Review Paper: Graph Neural Networks for Dynamic Graphs

Alberto Gonzalo Rodriguez Salgado [1]

Abstract

This paper provides an overview about the current research on dynamic Graph Neural Networks, which tasks need to be adressed in the dynamic setting and which models are currently available. Moreover, possible potential drawbacks of these models, new ideas and research directions will be adressed during the paper.

1. Introduction

Graphs are powerful representations of many kinds of data as for example proteins or social networks and have been used in applications as drug discovery [3] or recommender systems [12]. Most research has focused on *static graphs* i.e, graphs with a fixed number of nodes and edges which does not change over time. However, many real world applications show a dynamic behaviour where time plays an important role as for example in social network graphs where new users constantly sign in and add new friends i.e., the number of nodes and edges in the graph is constantly changing over time. Such graphs are refered to as *dynamic graphs*.

2. Graphs

A graph consists on a set of nodes G and a set of edges E. Every node i can be characterised by a feature rerpresentation v $\in R^D$. Specially, we will focus on the example of a social network graph where a node represents a user. In this case the node feature could represent e.g **v** = [*city of residence, university, work* ...]. During the paper we will consider the directed graph case i.e, $\mathbf{e}_{12} \neq \mathbf{e}_{21}$ necessarily. Moreover, an edge connecting two nodes may also be a vector $\mathbf{e}_{12} \in R^M$. In our example it could represent e.g $\mathbf{e}_{carlos,lara} =$ [*times they message daily, do they live in the same city* ...] or a vector representation of their messages produced by e.g BERT [2]. Consequently, we are dealing most times with $multigraphs$ since multiple edges may appear as in the example of user $Carlos$ messaging the user $Lara$ multiple times. As mentioned in the introduction, in dynamic graphs the node and edge sets may change over time. In the most general case, the node features v_i may also change over time. A special kind of graphs where only the node's features change over time are refered as *spatio temporal graphs* [7]. Note that the word *spatio* does not have a geometrical meaning. Last but not least, users may also delete their accounts or remove friends from their contacts. This events are refered to as $deletion\ events$ in the dynamic graphs literature [7]. Some methods as TGN [13] include them in their modeling, some others as DyREP [15] not.

Figure 1 illustrates a toy example with node creattion and edge creation events in the social network case setting.

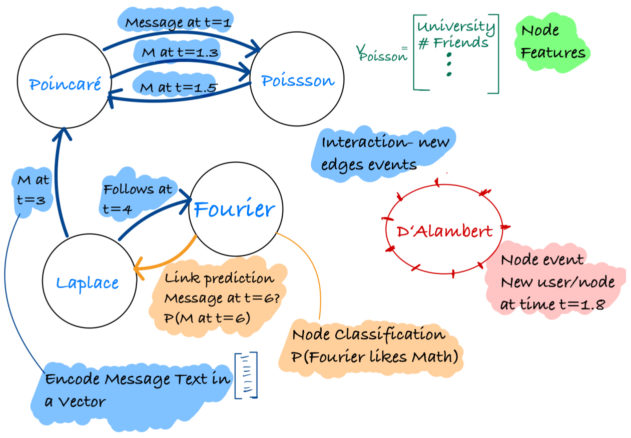

Figure 1. Social Network Dynamic Graph Example showing two event types: 1) Edge creation and 2) Node creation

3. Dynamic Graph Neural Networks

Tasks As with static graphs, in dynamic graphs one is also interested in performing *node classification* e.g does a social network user belong to the sports community? One is also interested in $link\ prediction$, predict whether two nodes n_i and n_j will interact with each other at time t i.e, in computing the probability $p\left((n_i, n_j) \mid t\right)$. Moreover, one could be also interested in predicting $when$ a node n_i is

[*]Equal contribution [1]Department of Computer Science, TUM, Munich, Germany. Correspondence to: Alberto Gonzalo Rodriguez Salgado <alberto.rodriguez-salgado@tum.de>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020. Copyright 2020 by the author(s).

likely to interact with user n_j. Last but not least, one may be also interested in predicting $which$ type of event will happen next between two nodes e.g, whether it will it be a text message or a like.

We are interested in solving to general types of tasks. First, a transdusctive task where the future link prediction and node classification are performed on nodes that have been seen by the model during training. Second, an inductive task where the model needs to generalize to nodes it has not seen during training. Popular graph neural network (GNN) as for example graph convolutional networks as GCN [8] most operate in a transductive settings and eventhough they can be modified to operate in inductive settings, these modifications are compuationally expensive [4]. Specifically in a dynamic graph, since new nodes may appea, a model needs to generalize to unseen nodes and be able to operate in an inductive setting.

