IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Foundations and modelling of dynamic networks using Dynamic Graph Neural Networks: A survey

JOAKIM SKARDING[1], BOGDAN GABRYS[1], AND KATARZYNA MUSIAL .[1]
[1]Complex Adaptive Systems Lab, Data Science Institute, University of Technology Sydney, Sydney, NSW 2007, Australia

Corresponding author: Joakim Skarding (e-mail: joakim.skarding@uts.edu.au).

This work was supported by the Australian Research Council, "Dynamics and Control of Complex Social Networks" under Grant DP190101087.

ABSTRACT Dynamic networks are used in a wide range of fields, including social network analysis, recommender systems and epidemiology. Representing complex networks as structures changing over time allow network models to leverage not only structural but also temporal patterns. However, as dynamic network literature stems from diverse fields and makes use of inconsistent terminology, it is challenging to navigate. Meanwhile, graph neural networks (GNNs) have gained a lot of attention in recent years for their ability to perform well on a range of network science tasks, such as link prediction and node classification. Despite the popularity of graph neural networks and the proven benefits of dynamic network models, there has been little focus on graph neural networks for dynamic networks. To address the challenges resulting from the fact that this research crosses diverse fields as well as to survey dynamic graph neural networks, this work is split into two main parts. First, to address the ambiguity of the dynamic network terminology we establish a foundation of dynamic networks with consistent, detailed terminology and notation. Second, we present a comprehensive survey of dynamic graph neural network models using the proposed terminology.

INDEX TERMS Dynamic network models, graph neural networks, link prediction, temporal networks.

I. INTRODUCTION

The bulk of network science literature focuses on static networks, yet every network existing in the real world changes over time. In fact, dynamic network structure has been frequently seen as a complication to be suppressed, to ease progress in the study of networks [1]. Since networks have been used as representations of complex systems in fields as diverse as biology and social science, advances in dynamic network analysis can have a large and far-reaching impact on any field using network analytics [2].

Dynamic networks add a new dimension to network modelling and prediction – time. This new dimension radically influences network properties which enable a more powerful representation of network data which in turn increases predictive capabilities of methods using such data [3], [4]. In fact, dynamic networks are not mere generalizations of static networks, they exhibit different structural and algorithmic properties [5].

This work is both broader and narrower in scope than previous works. The first part of this survey (section II) is broader in scope than related surveys and introduces dynamic networks and dynamic network models (referring to the 'foundations and modelling of dynamic networks' part of the title). The second part of this survey (section III and section IV) is narrower in scope and more detailed than related surveys, and is a survey on dynamic graph neural networks (referring to the 'using Dynamic Graph Neural Networks' part of the title).

Foundations of dynamic networks:

Dynamic networks suffer from a known terminology problem [6]. Complex networks which change over time have been referred to, among others, as; dynamic networks [7], [8], temporal networks [2], [9], evolutionary networks [3] or time-varying networks [10]. With models often working only on specific types of networks, a clear and more detailed terminology for dynamic networks is necessary. We describe dynamic networks foundations as well as propose and develop an associated taxonomy of dynamic networks to contextualize the models in this survey and enable a more thorough comparison between the models. We are unaware of any work with a comprehensive taxonomy of dynamic networks and therefore it can be considered as the first major contribution of this paper.

Dynamic networks is a vast and interdisciplinary field. Models of dynamic networks are designed by researchers from different disciplines and they usually use modelling

methods from their fields. This survey provides a cross-disciplinary overview of dynamic network models. This overview is not intended to be seen as a dynamic models survey, but rather as a context for dynamic graph neural networks and as a reference point for further exploration of the field of dynamic networks modelling.

We consider a dynamic network to be a network where nodes and edges appear and/or disappear over time. Due to the terminology problem establishing a terminology and a clear definition of a dynamic network is a necessity for a survey of any kind of dynamic network models such as dynamic graph neural networks. In the process, we introduce a specific and comprehensive terminology that enable future works to forego the extensive definition process and simply apply our terminology.

Related surveys [2], [6], [11] focus either on specific kinds of dynamic networks, for example, temporal networks [2], [6] or on specific types of models, for example, representation learning [11]–[13]. We are unaware of any work which gives as complete a picture of dynamic networks and dynamic network models as we do. The first section is thus broader in scope than other surveys that focus on only one network type or one type of network model.

Modelling dynamic networks using Dynamic Graph Neural Networks: A dynamic graph neural network (DGNN) is considered to be a neural network architecture that can encode a dynamic network and where the aggregation of neighbouring node features is part of the neural network architecture. DGNNs encode both structural and temporal patterns in dynamic networks. To encode structural patterns DGNNs often make use of a graph neural network (GNN) and for temporal patterns, they tend to use time series modules such as recurrent neural networks (RNN) or positional attention. Spatio-temporal networks (graphs where the topology is static and only node or edge features change [14]) are out of the scope of this survey and thus so are Spatio-temporal graph neural networks [14], [15].

DGNNs, like GNNs and other representation learning models, are versatile in which tasks they can be applied to. With different decoders and different data, different tasks are possible. In practice, so far DGNNs have been applied to similar tasks as GNNs, the most common of these tasks are node classification [16]–[19] and link prediction [16], [18]–[20], which both have diverse and interesting application across many disciplines. Link prediction may for example be applied in knowledge graph completion [21], [22] or by recommender systems [18], [19]. DGNNs have also been used for novel tasks such as predicting path-failure in dynamic graphs [23], quantifying scientific impact [24], and detecting dominance, deception and nervousness [25].

There are several surveys on graph neural networks [8], [26], [27] as well as surveys on network representation learning [28], [29], our work differs from theirs as we cover GNNs which encode dynamic networks. Kazemi et al. [11], Xie et al. [12] and Barros et al. [13] are the works most similar to this paper as they survey dynamic network representation learning. The distinction is that they survey the broader topic of representation learning on dynamic networks whereas we survey dynamic graph neural networks which is a subset of representation learning on dynamic networks. We thus survey a more narrow scope than dynamic representation learning surveys and a different network type from the GNN surveys which focus on static networks [8], [26], [27]. Wu et al. [27] and Zhou et al. [8] also survey spatio-temporal graph neural networks, which encode spatio-temporal networks (static networks with dynamic node attributes).

This survey's contributions are: (i) A conceptual framework and a taxonomy for dynamic networks, (ii) an overview of dynamic network models, (iii) a survey of dynamic graph neural networks (iv) an overview of how dynamic graph neural networks are used for prediction of dynamic networks (dynamic link prediction).

This work follows the encoder-decoder framework used by Hamilton et al. [28] and is split into three distinct sections each building upon the previous one.

1) Section II is a discussion on dynamic networks. It serves as a foundation to the following sections. In this section we explore different definitions of links and introduce a novel dynamic network taxonomy. We also give a brief overview of the dynamic network model landscape, which contextualizes the rest of the survey.
2) Section III is a survey of the deep learning models for encoding dynamic network topology. This covers dynamic network encoders.
3) Section IV is an overview of how the encoders from section III are used for prediction. This includes dynamic network decoders, loss functions and evaluation metrics.

II. DYNAMIC NETWORKS

A complex network is a representation of a complex system. A network that changes over time can be represented as a dynamic network. A dynamic network has both temporal and structural patterns, and these patterns are described by a dynamic network model.

The definition of a link is essential to any network representation. It is even more essential in dynamic networks, as it dictates when a link appears and disappears. Different link definitions affect network properties which in turn affect which models are capable of representing the dynamic network.

Dynamic networks are complex networks that change over time. Links and nodes may appear and disappear. With only this insight we can form a general definition for dynamic networks. Our definition is inspired by Rossetti and Cazabet [30].

Definition 1 (Dynamic Network) A Dynamic Network is a graph $G = (V, E)$ where: $V = \{(v, t_s, t_e)\}$, with v a vertex of the graph and t_s, t_e are respectively the start and end timestamps for the existence of the vertex (with

$t_s \leq t_e$). $E = \{(u, v, t_s, t_e)\}$, with $u, v \in V$ and t_s, t_e are respectively the start and end timestamps for the existence of the edge (with $t_s \leq t_e$).

This definition and any of the later definitions represent unlabeled and undirected networks, but they can however trivially be extended with both direction and labels taken into account.

Whereas dynamic networks are defined as complex networks where links and nodes may appear and disappear, dynamic network models are often designed to work on specific kinds of dynamic networks and specific dynamic network representations. It, therefore, makes sense to distinguish between different kinds of dynamic networks and how they are represented.

Table 7 an overview of the notation and Table 8 is an overview of the abbreviations used in this work.

There are several surveys on dynamic network methods [2], [3], [6], [11], [30]–[35]. These surveys focus either on specific kinds of dynamic networks or on a specific discipline and limit the scope of the survey to models in that discipline. To the best of our knowledge there is no comprehensive survey of dynamic networks, nor does any dynamic network model survey present a complete foundation or framework for dynamic networks. The aim of this section is to set the stage for the dynamic graph neural network survey by creating a conceptual framework for dynamic networks with more precise terminology and to add context by giving an overview of methods used for modelling dynamic network topology.

A. DYNAMIC NETWORK REPRESENTATIONS

Dynamic networks can be represented in different ways and there are advantages and disadvantages inherent to the different representation types.

Dynamic network representations can be grouped into four distinct levels ordered by temporal granularity: (i) static, (ii) edge-weighted, (iii) discrete, and (iv) continuous networks [36].

FIGURE 1: Network representations ordered by temporal granularity. Static networks are the most coarse-grained and continuous representations are the most fine-grained. With increasing temporal granularity comes increasing model complexity. The figure is inspired by Fig. 5.1 from Rossetti [36]

Fig. 1 shows those four representations with increasing model complexity as the model becomes more temporally fine-grained:

- **Static networks** have no temporal information.
- **Edge weighted networks** have temporal information included as labels on the edges and/or nodes of a static network. The most straightforward example of this is a static network with the edges labelled with the time they were last active.
- **Discrete networks** are represented in discrete time intervals. These can be represented by multiple snapshots of the network at different time intervals.
- **Continuous networks** have no temporal aggregation applied to them. This representation carries the most information but is also the most complex.

Static and edge-weighted networks are used to model stable patterns or the actual state of the network, whereas discrete and continuous methods are used for more dynamic modelling [30]. This work focuses on dynamic networks and will therefore only cover discrete and continuous representations.

Fine-grained representations can be trivially aggregated to produce coarser representations. For example, links in a continuous representation can be aggregated into snapshots (or time-windows) which is a discrete representation. Any discrete representation can combine the snapshots, yielding an edge-weighted representation and any edge-weighted representation can discard the weights thus yielding a static network.

1) Discrete Representation

Discrete representations use an ordered set of graphs (snapshots) to represent a dynamic graph.

$$DG = \{G^1, G^2, \ldots, G^T\}, \tag{1}$$

where T is the number of snapshots. Discrete representations, often simply referred to as "snapshots" is common for dynamic networks [2], [3], [9]. Using a discrete representation of the dynamic network allows for the use of static network analysis methods on each of the snapshots. Repeated use of the static methods on each snapshot can then collectively give insight into the network's dynamics.

There are other approaches that effectively use snapshots as well. Overlapping snapshots such as sliding time-windows [37] are also used in dynamic network analysis to have less radical change from one network snapshot to the next [38]. Discrete dynamic networks need not be represented as an ordered set of graphs, they may also be represented as a multi-layered network [39] or as a tensor [40].

2) Continuous Representation

Continuous network representations are the only representations that have exact temporal information. This makes them the most complex but also the representation with the most potential. We cover three continuous representations: (i) the event-based; (ii) the contact sequence; and (iii) the graph streams. The first two representations are taken from the temporal network literature and they are suitable for networks where links do not persist for long [2], [6], [9]. The third representation, i.e. the graph stream, is used in dynamic

networks where edges persist for longer [3]. The focus in these representations is on when edges are active, with no mention of change on nodes. All three representations are described in more detail below:

1) **The event-based representation** includes the time interval at which the edge on a graph is active [9]. An event is synonymous with a link in this case. It is a representation for dynamic networks focusing on link duration. The network is given by a time-ordered list of events which include the time at which the event appeared and the duration of the event.

$$EB = \{(u_i, v_i, t_i, \Delta_i); i = 1, 2, \ldots\}, \tag{2}$$

where u_i and v_i is a node pair on which the i-th event occurs, t_i is the timestamp for when the event starts and Δ_i is the duration of the event. This is very similar to, and serves the same purpose as, the interval graph [2]. The difference is that the interval graph has the time at which the event ends while the event-based representation has the duration of the event.

2) **The contact sequence representation** is a simplification of the event-based representation. In a contact, sequence the link is instantaneous and thus no link duration is provided.

$$CS = \{(u_i, v_i, t_i); i = 1, 2, \ldots\}, \tag{3}$$

It is common to consider event times in real systems instantaneous if the duration of the event is short or not important [2], [9]. Examples of systems where this representation is suitable, include message networks such as text message and email networks.

3) **The graph stream representation** is used to represent static graphs that are too large to fit in memory but can also be used as a representation of a dynamic network [32]. It is similar to the event-based representation, however, it treats link appearance and link disappearance as separate events.

$$GS = \{e_1, e_2, \ldots\}, \tag{4}$$

where $e_i = (u_i, v_i, t_i, \delta_i)$, and u_i and v_i is the node pair on which the i-th event occurs, t_i is the time at which the event occurs, and $\delta_i \in \{-1, 1\}$ where -1 represents an edge removal and 1 represents that an edge is added.

The original representation (used for large graphs) does not include timestamped information of when an edge is added/removed [32]. Timestamps will have to be added for retrieving temporal information.

Since graph streams are mostly used to circumvent hardware limitations rather than a limitation of network representations, we will not survey them in detail here. For a more in-depth discussion of the graph streams, we refer the interested reader to [3], [32], [34].

Which of the above representations is suitable for the network depends on the link duration with the intricacies of link duration covered in the next section.

B. LINK DURATION SPECTRUM

Dynamic networks go by many names and sometimes these names indicate specific types of dynamic networks. There is substantial literature on 'temporal networks' [2], [6], [9] which focuses on highly dynamic networks where links may represent events such as human interactions or a single email. On the other hand, there is also literature that refers to slowly evolving networks, where links represent persistent relations [3]. To the best of our knowledge, there are only two works that take note of this distinction, Rossetti and Cazabet [30], and Holme [6].

Rossetti and Cazabet [30] refer to temporal interaction and relational networks (our temporal and evolving networks respectively), but they do not categorize or make a formal distinction between the different networks.

Holme [6] suggests that temporal networks can be distinguished by two requirements: (i) The dynamics on the network being at the same or at a similar time scale as the dynamics of the network; and (ii) The dynamic network is non-trivial at any given time (an instantaneous snapshot yield little to no network structure).

The distinction manifests itself in networks even when not considering dynamics on the networks, and this work is limited to the dynamics of the network. Therefore we distinguish temporal networks purely based on network topology. We use the second requirement noted by Holme [6].

This work not only provides a way to distinguish between temporal networks and dynamic networks, but it also proposes a framework in which all networks of dynamic topology fit. We do this by introducing the link duration spectrum.

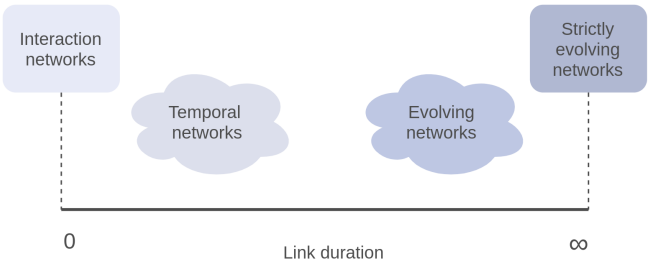

FIGURE 2: Temporal and evolving networks on the link duration spectrum. The spectrum go from 0 (links have no duration) to infinity (links last forever).

