# Temporal State Prediction in Mobile Ad-hoc Networks

**Barkin C.** [*1]

## Abstract

In this paper, we formulate temporal link prediction problem in dynamic networks. Specifically, we are interested in predicting node, edge, and graph level properties of dynamic mobile adhoc networks (MANETs) to aid in coordination of first responder teams in disaster scenarios.

## 1. Introduction

First-response teams operating in offline disaster scenarios typically utilize radio devices to communicate with one another due to the absence or unreliability of internet and phone signal access. Nevertheless, radio devices come with their own limitations, and responders in the field frequently act "blindly," without cognizance of how their trajectories and rescue plans may impact the reliability of the communication network. Quality of radio signal or probability that one radio device will be able to communicate with another is influenced by many factors, some of which rescuers will not have any control over, e.g. atmospheric conditions and obstructions in the general area (Whitehouse et al., 2007; Lymberopoulos et al., 2006; Luomala & Hakala, 2015). However, in most cases, responders can coordinate actions that in tandem keep the connectivity of communication network intact. Figure 1 conveys how one such coordination

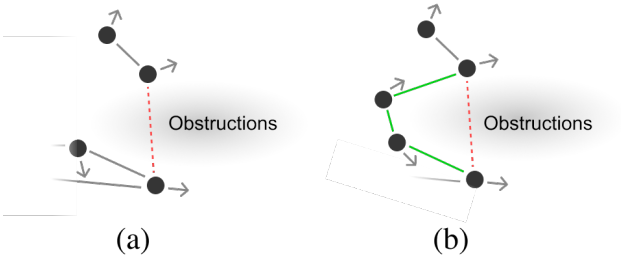


*Figure 1.* Various tasks to train GNNs for. Figure (a) is node-level task, (b) is edge-level task, and (c) is graph level task.

can prevent network disconnection. Figure 1(a) shows that if responders follow the trajectories they are on, indicated as arrows, the network will disconnect, leaving two responders in the dark. Fig 1(b), on the other hand, shows an example of a coordination around the obstructions that would preserve the network connectivity. The importance of connection between all responders becomes especially vital in dangerous scenarios such as natural disasters. Even though network reliability is an active research area, most studies do not take geographical factor into consideration and instead assume that network failures are random and independent (Liu et al., 2011), which, as our example suggests, is not always the case, and that network reliability is somewhat dependent on the environmental factors that responders operate in. Furthermore, they assume a static network and focus on proactive measures rather than both proactive and reactive ones. The type of network we are dealing with is a highly dynamic one (Raffelsberger & Hellwagner, 2012), and no proactive measure alone can cover all possible scenarios, which is why the approach we take to ensure network's reliability must also contain reactionary sub-components. In this paper, we will be studying a deep neural network architecture for temporal link prediction in a variant of mobile ad-hoc networks (MANETs) that informs members of the network about a potential disconnection based on their trajectories.

Various machine learning algorithms have been proposed in the field of communication networks. Adaptive channel equalization is one such example, where (Kumar et al., 1998; Zhang & Zhang, 2007; Erdogmus et al., 2001) propose various machine learning and deep learning methods to reduce potential information loss as much as possible due to dense and simultaneous communication. In similar vein, there could exist several types of interference in the same environment, therefore, it could be useful to discover the existence and types of such interference to decrease packet loss and transmission latency. Schmidt et al. use deep CNN based model to classify various types of interference (2017), whereas Grimaldi et al. use SVM-based methods for the same task (2017). In (Li et al., 2010), authors use reinforcement learning techniques to find an adaptive rate control strategy to optimize link layer performance and minimize power consumption. However, to our knowledge, there hasn't been as much research activity that deals with the problem of link prediction in MANETs. Even though link

[1]. Correspondence to: <>.

prediction in general has been a growing field especially in the context of traffic flow and social networks (Wang et al., 2020; Ahmed & Chen, 2016; Zhu et al., 2016), there remains a few major differences between the types of networks they operate on and MANETs. Specifically, first major difference is that our network is dynamic, therefore its topology is not static. Secondly, the topological rate of change in MANETs is significantly different than that of social networks, where most recent studies focus on. Whereas the topological rate of change in social networks can be in hours, days, or weeks, topological rate of change in MANETs especially in disaster response will be in seconds or minutes in most cases. Thirdly, MANETs that we are dealing with are particularly small, usually containing 10-20 nodes, whereas social networks and traffic networks can contain thousands of nodes. This is important because link prediction methods utilizing deep learning methods for large networks such as social networks won't translate that well to MANETs for reasons such as oversmoothing and overfitting. The only directly relevant research that we could find is (Shao et al., 2022), and we will use their performance results to compare against ours.

We summarize our main contributions in this paper as follows:

- We propose a novel GN block designed to learn spatial feature representation of a network using multi-level self-attention mechanisms and node, edge, and graph features for relational reasoning over our network.
- This block is further enhanced by the intrinsic properties of MANETs to circumvent the permutation-equivariance restriction imposed on some parts of a regular graph neural network. We integrate our GN block with GRU and Wasserstein GAN to complete the loop of temporal link prediction task.
- We argue that that our architecture can be generalized to other dynamic network types such as evolving social networks with few minor changes and achieve or match state-of-the-art performance. This is due to our model's ability to capture information at all levels of the graph. In this way, the architecture we propose can be used as a toolbox for dynamic networks.