Another important task is to propose a computationally efficient, scalable model for dynamic graphs. Especially, in dynamic graphs the number of neighbours of a node may be large since new nodes are constantly appearing over time as e.g, a social network user having more followers over time . Therefore, one important task consists on finding an optimal, computationally efficient neighbours aggregation and sampling strategy.

Graph Neural Networks (GNNs). The main idea behind GNNs [5] is to find first a learnable function that computes an embedding for every node $n_i \in G$. This is done typically in two steps in a message passing framework [5]. First, a message $\mathbf{m}_i$ is computed by aggregating information over n_i's neighbourhood N_i

$$\mathbf{m_i}^{(k)} = \sum_{u \in N_i} h\left(\mathbf{z}_i^{(k-1)}, \mathbf{z}_u^{(k-1)}, \mathbf{e}_{ui}\right) \tag{1}$$

taking into account the neighbours embedding at previous $k-1$ layer $\mathbf{z}_u^{(k-1)}$, it's own previous embedding $\mathbf{z}_i^{(k-1)}$ and the edge information $\mathbf{e}_{ui}$. $h\left(\right)$ is a learnable and differentiable function. Second, the new embedding $\mathbf{z}_i^k$ is computed as

$$\mathbf{z}_i^{(k)} = F\left(\mathbf{z}_i^{(k-1)}, \mathbf{m}_i^{(k)}\right) \tag{2}$$

taking into account the previous embedding state $\mathbf{z}_i^{(k-1)}$ and the computed message $\mathbf{m}_i^{(k)}$. Note that $F\left(\right)$ is again a learnable function. Last, in order to perform node classification based on the node embedding, popular architectures as Graph Convolutional Networks (GCN) [8] choose to directly apply a softmax activation function to the embedding, yielding the probability of each class as $p_i = softmax\left(\mathbf{z}_i\right)$.

Dynamic Graph Neural Networks In dynamic graphs the neighbourhood of a node is changing over time since new edges may appear or disappear. The main task is to find a mechanism that enables aggregating information over the temporal neighbourhood of n_i, $N\left(n_i, t\right)$ and take into account the dynamic graph behaviour when computing the embeddings. The TGN approach defines the node's embeddings update as

$$\mathbf{z}_i\left(t\right) = \sum_{j \in N_i^k[0,t|} h\left(\mathbf{s}_i\left(t\right), \mathbf{s}_j\left(t\right), \mathbf{e}_{ij}, \mathbf{v}_i\left(t\right), \mathbf{v}_j\left(t\right)\right). \tag{3}$$

We will later go into detail on each component of [3]. Note that in comparison to [1], now we need to aggregate information over the k-th hoop *temporal nieghbourhood* $N_i^k[0,t|$.

Aggregation Step, learnable (h) There are different ways of aggregating over the neighbours. One option proposed in GCN [8] is to use the adjancecy matrix with added self connections $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ to do the aggregation step yielding the embedding update as

$$\mathbf{Z}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right). \tag{4}$$

Note that the aggregation happens over all neighbours of a node n_i which might be computationally too expensive for large graphs.

A more general approach is provided by graph attention networks (GATs) [16] where the aggregation happens by attending over the nodes neighbours and it can be applied in inductive settings.

Taking time into account Most modern approaches for dynamic graphs as TGN [13] achieve their best scores by applying attention as the aggregation technique. Nevertheless, dynamic graph methods as TGN or TGAT[17] do not apply the same attention mechanism originally proposed in [16] since temporal information has to be considered when attending over the temporal nieghbouhood $N\left(n_i, t\right)$. Similarly to text sequences where the position of a word is represented by a positional encoding $\mathbf{p}_i$ and the attention happens over $v_i + p_i$ or $\mathbf{v}_i \parallel \mathbf{p}_i$, in a $temporal$ attention mechanism the aggregation happens over $\mathbf{v}_i + \mathbf{t}_i$ or $\mathbf{v}_i \parallel \mathbf{t}_i$, where $\mathbf{t}_i$ is a vector representation of time, a temporal encoding. Most recent works on dynamic grahs as TGN [13], DyREP [15] or DGNN [11] define the dynamic graph as an initial graph $G\left(t_0\right)$ together with a sequence of timestamped events $\{event(t_1)...event(t_N)\}$. Both TGN and DyREP distinguish two types of events: 1) New nodes are created or are deleted in the graph, in DyRep this is refered to as an association process and 2) New edges between nodes appear, in DyRep this is refered to as a communication process. One must note that in this setting the assumption is made that an event happens instantaneously e.g, when a social network user messages his friend. There may be applications where this modeling does not hold.

