Graph Neural Networks for temporal graphs: State of the art, open challenges, and opportunities

Antonio Longa[1,2] [0000−0003−0337−1838], Veronica Lachi[3] [0000−0002−6947−7304], Gabriele Santin[1] [0000−0001−6959−1070], Monica Bianchini[3] [0000−0002−8206−8142], Bruno Lepri[1] [0000−0003−1275−2333], Pietro Liò[4] [0000−0002−0540−5053], Franco Scarselli[3] [0000−0003−1307−0772], and Andrea Passerini[2] [0000−0002−2765−5395]

[1] Fondazione Bruno Kessler, Trento, Italy
[2] University of Trento, Trento, Italy
[3] University of Siena, Siena, Italy
[4] University of Cambridge, Cambridge, United Kingdom

Abstract. Graph Neural Networks (GNNs) have become the leading paradigm for learning on (static) graph-structured data. However, many real-world systems are dynamic in nature, since the graph and node/edge attributes change over time. In recent years, GNN-based models for temporal graphs have emerged as a promising area of research to extend the capabilities of GNNs. In this work, we provide the first comprehensive overview of the current state-of-the-art of temporal GNN, introducing a rigorous formalization of learning settings and tasks and a novel taxonomy categorizing existing approaches in terms of how the temporal aspect is represented and processed. We conclude the survey with a discussion of the most relevant open challenges for the field, from both research and application perspectives.

Keywords: Graph Neural Network · Temporal Graph · Temporal Graph Neural Network

1 Introduction

The ability to process temporal graphs is increasingly important in various fields such as recommendation systems [39], social network analysis [6], transportation systems [47] and face-to-face interactions [20]. Traditional graph-based models are not well-suited for analyzing temporal graphs as they assume a fixed structure and cannot capture temporal evolution. In recent years, several models have been developed to directly encode temporal graphs, including random walk-based methods [37], temporal motif-based methods [19], matrix factorization-based approaches [1], and deep learning models [24].

Graph Neural Networks (GNNs) [32] have emerged as the leading paradigm for static graph processing due to their ability to efficiently propagate information along the graph and learn node and graph representations. GNNs have

achieved state-of-the-art performance in tasks such as node classification [12], link prediction [48], and graph classification [42]. The success of GNNs highlights the importance of developing deep learning techniques for non-Euclidean data and their potential to revolutionize the analysis of graph-structured systems.

GNNs have also been successfully applied to temporal graphs with approaches ranging from attention-based methods [41] to Variational Graph-Autoencoder (VGAE) [11]. However, there is currently a lack of systematic literature review in this area. Existing surveys either discuss general techniques for learning over temporal graphs with only brief mentions of temporal extensions of GNNs [15,2,43,40] or focus on specific topics like temporal link prediction [27,33] or temporal graph generation [10].

This work aims to fill the gap by providing a systematic review of GNN-based methods for temporal graphs, referred to as Temporal GNNs (TGNNs), and formalizing the tasks being addressed. The main contributions include a coherent formalization of learning settings and tasks on temporal graphs, an organization of existing TGNN works into a comprehensive taxonomy based on time representation and the mechanism used, highlighting limitations of current TGNN methods, discussing open challenges for further investigation, and presenting real-world applications where TGNNs could provide substantial gains.

2 Temporal Graphs

We provide a formal definition of the different types of graphs analyzed in this work and we structure different existing notions in a common framework.

Definition 1 (Static Graph - SG). *A* Static Graph *is a tuple* $G = (V, E, X^V, X^E)$*, where* V *is the set of nodes,* $E \subseteq V \times V$ *is the set of edges, and* X^V, X^E *are* d_V*-dimensional node features and* d_E*-dimensional edge features.*

Node and edge features may be empty. Moreover, in the following, we assume that all graphs are directed, i.e., $(u, v) \in E$ does not imply that $(v, u) \in E$.

Extending [27], we define Temporal Graphs as follows.

Definition 2 (Temporal Graph - TG). *A* Temporal Graph *is a tuple* $G_T = (V, E, V_T, E_T)$*, where* V *and* E *are, respectively, the set of all possible nodes and edges appearing in a graph at any time, while*

$$V_T := \{(v, x^v, t_s, t_e) : v \in V, x^v \in \mathbb{R}^{d_V}, t_s \leq t_e\},$$
$$E_T := \{(e, x^e, t_s, t_e) : e \in E, x^e \in \mathbb{R}^{d_E}, t_s \leq t_e\},$$

are the temporal nodes and edges, with time-dependent features and initial and final timestamps. A set of temporal graphs is denoted as $\mathcal{G}_\mathcal{T}$*.*

