V}} = \{V_1, V_2, \ldots, V_k\}$ be the set of augmented VNs. The consider node u is connected to its corresponding VN $\hat{v}$. The final representation of node u, using g, can be composed as follows:

$$h'_u = h_u^{(\ell)} + \hat{h}_u,$$

where $h_u^{(\ell)}$ is related to f's computation (either through message-passing or temporal encoding) and $\hat{h}_u$ is related to the information gotten from the corresponding VN $\hat{v}$. We can consequently write:

$$\begin{aligned}
\|g_u(G^{t+1}) - g_u(G^t)\| &= \|h'_{u,t+1} - h'_{u,t}\| \\
&= \|h_{u,t+1}^{(\ell)} - h_{u,t}^{(\ell)} + \hat{h}_{u,t+1} - \hat{h}_{u,t}\| \\
&\leq \|h_{u,t+1}^{(\ell)} - h_{u,t}^{(\ell)}\| + \|\hat{h}_{u,t+1} - \hat{h}_{u,t}\|
\end{aligned}$$

In our analysis, we consider a unique event that happens within a community/cluster and that all the VNs are fully connected between them and to all their "child" nodes, denoted as the set $\mathcal{C}$, and finally that the information is propagated using an average-like aggregation. Therefore, we have the following:

$$\begin{aligned}\|\hat{h}_{u,t+1} - \hat{h}_{u,t}\| &= \|\sum_{\hat{j}\in\mathcal{S}_\mathcal{V}} \frac{h_{\hat{j},t+1}}{k} - \sum_{\hat{j}\in\mathcal{S}_\mathcal{V}} \frac{h_{\hat{j},t}}{k}\| \\ &= \|\sum_{\hat{j}\in\mathcal{S}_\mathcal{V}} \sum_{i\in\mathcal{C}_{\hat{j}}} \frac{h_{i,t+1}}{k\times m} - \sum_{\hat{j}\in\mathcal{S}_\mathcal{V}} \sum_{i\in\mathcal{C}_{\hat{j}}} \frac{h_{i,t}}{k\times m}\| \\ &\leq \sum_{\hat{j}\in\mathcal{S}_\mathcal{V}} \sum_{i\in\mathcal{C}_{\hat{j}}} \|\frac{h_{i,t+1} - h_{i,t}}{k\times m}\|\end{aligned}$$

Since we consider only a unique event, then there will be only two updates (each related to one of the nodes involved in the event). We note that here we consider updates related to the message-passing information and not the temporal projection. We can therefore write:

$$\begin{aligned}\|\hat{h}_{u,t+1} - \hat{h}_{u,t}\| &\leq \sum_{\hat{j}\in\mathcal{S}_\mathcal{V}} \sum_{i\in\mathcal{C}_{\hat{j}}} \|\frac{h_{i,t+1} - h_{i,t}}{k\times m}\| \\ &\leq \frac{\Delta_{t,t+1}(s_i) + \Delta_{t,t+1}(s_v)}{k\times m}\end{aligned}$$

Hence, we have:

$$\begin{aligned}&\|g_u(G^{t+1}) - g_u(G^t)\| = \|h'_{u,t+1} - h'_{u,t}\| \\ &\leq \|h^{(\ell)}_{u,t+1} - h^{(\ell)}_{u,t}\| + \|\hat{h}_{u,t+1} - \hat{h}_{u,t}\| \\ &\leq \|h^{(\ell)}_{u,t+1} - h^{(\ell)}_{u,t}\| + \frac{\Delta_{t,t+1}(s_i) + \Delta_{t,t+1}(s_v)}{k\times m}\end{aligned}$$

And consequently we have that g is (u, σ')-flowing with:

$$\sigma' = \sigma + \frac{\Delta_{t,t+1}(s_i) + \Delta_{t,t+1}(s_v)}{k\times m}$$

□

E Proof of Lemma 1

Lemma. *Let G be an undirected graph with adjacency matrix $A \in \mathbb{R}^{n\times n}$. Consider its rank-$k$ truncated singular value decomposition $A \approx U_k\Sigma_k U_k^\top$, with $\Sigma_k \in \mathbb{R}^{k\times k}$ a diagonal matrix containing k dominant singular values of A and $U_k \in \mathbb{R}^{n\times k}$ a semi-orthogonal matrix containing corresponding left singular vectors. Then, for any pair of nodes $u, v \in \mathcal{V}$, we have:*

$$\|A_{u,:} - A_{v,:}\| \approx \|((U_k)_{u,:} - (U_k)_{v,:})\Sigma_k\|,$$

where the error is bounded:

$$|\|A_{u,:} - A_{v,:}\| - \|((U_k)_{u,:} - (U_k)_{v,:})\Sigma_k\|| \leq \mathcal{O}(\sigma_{k+1}).$$