4. Approaches

Most models as Jodie [10], DyRep [15], TGN [13] or TGAT [17] focus on the idea of performing an aggregation over a temporal neighbourhood i.e, generalizing 1 to the dynamic setting.

	Handle Node Creation Event	Handle Node Deletion Event	Handle Edge Creation Event	Handle directed Graphs	Predict when event will happen
Jodie	✓	✗	✓	✗	✗
DyRep	✓	✗	✓	✗	✓
TGAT	✓	✗	✓	✗	✗
TGN	✓	✓	✓	✓	✗

Figure 2. Dynamic Graphs model's comparison

Note that even though TGN can not predict, when an event between two nodes is likely to happen, it is able to handle all other types of dynamic events as shown in 2. Moreover, as shown in [13], the other methods shown in 2 can be seen as a special case of TGN. For this reason the TGN approach will be discussed in more detail.

4.1. Temporal Graph Networks (TGN)

Memory state The main idea is to define a so called $memory\ state$ $\mathbf{s}_i$ for every node n_i which keeps track of what is happening to the node across time. The memory state $\mathbf{s}_i$ is updated every time an event involving node $\mathbf{v}_i$ occurs.

Message function Consider an edge interaction $\mathbf{e}_{ij}$ event involving nodes n_i and n_j, n_i being the source and n_j being the target, happens e.g user i messages user j, two messages $\mathbf{m}_i$ and $\mathbf{m}_j$ are computed to update both nodes. Note that TGN also handles directed graphs. The messages are computed by learnable functions $msg_d\left(\mathbf{s}_i\left(t^-\right), \mathbf{s}_j\left(t^-\right), \Delta t, \mathbf{e}_{ij}\right)$ and $msg_s\left(\mathbf{s}_i\left(t^-\right), \mathbf{s}_j\left(t^-\right), \Delta t, \mathbf{e}_{ij}\right)$, one for the source and one for the target node. Δt is the time since the last interaction of both nodes. In TGNs implementation, the authors define the message by concadenatin all inputs together as

$$\mathbf{m}_i = \mathbf{m}_j = \left[\mathbf{s}_j\left(t^-\right) \parallel \mathbf{s}_i\left(t^-\right) \parallel \mathbf{e}_{ij} \parallel \mathbf{\Phi}\left(t\right)\right]. \quad (5)$$

By doing this, it seems that they handle directed graphs as undirected graphs. Moreover, Δt is not explicitely used. Instead, the time vector encoding $\mathbf{\Phi}\left(t\right)$ is used. This time encoding will be later explained.

Messages batching For computational reasons, the TGN authors introduce a batching time system for messages. Consider a node n_i had n interactions with other nodes i.e, set of messages $\{\mathbf{m}_i\left(t_1\right) ... \mathbf{m}_i\left(t_n\right)\}$. The so called message aggregator agreggates these n messages into a single message $\tilde{\mathbf{m}}_i\left(t\right)$ by: 1) Aggregating only the most recent message i.e, $\tilde{\mathbf{m}}_i\left(t\right) = \mathbf{m}_i\left(t_n\right)$ or 2) Computing the mean over the messages. The idea that most recent messages matter the most is also present in [11]. Nevertheless, this assumption may not hold in some dynamic graphs since by only taking the most recent interaction into account, one may forget an older but much more important interaction.

Memory updater Once the aggregated message $\tilde{m}_i$ is computed, the memory state s_i of node n_i is updated as

$$\mathbf{s}_i\left(t\right) = mem\left(\tilde{\mathbf{m}}_i\left(t\right), \mathbf{s}_i\left(t^-\right)\right) \quad (6)$$

and *mem*() is a learnable funnction. In the TGN paper, the authors choose to use a GRU [1] unit as the learnable $mem\left(\right)$ function. By doing so, the memory module of TGN is supposed to be able to learn long term dependecies.