Fig. 2 shows different types of networks on the link duration spectrum. The scale goes from interactions with no link duration to links that have infinite link duration. No link ever disappears in a network with infinite link duration. Temporal networks reside on the lower end of the link

duration spectrum, whereas evolving networks reside on the higher end. The distinction is as follows:

- **Temporal networks.** Highly dynamic networks which are too dynamic to be represented statically. The network is at any given time non-trivial. These networks are studied in the temporal network literature [2], [9]. Network properties such as degree distribution and clustering coefficient cannot be adopted directly from static networks and are non-trivial to define. It is more natural to think of a link as an event with a duration.
- **Evolving networks.** Dynamic networks where events persist for long enough to establish a network structure. An instantaneous snapshot yields a well-defined network. Network properties such as degree distribution and clustering coefficient can be adopted from static networks and gradually updated. These are the networks most often referred to when the term dynamic network is used. Links persist for so long that it is more natural to think of link appearance as an event and link disappearance as another event.

Furthermore, there is one notable special case for each of the dynamic network types. These are types of networks that reside on the extreme ends of the link duration spectrum:

- **Interaction networks.** A type of temporal network where links are instantaneous events. These networks are studied in the temporal network literature and often represented as contact sequences [2], [9].
- **Strictly evolving networks.** A type of evolving network where events have infinite duration. This implies that the links never disappear.

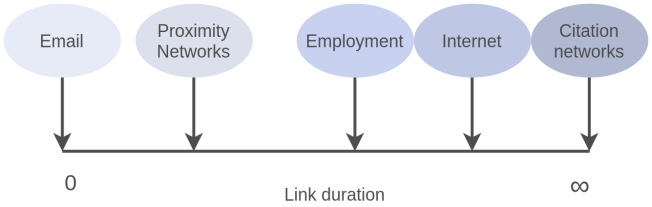

FIGURE 3: Examples of networks on the link duration spectrum.

Fig. 3 shows examples of networks on the link duration spectrum.

- An email is a nearly instantaneous event[1], an email network can therefore be considered an interaction network.
- Proximity networks are used as an example of a temporal network in [2]. The link is defined by who is close to whom at what time. Links require maintenance and do not typically last very long.
- Employment networks are social networks where links are formed between employees and employers. The link requires an action after it has been established (termination of contract) to change its state, but also maintenance (continued work from the employee). This network resides in the fuzzy area between temporal and evolving networks and can be treated as either.
- The Internet is an example of the network where we consider nodes linked if data-packets can flow between nodes. A link tends to persist for a long time once established and thus the internet can be thought of as an evolving network.
- Citation networks where links are defined as one paper citing another have the most persisting links. Once a paper cites another paper, the link lasts forever. This leads to a strictly growing network where no edges disappear. These networks have the additional special characteristic that edges only appear when new nodes appear.

Link definitions influence link duration, which in turn influences a network type. Links can be modified in ways that alter their link duration (also known as time to live, TTL [30]). An email network could define a link as: Actors have once sent an email between each other. This would modify the email link, which is usually nearly instant in duration to a link that will never disappear. This modification moves the network all the way to the right on the spectrum shown in Fig. 2. It transforms an interaction network into a strictly evolving one. Another example of a modification is to use a time-window to force forgetting. A time-window can be applied to a citation network such that only citations which occurred during the time-window appear as links. This will move the network to the left on the link duration spectrum. Depending on the size of the time-window the modified network may be either an evolving or a temporal network.

An additional theoretical special case that is not covered by this concept is a network where links may only disappear. This special case may justify another dimension along which dynamic networks should be distinguished.

C. NODE DYNAMICS

Another distinguishing factor among dynamic networks is whether nodes may appear or disappear. When modelling networks, it is sometimes simpler to assume that the number of nodes may not change so that the only possible new links are links between already existing nodes.

Many evolving network models assume that edges appear as a new node appears. These models include pseudo-dynamic models such as preferential attachment [41], forest fire [42] and GraphRNN [43]. This is fitting for a citation network where every node is a paper and the edges are cited papers, though, in many real-world networks, edges can appear and disappear regardless of whether nodes appear.

With respect to node change, we can distinguish between two kinds of networks.

- **Static** where the number of nodes stays static over time; and

[1]If you model information propagation then in practice it takes time from the moment an email is sent until it is read, so that case considering the email an instantaneous event is an approximation.

TABLE 1: Dynamic network types by node dynamics and link duration, excluding special cases.

Node dynamics	Link duration: Temporal	Link duration: Evolving
Static	Node-static temporal	Node-static evolving
Dynamic	Node-dynamic temporal	Node-dynamic evolving

- **Dynamic** where the nodes may appear and disappear.

A notable special case of node-dynamic networks are the networks where nodes may only appear:

- **Growing** networks are those where nodes may only appear. We consider this a special case of node-dynamic networks.

We are unaware of any real-world networks where nodes may only disappear. But it should be noted as at least a theoretical special case. Node growing networks on the other hand are rather common.

Any kind of node dynamics can be combined with any kind of link duration network. We can thus have, a growing evolving network or a node-static temporal network. Similarly to the edge duration spectrum, a node duration spectrum could theoretically be established, but it has no direct impact on dynamic network structure and we, therefore, chose to keep node dynamics a discrete distinction.

The node dynamics is an important consideration when modelling the network. Some models support node dynamics whereas others do not.

D. THE DYNAMIC NETWORK CUBE

Many models assume that nodes disappear when there are no longer any links connected to such nodes. This scheme can work for evolving networks, but in temporal networks, it is common that nodes have no links for the majority of the time. Thus for a temporal network, it makes sense to model node dynamics separately from link dynamics.

Different aspects of dynamic network representation have been covered in the previous sections. Section II-A defined different dynamic representations ordered by temporal granularity, section II-B defined network types by link duration and section II-C defined network types by node dynamics. This section will consider these previous sections jointly and discuss how the different network types fit together.

Table 3 includes a comprehensive list of the different dynamic network types. The types are grouped by node dynamic, temporal granularity and link duration type. Types of networks in each group can generally be combined, thus we can have a continuous node-static temporal network. The three groups can be thought of as dimensions of a space where different points in the space would represent different types of dynamic networks.

The 3D network type space resulting from excluding special cases is visualised in Fig. 4. When excluding special cases there are two types of networks along each dimension. The nodes are organised along three dimensions: temporal granularity (discrete and continuous) from Section II-A, the link duration spectrum (temporal and evolving) from Section II-B and node dynamics (node-dynamic and node-static) from Section II-C.

Additionally, Table 2 presents the suggested terminology for each of the dynamic network types. The precise dynamic network term column show the suggested terms for the different network types. These eight types represent domain-independent types of dynamic networks.

E. DYNAMIC NETWORK MODELS

This brief discussion on dynamic network models is intended to give a high-level overview of the dynamic model landscape without discussing different kinds of models in detail. For a detailed discussion, we refer to dedicated works. The aim of this section, is to give the reader the background and context needed to navigate through the field of dynamic network models.

A network model may model a variety of different network characteristics or dynamics. In this work, we focus on models of dynamic network structure. Many models define rules for how links are established [41], [42]. The rules are defined such that a network evolved with those rules express some desired features. These features are often observed in real-world networks and then included in models as a rule. The search for a good dynamic network model is thus also a search for accurate rules on link formation.

Network models might aim to replicate characteristics like node degree distribution or average shortest path between nodes [44]. The models define probabilistic rules for how links form such that the emerging network has certain distributions of given characteristics observed in real-world networks [44]. Some dynamic network models, particularly temporal network models, focus on temporal aspects. An example of a temporal characteristic is the distribution of inter-event times [9].

There are several use cases for network models. They may be used as reference models [2], [6] or as realistic models [45]–[47], and depending on their purpose there are several tasks the model can be used for. These include:

- Reference models are used in the analysis of static networks to study the importance and role of structural features of static networks. Reference models aim to preserve some characteristic such as node degree distribution and otherwise create maximally random networks. The goal is to determine how the observed network is different from a completely random network with the same characteristics. This approach has been adapted to temporal networks [2].
- Realistic models aim to replicate the change in the network as closely as possible. They can be used for several tasks such as network prediction [11], [47], [48] and community detection [30]. Examples include probabilistic models such as the dynamic stochastic block model [49] and representation learning based models such as E-LSTM-D [47]. Some realistic models aim to generate (simulate) realistic networks [43], [50].

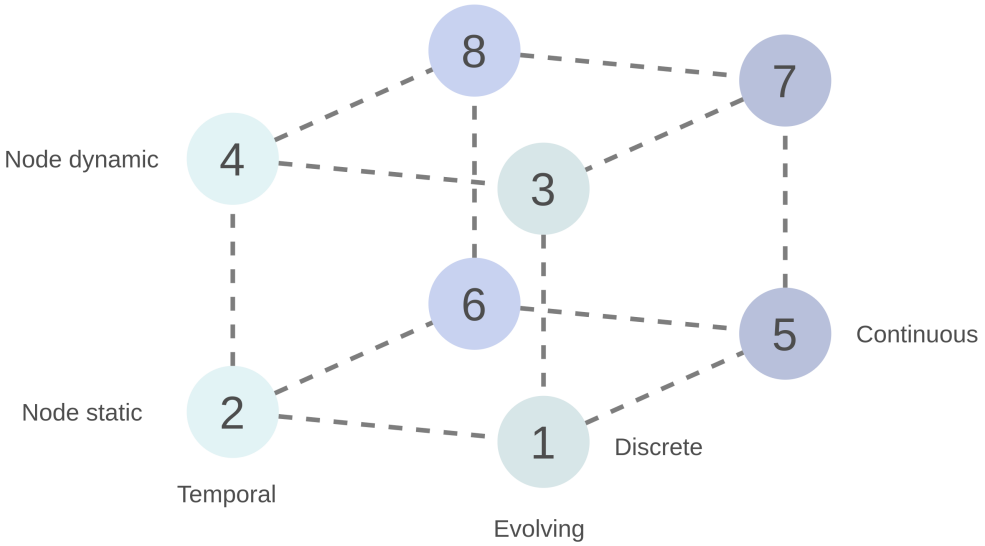

FIGURE 4: The dynamic network cube. The cube is a novel framework that succinctly represents different kinds of dynamic networks. Each node represents a specific type of dynamic networks. The nodes are organised along three dimensions: temporal granularity (discrete and continuous) from Section II-A, the link duration spectrum (temporal and evolving) from Section II-B and node dynamics (node-dynamic and node-static) from Section II-C. The complete list of terminology from the cube is presented in Table 2.

TABLE 2: Terminology of the dynamic network cube.

Node	Temporal granularity	Node dynamics	Link duration	Precise dynamic network term
1	Discrete	Node-static	Evolving	Discrete node-static evolving network
2			Temporal	Discrete node-static temporal network
3		Node-dynamic	Evolving	Discrete node-dynamic evolving network
4			Temporal	Discrete node-dynamic temporal network
5	Continuous	Node-static	Evolving	Continuous node-static evolving network
6			Temporal	Continuous node-static temporal network
7		Node-dynamic	Evolving	Continuous node-dynamic evolving network
8			Temporal	Continuous node-dynamic temporal network

TABLE 3: Types of dynamic networks along three dimensions. Static networks and edge-weighted networks are not dynamic networks, but they are included for completeness. If we exclude special cases, we are left with two elements in each dimension.

Dimension	Network types
Temporal granularity	Static, edge-weighted, discrete, continuous
Link duration	Interaction, temporal, evolving, strictly evolving
Node dynamics	Node-static, node-dynamic, node-appearing, node-disappearing

We establish a typology of models for dynamic network topology. The typology is based on the type of method used to model the network (see Fig. 5).

We group models intended for inference or identifying statistical regularities under statistical models. These include dynamic random graph models, probabilistic models, activity driven models and relational event models. Random graph models (RGM) and Exponential random graph models (ERGM) are random graph models which produce randomly connected graphs while following known common network topology [44]. Activity driven models are fit to interaction networks by modelling the activity of each node [51]. Relational event models are continuous-time models for interaction networks, they define the propensity for a future event to happen between node pairs.

Latent space models and stochastic block models are generative probabilistic models. Latent space models require the fitting of parameters with Markov chain Monte Carlo (MCMC) methods and are very flexible but scale to only a few hundred nodes [52]. Stochastic block models, on the other hand, scale to an order of magnitude larger networks, at a few thousand nodes [52].

Stochastic actor oriented models (SAOM) are continuous-time models which consider each node an actor and model actor behaviour. SAOMs learn to represent the dependencies between a network structure, the position of the actor and the actor behaviour [53].

Dynamic network representation learning includes a diverse set of methods that can be used to embed the dynamic graph in a latent space. Representation learning on dynamic networks includes models based on tensor decomposition, random walks and deep learning. Since latent space models and stochastic block models also generate variables in a

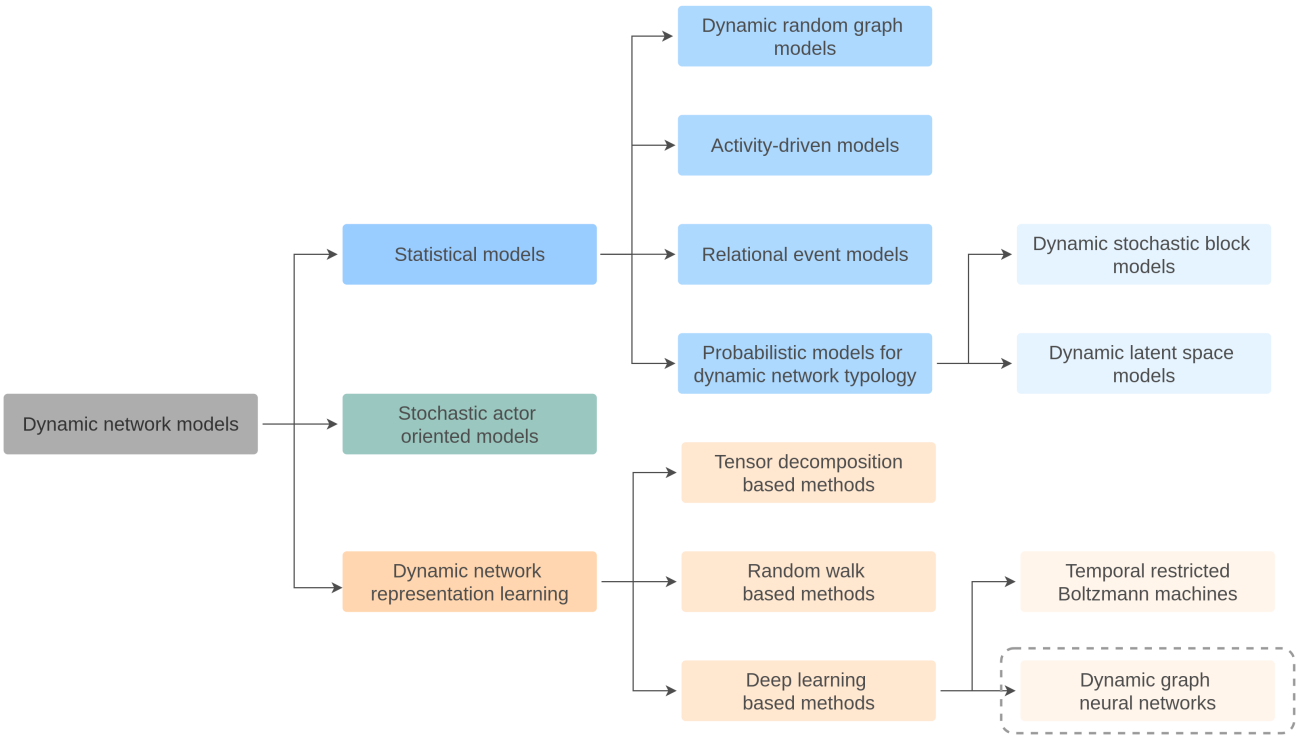

FIGURE 5: An overview of dynamic network models with dynamic graph neural networks outlined. Statistical models are models intended for inference or identifying statistical regularities in dynamic networks. Representation learning models are models which automatically detect features needed for the intended task. Stochastic actor oriented models are agent-based models. Dynamic network representation learning consist of shallow (tensor decomposition and random walk based) methods and deep learning based methods. This work explores dynamic graph neural networks in detail.

latent space they are closely related to dynamic network representation learning.