The definition implies that node and edge features are constant inside each interval $[t_s, t_e]$, but may otherwise change over time. Since the same node or edge may be listed multiple times, with different timestamps, we denote as $\bar{t}_s(v) = \min\{t_s : (v, x^v, t_s, t_e) \in V_T\}$ and $\bar{t}_e(v) = \max\{t_e : (v, x^v, t_s, t_e) \in V_T\}$ the time of

first and last appearance of a node, and similarly for $\bar{t}_s(e), \bar{t}_e(e), e \in E$. Moreover, we set $T_s(G_T) := \min\{\bar{t}_s(v) : v \in V\}$, $T_e(G_T) := \max\{\bar{t}_e(v) : v \in V\}$ as the initial and final timestamps in a TG G_T. For two TGs $G_T^i := (V^i, E^i, V_T^i, E_T^i)$, $i = 1, 2$, we write $G_T^1 \subseteq_V G_T^2$ to indicate the topological inclusion $V^1 \subseteq V^2$, while no relation between the corresponding timestamps is required.

General TGs have no restriction on their timestamps, which can take any value (for simplicity, we just assume that they are non-negative). However, in some applications, it makes sense to force these values to be multiples of a fixed timestep. This leads to the notion of Discrete Time Temporal Graphs, which are defined as follows.

Definition 3 (Discrete Time Temporal Graph - DTTG). *Let $\Delta t > 0$ be a fixed timestep and let $t_1 < t_2 < \cdots < t_n$ be timestamps with $t_{k+1} = t_k + \Delta t$. A* Discrete Time Temporal Graph *G_{DT} is a TG where for each $(v, x^v, t_s, t_e) \in V_T$ or $(e, x^e, t_s, t_e) \in E_T$, the t_s, t_e are taken from the set of fixed timestamps (i.e., $t_s, t_e \in \{t_1, t_2, \ldots, t_n\}$, with $t_s < t_e$).*

2.1 Representation of temporal graphs

Two main strategies can be found in the literature for the description of time-varying graphs, based on snapshots or on events. These different representations lead to different algorithmic approaches. Extending [10] we give a formal definition of these representation strategies.

Definition 4 (Snapshot-based Temporal Graph - STG). *Let $t_1 < t_2 < \cdots < t_n$ be the ordered set of all timestamps t_s, t_e occurring in a TG G_T. Set*

$$V_i := \{(v, x^v) : (v, x^v, t_s, t_e) \in V_T, t_s \leq t_i \leq t_e\},$$
$$E_i := \{(e, x^e) : (e, x^e, t_s, t_e) \in E_T, t_s \leq t_i \leq t_e\},$$

and define the snapshots *$G_i := (V_i, E_i)$, $i = 1, \ldots, n$. Then a* Snapshot-based Temporal Graph *representation of G_T is the sequence*

$$G_T^S := \{(G_i, t_i) : i = 1, \ldots, n\}$$

of time-stamped static graphs.

This representation is mostly used to describe DTTGs, where the snapshots represent the TG captured at periodic intervals (e.g., hours, days, etc.).

The event-based strategy is more appropriate when the focus is on the temporal evolution of individual nodes or edges. This leads to the following definition.

Definition 5 (Event-based Temporal Graph - ETG). *Let G_T be a TG, and let ε denote one of the following* event*:*

- Node insertion $\varepsilon_V^+ := (v, t)$*: the node v is added to G_T at time t, i.e., there exists $(v, x^v, t_s, t_e) \in V_T$ with $t_s = t$.*

- Node deletion $\varepsilon_V^- := (v, t)$*: the node* v *is removed from* G_T *at time* t*, i.e., there exists* $(v, x^v, t_s, t_e) \in V_T$ *with* $t_e = t$*.*
- Edge insertion $\varepsilon_E^+ := (e, t)$*: the edge* e *is added to* G_T *at time* t*, i.e., there exists* $(e, x^e, t_s, t_e) \in E_T$ *with* $t_s = t$*.*
- Edge deletion $\varepsilon_E^- := (e, t)$*: the edge* e *is removed from* G_T *at time* t*, i.e., there exists* $(e, x^e, t_s, t_e) \in E_T$ *with* $t_e = t$*.*

An Event-based Temporal Graph *representation of TG is a sequence of events*

$$G_T^E := \{\varepsilon : \varepsilon \in \{\varepsilon_V^+, \varepsilon_V^-, \varepsilon_E^+, \varepsilon_E^-\}\}.$$

Here it is implicitly assumed that node and edge events are consistent (e.g., a node deletion event implies the existence of an edge deletion event for each incident edge). In the case of an ETG, the TG structure can be recovered by coupling an insertion and deletion event for each temporal edge and node. ETGs are better suited than STGs to represent TGs with arbitrary timestamps.