Proof. Using the rank-k truncated singular value decomposition, the adjacency matrix can be decomposed as

$$A = A_k + A_\perp,$$

where

$$A_k = U_k \Sigma_k U_k^\top$$

is the best rank-k approximation of A, and $A_\perp$ contains the remaining singular values $\sigma_{k+1}, \ldots, \sigma_n$. For any nodes $u, v \in \mathcal{V}$, their row representations satisfy:

$$A_{u,:} = (U_k \Sigma_k U_k^\top)_{u,:} + (U_\perp \Sigma_\perp U_\perp^\top)_{u,:},$$
$$A_{v,:} = (U_k \Sigma_k U_k^\top)_{v,:} + (U_\perp \Sigma_\perp U_\perp^\top)_{v,:}.$$

Taking the row difference:

$$A_{u,:} - A_{v,:} = (U_k)_{u,:}\Sigma_k U_k^\top - (U_k)_{v,:}\Sigma_k U_k^\top + (U_\perp)_{u,:}\Sigma_\perp U_\perp^\top - (U_\perp)_{v,:}\Sigma_\perp U_\perp^\top.$$

Applying the triangle inequality:

$$|\|A_{u,:} - A_{v,:}\| - \|((U_k)_{u,:} - (U_k)_{v,:})\Sigma_k\|| \leq \|((U_\perp)_{u,:} - (U_\perp)_{v,:})\Sigma_\perp U_\perp^\top\|.$$

From the previous formulation, we can write:

$$\begin{aligned}
&|\|A_{u,:} - A_{v,:}\| - \|((U_k)_{u,:} - (U_k)_{v,:})\Sigma_k\|| \\
&\leq \|((U_\perp)_{u,:} - (U_\perp)_{v,:})\Sigma_\perp U_\perp^\top\| \\
&\leq \|(U_\perp)_{u,:}\Sigma_\perp U_\perp^\top - ((U_\perp)_{u,:}\Sigma_\perp U_\perp^\top)\| \\
&\leq \|(U_\perp)_{u,:}\Sigma_\perp U_\perp^\top\| + \|(U_\perp)_{u,:}\Sigma_\perp U_\perp^\top\| \\
&\leq 2\sigma_{k+1}
\end{aligned}$$

Since the residual term $A_\perp$ is at most σ_{k+1} in spectral norm, we obtain the bound:

$$|\|A_{u,:} - A_{v,:}\| - \|((U_k)_{u,:} - (U_k)_{v,:})\Sigma_k\|| \leq 2\sigma_{k+1} \tag{8}$$

Overall, this can also be written as:

$$|\|A_{u,:} - A_{v,:}\| - \|((U_k)_{u,:} - (U_k)_{v,:})\Sigma_k\|| \leq \mathcal{O}(\sigma_{k+1}).$$

From the topology perspective. The previous bound can also be upper-bound by a term depending on the graph's topology represented by its adjacency matrix A. By ordering the singular values: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. We have the following:

$$(k+1)\,\sigma_{k+1}^2 \;\leq\; \sum_{i=1}^{k+1} \sigma_i^2 \;\leq\; \sum_{i=1}^{n} \sigma_i^2 \;=\; \|A\|_F^2,$$

Hence, using Equation 8, we have:

$$|\|A_{u,:} - A_{v,:}\| - \|((U_k)_{u,:} - (U_k)_{v,:})\Sigma_k\|| \leq 2\frac{\sqrt{2 \mid E \mid}}{k+1}$$

□

F Practical Implementation

In the following section, we aim to provide practical implementation guidelines for our proposed approach k-TVNs, which is based on an adaptation of the k-means algorithm. Algorithm 2 outlines the iterative procedure of this adapted k-means method. Specifically, at each individual iteration, each node (represented by its adjacency-matrix row) is assigned to the nearest cluster center in Euclidean distance, then update each center as the mean of its assigned rows. Since each row in the adjacency matrix encodes a node's connectivity pattern, row-wise comparisons naturally capture both the number and identity of shared neighbors. These shared neighborhoods, in turn, drive cluster formation, effectively capturing the graph's underlying communities. By dynamically updating cluster centers based on row-wise distances, the algorithm groups nodes with similar adjacency patterns, i.e., those with strongly overlapping neighbor sets.