Tensor decomposition is analogous to matrix factorization where the extra dimension is time [11]. Random walk approaches for dynamic graphs are generally extensions of random walk based embedding methods for static graphs or they apply temporal random walks [9]. Deep learning models include deep learning techniques to generate embeddings of the dynamic network. Deep models can be contrasted with the other networks representation learning models which are shallow models. We distinguish between two types of deep learning models: (i) Temporal restricted Boltzmann machines and (ii) Dynamic graph neural networks. Temporal restricted Boltzmann machines are probabilistic generative models which have been applied to the dynamic link prediction problem [4], [54]–[56]. Dynamic graph neural networks combine deep time series encoding with the aggregation of neighbouring nodes. Often discrete versions of these models take the form of a combination of a GNN and an RNN. Continuous versions of dynamic graph neural networks cannot make direct use of a GNN since a GNN require a static graph. Continuous DGNNs must therefore modify how node aggregation is done.

A detailed survey of all kinds of dynamic network models is too broad a topic to cover in detail by one survey. Deep learning based models for dynamic networks is a rapidly growing and exciting field, however, no existing survey focuses exclusively on dynamic graph neural networks (Kazemi et al. [11], Xie et al. [12] and Barros et al. [13] being the closest).

For the models not discussed in section III there are several works describing and discussing them in detail. Random reference models for temporal networks are surveyed in [2] and [6]. For activity-driven models see Perra et al. [51] and for an introduction to the Relational Event Model (REM) see Butts [57]. See Hanneke et al. [58] for Temporal ERGMs (TERGM) on discrete dynamic networks. Block et al. [59] provides a comparison of TERGM and SAOM. Fritz et al. [33] provide a comparison of a discrete-time model, based on the TERGM, and the Relational Event Model (REM), a continuous-time model. Goldenberg et al. [60] survey dynamic network models and their survey include dynamic random graph models and probabilistic models. Kim et al. [31] surveys latent space models and stochastic block models for dynamic networks. For an introduction to SOAM see Snijders et al. [53]. For surveys of representation learning on dynamic networks see Kazemi et al. [11], Xie et al. [12] and Barros et al. [13], and for a survey of dynamic link prediction, including Temporal restricted Boltzmann machines, see Divakaran et al. [54].

F. DISCUSSION AND SUMMARY

We have given a comprehensive overview of dynamic networks. This establishes a foundation on which dynamic network models can be defined and thus sets the stage for the survey on dynamic graph neural networks. Establishing this foundation included the introduction of a new taxonomy for dynamic networks and an overview of dynamic network models.

Section II-A presents representations of dynamic networks and distinguishes between discrete and continuous dynamic networks. In section II-B we introduce the link duration spectrum and distinguish between temporal and evolving networks, and in section II-C node dynamics is discussed, we distinguish between node-static and node-dynamic networks. Section II-D brings together the previous sections to arrive at a comprehensive dynamic network taxonomy.

Discrete representations have seen great success in use on evolving networks with slow dynamics. Graph streams are used on evolving networks that update too frequently to be represented well by snapshots [3]. Both discrete and continuous representations are used to represent temporal networks [2], [9]. Table 4 combines information from section II-A and section II-B and summarizes the existing representations in terms of temporal granularity and link duration.

TABLE 4: Suitable dynamic network representations for temporal and evolving networks.

Temporal granularity	Temporal network	Evolving network
Continuous	Event-based representation or Contact sequence	Graph stream
Discrete	Time-windows	Snapshots

Discrete representations have several advantages. A model which works on the static network case can be extended to dynamic networks by applying it on each snapshot and then aggregating the results of the model [11], [31]. This makes it relatively easy, compared to the continuous representation to design dynamic network models. Furthermore, the distinction between an evolving and a temporal network is less important. If modelling a temporal network, one only needs to make sure that a time-window size is large enough that the network structure emerges in each snapshot. However, the discrete representations have their disadvantages too. Chief among them is coarse-grained temporal granularity. When modelling a temporal network the use of a time-window is a must. By using a time-window the appearance order of the links and temporal clustering (links appearing frequently together) is lost.

Reducing the size of the time-window or the interval between snapshots is a way to increase temporal granularity. There are however some fundamental problems with this. In the case of a temporal network, a small time-window will eventually yield a snapshot with no network structure. In the case of an evolving network, we will have a sensible network no matter how small the time-window, however, there is a trade-off with run-time complexity. Discrete models tend to process the entire graph in each snapshot. In which case the run-time will increase linearly with the number of snapshots. The run-time problem is exacerbated by the fact that a lot of real-world graphs are huge which make the run-time on each snapshot significant.

Continuous representations offer superior temporal granularity and thus theoretically a higher potential to model dynamic networks. However, continuous-time models tend to be more complex and require either completely new models or significant changes to existing ones to work on the continuous representation. Continuous models are less common than discrete-time models [3], [11], [30]. This is likely due to continuous methods being significantly more difficult to develop than discrete methods [3].

When modelling dynamic networks in continuous time it is essential to specify which kind of network is being modelled. As models for temporal and evolving networks may not be mutually exclusive and many models work on only specific types of networks. In these cases, it might be possible to modify the link duration of a network to run a model on the network. This modification may come at the loss of information, for example when modifying an interaction network to a strictly evolving network, any reappearing link will be removed.

This entire background section establishes a foundation and a conceptual framework in which dynamic networks can be understood. By providing an overview of dynamic network models, it maps out the landscape around deep learning on dynamic graphs thus providing the necessary context. The following sections will explore dynamic graph neural networks in detail.

III. DYNAMIC GRAPH NEURAL NETWORKS

Network representation learning and Graph Neural Networks (GNN) have seen rapid progress recently and they are becoming increasingly important in complex network analysis. Most of the progress has been done in the context of static networks, with some advances being extended to dynamic networks. Particularly GNNs have been used in a wide variety of disciplines such as chemistry [61], [62], recommender systems [63], [64] and social networks [65], [66].

GNNs are deep neural network architectures that encode graph structures. They do this by aggregating features of neighbouring nodes together. One might think of this node aggregation as similar to the convolution of pixels in convolutional neural networks (CNN). By aggregating features of neighbouring nodes together GNNs can learn to encode both local and global structure.

Several surveys exist of works on static graph representation learning [29], [67] and static graph neural networks [8], [26], [27]. Time-series analysis is relevant for work on dynamic graphs, thus recent advances in this domain is of relevance. For and up to date survey of deep learning on time series we refer to Fawaz et al. [68].

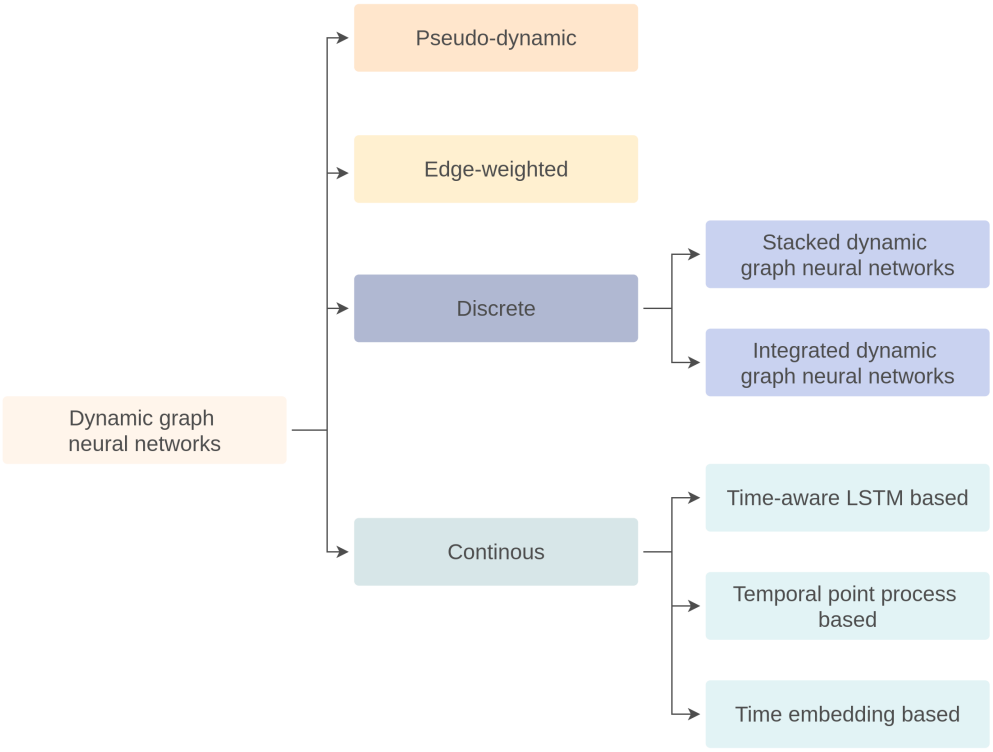

FIGURE 6: An overview of the different types of dynamic graph neural networks. This is an extension of Fig 5 where we zoom in on graph neural networks. Different models are first grouped by which type of network they encode (pseudo-dynamic, edge-weighted, discrete or continuous). Discrete models are grouped by whether the structural layers and temporal layers are stacked, or integrated into one layer. Continuous models are grouped by how they encode temporal patterns.

If dealing with an evolving graph, a static graph algorithm can be used to maintain a model of the graph. Minor changes to the graph would most likely not change the predictions of a static model too much, and the model can then be updated at regular intervals to avoid getting too outdated. We suspect that a spatial GNN is likely to stay accurate for longer than a spectral GNN, since the spectral graph convolution is based on the graph laplacian which will go through more changes than the local changes in a spatial GNN.

It is important to define what we mean by a dynamic graph neural network (DGNN). Informally we can say that a DGNN is a neural network that encodes a dynamic graph. However, there are some representation learning models for dynamic graphs using deep methods, which we do not consider dynamic graph neural networks. A key characteristic of a graph neural network is an aggregation of neighbouring node features (also known as message passing) [8]. Thus, if a deep representation learning model aggregates neighbouring nodes as part of its neural architecture we call it a dynamic graph neural network. In the discrete case, a DGNN is a combination of a GNN and a time series model. Whereas in the continuous case we have more variety since the node aggregation can no longer be done using traditional GNNs. Given this definition of representation learning, network models where RNNs are used but network structure is learned using other methods than node aggregation (temporal random walks for example), are not considered DGNNs.

The previous section (Section II) introduced a framework for dynamic networks and an overview of dynamic network models. The overview presented in Fig. 5 shows dynamic graph neural networks to be a part of deep representation learning, which in turn is part of dynamic network representation learning. We further extend the overview in Fig. 5 to show a hierarchical overview of dynamic graph neural networks, Fig. 6.

An overview of the types of DGNN encoders is seen in Fig. 6. The encoders are grouped first by which type of network they encode, then by model type. The pseudo-dynamic approaches model a network with changing topology, but not time. Discrete DGNNs model discrete networks and continuous DGNNs model continuous networks. A discrete DGNNs encode the network snapshot by snapshot and encode a snapshot all at once, similar to how a GNN encode a static network. A continuous DGNN iterate over the network edge by edge and is thus completely independent of any snapshot size.

Common to all DGNNs is that the encoders aim to capture both structural and temporal patterns and store these patterns in embeddings. A stacked DGNNs separate encoding of

structural and temporal patterns in separate layers, having one layer for structural patterns (using a static GNN) and one layer for temporal patterns (often using some form of an RNN), these models often make use of existing layers and combine them in new ways to encode dynamic networks. Integrated DGNNs combine structural and temporal patterns in one layer. This means that integrated DGNNs require the design of new layers, not just a combination of existing layers. The continuous DGNNs consist of RNN, Temporal point process (TPP) and time embedding based methods.

A timeline of dynamic network models with a focus on DGNNs is shown in Fig. 7. The timeline includes the first appearance of each of the models found in Fig. 5, significant network embedding models preceding DGNNs and DGNNs.

We consider the Albert-Barabasi model [45] the first dynamic network model, although it is only a pseudo-dynamic model (see section III-A). The Dynamic Social Network in Latent space" (DSNL) model [70] is the first dynamic latent space model [31]. The Temporal Exponential Random Graph Model (TERGM) [58] a type of dynamic random graph model was introduced in 2009. Snijders et al. introduced Stochastic Actor Oriented Models (SAOM) [53] for dynamic networks in 2010. The first dynamic stochastic block model (DSBM) was introduced by Yang et al. [71]. The first restricted boltzmann machine (RBM) for static social networks [56] in 2013 was shortly followed by the first RBM for dynamic networks, the Temporal Restricted Boltzmann Machine (TRBM) in 2014.

Prior to DGNNs there were several influential static embedding methods and graph neural networks. The first GNN [72] was introduced in 2008. Deepwalk [73], a highly influential node embedding fueled by random walks was introduced in 2014. Some Graph Convolutional Neural networks (GCN) [74], [75] which function as building blocks and inspiration for several DGNNs were released in 2016.

The first DGNNs were discrete DGNNs. First (GCRN-M1 & GCRN-M2) was introduced by Seo et al. [69], followed by Manessi et al. [76] a few months later. Know-Evolve [21] a TPP based model was the first continuous model, which in turn directly inspired DyREP [48] by the same author. JODIE [77] is notable as the RNN based DGNN, and it was quickly followed by Streaming GNN [78] which was the first DGNN for continuous strictly evolving networks. DySAT [17] introduced the first discrete DGNN which was based solely on attention, thus not using an RNN. EvolveGCN [16] introduced the first design that had an RNN feed into a GCN, rather than what the previous models did, which was to have a GCN feed into an RNN. The first pseudo-dynamic GNN, G-GCN was introduced in early 2019. TGAT [18] is the first DGNN to encode inter-event time as a vector, while TGN [19] adds a memory module to TGAT. HDGNN showed how to use DGNNs for encoding discrete heterogeneous dynamic networks and TDGNN although simple was the first GNN to explicitly weight the edges to enable interaction network encoding.

This section surveys DGNNs, identifies different types of DGNNs and covers how embeddings are encoded. The next section (Section IV) covers decoding of the embeddings.

A. PSEUDO-DYNAMIC MODELS

Goldenberg et al. [60] refer to network models as "pseudo-dynamic" when they contain dynamic processes, but the dynamic properties of the model are not fit to the dynamic data. A well-known example of a non-DGNN pseudo-dynamic model is the Barabasi-Albert model [45].

G-GCN [79] can be seen as an extension of the Variational Graph Autoencoder (VGAE) [80] which is able to predict links for nodes with no prior connections, the so-called cold start problem. It uses the same encoder and decoder as VGAE, namely a GCN [75] for encoding and the inner product between node embeddings as a decoder. The resulting model learns to predict links of nodes that have only just appeared.

B. EDGE-WEIGHTED MODELS

As noted earlier in Section II-A, dynamic network representations can be simplified. One way to simplify the modelling is to convert the dynamic network to an edge-weighted network and then use a static GNN on the edge-weighted network. This is exactly what Temporal Dependent GNN (TDGNN) does [81]. They convert an interaction network to an edge weighted network by using an exponential distribution. An edge which appeared more recently gets a high weight and one that appeared long ago gets a low weight. After the conversion an standard GCN [75] is applied to the edge-weighted network. While the conversion from interaction network (a continuous network) to edge-weighted is done as part of the model in the original work, there appears to be is no reason why it cannot be done as a pre-processing step and thus we classify it as an edge-weighted model.

C. DISCRETE DYNAMIC GRAPH NEURAL NETWORKS

Modelling using discrete graphs has the advantage that static graph models can be used on each snapshot of the graph. Discrete DGNNs use a GNN to encode each graph snapshot. We identify two kinds of discrete DGNNs: Stacked DGNNs and Integrated DGNNs.

Autoencoders use either static graph encoders or DGNN encoders, however since they are trained a little differently from DGNNs and generally make use of (and thus extend) a DGNN encoder they are here distinguished from other models.

A discrete DGNN combines some form of deep time-series modelling with a GNN. The time-series model often comes in the form of an RNN, but self-attention has also been used.