We will use the general notion of TG, which comprises both STG and ETG, in formalizing learning tasks in the next section. On the other hand, we will revert to the STG and ETG notions when introducting the taxonomy of TGNN methods in Section 4, since TGNNs use one or the other representation strategy in their algorithmic approaches.

3 Learning tasks on temporal graphs

Thanks to their learning capabilities, TGNNs are extremely flexible and can be adapted to a wide range of tasks on TGs. Some of these tasks are straightforward temporal extensions of their static counterparts. However, the temporal dimension has some non-trivial consequences in the definition of learning settings and tasks, some of which are often only loosely formalized in the literature. We start by formalizing the notions of transductive and inductive learning for TGNNs, and then describe the different tasks that can be addressed.

3.1 Learning settings

The machine learning literature distinguishes between inductive learning, in which a model is learned on training data and later applied to unseen test instances, and transductive learning, in which the input data of both training and test instances are assumed to be available, and learning is equivalent to leveraging the training inputs and labels to infer the labels of test instances given their inputs. This distinction becomes extremely relevant for graph-structured data, where the topological structure gives rise to a natural connection between nodes, and thus to a way to propagate the information in a transductive fashion. Roughly speaking, transductive learning is used in the graph learning literature when the node to be predicted and its neighborhood are known at training time — and is typical of node classification tasks —, while inductive learning indicates

that this information is not available — and is most often associated to graph classification tasks.

However, when talking about GNNs with their representation learning capabilities, this distinction is not so sharp. For example, a GNN trained for node classification in transductive mode could still be applied to an unseen graph, thus effectively performing inductive learning.

The temporal dimension makes this classification even more elusive, since the graph structure is changing over time and nodes are naturally appearing and disappearing. Defining node membership in a temporal graph is thus a challenging task in itself.

Below, we provide a formal definition of transductive and inductive learning for TGNNs which is purely topological, i.e. linked to knowing or not the instance to be predicted at the training time, and we complete it with a temporal dimension, which distinguishes between past and future prediction tasks.

Definition 6 (Learning settings). *Assume that a model is trained on a set of* $n \geq 1$ *temporal graphs* $\mathcal{G}_{\mathcal{T}} := \{G_T^i := (V_i, E_i, X_i^V, X_i^E),\ i = 1, \ldots, n\}$. *Moreover, let*

$$T_e^{all} := \max_{i=1,\ldots,n} T_e(G_T^i),\ V^{all} := \cup_{i=1}^n V_i,\ E^{all} := \cup_{i=1}^n E_i,$$

be the final timestamp and the set of all nodes and edges in the training set. Then, we have the following settings:

- Transductive learning*: inference can only be performed on* $v \in V^{all}$, $e \in E^{all}$, *or* $G_T \subseteq_V G_T^i$ *with* $G_T^i \in \mathcal{G}_{\mathcal{T}}$.
- Inductive learning*: inference can be performed also on* $v \notin V^{all}$, $e \notin E^{all}$, *or* $G_T \not\subseteq_V G_T^i$, *for all* $i = 1, \ldots, n$.
- Past prediction*: inference is performed for* $t \leq T_e^{all}$.
- Future prediction*: inference is performed for* $t > T_e^{all}$.

We remark that all combinations of topological and temporal settings are meaningful, except for the case of inductive graph-based tasks. Indeed, the measure of time used in TGs is relative to each single graph. Moving to an unobserved graph would thus make the distinction between past and future pointless. Moreover, let us observe that, in all other cases, the two temporal settings are defined based on the final time of the entire training set, and not of the specific instances (nodes or edges), since their embedding may change also as an effect of the change of their neighbors in the training set.

We will use this categorization to describe supervised and unsupervised learning tasks in Section 3.2-3.3, and to present existing models in Section 4.

3.2 Supervised learning tasks

Supervised learning tasks are based on a dataset where each object is annotated with its label (or class), from a finite set of possible choices $\mathcal{C} := \{C_1, C_2, \ldots, C_k\}$.

Classification

Definition 7 (Temporal Node Classification). *Given a TG* $G_T = (V, E, V_T, E_T)$*, the node classification task consists in learning the function*

$$f_{NC} : V \times \mathbb{R}^+ \to \mathcal{C}$$

which maps each node to a class $C \in \mathcal{C}$*, at a time* $t \in \mathbb{R}^+$*.*

This is one of the most common tasks in the TGNN literature. For instance, [25,41,36,49,30] focus on a future-transductive (FT) setting, i.e., predicting the label of a node in future timestamps. TGAT [41] performs future-inductive (FI) learning, i.e. it predicts the label of an unseen node in the future. Finally, DGNN [22] is the only method that has been tested on a past-inductive (PI) setting, i.e., predicting labels of past nodes that are unavailable (or masked) during training, while no approach has been applied to the past-transductive (PT) one. A significant application may be in epidemic surveillance, where contact tracing is used to produce a TG of past human interactions, and sample testing reveals the labels (infection status) of a set of individuals. Identifying the past infection status of the untested nodes is a PT task.