Algorithm 2 k-Means-Based Clustering

Require: Adjacency matrix $A \in \mathbb{R}^{n \times n}$, number of clusters $K_0 \in \mathbb{N}$, number of iterations T

1: Interpret each row of A as a data point:

$$\mathbf{x}_i \;=\; (A_{i1},\, A_{i2},\, \ldots,\, A_{in}) \quad \text{for } i = 1, 2, \ldots, n.$$

2: Initialize the cluster centers $\mathbf{c}_1^{(0)}, \mathbf{c}_2^{(0)}, \ldots, \mathbf{c}_{K_0}^{(0)} \;\in\; \mathbb{R}^n$

3: **repeat**

4: **(1) Assignment step:**

5: **for** $i = 1, \ldots, n$ **do**

6: Assign $\mathbf{x}_i$ to cluster: $k = \arg\min_{1 \le j \le K_0} \|\mathbf{x}_i - \mathbf{c}_j^{(t)}\|$

7: **(2) Update step:**

8: **for** $k = 1, \ldots, K_0$ **do**

9: Recompute center $\mathbf{c}_k^{(t+1)}$ as the mean:

$$\mathbf{c}_k^{(t+1)} \;=\; \frac{1}{|C_k(\mathbf{c}^{(t)})|} \sum_{\mathbf{x}_i \,\in\, C_k(\mathbf{c}^{(t)})} \mathbf{x}_i.$$

10: **until** convergence

11: **return** Cluster assignments.

F.1 Time Complexity Analysis

While our proposed *k*-TVNs, which is based on a clustering method, increases the performance of the an underlying TGN model, it comes with a certain price in terms of complexity as previously discussed in Section 5. In this perspective, we investigate the added complexity in terms of training time of a TGN with its counter-part of an augmented version using our proposed method. Note that we used the same set of hyper-parameters for both models and that all the experiments have been run on a NVIDIA L4 and A100 (for the larger datasets `tgbl-comment` and `tgbl-flight`) GPU. Table 4 reports the average training time in seconds (with the corresponding standard deviation) for each method on the considered benchmark datasets.

Method	TGN	k-TVNs	Louvain	Random
Train	15.4 ± 0.1	17.6 ± 0.4	783.9 ± 6.9	15.8 ± 0.1
Validation	77.5 ± 0.2	78.9 ± 0.5	188.3 ± 2.7	78.1 ± 0.3
Test	77.7 ± 0.3	79.3 ± 0.5	205.9 ± 2.1	78.2 ± 0.3

Table 4: Average training, validation and test time (in s) per epoch ($\pm$ denotes standard deviation) for our proposed *k*-TVN in comparison to the original TGN model and the other considered clustering approaches.

F.2 Hyper-parameters Analysis

To better understand the effect of the different hyper-parameters involved in the Algorithm, we provide an empirical analysis consisting of changing the values and discussing their resulting validation and test performance.

Effect of number of iterations. We start by analyze the effect of number of iterations (denoted as T in the Algorithm). Table 5 provides the validation and test MRR for different number of iterations (ranging from 3 to 50). We see that there seem to be a sweet spot point at $T = 10$, where the maximal validation and test values are reached. Interestingly, when increasing the number of iterations beyond that point, both the validation and test MRR value decreases. In particular, we see that the gap between validation and test becomes much bigger.

Effect of number of clusters. While some clustering methods doesn't necessary need a pre-defined number of clusters, in the proposed Algorithm which is based on k-means, we need to define this number. In this perspective, in this analysis, we aim to analyze how such choice affects the performance of the model in terms of downstream performance. We note that the majority of the used

	tgbl-wiki				
	$T = 3$	$T = 5$	$T = 10$	$T = 50$	$T = 100$
Val MRR	60.8	60.9	61.3	60.5	59.5
Test MRR	56.1	56.6	57.2	56.0	55.7

Table 5: Validation and MRR for different number of iterations T on the `tgbl-wiki` dataset.

	tgbl-wiki			
	$K_0 = 2$	$K_0 = 3$	$K_0 = 5$	$K_0 = 10$
Val MRR	61.3	56.2	58.9	58.6
Test MRR	57.2	54.2	55.6	54.9

Table 6: Validation and MRR for different number of clusters K_0 on the `tgbl-wiki` dataset.

graphs are bi-partite and therefore the ultimate choice would be to choose $K_0 = 2$. Table 6 provides the Test and Validation MRR values for the tgbl-wiki dataset. We see that $K_0 = 2$ is the one yielding the best performance, which was expected given the bi-partite aspect of the graphs.