Given a discrete graph $DG = \{G^1, G^2, \ldots, G^T\}$ a discrete DGNN using a function f for temporal modelling

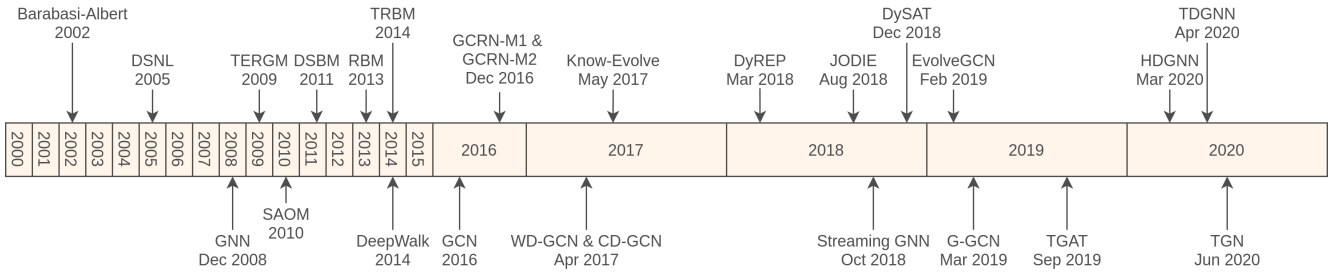

FIGURE 7: Timeline of dynamic graph models and dynamic graph neural networks. The timeline shows the first dynamic network models of each type of model from Fig 5 and significant representation learning models leading up to the first DGNN. After the first DGNNs (GCRN-M1 and GCRN-M2 [69]) in Dec 2016, only DGNNs are marked on the timeline. DGNNs are marked by the month they were first publicised as they appeared in tight succession. The timeline indicates when a model was first publicized (the timeline may therefore show a different year than that in the citation if the paper was pre-published)

can be expressed as:

$$\begin{aligned} z_1^t, \ldots, z_n^t &= \text{GNN}\left(G^t\right) \\ h_j^t &= f\left(h_j^{t-1}, z_j^t\right) \quad \text{for } j \in [1, n] \end{aligned} \tag{5}$$

where f is a neural architecture for temporal modelling (in the methods surveyed f is almost always an RNN but can also be self-attention), $z_i^t \in \mathbb{R}^l$ is the vector representation of node i at time t produced by the GNN, where l is the output dimension of the GNN. Similarity $h_i^t \in \mathbb{R}^k$ is the vector representation produced by f, where k is the output dimension of f.

This can also be written as:

$$\begin{aligned} Z^t &= \text{GNN}\left(G^t\right) \\ H^t &= f\left(H^{t-1}, Z^t\right) \end{aligned} \tag{6}$$

Informally we can say that the GNN is used to encode each network snapshot and f (the RNN or self-attention) encodes across the snapshots.

Seo et al. [69] introduce two deep learning models which encode a static graph with dynamically changing attributes. Whereas the modelling of this kind of graph is outside the scope of the survey, the two models they introduced are, to the best of our knowledge, the first DGNNs. They introduce both a stacked DGNN and an integrated DGNN: (i) Graph Convolutional Recurrent Network Model 1 (GCRN-M1) and (ii) GCRN model 2 (GCRN-M2) respectively. Very similar encoders have been used in later publications for dynamic graphs.

1) Stacked Dynamic Graph Neural Networks

The most straightforward way to model a discrete dynamic graph is to have a separate GNN handle each snapshot of the graph and feed the output of each GNN to a time series component, such as an RNN. We refer to a structure like this as a stacked DGNN.

There are several works using this architecture with different kinds of GNNs and different kinds of RNNs. We'll use GCRN-M1 [69] as an example of a stacked DGNN. This model stacks the spectral GCN from [74] and a standard peephole LSTM [82]:

$$\begin{aligned} z_t &= \text{GNN}\left(X_t\right) \\ i &= \sigma\left(W_i z_t + U_i h_{t-1} + w_i \odot c_{t-1} + b_i\right) \\ f &= \sigma\left(W_f z_t + U_f h_{t-1} + w_f \odot c_{t-1} + b_f\right) \\ c_t &= f_t \odot c_{t-1} \\ &\quad + i_t \odot \tanh\left(W_c z_t + U_c h_{t-1} + b_c\right) \\ o &= \sigma\left(W_o z_t + U_o h_{t-1} + w_o \odot c_t + b_o\right) \\ h_t &= o \odot \tanh\left(c_t\right) \end{aligned} \tag{7}$$

Let $X_t \in \mathbb{R}^{n \times d}$, $W \in \mathbb{R}^{k \times nl}$, $U \in \mathbb{R}^{k \times k}$ and $h, w, c, b, i, f, o \in \mathbb{R}^k$. The gates which are normally vectors in the LSTM are now matrices. Also, $z_t \in \mathbb{R}^{nl \times 1}$ is a vector and not a matrix. Even though the GNN used by Seo et al. [69] can output features with the same structure as the input, they reshaped the matrix into a vector. This allows them to use a one-dimensional LSTM to encode the entire dynamic network.

Whereas [69] use a spectral GCN and a peephole LSTM this is not a limitation of the architecture as any GNN and RNN can be used. Other examples of stacked DGNNs are: RgCNN [83] which use the Spatial GCN, PATCHY-SAN [84] stacked with a standard LSTM and DyGGNN [85] which uses a gated graph neural network (GGNN) [86] combined with a standard LSTM.

Manessi et al. [76] present two stacked DGNN encoders: Waterfall Dynamic-GCN (WD-GCN) and Concatenated Dynamic-GCN (CD-GCN). These architectures are distinct in that they use a separate LSTM per node (although the weights across the LSTMs are shared). The GNN in this case is a GCN [75] stacked with an LSTM per node. The WD-GCN encoder with a vertex level decoder is shown in Fig. 8. WD-GCN and CD-GCN differ only in that CD-GCN adds skip-connections past the GCN. The equations below are for the WD-GCN encoder.

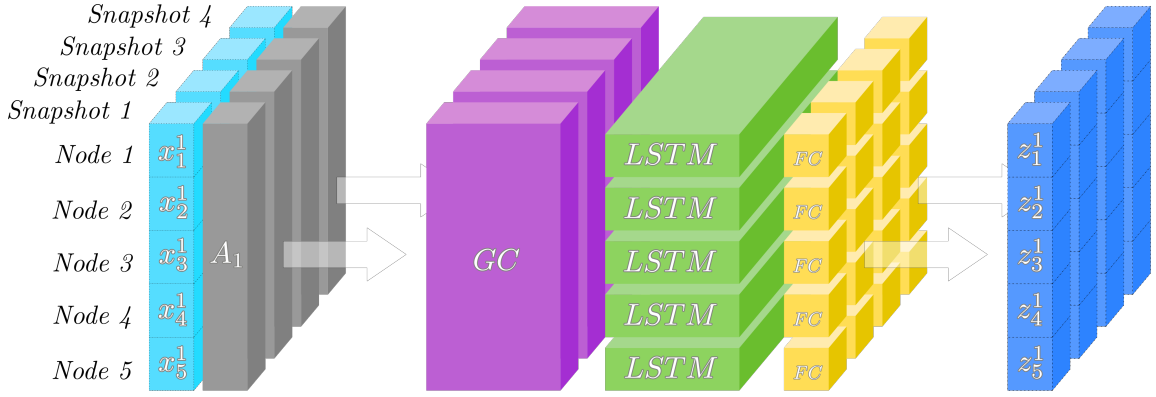

FIGURE 8: Stacked DGNN structure from Manessi *et al.* [76]. The graph convolution layer (GC) encode the graph structure in each snapshot while the LSTMs encode temporal patterns.

$$\begin{aligned} Z_1, \ldots, Z_t &= \text{GNN}(A_1, X_1), \ldots, \text{GNN}(A_t, X_t) \\ H &= \text{v-LSTM}_k(Z_1, \ldots, Z_t) \end{aligned} \tag{8}$$

Let $A \in \mathbb{R}^{n \times n}$ be the adjacency matrix, n be the number of nodes, d be the number of features per node and $X_t \in \mathbb{R}^{n \times d}$ be the matrix describing the features of each node at time t. $Z_t \in \mathbb{R}^{n \times l}$ where l is the output size of the GNN and $H \in \mathbb{R}^{k \times n \times t}$ where k is the output size of the LSTMs.

$$\text{v-LSTM}_k(Z_1, \ldots, Z_t)) = \begin{pmatrix} \text{LSTM}_k(V_1' Z_1, \ldots, V_1' Z_t) \\ \vdots \\ \text{LSTM}_k(V_n' Z_1, \ldots, V_n' Z_t) \end{pmatrix} \tag{9}$$

where LSTM is a normal LSTM [87] and $V_p \in \mathbb{R}^n$ is defined as $V_p = \delta_{pi}$ where δ is the Kronecker delta. Due to the v-LSTM layer the encoder can store a hidden representation per node.

Since a set of snapshots is a time-series, one is not restricted to the use of RNNs and other works have stacked GNNs with other types of deep time-series models. Sankar et al. [17] present a stacked architecture that consists completely of self-attention blocks. They use attention along the spatial and temporal dimensions. For the spatial dimension, they use the Graph Attention Network (GAT) [88] and for the temporal dimension, they use a transformer [89]. Wang et al. [25], [90] stacks a GNN with 1D temporal convolution (TNDCN) similar to the dilated convolution in WaveNet [91].

Stacked DGNN architectures also exist for specific types of dynamic networks. There is HDGNN [24] for heterogeneous dynamic networks and TeMP [22] for knowledge networks.

When encoding graphs one option is to split the graph into sub-graphs and use a GNN to project each sub-graph as done by Zhang et al. [92] for static GNNs. This approach has also been applied to DGNNs by Cai et al. [93], where they split each snapshot into sub-graphs and use a stacked DGNN for anomaly detection.

2) Integrated Dynamic Graph Neural Networks

Integrated DGNNs are encoders that combine GNNs and RNNs in one layer and thus combine modelling of the spatial and the temporal domain in that one layer.

Inspired by convLSTM [94] Seo et al. [69] introduced GCRN-M2. GCRN-M2 amounts to convLSTM where the convolutions are replaced by graph convolutions. ConvLSTM uses a 3D tensor as input whereas here we are using a two-dimensional signal since we have a feature vector for each node.

$$\begin{aligned} f_t &= \sigma\left(W_f *_{\mathcal{G}} X_t + U_f *_{\mathcal{G}} h_{t-1} + w_f \odot c_{t-1} + b_f\right) \\ i_t &= \sigma\left(W_i *_{\mathcal{G}} X_t + U_i *_{\mathcal{G}} h_{t-1} + w_i \odot c_{t-1} + b_i\right) \\ c_t &= f_t \odot c_{t-1} \\ &\quad + i_t \odot \tanh\left(W_c *_{\mathcal{G}} X_t + U_c *_{\mathcal{G}} h_{t-1} + b_c\right) \\ o_t &= \sigma\left(W_o *_{\mathcal{G}} X_t + U_o *_{\mathcal{G}} H_{t-1} + w_o \odot c_t + b_o\right) \\ h_t &= o \odot \tanh\left(c_t\right) \end{aligned} \tag{10}$$

where $x_t \in \mathbb{R}^{n \times d}$, n is the number of nodes and x_i is a signal for the i-th node at time t. $W \in \mathbb{R}^{K \times k \times l}$ and $U \in \mathbb{R}^{K \times k \times k}$ where k is the size of the hidden layer and K is the number of Chebyshev coefficients . $W_f *_{\mathcal{G}} x_t$ denotes the graph convolution on x_t.

EvolveGCN [16] integrates an RNN into a GCN. The RNN is used to update the weights W of the GCN. [16] name their layer the Evolving Graph Convolution Unit (EGCU) and present two versions of it: (i) EGCU-H where the weights W are treated as the hidden layer of the RNN and (ii) EGCU-O where the weights W are treated as the input and output of the RNN. In both EGCU-H and EGCU-O, the RNN operate on matrices rather than vectors as in the standard LSTM. The EGCU-H layer is given by the following equations, where (l) indicates the neural network layer:

$$\begin{aligned} W_t^{(l)} &= \text{GRU}\left(H_t^{(l)}, W_{t-1}^{(l)}\right) \\ H_t^{(l+1)} &= \text{GNN}\left(A_t, H_t^{(l)}, W_t^{(l)}\right) \end{aligned} \tag{11}$$

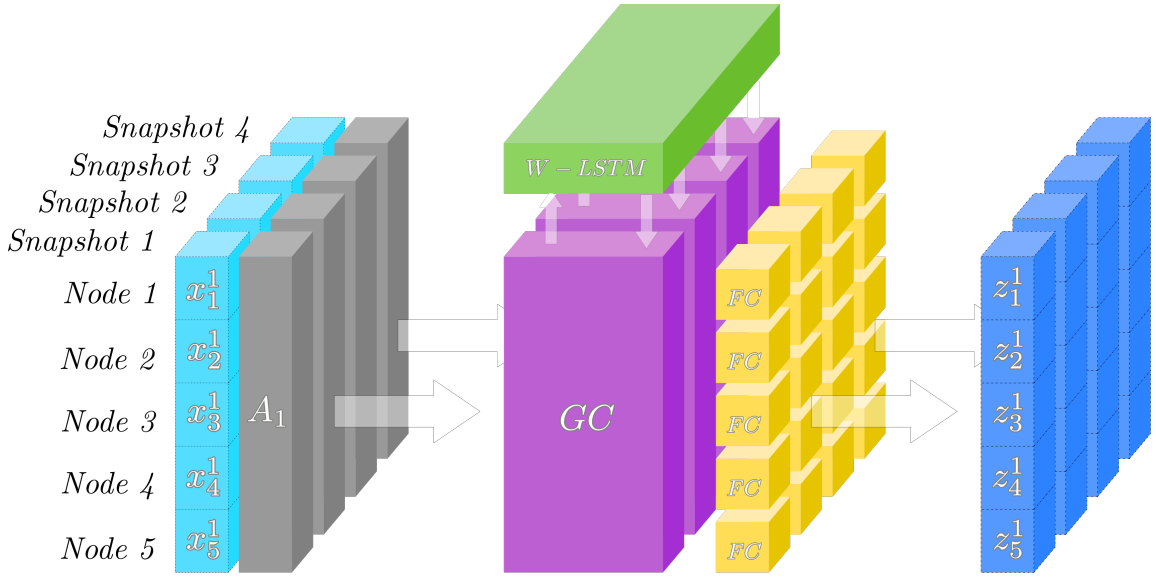

FIGURE 9: Integrated DGNN structure of EvolveGCN with an EGCU-O layer [16]. The EGCU-O layer constitutes the GC (graph convolution) and the W-LSTM (LSTM for GC weights). W-LSTM is used to initialize the weights of the GC.

And the EGCU-O layer is given by the equations:

$$\begin{aligned} W_t^{(l)} &= \text{LSTM}\left(W_{t-1}^{(l)}\right) \\ H_t^{(l+1)} &= \text{GNN}\left(A_t, H_t^{(l)}, W_t^{(l)}\right) \end{aligned} \tag{12}$$

The RNN in both layers can be replaced with any other RNN, and the GCN [75] can be replaced with any GNN given minor modifications.

Other integrated DGNN approaches are similar to GCRN-M2. They may differ in which GNN and/or which RNN they use, the target use case or even the kind of graph they are built for, but the structures of the neural architecture are similar. Examples of these include GC-LSTM [20], LRGCN [23], RE-Net [95] and TNA [96].

Chen et al. [20] present GC-LSTM, an encoder very similar to GCRN-M2. GC-LSTM takes the adjacency matrix A_t at a given time as an input to the LSTM and performs a spectral graph convolution [74] on the hidden layer. In contrast, GCRN-M2 runs a convolution on both the input and the hidden layer.

LRGCN [23] integrates an R-GCN [97] into an LSTM as a step towards predicting path failure in dynamic graphs.

RE-Net [95] encodes a dynamic knowledge graph by integrating an R-GCN [97] in several RNNs. Other modelling changes enable them to encode dynamic knowledge graphs, thus extending the use of discrete DGNNs to knowledge graphs.

A temporal neighbourhood aggregation (TNA) layer [96] stacks a GCN, a GRU and a linear layer. Bonner et al. designs an encoder that stacks two TNA layers, to achieve a 2-hop convolution and employs variational sampling for use on link prediction. This architecture is arguably a stacked DGNN, but since the authors define the TNA as one layer, we classify it as an integrated DGNN, despite the layer itself being stacked.