Definition 8 (Temporal Edge Classification). *Given a TG* $G_T = (V, E, V_T, E_T)$*, the temporal edge classification task consists in learning a function*

$$f_{EC} : E \times \mathbb{R}^+ \to \mathcal{C}$$

which assigns each edge to a class at a given time $t \in \mathbb{R}^+$*.*

Temporal edge classification has been less explored in the literature. Existing methods have focused on FT learning [25,36], while FI, PI and PT have not been tackled so far. An example of PT learning consists in predicting the unknown past relationship between two acquaintances in a social network given their subsequent behaviour. For FI, one may predict if a future transaction between new users is a fraud or not.

In the next definition we use the set of real and positive intervals $I^+ := \{[t_s, t_e] \subset \mathbb{R}^+\}$.

Definition 9 (Temporal Graph Classification). *Let* $\mathcal{G}_\mathcal{T}$ *be a domain of TGs. The graph classification task requires to learn a function*

$$f_{GC} : \mathcal{G}_\mathcal{T} \times I^+ \to \mathcal{C}$$

that maps a temporal graph, restricted to a time interval $[t_s, t_e] \in I^+$*, into a class.*

The definition includes the classification of a single snapshot (i.e., $t_s = t_e$). As mentioned above, in the inductive setting the distinction between past and future predictions is pointless. In the transductive setting, instead, a graph $G_T \in \mathcal{G}_\mathcal{T}$ may be classified in a past mode if $[T_s(G_T), T_e(G_T)] \subseteq [t_s, t_e]$, or in the future mode, otherwise.

None of the existing methods tackles temporal graph classification tasks, possibly for the lack of suitable datasets (see also Section 5). The task has however numerous relevant applications. An example of inductive temporal graph classification is predicting mental disorders from the analysis of the brain connectome [13]. Instead, detecting stages of disease progression from gene expression profiles [9] can be framed as a PT graph classification task.

Regression The tasks introduced for classification can all be turned into corresponding regression tasks, simply by replacing the categorical target $\mathcal{C}$ with the set $\mathbb{R}$. We omit the formal definitions for the sake of brevity. To the best of our knowledge, no existing TGNN work addresses this kind of problem, even if the application of static GNNs alone has already shown outstanding results in this setting, e.g. in weather forecasting [16] and earthquake location and estimation [23].

Link prediction Link prediction requires the model to predict the relation between two given nodes, and can be formulated by taking as input any possible pair of nodes. Thus, we consider the setting to be transductive when both node instances are known at training time, and inductive otherwise.

Definition 10 (Temporal Link Prediction). *Let $G_T = (V, E, V_T, E_T)$ be a TG. The temporal link prediction task consists in learning a function*

$$f_{LP} : V \times V \times \mathbb{R}^+ \to [0, 1]$$

which predicts the probability that, at a certain time, there exists an edge between two given nodes.

The domain of the function f_{LP} is the set of all feasible pairs of nodes, since it is possible to predict the probability of future interactions between nodes that have been connected in the past or not, as well as the probability of missing edges in a past time. Most TGNN approaches for temporal link prediction focus on future predictions, forecasting the existence of an edge in a future timestamp between existing nodes (FT is the most common setting) [25,31,11,46,41,21,36,22,30,49], or unseen nodes (FI) [11,41,30]. The only model that investigates past temporal link prediction is [21], which devises a PI setting by masking some nodes and predicting the existence of a past edge between them. Note that predicting past temporal links can be extremely useful for predicting, e.g., missing interactions in contact tracing for epidemiological studies.

Definition 11 (Event Time Prediction). *Let $G_T = (V, E, V_T, E_T)$ be a TG. The aim of the event time prediction task is to learn a function*

$$f_{EP} : V \times V \to \mathbb{R}^+$$

that predicts the time of the first appearance of an edge.

None of the existing methods address this task. Potential FT applications of event time prediction include predicting when a customer will pay an invoice to its supplier, or how long it takes to connect two similar users in a social network.

3.3 Unsupervised learning tasks

In this section, we formalize unsupervised learning tasks on temporal graphs, an area that has received little to no attention in the TGNN literature so far.

Clustering Temporal graphs can be clustered at the node or graph level, with edge-level clustering being a minor variation of the node-level one. Some relevant applications can be defined in terms of temporal clustering.