Embeedding computation Finally, the node's v_i embedding is computed as

$$\mathbf{z}_i\left(t\right) = \sum_{j \in N_i^k[0,t|} h\left(\mathbf{s}_i\left(t\right), \mathbf{s}_j\left(t\right), \mathbf{e}_{ij}, \mathbf{v}_i\left(t\right), \mathbf{v}_j\left(t\right)\right) \quad (7)$$

by aggregating over the neighbour's and own memory states $\mathbf{s}_j, \mathbf{s}_i$, node's features $\mathbf{v}_j, \mathbf{v}_i$ and edge's features $\mathbf{e}_{ij}$. Specifically, $N_i^k[0,t|$ refers to the k-th hop temporal neighbourhoood of node n_i in the time span $[0,t]$ i.e, all the nodes it has interacted with till time t. $h\left(\right)$ is again a possible learnable function. The TGN authors [13] accomplish their best performance by using $Temporal\ Graph\ Attention$ as $h\left(\right)$.

Encoding time as a vector The *temporal self-attention* idea presented in [6] is to replace the common positional encoding one finds for text sequences by a time encoding i.e, time encoded as a vector. The proposed time encoding is

$$\mathbf{\Phi}\left(t\right) = \left[\omega_0 + \phi_0, \Psi\left(\omega_1 t + \phi_1\right) ... \Psi\left(\omega_d t + \phi_d\right)\right] \quad (8)$$

where $\Psi\left(\right)$ is some periodic function. Note that each ω encodes a periocity e.g "how often does a user message other user". The more frequencies we include in the time encoding, the more periodic events we can take into account. The first term captures non periodic behaviour. Moreover, this time encoding is invariant to time rescaling. Note that the frequencies w_i can be learned during training [13] via standard back propagation since they are implemented as the weights of a one layer MLP. The corresponding shifts ϕ_i are also learned as the bias term of this MLP. Another possible ways of learning or sampling the frequencies ω via e.g normalizing flows are explained in [17].

The TGN authors [13] use $\mathbf{\Psi}\left(t\right) = cos\left(t\right)$ and do not include the non-periodic term $\omega_0 + \phi_0$ in their implementation which might result in not being able to take into account non periodic behaviours.

For node n_i with N neighbours,the input to the self attention layer is given by

$$\mathbf{Z} = \left[\mathbf{z}_i \parallel \mathbf{\Phi}\left(t_i\right), \mathbf{z}_1 \parallel \mathbf{\Phi}\left(t_1\right) ... \mathbf{z}_N \parallel \mathbf{\Phi}\left(t_N\right)\right]. \tag{9}$$

4.2. Other approaches

Jodie Jodie [10] seems to have introduced the concept of a memory state vector which is refered to as the dynamic embedding i.e, an embedding that is updated when the node is involved in an event and a static embedding vector which does not change over time. The main difference to TGN[13] lies in the fact that Jodie does not aggregate over the neighbours to compute the final embedding but computes it by linearly projecting it's memory state as $\mathbf{z_i} = (1 + \Delta t\mathbf{w})\,\mathbf{s_i}$ with learnable $\mathbf{w}$. Δt indicates the time since last node's interaction in an event. Note that if the last node interaction is very recent i.e, $\Delta t \approx 0$, then $\mathbf{z_i} = \mathbf{s_i}$. On the other side, even though a user might have been inactive a long time i.e, $\Delta t >> 0$ and it's memory state has not been updated in a long time, the linear projection will make sure that the embedding does change with the term $\Delta t\mathbf{w}$.

TGAT The *Temporal Graph Attention* (TGAT) model [17] can be seen as a special case of TGN where no node's memory representations s_i are used. . TGAT solely relies on the time encoding vector to handle the dynamic setting. The embeddings are computed as in 3 using temporal self-attention without the memory state vectors.

DyRep DyRep takes a different approach and aims to capture the graphs dynamics with temporal point processes (TPP). The authors define the TPP conditional distribution parametrizing the intensity distribution as a learnable softplus function. Then, one can compute the likelihoods of events happening between two nodes with the TPP model. Note that using TPPs also allows us to know *when* an event is likely to occur. The embeddings are computed via a temporal attention mechanism. Note that this temporal attention is not the same as in TGN. A detailed explanaition is provided in [15]. Moreover, whenever an event happens, the embeddings from the node involved in the events are updated. In comparison, in TGN only the memory states are updated in this scenario.