3) Dynamic graph autoencoders and generative models

The Dynamic Graph Embedding model (DynGEM) [98] uses a deep autoencoder to encode snapshots of discrete node-dynamic graphs. Inspired by an autoencoder for static graphs [99] DynGEM makes some modifications to improve computation on dynamic graphs. The main idea is to have the autoencoder initialized with the weights from the previous snapshot. This speeds up computation significantly and makes the embeddings stable (i.e. no major changes from snapshot to snapshot). To handle new nodes the Net2WiderNet and Net2DeeperNet approaches from [100] are used to add width and depth to the encoder and decoder while the embedding layer stays fixed in size. This allows the autoencoder to expand while approximately preserving the function the neural network is computing.

Dyngraph2vec [101] is a continuation of the work done on DynGEM. dyngraph2vec considers the last l snapshots in the encoding and can thus be thought of as a sliding time-window. The adjacency matrices $A^t, \ldots, A^{t+l}$ are used to predict A^{t+l+1}, it is assumed that no new nodes are added. The architecture comes in three variations: (1) dyngraph2vecAE, an autoencoder similar to DynGEM except that it leverages information from the past to make the future prediction; (2) dyngraph2vecRNN, where the encoder and decoder consist of stacked LSTMs; (3) dyngraph2vecAERNN, where the encoder has first a few dense feed-forward layers followed by LSTM layers and the decoder is similar to dyngraph2vecAE, namely a deep feed-forward network.

E-LSTM-D [47] like DynGEM, encode and decode with

dense layers, however, they run an LSTM on the encoded hidden vector to predict the new embeddings. Although trained like an autoencoder, the model aims to perform a dynamic link prediction.

Hajiramezanali et al. [102] introduce two variational autoencoder versions for dynamic graphs: the Variational Graph Recurrent Neural Network (VGRNN) and Semi-implicit VGRNN (SI-VGRNN). They can operate on node-dynamic graphs. Both models use a GCN integrated into an RNN as an encoder (similar to GCRN-M2 [69]) to keep track of the temporal evolution of the graph. VGRNN uses a VGAE [80] on each snapshot that is fed the hidden state of the RGNN h_{t-1}. This is to help the VGAE take into account how the dynamic graph changed in the past. Each node is represented in the latent space and the decoding is done by taking the inner product decoder of the embeddings [80]. By integrating semi-implicit variational inference [103] with VGRNN they create SI-VGRNN. Both models aim to improve dynamic link prediction.

Generative adversarial networks (GAN) [104] have proven to be very successful in the computer vision field [105]. They have subsequently been adapted for dynamic network generation as well. GCN-GAN [106] and DynGraphGAN [107] are two such models. Both models are aimed towards the dynamic link prediction task. The generator is used to generate an adjacency matrix and the discriminator tries to distinguish between the generated and the real adjacency matrix. The aim is to have the generator, generate realistic adjacency matrices which can be used as a prediction for the next time step.

GCN-GAN use a stacked DGNN as a generator and a dense feed-forward networks as a discriminator [106] and DynGraphGAN use a shallow generator and a GCN [75] stacked with a CNN as a discriminator [107].

D. CONTINUOUS DYNAMIC GRAPH NEURAL NETWORKS

Currently, there are three DGNN approaches to continuous modelling. RNN based approaches where node embeddings are maintained by an RNN based architecture, temporal point based (TPP) approaches where temporal point processes are parameterized by a neural network and time embedding approaches where positional embedding of the time is used to represent time as a vector.

1) RNN based models

These models use RNNs to maintain node embeddings in a continuous fashion. A common characteristic for these models is that as soon as an event occurs or there is a change to the network, the embeddings of the interacting nodes are updated. This enables the embeddings to stay up to date continuously. There are two models in this category, Streaming graph neural networks (SGNN) [78] which encode directed strictly evolving networks and JODIE [77] which encodes interaction networks.

The Streaming graph neural network [78] maintains a hidden representation in each node. The architecture consists of two components: (i) an update; and (ii) a propagation component. The update component is responsible for updating the state of the nodes involved in an interaction and the propagation component propagates the update to the involved node's neighbours.

The update and propagation component consist of 3 units each: (i) the interact unit; (ii) the update / propagate unit; and (iii) the merge unit. The difference between the update component and the propagation component is thus the second unit where the update component makes use of the update unit and the propagate component makes use of the propagate unit.

The model maintains several vectors for each node. Among them are: (i) a hidden state for the source role of the node; and (ii) a hidden state of the target role of the node. This is required to treat source and target nodes differently. The model also contains a hidden state which is based on both the source and target state of the node. The interact unit and merge units can be thought of as wrappers that handle many node states. The interact unit generates an encoding based on the interacting nodes and this can be thought of as an encoding of the interaction. The merge unit updates the combined hidden state of the nodes based on the change done to the source and target hidden states by the middle unit.

The middle units and core of the update and propagate components are the update and the propagate units. The update unit generates a new hidden state for the interacting nodes. It is based on a Time-aware LSTM [108], which is a modified LSTM that works on time-series with irregular time intervals. The propagate unit updates the hidden states of the neighbouring nodes. It consists of an attention function f, a time decay function g and a time based filter h. f estimates the importance between nodes, g gauges the magnitude of the update based on how long ago it was and h is a binary function which filters out updates when the receiving node has too old information. h has the effect of removing noise as well as making the computation more efficient.

By first running the update component and afterwards propagating, information of the edge update is added to the hidden states of the local neighbourhood.

The second method is JODIE [77]. JODIE embeds nodes in an interaction network. It is however targeted towards recommender systems and built for user-item interaction networks. The intuition is that with minor modifications this model can work on general interaction networks.

JODIE uses an RNN architecture to maintain the embeddings of each node. With one RNN for users (RNN_u) and one RNN for items (RNN_i), the formula for each RNN is identical except that they use different weights. When an interaction happens between a user and an item, each of the embeddings is updated according to equation 13.

$$u(t) = \sigma\left(W_1^u u\left(\bar{t}\right) + W_2^u i\left(\bar{t}\right) + W_3^u f + W_4^u \Delta_u\right)$$
$$i(t) = \sigma\left(W_1^i i\left(\bar{t}\right) + W_2^i u\left(\bar{t}\right) + W_3^i f + W_4^i \Delta_i\right) \quad (13)$$

where $u(t)$ is the embedding of the interacting user, $i(t)$ the embedding of the interacting item, $u(\bar{t})$ the embedding of the user just before the interaction and similarly $i(\bar{t})$ is the embedding of the item just before the interaction. The superscript on the weights indicates which RNN they are parameters of, so W_1^u is a parameter of RNN_u. f is the feature vector of the interaction and Δ_u is the time since the user interacted with an item and similarly for $Delta_i$.

An additional functionality of JODIE is the projection component of their architecture. It is used to predict the trajectory of the dynamic embeddings. The model predicts the future position of the user or item embedding and is trained to improve this prediction.

2) Temporal point process based models

Know-Evolve [21] is the precursor to the rest of the dynamic graph temporal point process models discussed in this section. It models knowledge graphs in the form of interaction networks by parameterizing a temporal point process (TPP) by a modified RNN. With some minor modifications, the model should be applicable to any interaction network, but since the original model is specifically for knowledge graphs we will rather focus on its successor, DyREP [48].

DyREP uses a temporal point process model which is parameterised by a recurrent architecture [48]. The temporal point process can express both dynamics "of the network" (structural evolution) and "on the network" (node communication). By modelling this co-evolution of both dynamics they achieve a richer representation than most embeddings.

The temporal point process (TPP) is modelled by events (u, v, t, k) where u and v are the interacting nodes, t is the time of the event and $k \in \{0, 1\}$ indicates whether the event is a structural evolution, $k = 0$ (edge added) or a communication $k = 1$.

The conditional intensity function λ describes the probability of an event happening. λ is parameterised by two functions f and g.

$$\lambda_k^{u,v} f_k(g_k^{u,v}(\bar{t})) \quad (14)$$

where $\bar{t}$ is the time just before the current event, g is a weighted concatenation of node embeddings z, $g_k^{u,v}(\bar{t}) = \boldsymbol{\omega}_k^T \cdot [z^u(\bar{t}); z^v(\bar{t})]$. f is a modified softplus, $f_k(x) = \psi_k \log\left(1 + \exp\left(x/\psi_k\right)\right)$, ω_k and ψ_k are four parameters which enable the temporal point process to be modelled on two different time scales.

The TPP is parameterised by an RNN. The RNN incorporates aggregation of local node embeddings, the previous embedding of the given node and an exogenous drive.

$$z^v\left(t_p\right) = \sigma(W^{struct} h_{struct}^u(\bar{t}_p) + W^{rec} z^v(\bar{t}_p^v) W^t(t_p - \bar{t}_p^v)) \quad (15)$$

where h_{struct}^u is given by an attention mechanism that aggregates embeddings of neighbours of u. The attention mechanism uses an attention matrix S which is calculated and maintained by the adjacency matrix A and the intensity function λ. In short, the λ parameterises the attention mechanism used by the RNN which in turn is used to parameterise λ. Thus λ influences the parameterisation of itself.

With λ well parameterised it serves as a model for the dynamic network and its conditional intensity function can be used to predict link appearance and time of link appearance.

Latent dynamic graph (LDG) [109] uses Kipf et al.'s Neural Relational Inference (NRI) model [110] to extend DyREP. The idea is to re-purpose NRI to encode the interactions on the graph, generate a temporal attention matrix which is then used to improve upon self-attention originally used in DyREP.

Graph Hawkes Network (GHN) [111] is another method that parameterizes a TPP through a deep neural architecture. Similarly to Know-Evolve [21], it targets temporal knowledge networks. A part of the architecture, the Graph Hawkes Process, is an adapted continuous-time LSTM for Hawkes processes [112].

3) Time embedding based models

Some continuous models rely on time embedding methods. This includes using positional encoding to represent the time dimension as introduced by Vaswani et al. [89]. An example of a time embedding method is time2vec [113]. This is a positional encoding, similar to the transformer but especially focused on encoding temporal patterns. Another example, is the functional time embedding introduced by Xu et al. [114] which converts learning temporal patterns to the kernel learning problem and learns the kernel function. They apply classical functional analysis to enable functional learning. These time embedding methods are particularly aimed at capturing temporal difference $t_i - t_j$, which is of substantial benefit when modelling interaction networks since it enables them to effectively capture inter-event time.

Temporal Graph Attention (TGAT) [18] was the first continuous DGNN to use a time embedding. The authors use the functional time embedding they introduced separately [114], however when comparing different versions of the embedding they end up using a non-parametric version (Equation 16) which is near identical to time2vec [113].

$$\Phi_d(t, t_1) = \left[\cos\left(\omega_1\left(t - t_1\right) + \varphi_1\right), \ldots, \cos\left(\omega_d\left(t - t_1\right) + \varphi_d\right)\right] \quad (16)$$

Where ω_i and φ_i are learned weights and d is the size of the time embedding.

A TGAT layer concatenates together the node features, edge features (optional) and time features of each neighbouring node as well as the target node. It then applies masked-attention similar to the attention in GAT [88]. For each layer added an additional hop of neighbours is added. The authors

found 2 layers (2 hops) to be optimal, as additional hops exponentially increase run-time.

$$Z(t) = \Big[\tilde{h}_0^{(l-1)}(t) \,\|e_{0,0}(t_0) \,\|\Phi_{d_T}(0), \\ \tilde{h}_1^{(l-1)}(t_1) \,\|e_{0,1}(t_1) \,\|\Phi_{d_T}(t-t_1), \\ \ldots, \\ \tilde{h}_N^{(l-1)}(t_N) \,\|e_{0,N}(t_N) \,\|\Phi_{d_T}(t-t_N)\Big]^\top \tag{17}$$

$Z(t)$ is an entity-temporal feature matrix which include features of nodes, edges and inter-event time. l is the layer. In line with self-attention $Z(t)$ is linearly projected to obtain the 'query', 'key' and 'value'.

$$\begin{aligned} q(t) &= [Z(t)]_0 W_Q \\ K(t) &= [Z(t)]_{1:N} W_K \\ V(t) &= [Z(t)]_{1:N} W_V \end{aligned} \tag{18}$$

$[Z(t)]_0$ is the features of the target node (the node we want to compute the embedding for) and $[Z(t)]_{1:N}$ is the features of its neighbours. TGAT applies its attention to $Z(t)$ to obtain $h(t)$, the hidden representation of the node.

$$h(t) = \text{softmax}(\frac{q(t)K(t)}{\sqrt{d_k}})V(t) \tag{19}$$

Finally, the hidden representation is concatenated with the (static) node embedding of the target node, x_0, and passed to a feed-forward network.

$$\tilde{h}_0^{(l)}(t) = \text{FFN}\left(h(t) \| x_0\right) \tag{20}$$

Temporal Graph Networks (TGN) [19] extends TGAT by adding a memory module. The memory module embeds the history of the node. The memory vector is added to $Z(t)$ in Equation 17.

E. DISCUSSION AND SUMMARY

Deep learning on dynamic graphs is still a new field, however, there are already promising methods that show the capacity to encode dynamic topology. This section has provided a comprehensive and detailed survey of deep learning models for dynamic graph topology encoding.

The encoders are summarised and compared in Table 6. Models are listed based on their encoders and the encoders capacity to model link and node dynamics. Any model which cannot model link deletion or link duration can only model strictly evolving networks or interaction networks (see section II-B).

Table 6 list many models as not supporting link deletion, it is possible to model link deletion by link deletion events and thus an interaction network can model a persistent link disappearing. Any continuous model should also be able to model node deletion by removing the node from node neighbourhood aggregation to effectively delete it. However, while these ways of modelling dynamics have been discussed by earlier works [19], to the best of our knowledge, they have not been implemented in practice.

TABLE 5: DGNN model types and network types. All continuous DGNNs work on specific types of networks, such as directed or knowledge networks, therefore there are no continuous DGNNs for any general purpose dynamic network.

DGNN model type	Network type			
	Interaction	Temporal	Evolving	Strictly evolving
Discrete	Yes	Yes	Yes	Yes
Continuous	Yes	No	No	No

Most methods focus on discrete graphs which enable them to leverage recent advances in graph neural networks. This allows for modelling of diverse graphs, including node-dynamic graphs, dynamic labels on nodes and due to the use of snapshots, temporal networks can also be handled. Continuous models currently exist for strictly growing networks and interaction networks. This leaves many classes of dynamic graphs unexplored. Since continuous models have some inherent advantages over discrete graphs (see section II-F), expanding the repertoire of dynamic network classes for continuous models, is a promising future direction.

All discrete DGNNs use a GNN to model graph topology and a deep time-series model, typically an RNN, to model the time dependency. Two types of architectures can be distinguished: (i) the stacked DGNN and (ii) the integrated DGNN. Different stacked DGNNs only differ in which spatial and temporal layers are used to stack (which GNN they use and which time series layer), while the integrated DGNNs may differ not only by how they model spatial and temporal patterns but also in how they integrate the spatial and temporal modules. Given the same graph, a stacked DGNN would generally have fewer parameters than a typical integrated DGNN (such as GCRN-M2 [69]). Both approaches offer great flexibility in terms of which GNN and RNN can be used. They also are rather flexible in that they can model networks with both appearing and disappearing edges as well as dynamic labels.

Discrete models tend to treat every snapshot as a static graph, thus the complexity of the model is proportional to the size of the graph in each snapshot and the number of snapshots. Whereas a continuous model complexity is generally proportional to the number of changes in the graph. If a discrete approach creates snapshots using time-windows, then it can trade off temporal granularity (and thus theoretically modelling accuracy) for faster computation by using larger time-windows for each snapshot.

Table 6 shows that every continuous DGNN is aimed at a special type of continuous network. This is reflected in Table 5 which shows that there is, as of yet, no continuous DGNN encoder for any general-purpose dynamic network.

So which one should you chose? Converting the dynamic network to an edge-weighted network is a simple, and depending on the application, possibly "good enough" approach. A practitioner only need to come up with some

scheme to weight edges, and then feed that to an optimized implementation of a standard GNN, e.g. GCN [75] or GAT [88]. TDGNN [81] shows a good example of such a scheme by weighting the edges using an exponential distribution, which weights more recent edges higher than old edges.