Definition 12 (Temporal Node Clustering). *Given a TG $G_T = (V, E, V_T, E_T)$, the temporal node clustering task consists in learning a time-dependent cluster assignment map*

$$f_{NCl} : V \times \mathbb{R}^+ \to \mathcal{P}(V),$$

where $\mathcal{P}(V) := \{p_1, p_2, \ldots, p_k\}$ is a partition of the node set V, i.e., $p_i \subset V_T$, $p_i \cap p_j = \emptyset$, if $i \neq j$, $\cup_{i=1}^{N} p_i = V_T$.

While node clustering in SGs is a very common task, its temporal counterpart has not been explored yet for TGNNs, despite its potential relevance in application domains like epidemic modelling (identifying groups of exposed individuals, in both inductive and transductive settings), or trend detection in customer profiling (mostly transductive).

Definition 13 (Temporal Graph Clustering). *Given a set of temporal graphs $\mathcal{G}_\mathcal{T}$, the temporal graph clustering task consists in learning a cluster-assignment function*

$$f_{GCl} : \mathcal{G}_\mathcal{T} \times I^+ \to \mathcal{P}(\mathcal{G}_\mathcal{T}),$$

where $\mathcal{P}(\mathcal{G}_\mathcal{T}) := \{p_1, \ldots, p_k\}$ is a partition of the set of temporal graphs in the given time interval.

Relevant examples of tasks of inductive temporal graph clustering are grouping diseases in terms of similarity between their spreading processes [7].

Low-dimensional embedding (LDE) LDEs are especially useful in the temporal setting, e.g. to visually inspect temporal dynamics of individual nodes or entire graphs, and identify relevant trends and patterns.

Definition 14 (Low-dimensional temporal node embedding). *Given a TG $G_T = (V, E, V_T, E_T)$, the low-dimensional temporal node embedding task consists in learning a map*

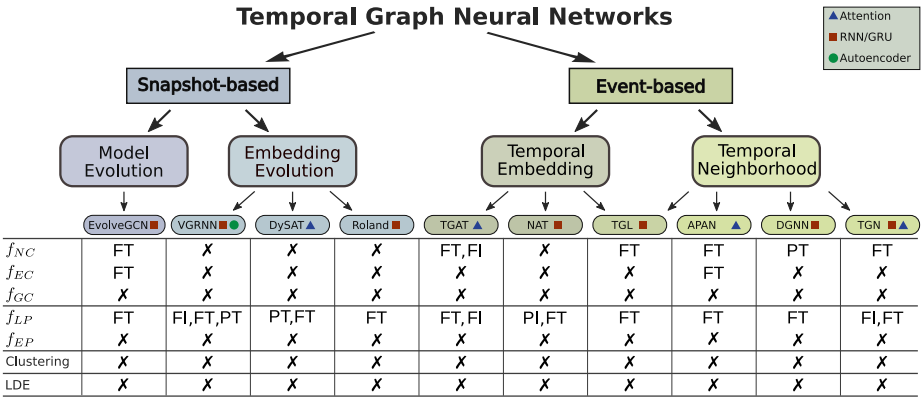
$$f_{NEm} : V \times \mathbb{R}^+ \to \mathbb{R}^d$$

to map a node, at a given time, into a low dimensional space.

Definition 15 (Low-dimensional temporal graph embedding). *Given a domain of TGs $\mathcal{G}_\mathcal{T}$, the low-dimensional temporal graph embedding task aims to learn a map*

$$f_{GEm} : \mathcal{G}_\mathcal{T} \times I^+ \to \mathbb{R}^d,$$

which represents each graph as a low dimensional vector in a given time interval.



	EvolveGCN	VGRNN	DySAT	Roland	TGAT	NAT	TGL	APAN	DGNN	TGN
f_{NC}	FT	✗	✗	✗	FT,FI	✗	FT	FT	PT	FT
f_{EC}	FT	✗	✗	✗	✗	✗	✗	FT	✗	✗
f_{GC}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
f_{LP}	FT	FI,FT,PT	PT,FT	FT	FT,FI	PI,FT	FT	FT	FT	FI,FT
f_{EP}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Clustering	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
LDE	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Fig. 1: **The proposed TGNN taxonomy and an analysis of the surveyed methods.** The top panel shows the new categories introduced in this work with the corresponding model instances (Section 4), where the colored bullets additionally indicate the main technology that they employ. The bottom table maps these methods to the task (Section 3) to which they have been applied in the respective original paper, with an additional indication of their use in the future (F), past (P), inductive (I), or transductive (T) settings (Section 3.1).

4 A taxonomy of TGNNs

This section describes the taxonomy with which we categorize existing TGNN approaches (see Figure 1). Following the representation strategies outlined in Section 2.1, the first level groups methods into *Snaphost-based* and *Event-based*. The second level of the taxonomy further divides these two macro-categories based on the techniques used to manage the temporal dependencies. The leaves of the taxonomy in Figure 1 correspond to the individual models, with a colored symbol indicating their main underlying technology.