5. Perfomance and datasets

5.1. Datasets

It seems that only a few open source dynamic graph datasets are available. The most common datasets used for evaluation seem to be the Wikipedia and Reddit bipartite graph datasets [9]. Both of them are open source. Moreover, the TGN authors use the non-bipartite Twitter dataset [13] which is not open source. Neither the Reddit, Wikipedia or Twitter datasets contain nodes creation or deletion events. Moreover, none of these data sets contain node features.

5.2. Perfomance

TGN acomplishes the best performance on the Wikipedia, Reddit and Twitter datasets, both in transductive and inductive link prediction tasks where performance is measured by *average precision* . Specifically, TGN acomplishes on the Twitter data set 94.52% and 91.37% accuracy respectively in the transductive and inductive settings while the second best approach Jodie [10] acomplishes 85.20% and 79.83%. The exact scores are provided in tables 2 and 3 in [13].

6. Ideas and future research

Message aggregator Aggregating by only choosing the most recent interaction/message, seems to be a strong assumption. The TGN authors [13] propose to investigate the option of using an attention mechanism to compute the new message $\tilde{\mathbf{m}}_\mathbf{i}$.

Different Neighbourhood Sampling Strategy (I) The TGN authors compute a node's embedding by aggregating over the 10 most recent neighbours or by uniformly sampling 10 temporal neighbours. Note that eventhough they acomplish their best results by sampling the 10 most recent neighbours, this enforces even more the assumption that most recent matters the most. An alternative approach to consider in order to sample node n_i's neighbours could be: from the most recent N neighbours, sample the L neighbours with which node v_i has intereacted the most.

Message function In TGN, the message when an event happens is computed by stacking together all the inputs [5]. A learnable message function could give the model more expressive power to compute these messages in a more flexible way. For this reason, building such a learnable mechanism could be considered as a future research topic to look into.

Predicting *when* with a neural TPP In order to predict *when* an event willl happen, the event time t_i, and *which* type of event it will be, its so called mark m_i, a neural temporal point process (TPP) using e.g an LSTM could be used. A review on the possible model choices in this scenario is provided in [14].

Lack of data Unfortuately, it is not possible to establish how well methods as TGN, which claim to handle node features , node creation and node deletion events, perform on such data. Therefore, releasing public dynamic graphs data sets is important. Creating or releasing such datasets would greatly help to evaluate methods as TGN.

References

[1] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS*, 2017.

[2] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv:1810.04805 [cs.CL]*, 2018.

[3] Gaudelet, T., Day, B., Jamasba, A. R., Soman, J., Regep, C., Liu, G., Hayter, J. B. R., Richard Vicker and, C. R., Tang, J., Roblin, D., Blundell, T. L., Bronstein, M. M., and Taylor-King, J. P. Utilising graph machine learning within drug discovery and development. In *arXiv:2012.05716 [q-bio.QM]*, 2021.

[4] Hamilton, W., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Conference on Neural Information Processing*, 2017.

[5] Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. In *arXiv:1709.05584*, 2017.

[6] Kazemi, S., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P., and Brubaker, M. Time2vec: Learning a vector representation of time. In *arXiv:1907.05321 [cs.LG]*, 2019.

[7] Kazemi, S. M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., and Poupart, P. Representation learning for dynamic graphs: A survey. In *JMLR*, 2020.

[8] Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[9] Kumar, S., Zhang, X., and Leskovec, J. Predicting dynamic embedding trajectory in temporal interaction networks. In *SIGKDD 2019*, 2019.

[10] Kumar, S., Zhang, X., and Leskovec, J. Jodie: Predicting dynamic embedding trajectory in temporal interaction networks. In *SIGKDD*, 2019.

[11] Ma, Y., Guo, Z., Ren, Z., Zhao, E., Tang, J., and Yin, D. Streaming graph neural networks. In *SIGIR*, 2020.

[12] Monti, F., Bronstein, M. M., and Bresson, X. Geometric matrix completion with recurrent multi-graph neural networks. In *NIPS*, 2017.

[13] Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., and Bronstein, M. Temporal graph networks for deep learning on dynamic graphs. In *ICML 2020 Workshop on Graph Representation Learning*, 2020.

[14] Shchur, O., Türkmen, A. C., Januschowski, T., and Günnemann, S. Neural temporal point processes: A review. In *IJCAI*, 2021.

[15] Trivedi, Farajtabar, Biswal, and Zha. Dyrep: Learning representations over dynamic graphs, May 2019.

[16] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.

[17] Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., and Achan, K. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations (ICLR)*, 2020.