Another approach which should be considered before trying any large DGNN model is whether applying a static GNN on a discrete representation might yield good enough results. Given the same number of features and layer size, it will train faster and generally be a simpler model.

The choice between discrete and continuous depends on the data and the intended problem. If temporal granularity and performance is not a concern then one of the advanced discrete approaches such as DySAT or EvolveGCN will likely be a great fit for most dynamic network problems. Since they naturally support link deletion, node addition and node deletion, they provide good general-purpose functionality.

The Discrete DGNNs covered in this work all iterate over snapshots to encode, while the continuous DGNNs iterate edge-by-edge. The continuous therefore tend to take longer to train compared to the discrete models. This is especially true if the network is rather dense.

Evolving networks are well served by any discrete approach, however, with the recent dominance of attention architectures [89], we would expect DySAT to do well in a comparative test. EvolveGCN is expected to train fast on an evolving network with little change between snapshots. The discrete methods are also suited for temporal networks given that the length of the time-windows covered by snapshots is well selected.

If node dynamics is an important feature of the network you wish to model, then you should choose a model that can encode node dynamics such as DySAT [17], EvolveGCN [16] or HDGNN [24].

If you have an interaction network with detailed timestamps, then TGAT [18] or TGN [19] are likely good fits. If run-time complexity and time granularity are essential to the dynamic complex network at hand (for example in the case of a temporal network), then non-deep learning methods that are not covered by this survey are recommended. Those methods can be explored in the literature referred to in section II-E.

IV. DEEP LEARNING FOR PREDICTION OF NETWORK TOPOLOGY

Any embedding method can be thought of as a concatenation of an encoder and a decoder [28]. Until now, we have discussed encoders, but the quality of embeddings depend on the decoder and the loss function as well. While the encoders in Section III can be paired with a variety of decoders and loss functions depending on the intended task, we focus in this section on one of the most commonly tackled problems - link prediction.

Prediction problems can be defined for many different contexts and settings. In this survey, we refer to the prediction of the future change to the network topology. Much work has been done on the prediction of missing links in networks, which can be thought of as an interpolation task. This section explores how dynamic graph neural networks can be used for link prediction and deal exclusively with the extrapolation (future link prediction) task.

Predictions can be done in a time-conditioned or time-predicting manner [11]. Time-predicting means that a method predicts when an event will occur and time-conditioned means that a method predicts whether an event will occur at a given time t. For example, if the method predicts the existence of a link in the next snapshot, it is a time-conditioned prediction. If it predicts when a new link between nodes will appear, it is a time-predicting prediction.

Prediction of links often focuses only on the prediction of the appearance of a link. However, link disappearance is less explored but also important for the prediction of network topology. We refer to link prediction based on a dynamic network as dynamic link prediction.

For embedding methods, what is predicted and how is decided by the decoder. You can have both time-predicting and time-conditioned decoders. The prediction capabilities will depend on the information captured by the embeddings. Thus, an embedding that captures continuous-time information has a higher potential to model temporal patterns. Well modelled temporal and structural embeddings offer a better foundation for a decoder and thus potentially better predictions.

If dealing with discrete data and few timestamps, a time-conditioned decoder can be used for time prediction. This can be done by applying the time-conditioned decoder to every candidate timestamp t and then consider the t where the link has the highest probability of appearing.

The rest of this section is a description of how the surveyed models from the previous section can be used to perform predictions. This includes mainly a discussion on decoders and loss functions. Since the surveyed models aim to predict the time-conditioned existence of links, the focus will be on the dynamic link prediction task.

Autoencoders can use the same decoders and loss functions as other methods. Their aim is typically a little different. The decoder is targeted at the already observed network and tries to recreate the snapshot. A prediction for a snapshot at time $t+1$ is marginally different from the decoder of an autoencoder which is targeted at already observed snapshots.

A. DECODERS

Of the surveyed approaches which apply a predicting decoder, almost all apply a time-conditioned decoder. A prediction is then often an adjacency matrix $\hat{A}^{\tau}$ which indicates the probabilities of an edge at time τ. Often $\tau = t+1$.

We consider decoders to be the part of the architecture that produces $\hat{A}^{\tau}$ from Z the dynamic graph embeddings.

Since encoders make node embeddings and predicting a link involves two nodes decoders tend to aggregate two

TABLE 6: Deep encoders for dynamic network topology. While we note which GNNs are used in each of the discrete models it is usually trivial to replace it with another GNN.

Model type	Model name	Encoder	Link addition	Link deletion	Node addition	Node deletion	Network type
Discrete networks							
Stacked DGNN	GCRN-M1 [69]	Spectral GCN [74] & LSTM	Yes	Yes	No	No	Any
	WD-GCN [76]	Spectral GCN [75] & LSTM	Yes	Yes	No	No	Any
	CD-GCN [76]	Spectral GCN [75] & LSTM	Yes	Yes	No	No	Any
	RgCNN [83]	Spatial GCN [84] & LSTM	Yes	Yes	No	No	Any
	DyGGNN [85]	GGNN [86] & LSTM	Yes	Yes	No	No	Any
	DySAT [17]	GAT [88] & temporal attention [89]	Yes	Yes	Yes	Yes	Any
	TNDCN [25], [90]	Spectral GCN [25] & TCN [25]	Yes	Yes	No	No	Any
	StrGNN [93]	Spectral GCN [75] & GRU	Yes	Yes	No	No	Any
	HDGNN [24]	Spectral GCN [75] & A variety of RNNs	Yes	Yes	Yes	Yes	Heterogeneous
	TeMP [22]	R-GCN [97] stacked with either GRU or attention	Yes	Yes	No	No	Knowledge
Integrated DGNN	GCRN-M2 [69]	GCN [74] integrated in an LSTM	Yes	Yes	No	No	Any
	GC-LSTM [20]	GCN [74] integrated in an LSTM	Yes	Yes	No	No	Any
	EvolveGCN [16]	LSTM integrated in a GCN [75]	Yes	Yes	Yes	Yes	Any
	LRGCN [23]	R-GCN [97] integrated in an LSTM	Yes	Yes	No	No	Any
	RE-Net [95]	R-GCN [97] integrated in several RNNs	Yes	Yes	No	No	Knowledge
	TNA [96]	GCN [75] stacked with a GRU and a linear layer	Yes	Yes	No	No	Any
Continuous networks							
RNN based							
	Streaming GNN [78]	Node embeddings maintained by architecture consisting of T-LSTM [108]	Yes	No	Yes	No	Directed strictly evolving
	JODIE [77]	Node embeddings maintained by an RNN based architecture	Yes	No	No	No	Bipartite, interaction
TTP based							
	Know-Evolve [21]	TPP parameterised by an RNN	Yes	No	No	No	Interaction, knowledge network
	DyREP [48]	TPP parameterised by an RNN aided by structural attention	Yes	No	Yes	No	Interaction combined with strictly evolving
	LDG [109]	TPP, RNN and self-attention	Yes	No	Yes	No	Interaction combined with strictly evolving
	GHN [111]	TPP parameterised by a continuous time LSTM [112]	Yes	No	No	No	Interaction, knowledge network
Time embedding based							
	TGAT [18]	Temporal [113] and structural [88] attention	Yes	No	Yes	No	Directed or undirected interaction
	TGN [19]	Temporal [113] and structural [88] attention with memory	Yes	No	Yes	No	Directed or undirected interaction

node embeddings to predict a link. The simplest way to aggregate is to apply an operator, e.g. the inner product [80] (shown in Equation 21), concatenation, mean or Hadamard product [81]. This combines the node embeddings and gives a probability of a link appearing. These simple approaches require that the encoder is able to embed the nodes in a space such that nodes that are likely to connect are close to each other or otherwise able to be decoded by the simple decoder.

Another simple decoder is to use a simple feed-forward network. The decoder as before receives two node embeddings and gives out a probability for whether the link appeared or didn't appear. This approach is used by several models for link prediction [16], [47], [101]. While this requires more parameters, the decoder is can easily be dwarfed in size by the encoder and it enables decoding of non-linear relationships between node embeddings.

$$p\left(A_{ij}^t = 1 | z_i^t, z_j^t\right) = \sigma\left((z_i^t)^\top z_j^t\right) \tag{21}$$

Where z_k is the node embedding of node k. An inner product decoder works well if we only want to predict or reproduce the graph topology. If we would like to decode the feature matrix then a neural network should be used [102].

Wu et al. [115] uses GraphRNN, a deep sequential generative model as a decoder [43]. What is unique with GraphRNN is that it reframes the graph generation problem as a sequential problem. The GraphRNN authors claim increased performance over feed-forward auto-encoders.

In general, there are many options for how decoding can be done. A decoder might be viable as long as the probability for each edge is produced from the latent variables and the architecture can be efficiently optimized with back-propagation.

The only surveyed method using a time-predicting decoder is DyRep [48]. DyRep uses the conditional intensity function of its temporal point process to model the dynamic network.

While the focus in this section is on decoders that are used directly for the forecasting task, it is important to note that downstream learning can also be used. This is the DGNN trained on a task and the node embeddings are used for a different task. For example, the DGNN can be trained on node classification and then the node embeddings are used later for link prediction. An example of this is seen in [17], where a logistic regression classifier is trained on the node embeddings of snapshot t to predict links at $t + 1$.

B. LOSS FUNCTIONS

The loss function is central to any deep learning method, as it is the equation that is being optimized. Regarding loss functions, we can make a distinction between (i) link prediction optimizing methods; and (ii) autoencoder methods. As the prediction methods optimize towards link prediction directly, an autoencoder optimizes towards the recreation of the dynamic graph. Despite have slightly different aims, both approaches have been used for link prediction and have been shown to perform well.

1) Link prediction

Prediction of edges is seen as a binary classification task. Traditional link prediction is well known for being extremely unbalanced [52], [116]. For predicting methods the loss function is often simply the binary cross-entropy [16], [17], [85].

Some models use negative sampling [16], [17]. This transforms the problem of link prediction from a multiple output classification (a prediction for each link) to a binary classification problem (is the link a "good" link or a "bad" link). This speeds up computation and deals with the well-known class imbalance problem in link prediction. The rate of negative samples used vary from work to work, EvolveGCN [16] use 1 to 100 for training, while TGAT [18] and TGN [19] use 1 to 1.

$$\mathcal{L}_{CE} = \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}^{t}\log(\hat{A}_{ij}^{t}) \tag{22}$$

Equation 22 is an example of a binary cross entropy loss adapted from [20].

DySAT [17] sums the loss function only over nodes that are in the same neighbourhood at time t. The neighbourhoods are extracted by taking nodes that co-occur in random walks on the graph. The inner product is calculated as a part of the summation in the loss function. This means that the inner product will be calculated only for the node pairs that the loss is computed on. Together it reduces the number of nodes that are summed up and should result in a training speed up. Any accuracy trade-off is not discussed by the authors.

2) Autoencoders

Autoencoder approaches [47], [98], [101] aim to reconstruct the dynamic network. All surveyed autoencoders operate on discrete networks. Therefore the reconstruction of the network is reduced to the reconstruction of each snapshot. This entails creating a loss function that penalizes wrong reconstruction of the input graph. Variational autoencoder approaches [79], [102] also aim to be generative models. To be generative, they need to enable interpolation in latent space. This is achieved by adding a term to the loss function which penalizes the learned latent variable distribution for being different from a normal distribution. It is also common to add regularization to the loss functions to avoid overfitting.

$$\mathcal{L} = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(A_{ij}^{t} - \hat{A}_{ij}^{t}\right) * P_{ij} \tag{23}$$

Equation 23 is the reconstruction penalizing component of E-LSTM-D's loss function [47]. P is a matrix which increases the focus on existing links. $p_{ij} = 1$ if $A_{ij}^{t} = 0$ and $p_{ij} = \beta > 1$ if $A_{ij}^{t} = 1$.

3) Temporal Point Processes

DyRep [48] models a dynamic network by parameterising a temporal point process. Its loss function influences how the temporal point process is optimized.

$$\mathcal{L} = -\sum_{p=1}^{P}\log\left(\lambda_p(t)\right) + \int_{0}^{T}\Lambda(\tau)d\tau \tag{24}$$

where P is the set of observed events, λ is the intensity function and $\Lambda(\tau) = \sum_{u=1}^{n}\sum_{v=1}^{n}\sum_{k\in\{0,1\}}\lambda_k^{u,v}(\tau)$ is the survival probability for all events that did not happen. Survival probability indicates the probability of an event not happening [117]. The first term thus rewards a high intensity when an event happens, whereas the second term rewards a low intensity (high survival) of events that do not happen.

Trivedi et al. [48] further identify that calculating the integral of Λ is intractable. They get around that by sampling non-events and estimating the integral using Monte Carlo estimation, this is done for each mini-batch.

4) Regularization

There are several different approaches for adding regularization to loss functions to avoid overfitting. The total loss function (equation 25) is composed of the reconstruction loss and the regularization with an optional constant α to balance the terms. Here we cover the methods that use regularization, however many models chose to not use regularization as they find that they don't have a problem with overfitting [16], [18], [19], [48].

$$\mathcal{L}_{\text{total}} = \mathcal{L} + \alpha\mathcal{L}_{reg} \tag{25}$$

A common way to regularize is through summing up all the weights of the model, thus keeping the weights small and the model less likely to overfit. The L_2 norm is commonly used for this [20], [47].

The variational autoencoder methods use a different regularizer. They normalize the node embeddings compared to a prior. In traditional variational autoencoders, this prior is a Normal distribution with mean 0 and standard deviation 1. In dynamic graph autoencoders [79], [102], the prior is still a Gaussian, but it is parameterised by previous observations. Equation 26 is the regularization term from [102].

$$\begin{aligned}&\mathbf{KL}(q\left(Z^{t}|A^{\leq t}, X^{\leq t}, Z^{<t}\right)\\ &\quad\|p\left(Z^{t}|A^{<t}, X^{<t}, Z^{<t}\right))\end{aligned} \tag{26}$$

where q is the encoder distribution and p is the prior distribution. $\mathbf{KL}$ is the Kullback-Leibler divergence which measures the difference between two distributions. The $A^{<t}$ indicate all adjacency matrices up to, but not including t and similarly for the other matrices. We can see that the prior is influenced by previous snapshots, but not by the current. Whereas the encoder is influenced by the previous and the current snapshot.

C. EVALUATION METRICS

Link prediction is plagued by high class imbalance. It is a binary classification, a link either exists or not and most links will not exist. In fact, actual links tend to constitute less than 1% of all possible links [118]. AUC and precision@k are two commonly used evaluation metrics in static link prediction [116], [119]. If dynamic link prediction requires the prediction of both appearing and disappearing edges, the evaluation metric needs to reflect that. Furthermore,

traditional link prediction metrics have shortcomings when used in a dynamic setting [52].

For a detailed discussion on the evaluation of link prediction, we refer to Yang et al. [116] for static link prediction and Junuthula et al. [52] for dynamic link prediction evaluation.