4.1 Snapshot-based models

Snaphot-based models are specifically tailored for STGs (see Def. 4) and thus, consistently with the definition, they are equipped with a suitable method to process the entire graph at each point in time, and with a mechanism that learns the temporal dependencies across timesteps. Based on the mechanism used, we can further distinguish between *Model Evolution* and *Embedding Evolution* methods.

Model Evolution methods We call *Model Evolution* the evolution of the parameters of a static GNN model over time. This mechanism is appropriate for modelling STG, as the evolution of the model is performed at the snapshot level. The only existing method belonging to this category is **EvolveGCN** [25]. This model utilizes a RNN to update the GCN [18] parameters at each timestep, allowing for model adaptation that is not constrained by the presence or absence of nodes. The method can handle new nodes without prior historical information.

Embedding Evolution methods Rather than evolving the parameters of a static GNN model, *Embedding Evolution* methods focus on evolving the embeddings produced by a static model. There are several different TGNN models that fall under this category. These networks differ from one another in the techniques used for processing both the structural information and the temporal dynamics of the STGs. **DySAT** [31] introduced a generalization of Graph Attention Network (GAT) [35] for STGs. First, it uses a self-attention mechanism to generate static node embeddings at each timestamp. Then, it uses a second self-attention block to process past temporal embeddings for a node to generate its novel embedding. The **VGRNN** model [11] uses VGAE [17] coupled with SIVI [45] to handle the variation of the graph over time. The learned latent representation is then evolved through an LSTM conditioned on the previous time's latent representation, allowing the model to predict the future evolution of the graph. Finally, in **ROLAND** [46] hierarchical node states are updated based on newly observed nodes and edges through a GRU update module [4].

4.2 Event-based models

Models belonging to the *Event-based* macro category are designed to process ETGs (see Def. 5). These models are able to process streams of events by incorporating techniques that update the representation of a node whenever an event involving that node occurs. The models that lie in this macro category can be further classified in *Temporal Embedding* and *Temporal Neighborhood* methods, based on the technology used to learn the time dependencies. In particular, the *Temporal Embedding* models use recurrent or self-attention mechanisms to model sequential information from streams of events, while also incorporating a time encoding. This allows for temporal signals to be modeled by the interaction between time embedding, node features and the topology of the graph. *Temporal Neighborhood* models, instead, use a module that stores functions of events involving a specific node at a given time. These values are then aggregated and used to update the node representation as time progresses.

Temporal Embedding methods *Temporal embedding* methods model TGs by combining time embedding, node features, and graph topology. These models use an explicit functional time encoding, i.e., a translation-invariant vectorial embedding of time based on Random Fourier Features (RFF) [28]. **TGAT** [41], for example, introduces a graph-temporal attention mechanism which works on the embeddings of the temporal neighbours of a node, where the positional encoding is replaced by a temporal encoding based on RFFs. On the other hand, **NAT** [21] collects the temporal neighbours of each node into dictionaries, and then it learns the node representation with a recurrent mechanism, using the historical neighbourhood of the current node and a RFF based time embedding.

Temporal Neighborhood methods The *Temporal Neighborhood* class includes all TGNN models that make use of a special *mailbox* module to update

node embeddings based on events. When an event occurs, a function is evaluated on the details of the event to compute a *mail* or a *message*. For example, when a new edge appears between two nodes, a message is produced, taking into account the time of occurrence of the event, the node features, and the features of the new edge. The node representation is then updated at each time by aggregating all the generated messages. Several existing TGNN methods belong to this category. In **APAN** [36] an attention-based encoder maps the content of the mailbox to a latent representation of each node, which is decoded by an MLP adapted to the downstream task. **DGNN** [22] combines an *interact* module — which generates an encoding of each event based on the current embedding of the interacting nodes and its history of past interactions — and a *propagate* module — which transmits the updated encoding to each neighbors of the interacting nodes. The aggregation of the current node encoding with those of its temporal neighbors uses a modified LSTM. **TGN** [30] is an inductive framework made up of separate and interchangeable modules. Each node the model has seen so far is characterized by a memory vector, which is a compressed representation of all its past interactions. Given a new event, a mailbox module computes a mail for every node involved. Mails will then be used to update the memory vector. To overcome the so-called staleness problem [15], an embedding module computes, at each timestamp, the node embeddings using their neighbourhood and their memory states. Finally, in **TGL** [49] a mailbox module is used to store a limited number of the most recent interactions, called mails. When a new event occurs, the node memory of the relevant nodes is updated using the messages in the mailbox. The mailbox is updated after the node embeddings are calculated.