G Datasets and Implementation Details

G.1 Datasets

Real-world Datasets. Table 7 presents an overview of the datasets used in our experimental evaluation. As detailed in the main paper, we conduct experiments on the datasets constituting the TGB benchmark. Specifically, we consider the following datasets:

- `tgbl-wiki`: The dataset is a bi-partite network where Wiki pages and editors are nodes, while an edge is added when a user edits a page at a specific timestamp.
- `tgbl-review`: It is also a bi-partite network where nodes are Amazon product and users, and edges represents a particular review from a user to a product at a given time.
- `tgbl-coin`: This a network representing different cryptocurrency transactions. Specifically, the nodes represent an address and each edge represents the transfer of funds from one adress to another at a time.
- `tgbl-comment`: The graph compromises replies from Reddit where users reply to each other's threads.
- `tgbl-flight`: The dataset represents flight network, where nodes represent airports while the edges are flights between airports at a given day.

Synthetic Datasets. We also assess the benchmark's performance on graphs requiring long-range dependencies, where interactions among distant nodes play a crucial role. For this purpose, we consider a family of graphs derived from the static PascalVOC-SP dataset. In this family, each node corresponds to a region in an image associated with a particular class, and the temporal dimension is introduced by sequentially revealing nodes from the top-left to the bottom-right of the image. Further details can be found in the original work by Gravina et al. [25].

Table 7: Statistics of the link prediction datasets used in our experiments.

	# Nodes	# Edges	# Edge ft.	Split	Surprise Index
`tgbl-wiki`	9,227	157,474	172	70/15/15, Chronological	0.108
`tgbl-review`	352,637	4,873,540	-	70/15/15, Chronological	0.987
`tgbl-coin`	638,486	22,809,486	-	70/15/15, Chronological	0.120
`tgbl-comment`	994,790	44,314,507	-	70/15/15, Chronological	0.823
`tgbl-flight`	18,143	67,169,570	-	70/15/15, Chronological	0.024
T-PascalVOC$_{10}$	2,671,704	2,660,352	14	70/15/15	1.0
T-PascalVOC$_{30}$	2,990,466	2,906,113	14	70/15/15	1.0

G.2 Clustering Specification

For our proposed k-TVNs, we set the number of clusters $k = 2$ and limit the maximum number of iterations to 10. Clustering is performed at every batch: for each batch of events, we apply clustering to the nodes involved in that batch and subsequently construct the VNs. These nodes are fully connected to one another and also linked to their respective child nodes (i.e., the original nodes in the cluster).

We cluster every batch to adapt to non-stationary temporal neighborhoods; this keeps VN assignments aligned with the most recent local structure. While less frequent reclustering reduces compute, it can lag structural drift; we therefore prioritize per-batch updates in our main results.

Note that Algorithm 2 is not required to converge to an optimal solution; instead, we only require a sufficiently good clustering outcome, as discussed in Appendix F.2. To accommodate this, we introduce a tolerance threshold when comparing distances during clustering. In our experiments, this threshold is set to 10^{-2}.

The aggregation within each cluster, used to compute the embedding of the corresponding VN, is implemented as a weighted average, where weights are determined by the degree of each node. While alternative aggregation methods (e.g., attention-based mechanisms) are possible, we adopt the weighted average approach for our empirical evaluations, aligning with the focus of our theoretical analysis.

G.3 Experimental Setup

Hyper-parameters. For all experiments, we used the Adam optimizer [45] with a learning rate of 1×10^{-4} and a batch size of 200 for training, validation, and testing. The training process consisted of 50 epochs with a patience threshold of 5. We set the embedding dimension to 256 as the original parameter used for the TGN.

Implantation Details. Our code is provided in the supplementary materials section (and will be publicly available afterwards upon publication). Our implementation is built on top of two main publicly available implementations: **(i)** the TGN implementation, which is publicly available from the TGB Github and was used to do the experiments on the TGB benchmark graphs, and on the **ii** the CTAN [25] implementation for the long-range synthetic datasets.