1) **Area under the curve (AUC).** The area under the curve (AUC) is used to evaluate a binary classification and has the advantage of being independent of the classification threshold. The AUC is the area under the receiver operating characteristic (ROC) curve. The ROC is a plot of the true positive rate and the false positive rate.
   The AUC evaluates predictions based on how well the classifier ranks the predictions, this provides a measure that is invariant of the classification threshold. In link prediction, there has been little research into finding the optimal threshold [120], using the AUC for evaluation avoids this problem.
   Yang et al. [116] note that AUC can show deceptively high performance in link prediction due to the extreme class imbalance. They recommend the use of PRAUC instead.
2) **PRAUC.** The PRAUC is similar to the AUC except that it is the area under the precision-recall curve. The metric is often used in highly imbalanced information retrieval problems [52].
   PRAUC is recommended by Yang et al. [116] as a suitable metric for traditional (static) link prediction due to the deceptive nature of the ROC curve and because PRAUC shows a more discriminative view of classification performance. And recommended for the same reasons by Li et al. for dynamic link prediction [118].
   One way of calculating the PRAUC is by using Mean Average Precision (MAP). MAP is the mean of the average precision (AP) per node.
3) **Fixed-threshold metrics.** One of the most common fixed threshold metrics in traditional link prediction is Precision@k. It is the ratio of items that are correctly predicted. From the ranking prediction the top k predictions are selected, then precision is the ratio $\frac{k_r}{k}$, where k_r is the number of correctly predicted links in the top k predictions.
   While a higher precision indicates a higher prediction accuracy, it is dependent on the parameter k. k might be given on web-scale information retrieval, where we care about the accuracy of the highest k ranked articles, in link prediction it is difficult to find the right cut-off [120].
   A fixed-threshold can be applied to other common metrics including accuracy, recall and F1 among others [116]. These methods suffer from instability in their predictions, where a change of thresholds can lead to contradictory results [116]. This problem is also observed in dynamic link prediction [52]. Fixed-threshold metrics are not recommended unless the targeted problem has a natural threshold [116].
4) **Sum of absolute differences (SumD).** Li et al. [118] pointed out that models often have similar AUC scores and suggested SumD as a stricter measurement of accuracy. It is simply, the number of mispredicted links. The metric has different meanings depending on how many values are predicted since it is not normalized according to the total number of links. Chen et al. considers SumD misleading for this reason [47]. The metric strictly punishes false positives, since there are so many links not appearing, a slightly higher rate of false positives will have a large impact on this metric.
5) **Error rate.** Since SumD suffers from several drawbacks an extension is suggested by Chen et al. [47]. Error rate normalizes SumD by the total number of existing links.

$$\text{Error Rate} = \frac{N_{\text{false}}}{N_{\text{true}}} \tag{27}$$

   where N_{false} is the number of mispredicted links and N_{true} is the number of existing links. The error rate is very similar to recall, except that recall focuses on true positives, where the error rate focuses on false positives. Another difference between recall and error rate is that recall is normalized between 0 and 1, while the error rate may be above 1 if the number of mispredicted links outnumber the number of existing links The error rate is a good metric if the number of false positives is a major concern. In dynamic link prediction, false positives become a major issue due to the massive class imbalance of the prediction problem.
6) **GMAUC.** After a thorough investigation of evaluation metrics for dynamic link prediction, Junuthula et al. suggests GMAUC as an improvement over other metrics [52]. The key insight is that dynamic link prediction can be divided into two sub-problems: (i) predicting the disappearance of links that already exist or the appearance of links that have once existed; and (ii) predicting links that have never been seen before. When the problem is divided in this way, each of the sub-problems takes on different characteristics.
   Prediction of links that have never been seen before is equivalent to traditional link prediction, for which PRAUC is a suitable metric [116]. Prediction of already existing links is both the prediction of once seen links appearing and existing links disappearing. This is a more balanced problem than traditional link prediction, thus AUC is a suitable measure. [52] note that both the mean and the harmonic mean will lead to either the AUC or the PRAUC to dominate, thus the geometric mean is used to form a unified metric.

$$\text{GMAUC} = \sqrt{\frac{\text{PRAUC}_{\text{new}} - \frac{P}{P+N}}{1 - \frac{P}{P+N}} \cdot 2\left(\text{AUC}_{\text{prev}} - 0.5\right)} \tag{28}$$

$PRAUC_{new}$ is the PRAUC score of new links, AUC_{prev} is the AUC score of previously observed links.

The authors note the advantages of GMAUC:

- Based on threshold curves, thus avoids the pitfall of fixed-threshold metrics
- Accounts for differences between predicting new and already observed edges without having the metric to be dominated by either sub-problem.
- Any predictor that predicts only new edges or only previously observed edges gets a score of 0.

However, it does hinge on the assumption that reoccurring edges is a balanced enough prediction problem that AUC is suitable. And that is not necessarily the case. Many real-world networks are much more sparse than the two networks used by Junuthula et al. [52].

D. DISCUSSION AND SUMMARY

In this section we have provided an overview of how, given a dynamic network encoder, one can perform network topology prediction. The overview includes how methods from section III use their embeddings for prediction. This completes the journey from establishing a dynamic network, to encoding the dynamic topology, to predicting changes in the topology.

Prediction using a deep model requires decoding and the use of a loss function that captures temporal and structural information. Prediction is largely focused on time-conditioned link prediction and the two main modelling approaches are (1) an architecture directly aimed at prediction; and (2) an architecture aimed at generating node embeddings which are then used for link prediction in a downstream step. Most dynamic network models surveyed fall into the second category, including all autoencoder approaches. All else being equal we would expect an architecture directly aimed at prediction to perform better than a two step architecture. This is because the first case will allow the entire architecture to optimize itself towards the prediction task.

The massive class imbalance makes the evaluation of dynamic link prediction is non-trivial. If the target problem has a natural fixed threshold, then adding a fixed threshold to a common metric such as F1 is likely a good fit. PRAUC (MAP) and Error rate are good metrics that avoids the class imbalance problem and are suitable for both link prediction and dynamic link prediction. The GMAUC metric incorporates the observation that reappearing and disappearing links are not an imbalanced classification. Usage of GMAUC however hinges on the assumption that reoccurring links are a reasonably balanced classification, this is not necessarily true and depends on the data. An evaluation of new methods should report the PRAUC of newly appearing links and the PRAUC or AUC of reappearing links separately. The combined score should also be reported as either the PRAUC, Error rate or GMAUC.

Prediction on dynamic networks is in its infancy. Deep models are largely focused on unattributed time-conditioned discrete link appearance prediction. This leaves opportunities for future work in a large range of prediction tasks, with some types of prediction still unexplored. Prediction based on continuous-time encoders is a particularly interesting frontier due to the representations inherent advantages and due to the limited amount of works in that area.

V. CHALLENGES AND FUTURE WORK

There are plenty of challenges and multiple avenues for the improvement of deep learning for both modelling and prediction of network topology.

Expanding modelling and prediction repertoire. In this work we have exclusively focused on dynamic network topology. However, complex networks are diverse and not only topology may vary. Topology dynamics can be represented as a 3-dimensional cube (Section II-D). However, real networks can be much more complex. Complex networks may have dynamic node and edge attributes, they may have directed and/or signed edges, be heterogeneous in terms of nodes and edges and be multilayered or multiplex. Each of these cases can be considered another dimension in the dynamic network hypercube. Designing deep learning models for encoding these network cases expand the repertoire of tasks on which deep learning can be applied. Which types of networks can be encoded can be expanded as well as an expansion of what kind of predictions can be made on those networks. For example, most DGNN models (and most GNN models) encode attributed dynamic networks but predict only graph topology without the node attributes.

Adoption of advances in closely related fields. Dynamic graph neural networks are based on GNNs and thus advances made to GNNs trickle down and can improve DGNNs. Challenges for GNNs include increasing modelling depth as GNNs struggle with vanishing gradients [121] and increasing scalability for large graphs [67]. As advancements are made in deep neural networks for time series and in GNNs these advancements can be applied to dynamic network modelling and prediction to improve performance. Similarly, improvements in deep time-series modelling can easily be adapted to improve DGNNs.

Continuous DGNNs. Modelling temporal patterns is what distinguishes modelling dynamic graphs from modelling static graphs. Capturing these temporal patterns is key to making accurate predictions. However, most models rely on snapshots which are coarse-grained temporal representations. Methods modelling network change in continuous time will offer fine-grained temporal modelling. Future work

is needed for modelling and prediction of continuous-time dynamic networks.

Scalability. Large scale datasets is a challenge for dynamic network modelling. Real-world datasets tend to be so large that modelling becomes prohibitively slow. Dynamic networks either use a discrete representation in the form of snapshots, in which case processing of each snapshot is the bottleneck or continuous-time modelling which scales with the number of interactions. A snapshot model will need to have frequent snapshots in order to achieve high temporal granularity. In addition, frequent snapshots might undermine the capacity to model a temporal network. Improvements in continuous-time modelling are likely to improve the performance of deep learning modelling on dynamic networks both in terms of temporal modelling capacity and ability to handle large networks.

Dynamic graph neural networks is a new exciting research direction with a broad area of applications. With these opportunities, the field is ripe with potential for future work.

REFERENCES

[1] Steven H. Strogatz. Exploring complex networks. Nature, 410(6825):268, March 2001.

[2] Petter Holme and Jari Saramäki. Temporal networks. Physics reports, 519(3):97–125, 2012.

[3] Charu Aggarwal and Karthik Subbian. Evolutionary network analysis: A survey. ACM Computing Surveys (CSUR), 47(1):10, 2014.

[4] Taisong Li, Bing Wang, Yasong Jiang, Yan Zhang, and Yonghong Yan. Restricted Boltzmann Machine-Based Approaches for Link Prediction in Dynamic Networks. IEEE Access, 6:29940–29951, 2018.

[5] Othon Michail and Paul G Spirakis. Elements of the theory of dynamic networks. Communications of the ACM, 61(2):72–81, 2018.

[6] Petter Holme. Modern temporal network theory: A colloquium. The European Physical Journal B, 88(9):234, September 2015.

[7] Kathleen M Carley, Jana Diesner, Jeffrey Reminga, and Maksim Tsvetovat. Toward an interoperable dynamic network analysis toolkit. Decision Support Systems, 43(4):1324–1347, 2007.

[8] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph Neural Networks: A Review of Methods and Applications. arXiv:1812.08434 [cs, stat], December 2018.

[9] Naoki Masuda and Renaud Lambiotte. A Guide to Temporal Networks. World Scientific, 2016.

[10] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-Varying Graphs and Dynamic Networks. In Hannes Frey, Xu Li, and Stefan Ruehrup, editors, Ad-Hoc, Mobile, and Wireless Networks, pages 346–359, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[11] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Relational Representation Learning for Dynamic (Knowledge) Graphs: A Survey. arXiv:1905.11485 [cs, stat], May 2019.

[12] Yu Xie, Chunyi Li, Bin Yu, Chen Zhang, and Zhouhua Tang. A survey on dynamic network embedding. arXiv preprint arXiv:2006.08093, 2020.

[13] Claudio DT Barros, Matheus RF Mendonça, Alex B Vieira, and Artur Ziviani. A survey on embedding dynamic graphs. arXiv preprint arXiv:2101.01229, 2021.

[14] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In Proceedings of the ieee conference on computer vision and pattern recognition, pages 5308–5317, 2016.

[15] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875, 2017.

[16] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, and Charles E. Leisersen. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. CoRR, abs/1902.10191, 2019.

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

[18] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. arXiv preprint arXiv:2002.07962, 2020.

[19] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. arXiv preprint arXiv:2006.10637, 2020. tex.ids= rossiTemporalGraphNetworks2020.

[20] Jinyin Chen, Xuanheng Xu, Yangyang Wu, and Haibin Zheng. GC-LSTM: graph convolution embedded LSTM for dynamic link prediction. CoRR, abs/1812.04206, 2018.

[21] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In Doina Precup and Yee Whye Teh, editors, Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of Proceedings of Machine Learning Research, pages 3462–3471. PMLR, 2017.

[22] Jiapeng Wu, Meng Cao, Jackie Chi Kit Cheung, and William L Hamilton. TeMP: Temporal message passing for temporal knowledge graph completion. arXiv preprint arXiv:2010.03526, 2020.

[23] Jia Li, Zhichao Han, Hong Cheng, Jiao Su, Pengyun Wang, Jianfeng Zhang, and Lujia Pan. Predicting Path Failure In Time-Evolving Graphs. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, pages 1279–1289, New York, NY, USA, 2019. ACM.

[24] Fan Zhou, Xovee Xu, Ce Li, Goce Trajcevski, Ting Zhong, and Kunpeng Zhang. A heterogeneous dynamical graph neural networks approach to quantify scientific impact. arXiv preprint arXiv:2003.12042, 2020.

[25] Yanbang Wang, Pan Li, Chongyang Bai, VS Subrahmanian, and Jure Leskovec. Generic representation learning for dynamic social interaction. In The 26th ACM SIGKDD international conference on knowledge discovery & data mining, MLG workshop, 2020.

[26] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep Learning on Graphs: A Survey. arXiv:1812.04202 [cs, stat], December 2018.

[27] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, pages 1–21, 2020.

[28] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. arXiv preprint arXiv:1709.05584, 2017.

[29] Palash Goyal and Emilio Ferrara. Graph Embedding Techniques, Applications, and Performance: A Survey. Knowledge-Based Systems, 151:78–94, July 2018.

[30] Giulio Rossetti and Rémy Cazabet. Community Discovery in Dynamic Networks: A Survey. ACM Computing Surveys, 51(2):1–37, February 2018.

[31] Bomin Kim, Kevin H Lee, Lingzhou Xue, and Xiaoyue Niu. A review of dynamic network models with latent variables. Statistics surveys, 12:105—135, 2018.

[32] Jian Zhang. A survey on streaming algorithms for massive graphs. In Managing and Mining Graph Data, pages 393–420. Springer, 2010.

[33] Cornelius Fritz, Michael Lebacher, and Göran Kauermann. Tempus volat, hora fugit: A survey of tie-oriented dynamic network models in discrete and continuous time. Statistica Neerlandica, 2019.

[34] Andrew McGregor. Graph stream algorithms: A survey. ACM SIGMOD Record, 43(1):9–20, 2014.

[35] Saoussen Aouay, Salma Jamoussi, Faïez Gargouri, and Ajith Abraham. Modeling dynamics of social networks: A survey. In CASoN, pages 49–54. IEEE, 2014.

[36] Giulio Rossetti. Social Network Dynamics. 2015.

[37] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining Stream Statistics over Sliding Windows. SIAM Journal on Computing, 31(6):1794–1813, January 2002.

[38] Matteo Morini, Patrick Flandrin, Eric Fleury, Tommaso Venturini, and Pablo Jensen. Revealing evolutions in dynamical networks. July 2017.

[39] S. Boccaletti, G. Bianconi, R. Criado, C. I. del Genio, J. Gómez-Gardeñes, M. Romance, I. Sendiña-Nadal, Z. Wang, and M. Zanin. The structure and dynamics of multilayer networks. Physics Reports, 544(1):1–122, November 2014.

[40] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. ACM Transactions on Knowledge Discovery from Data (TKDD), 5(2):10, 2011.

[41] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. science, 286(5439):509–512, 1999.

[42] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pages 177–187. ACM, 2005.

[43] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. GraphRNN: A Deep Generative Model for Graphs. CoRR, abs/1802.08773, 2018.

[44] Mark Newman. Networks. Oxford university press, 2018.

[45] Albert-Laszlo Barabâsi, Hawoong Jeong, Zoltan Néda, Erzsebet Ravasz, Andras Schubert, and Tamas Vicsek. Evolution of the social network of scientific collaborations. Physica A: Statistical mechanics and its applications, 311(3-4):590–614, 2002.

[46] Kevin Xu. Stochastic block transition models for dynamic networks. In Artificial Intelligence and Statistics, pages 1079–1087, 2015.

[47] Jinyin Chen, Jian Zhang, Xuanheng Xu, Chengbo Fu, Dan Zhang, Qingpeng Zhang, and Qi Xuan. E-LSTM-D: A Deep Learning Framework for Dynamic Network Link Prediction. arXiv preprint arXiv:1902.08329, 2019.

[48] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. DyRep: Learning representations over dynamic graphs. In International Conference on Learning Representations, 2019.

[49] Kevin S. Xu and Alfred O. Hero III. Dynamic stochastic blockmodels for time-evolving social networks. IEEE Journal of Selected Topics in Signal Processing, 8(4):552–562, August 2014.

[50] Akanda Wahid-Ul Ashraf, Marcin Budka, and Katarzyna Musial. Simulation and Augmentation of Social Networks for Building Deep Learning Models. May 2019.

[51] Nicola Perra, Bruno Gonçalves, Romualdo Pastor-Satorras, and Alessandro Vespignani. Activity driven modeling of time varying networks. Scientific reports, 2(1):1–7, 2012. Publisher: Nature Publishing Group.

[52] Ruthwik R Junuthula, Kevin S Xu, and Vijay K Devabhaktuni. Evaluating link prediction accuracy in dynamic networks with added and removed edges. In 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom), pages 377–384. IEEE, 2016.

[53] Tom A. B. Snijders, Gerhard G. van de Bunt, and Christian E. G. Steglich. Introduction to stochastic actor-based models for network dynamics. Social Networks, 32(1):44–60, January 2010.

[54] Aswathy Divakaran and Anuraj Mohan. Temporal link prediction: A survey. New Generation Computing, pages 1–46, 2019.