5 Open challenges

Several open challenges still need to be faced to fully exploit the potential of TGNNs. We discuss the ones we believe are the most relevant in the following.

Evaluation The evaluation of GNN models has been greatly enhanced by the Open Graph Benchmark (OGB) [14], which provides a standardized evaluation protocol and a collection of graph datasets enabling a fair and consistent comparison between GNN models. An equally well-founded standardized benchmark for evaluating TGNNs does not currently exist. As a result, each model has been tested on its own selection of datasets, making it challenging to compare and rank different TGNNs on a fair basis. The variety and the complexity of learning settings and tasks described in Section 3 makes a standardization of tasks, datasets and processing pipelines especially crucial to allow a fair assessment of the different approaches and foster innovation in the field.

Expressiveness The expressive power of TGNNs is still far from being fully explored, and the design of new WL tests, suitable for TGNNs, is a crucial step towards this aim. This is a challenging task since the definition of a node neighbourhood in temporal graphs is not as trivial as for static graphs, due to the appearing/disappearing of nodes and edges. In [3], a new version of the

WL test for temporal graphs has been proposed, applicable only to DTTGs. Instead, [34] proposed a novel WL test for ETGs, and the TGN model [30] has been proved to be as powerful as this test. A complete theory of the WL test for the different TG representations is still lacking. Moreover, no efforts have been made to incorporate higher-order graph structures to enhance the expressiveness of TGNNs. This task is particularly demanding, since it requires not only the definition of the temporal counterpart of the k-WL test but also some techniques to scale to large datasets.

Learnability Training standard GNNs over large and complex graph data is highly non-trivial, often resulting in problems such as over-smoothing and over-squashing. More intuitively, we yet do not know how to reproduce the breakthrough obtained in training very deep architectures over vector data when training deep GNNs. Such a difficulty is even more challenging with TGNNs, because the typical long-term dependency of TGs poses additional problems to those due to over-smoothing and over-squashing. Modern static GNN models use techniques such as dropout and neighbor sampling, but a general solution is far from being reached. The extension of these techniques to TGNNs are open challenges and we are aware of only one work towards this goal [44].

Real-world applications The analysis of the tasks in Section 3 revealed several opportunities for the use of TGNNs far beyond their current scope of application. A challenging direction for the application of TGNNs is the learning of dynamical systems through the combination of machine learning and physical knowledge [38]. Physic Informed Neural Networks (PINNs) [29] are already revolutionizing the field of scientific computing [5], and static GNNs have been employed in this framework with great success [26,8]. Adapting TGNNs to this field may enable to carry over these results to the treatment of time-dependent problems. Climate science and epidemics studies are other topics of enormous everyday impact that may be explored throw the lens of TGNNs.

6 Conclusion

GNN based models for temporal graphs have become a promising research area. However, we believe that the potential of GNNs in this field has only been partially explored. In this work, we propose a systematic formalization of tasks and learning settings for TGNNs, which was lacking in the literature, and a comprehensive taxonomy categorizing existing methods and highlighting unaddressed tasks. Building on this systematization of the current state-of-the-art, we discuss open challenges that need to be addressed to unleash the full potential of TGNNs.

Acknowledgments

This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

References

1. N.M. Ahmed, L. Chen, Y. Wang, B. Li, Y. Li, and W. Li. Deepeye: Link prediction in dynamic networks based on non-negative matrix factorization. *Big Data Mining and Analytics*, 2018.
2. C.D.T. Barros, M.R.F. Mendonça, A. Vieira, and A. Ziviani. A survey on embedding dynamic graphs. *ACM CSUR*, 2021.
3. S. Beddar-Wiesing, G.A. D'Inverno, C. Graziani, V. Lachi, A. Moallemy-Oureh, F. Scarselli, and J.M. Thomas. Weisfeiler–Lehman goes dynamic: An analysis of the expressive power of graph neural networks for attributed and dynamic graphs. *arXiv preprint arXiv:2210.03990*, 2022.
4. J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
5. S. Cuomo, V.S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F Piccialli. Scientific machine learning through physics–informed neural networks: where we are and what's next. *J. of Scientific Comp.*, 2022.
6. S. Deng, H. Rangwala, and Y. Ning. Learning dynamic context graphs for predicting social events. In *ACM SIGKDD*, 2019.
7. J. Enright and R.K. Rowland. Epidemics on dynamic networks. *Epidemics*, 2018.
8. H. Gao, M.J. Zahr, and J. Wang. Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 2022.
9. R. Gao, J. Yan, P. Li, and L. Chen. Detecting the critical states during disease development based on temporal network flow entropy. *Briefings in Bioinformatics*, 2022.
10. S. Gupta and S. Bedathur. A survey on temporal graph representation learning and generative modeling. *arXiv preprint arXiv:2208.12126*, 2022.
11. E. Hajiramezanali, A. Hasanzadeh, K. Narayanan, N. Duffield, M. Zhou, and X. Qian. Variational graph recurrent neural networks. *NeurIPS*, 32, 2019.
12. W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *NeurIPS*, 30, 2017.
13. M.P. Van Den Heuvel, R.C.W. Mandl C. W., C.J. Stam., R.S. Kahn, P. Hulshoff, and E. Hilleke. Aberrant frontal and temporal complex network structure in schizophrenia: A graph theoretical analysis. *J of Neuroscience*, 2010.
14. W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open Graph Benchmark: Datasets for machine learning on graphs. *NeurIPS*, 33:22118–22133, 2020.
15. S.M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart. Representation learning for dynamic graphs: A survey. *J. Mach. Learn. Res.*, 2020.
16. R. Keisler. Forecasting global weather with graph neural networks. *arXiv preprint arXiv:2202.07575*, 2022.
17. T.N. Kipf and M. Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

18. T.N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR 2016*, 2018.
19. A. Longa, G. Cencetti, S. Lehmann, A. Passerini, and B. Lepri. Neighbourhood matching creates realistic surrogate temporal networks. *arXiv preprint arXiv:2205.08820*, 2022.
20. A. Longa, G. Cencetti, B. Lepri, and A. Passerini. An efficient procedure for mining egocentric temporal motifs. *KDD*, 2022.
21. Y. Luo and P. Li. Neighborhood-aware scalable temporal network representation learning. *arXiv preprint arXiv:2209.01084*, 2022.
22. Y. Ma, Z. Guo, Z. Ren, J. Tang, and D. Yin. Streaming graph neural networks. In *ACM SIGIR*, 2020.
23. I.W. McBrearty and G.C. Beroza. Earthquake location and magnitude estimation with graph neural networks. In *IEEE ICIP*, 2022.
24. F. L. Opolka, A. Solomon, C. Cangea, P. Veličković, P. Liò, and R.D. Hjelm. Spatio-temporal deep graph infomax. *arXiv preprint arXiv:1904.06316*, 2019.
25. A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson. EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In *AAAI*, 2020.
26. T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P .Battaglia. Learning mesh-based simulation with graph networks. In *ICLR*, 2021.
27. M. Qin and D Yeung. Temporal link prediction: A unified framework, taxonomy, and review. *arXiv preprint arXiv:2210.08765*, 2022.
28. A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NeurIPS*, 2008.
29. M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.
30. E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.
31. A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *WSDM*, 2020.
32. F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 2008.
33. J. Skarding, B. Gabrys, and K. Musial. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access*, 9:79143–79168, 2021.
34. A.H. Souza, D. Mesquita, S. Kaski, and V. Garg. Provably expressive temporal graph networks. *NeurIPS*, 2022.
35. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
36. X. Wang, D. Lyu, M. Li, Y. Xia, Q. Yang, X. Wang, Xinguang Wang, P. Cui, Y. Yang, B. Sun, et al. APAN: Asynchronous propagation attention network for real-time temporal graph embedding. In *SIGMOD*, 2021.
37. Y. Wang, Y. Chang, Y. Liu, J. Leskovec, and P. Li. Inductive representation learning in temporal networks via causal anonymous walks. *arXiv preprint arXiv:2101.05974*, 2021.
38. J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar. Integrating scientific knowledge with machine learning for engineering and environmental systems. *ACM Comput. Surv.*, 55(4), 2022.

39. S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui. Graph neural networks in recommender systems: a survey. *ACM CSUR*, 2022.
40. Y. Xie, C. Li, B. Yu, C. Zhang, and Z. Tang. A survey on dynamic network embedding. *arXiv preprint arXiv:2006.08093*, 2020.
41. D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020.
42. K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.
43. G. Xue, M. Zhong, J. Li, J. Chen, C. Zhai, and R. Kong. Dynamic network embedding survey. *Neurocomputing*, 2022.
44. M. Yang, Z. Meng, and I. King. Featurenorm: L2 feature normalization for dynamic graph embedding. In *ICDM*, 2020.
45. M. Yin and M. Zhou. Semi-implicit variational inference. In *ICML*, 2018.
46. J. You, T. Du, and J. Leskovec. ROLAND: graph learning framework for dynamic graphs. In *ACM SIGKDD*, 2022.
47. B. Yu, H. Yin, and Z. Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.
48. M. Zhang and Y. Chen. Link prediction based on graph neural networks. *NeurIPS*, 2018.
49. H. Zhou, D. Zheng, I. Nisa, V. Ioannidis, X. Song, and G. Karypis. TGL: A general framework for temporal GNN training on billion-scale graphs. *arXiv preprint arXiv:2203.14883*, 2022.