[55] Xiaoyi Li, Nan Du, Hui Li, Kang Li, Jing Gao, and Aidong Zhang. A deep learning approach to link prediction in dynamic networks. In Proceedings of the 2014 SIAM International Conference on Data Mining, pages 289–297. SIAM, 2014.

[56] Feng Liu, Bingquan Liu, Chengjie Sun, Ming Liu, and Xiaolong Wang. Deep learning approaches for link prediction in social network services. In International conference on neural information processing, pages 425–432, 2013. tex.organization: Springer.

[57] Carter T. Butts and Christopher Steven Marcum. A relational event approach to modeling behavioral dynamics. In Andrew Pilny and Marshall Scott Poole, editors, Group processes: Data-driven computational approaches, pages 51–92. Springer International Publishing, Cham, 2017.

[58] Steve Hanneke, Wenjie Fu, and Eric P. Xing. Discrete temporal models of social networks. Electronic Journal of Statistics, 4:585–605, 2010.

[59] Per Block, Johan Koskinen, James Hollway, Christian Steglich, and Christoph Stadtfeld. Change we can believe in: Comparing longitudinal network models on consistency, interpretability and predictive power. Social Networks, 52:180–191, January 2018.

[60] Anna Goldenberg, Alice X. Zheng, Stephen E. Fienberg, and Edoardo M. Airoldi. A Survey of Statistical Network Models. Foundations and Trends® in Machine Learning, 2(2):129–233, February 2010.

[61] 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.

[62] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In Advances in Neural Information Processing Systems, pages 2224–2232, 2015.

[63] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 974–983. ACM, 2018.

[64] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In Advances in Neural Information Processing Systems, pages 3697–3707, 2017.

[65] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. DeepInf: Social Influence Prediction with Deep Learning. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '18, pages 2110–2119, 2018.

[66] Yozen Liu, Xiaolin Shi, Lucas Pierce, and Xiang Ren. Characterizing and forecasting user engagement with in-app action graph: A case study of snapchat. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 2023–2031, 2019.

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

[68] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: A review. Data Mining and Knowledge Discovery, 33(4):917–963, July 2019.

[69] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured Sequence Modeling with Graph Convolutional Recurrent Networks. In Long Cheng, Andrew Chi Sing Leung, and Seiichi Ozawa, editors, Neural Information Processing, Lecture Notes in Computer Science, pages 362–373. Springer International Publishing, 2018.

[70] Purnamrita Sarkar and Andrew Moore. Dynamic social network analysis using latent space models. In Y. Weiss, B. Schölkopf, and J. Platt, editors, Advances in neural information processing systems, volume 18. MIT Press, 2006.

[71] Tianbao Yang, Yun Chi, Shenghuo Zhu, Yihong Gong, and Rong Jin. Detecting communities and their evolutions in dynamic social networks—a Bayesian approach. Machine learning, 82(2):157–189, 2011. Publisher: Springer.

[72] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. IEEE transactions on neural networks, 20(1):61–80, 2008. Publisher: IEEE.

[73] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online Learning of Social Representations. CoRR, abs/1403.6652, 2014.

[74] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, Advances in Neural Information Processing Systems 29, pages 3844–3852. Curran Associates, Inc., 2016.

[75] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017.

[76] Franco Manessi, Alessandro Rozza, and Mario Manzo. Dynamic graph convolutional networks. Pattern Recognition, 97:107000, January 2020.

[77] 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 & Data Mining, pages 1269–1278, 2019.

[78] Yao Ma, Ziyi Guo, Zhaochun Ren, Eric Zhao, Jiliang Tang, and Dawei Yin. Streaming Graph Neural Networks. arXiv:1810.10627 [cs, stat], October 2018.

[79] Da Xu, Chuanwei Ruan, Kamiya Motwani, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Generative Graph Convolutional Network for Growing Graphs. ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3167–3171, May 2019.

[80] Thomas N. Kipf and Max Welling. Variational graph auto-encoders. CoRR, abs/1611.07308, 2016.

[81] Liang Qu, Huaisheng Zhu, Qiqi Duan, and Yuhui Shi. Continuous-time link prediction via temporal dependent graph neural network. In Proceedings of the web conference 2020, WWW '20, pages 3026–3032, New York, NY, USA, 2020. Association for Computing Machinery. Number of pages: 7 Place: Taipei, Taiwan.

[82] Felix A Gers, Nicol N Schraudolph, and Jurgen Schmidhuber. Learning Precise Timing with LSTM Recurrent Networks. page 29.

[83] Apurva Narayan and Peter H. O'N Roe. Learning Graph Dynamics using Deep Neural Networks. IFAC-PapersOnLine, 51(2):433–438, January 2018.

[84] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, volume 48 of JMLR Workshop and Conference Proceedings, pages 2014–2023. JMLR.org, 2016.

[85] Aynaz Taheri, Kevin Gimpel, and Tanya Berger-Wolf. Learning to Represent the Evolution of Dynamic Graphs with Recurrent Models. In Companion Proceedings of The 2019 World Wide Web Conference, WWW '19, pages 301–307, New York, NY, USA, 2019. ACM.

[86] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. In Yoshua Bengio and Yann LeCun, editors, 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, 2016.

[87] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. Neural Comput., 9(8):1735–1780, November 1997.

[88] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018.

[89] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 5998–6008. Curran Associates, Inc., 2017.

[90] Yanbang Wang, Pan Li, Chongyang Bai, and Jure Leskovec. TEDIC: Neural modeling of behavioral patterns in dynamic social interaction networks. 2020.

[91] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. In Arxiv, 2016.

[92] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In Advances in neural information processing systems, pages 5165–5175, 2018.

[93] Lei Cai, Zhengzhang Chen, Chen Luo, Jiaping Gui, Jingchao Ni, Ding Li, and Haifeng Chen. Structural temporal graph neural networks for anomaly detection in dynamic graphs. arXiv preprint arXiv:2005.07427, 2020.

[94] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pages 802–810, 2015.

[95] Woojeong Jin, He Jiang, Meng Qu, Tong Chen, Changlin Zhang, Pedro Szekely, and Xiang Ren. Recurrent event network: Global structure inference over temporal knowledge graph. arXiv: 1904.05530, 2019.

[96] Stephen Bonner, Amir Atapour-Abarghouei, Philip T Jackson, John Brennan, Ibad Kureshi, Georgios Theodoropoulos, Andrew Stephen McGough, and Boguslaw Obara. Temporal neighbourhood aggregation: Predicting future links in temporal graphs via recurrent variational graph convolutions. In 2019 IEEE international conference on big data (big data), pages 5336–5345, 2019. tex.organization: IEEE.

[97] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, The Semantic Web, Lecture Notes in Computer Science, pages 593–607. Springer International Publishing, 2018.

[98] Palash Goyal, Sujit Rokka Chhetri, Ninareh Mehrabi, Emilio Ferrara, and Arquimedes Canedo. DynamicGEM: A library for dynamic graph embedding methods. arXiv preprint arXiv:1811.10734, 2018.

[99] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural Deep Network Embedding. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, pages 1225–1234, New York, NY, USA, 2016. ACM.

[100] Tianqi Chen, Ian J. Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. In Yoshua Bengio and Yann LeCun, editors, 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, 2016.

[101] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. Dyngraph2vec: Capturing Network Dynamics using Dynamic Graph Representation Learning. Knowledge-Based Systems, page S0950705119302916, July 2019.

[102] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna R. Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. Variational graph recurrent neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada, pages 10700–10710, 2019.

[103] Mingzhang Yin and Mingyuan Zhou. Semi-Implicit Variational Inference. In International Conference on Machine Learning, pages 5660–5669, July 2018.

[104] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 2672–2680. Curran Associates, Inc., 2014.

[105] Zhengwei Wang, Qi She, and Tomas E Ward. Generative adversarial networks: A survey and taxonomy. arXiv preprint arXiv:1906.01529, 2019.

[106] Kai Lei, Meng Qin, Bo Bai, Gong Zhang, and Min Yang. Gcn-gan: A non-linear temporal link prediction model for weighted dynamic networks. In IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pages 388–396. IEEE, 2019.

[107] Yun Xiong, Yao Zhang, Hanjie Fu, Wei Wang, Yangyong Zhu, and S Yu Philip. Dyngraphgan: Dynamic graph embedding via generative adversarial networks. In International Conference on Database Systems for Advanced Applications, pages 536–552. Springer, 2019.

[108] Inci M. Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K. Jain, and Jiayu Zhou. Patient Subtyping via Time-Aware LSTM Networks. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17, pages 65–74, Halifax, NS, Canada, 2017. ACM Press.

[109] Boris Knyazev, Carolyn Augusta, and Graham W Taylor. Learning temporal attention in dynamic graphs with bilinear interactions. arXiv preprint arXiv:1909.10367, 2019.

[110] Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. In Jennifer G. Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research, pages 2693–2702. PMLR, 2018.

[111] Zhen Han, Yuyi Wang, Yunpu Ma, Stephan Günnemann, and Volker Tresp. The graph hawkes network for reasoning on temporal knowledge graphs. arXiv preprint arXiv:2003.13432, 2020.

[112] Hongyuan Mei and Jason M Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In Advances in neural information processing systems, pages 6754–6764, 2017.

[113] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time. arXiv preprint arXiv:1907.05321, 2019.

[114] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Self-attention with functional time representation learning. In

H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché Buc, E. Fox, and R. Garnett, editors, Advances in neural information processing systems, volume 32. Curran Associates, Inc., 2019.
[115] Changmin Wu, Giannis Nikolentzos, and Michalis Vazirgiannis. EvoNet: A neural network for predicting the evolution of dynamic graphs. arXiv preprint arXiv:2003.00842, 2020.
[116] Yang Yang, Ryan N. Lichtenwalter, and Nitesh V. Chawla. Evaluating link prediction methods. Knowl. Inf. Syst., 45(3):751–782, 2015.
[117] Odd Aalen, Ornulf Borgan, and Hakon Gjessing. Survival and Event History Analysis: A Process Point of View. Springer Science & Business Media, 2008.
[118] Taisong Li, Jiawei Zhang, S Yu Philip, Yan Zhang, and Yonghong Yan. Deep dynamic network embedding for link prediction. IEEE Access, 6:29219–29230, 2018.
[119] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. Physica A, 390(6):11501170, 2011.
[120] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A Survey of Link Prediction in Complex Networks. ACM Comput. Surv., 49(4):69:1–69:33, December 2016.
[121] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, pages 9266–9275. IEEE, 2019.

JOAKIM SKARDING received the M.Sc degree in computer science from the Norwegian University of Science and Technology (NTNU) in 2015.

He worked as a software developer for TomTom in Berlin and since 2018 he has been pursuing his PhD at the University of Technology Sydney. His research interest include modelling and prediction of dynamic complex networks.

BOGDAN GABRYS received the M.Sc. degree in electronics and telecommunication from Silesian Technical University, Gliwice, Poland, in 1994, and the Ph.D. degree in computer science from Nottingham Trent University, Nottingham, U.K., in 1998.

Over the last 25 years, he has been working at various universities and research and development departments of commercial institutions. He is currently a Professor of Data Science and the Director of the Advanced Analytics Institute at the University of Technology Sydney, Sydney, Australia. His research activities have concentrated on the areas of data science, complex adaptive systems, computational intelligence, machine learning, predictive analytics, and their diverse applications. He has published over 180 research papers, chaired conferences, workshops, and special sessions, and been on program committees of a large number of international conferences with the data science, computational intelligence, machine learning, and data mining themes. He is also a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE), a Member of IEEE Computational Intelligence Society and a Fellow of the Higher Education Academy (HEA) in the UK. He is frequently invited to give keynote and plenary talks at international conferences and lectures at internationally leading research centres and commercial research labs. More details can be found at: http://bogdan-gabrys.com

KATARZYNA MUSIAL received the M.Sc. degree in computer science from the Wrocław University of Science and Technology (WrUST), Poland, the M.Sc. degree in software engineering from the Blekinge Institute of Technology, Sweden, in 2006, and the Ph.D. from WrUST, in November 2009.

In November 2009, she was appointed as a Senior Visiting Research Fellow with Bournemouth University (BU), where she has been a Lecturer in informatics, since 2010. In November 2011, she joined Kings as a Lecturer in computer science. In September 2015, she returned to Bournemouth University as a Principal Academic in Computing, where she was a member of the Data Science Institute. In September 2017, she joined as an Associate Professor in network science with the School of Software, University of Technology Sydney, where she is currently a member of the Advanced Analytics Institute. Her research interests include complex networked systems, analysis of their dynamics and its evolution, adaptive and predictive modeling of their structure and characteristics, as well as the adaptation mechanisms that exist within such systems are in the center of her research interests.

TABLE 7: A summary of notation used in this work.

Notation	Description
$\odot$	Element wise product
G	Static graph
G^t	Static graph at time t
DG	Discrete dynamic graph
CG	Continuous dynamic graph
V	The set of nodes in a graph
E	The set of edges in a graph
v	A node $v \in V$
e	An edge $e \in E$
t	Time step / event time
$\bar{t}$	Time step just before time t
$< t$	All time steps up until time t
Δ	Duration
n	Number of nodes
d	Dimensions of a node feature vector
l	Dimensions of a GNN produced hidden feature vector
k	Dimensions of a RNN/self-attention produced hidden feature vector
X^t	Feature matrix at time t
A^t	Adjacency matrix at time t
$\hat{A}$	Predicted adjacency matrix
z_u^t	GNN produced hidden feature vector of node u at time t
h_u^t	RNN/self-attention produced hidden feature vector of node u at time t
Z^t	GNN produced hidden feature matrix at time t
H^t	RNN/self-attention produced hidden feature matrix at time t
λ	Conditional intensity function
σ	The sigmoid function
$W, U, w, b, \omega, \psi, \varphi$	Learnable model parameters

TABLE 8: A summary of abbreviations used in this work. Some model names in the text look like abbreviations but are in fact simply the name of the model (or the authors do not explicitly state what the abbreviation stand for). These include: PATCHY-SAN, DyGGNN, RgGNN, StrGNN, EvolveGCN, JODIE, GC-LSTM, GCN-GAN, DynGraphGAN and DyREP.

Abbreviation	Description
GNN	Graph neural network
DGNN	Dynamic graph neural network
RNN	Recurrent neural network
LSTM	Long-term short term memory
GRU	Gated recurrent unit
GAN	Generative adversarial network
CNN	Convolutional neural network
TPP	Temporal point process
RGM	Random graph model
ERGM	Exponential random graph model
TERGM	Temporal exponential random graph model
SAOM	Stochastic actor oriented model
REM	Relational event model
GCN	Graph Convolutional Network [75]
GAT	Graph Attention Network [88]
GGNN	Gated Graph Neural Network [86]
R-GCN	Relational Graph Convolutional Network [97]
convLSTM	Convolutional LSTM [94]
GraphRNN	Graph recurrent neural network [43]
G-GCN	Generative graph convolutional network [79]
VGAE	Variational graph autoencoder [80]
GCRN-M1	Graph convolutional recurrent network model 1 [69]
GCRN-M2	Graph convolutional recurrent network model 2 [69]
WD-GCN	Waterfall dynamic graph convolutional network [76]
CD-GCN	Concatenated dynamic graph convolutional network [76]
DySAT	Dynamic Self-Attention Network [17]
TNDCN	Temporal network-diffusion convolution networks [25] [90]
HDGNN	Heterogeneous Dynamical Graph Neural Network [24]
TeMP	Temporal Message Passing [22]
LRGCN	Long Short-Term Memory R-GCN [23]
RE-Net	Recurrent Event Network [95]
TNA	Temporal Neighbourhood Aggregation [96]
TDGNN	Temporal Dependent Graph Neural Network [81]
DynGEM	Dynamic graph embedding model [98]
E-LSTM-D	Encode-LSTM-Decode [47]
VGRNN	Variational graph recurrent neural network [103]
SI-VGRNN	Semi-implicit VGRNN [103]
SGNN	Streaming graph neural network [78]
LDG	Latent dynamic graph [109]
GHN	Graph Hawkes network [111]
TGAT	Temporal graph attention [18]
TGN	Temporal Graph Networks [19]