## 2. Preliminaries

Generally, a graph can be defined as a tuple $G = (V, E)$ where V is the set of nodes and E is the set of edges. In some cases such as ours, we might want to attribute some features to nodes, e.g. node degrees, and edges, e.g. distance between two nodes. Using graph neural networks (GNNs) (Scarselli et al., 2008), we can make inferences on nodes, edges, and whole graphs such as node classification (Kipf & Welling, 2016), link prediction (Zhang & Chen, 2018; Long et al., 2022; Li et al., 2014) and graph classification (Zhang et al., 2018; Lee et al., 2018; Gao & Ji, 2019).

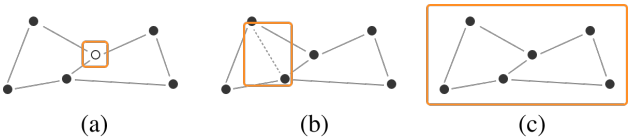


*Figure 2.* Various tasks to train GNNs for. Figure (a) is node-level task, (b) is edge-level task, and (c) is graph level task.

### 2.1. Graph Neural Networks

The discovery of GNNs were motivated in part due to the incompatibility of graph data with classical deep learning models, such as convolutional neural networks (CNNs) as they strictly require grid-structured inputs. This follows from the unordered property of $V$, the set of nodes in a graph. In other words, any learnable function $f$ that operates on graphs must be permutation equivariant to the order of its input, since a graph $G$ with adjacency matrix $A$ can be represented by multiple different adjacency matrices $\{P_1A, P_2A, ..., P_nA\}$ where $P_i$ is any permutation matrix on $A$. This means that $f(A) = f(P_iA)$ for $i \in [1, n]$.

#### 2.1.1. Graph Convolutional Network (GCN)

There are various GNN models. However, it is usually expected that any GNN model contains 2 fundamental operations. AGGREGATE collects information from node $u$'s neighborhood, $\mathcal{N}_u$, and UPDATE combines this aggregated information into a hidden embedding for $u$. One such model that has been increasingly popular is GCN model

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^lW^l) \tag{1}$$

where $\tilde{A} = A + I$, $\tilde{D}$ is degree matrix of $\tilde{A}$, $H^l$ is hidden node embeddings and $W^l$ is a trainable weight matrix for current layer $l$ (Kipf & Welling, 2016). This operation is first-order approximation of localized spectral filters on graphs. In essence, GCN is performing aggregation of features per node $u$ *normalized* by its degree and that of its neighbor as $\sqrt{|\mathcal{N}_u||\mathcal{N}_v|}$.

#### 2.1.2. Graph Attention Network (GAT)

Another hugely successful variant of GNNs is graph attention network, which incorporates attention weights to aggregator function. Specifically, hidden embedding of node $u$ is

calculated as

$$\alpha_{uv} = \frac{\exp(\sigma_n(a(\mathbf{W}h_u\mathbf{W}h_v)))}{\sum_{w\in\mathcal{N}_u}\exp(\sigma_n(a(\mathbf{W}h_u\mathbf{W}h_w)))} \quad (2)$$

$$h_u = \sigma_a(\sum_{v\in\mathcal{N}_u}\alpha_{uv}\mathbf{W}h_v) \quad (3)$$

where $\alpha_{uv}$ can be thought of as node $v$'s *influence* on node $u$, $\mathbf{W}$ is weight matrix that parameterize shared attentional mechanism $a$, and $\sigma_n$ and $\sigma_a$ are non-linearity functions such as LeakyReLU (Veličković et al., 2017).

#### 2.1.3. Set Aggregator

There is a growing body of work on set aggregator functions to improve the performance of GNNs. Some notable ones include DeepSets (Zaheer et al., 2017) and JanossyPooling (Murphy et al., 2018), which inspired our aggregator function explained in Section 3. Specifically, DeepSets solves the problem of permutation-equivariance required by aggregator function by transforming each element $x$ in a set $S$ using a function of choice, adding up all transformations of elements and processing the output using another function of choice. Since $x$ is expected to be a vector, we can use any function we want, including any deep network. JanossyPooling takes a different approach, and provides two ways we can employ to improve set aggregator function: (1) We generate all permutations of a given set, apply a function of choice to each one of permutations, and average the results, or (2) we find a canonical ordering on set $S$, and apply a function of choice on the ordered input. However, it is expected that this ordering is not random and makes use of properties inherent to elements in the set.

### 2.2. Temporal Link Prediction

Recently, learning over temporal graphs (or dynamic networks) has become a major research topic as it can help us better understand how graph data modeling complex relations changes over time. Earlier works applied matrix factorization or other types of aggregations to capture the temporal dimension (Dunlavy et al., 2011; Yu et al., 2017). While more recent works turned to deep neural networks for better performance. For example, Singer et al. (2019) proposed an alignment method over different snapshots and used an LSTM layer to learn the evolution of each node over time. da Xu et al. (2020) extended Graph Attention Networks with a temporal dimension when aggregating the neighbors. This was further extended in (Singer et al., 2022) by tBDFS which added a new layer that is responsible for capturing Depth-First Search patterns in a temporal graph. Moreover, Spatial-Temporal Graph Neural Networks (STGNNs) are being used to model the complex spatial-temporal correlations such as in traffic, social network, and pedestrian trajectories data (Wang et al., 2020; Peng et al., 2020; Min et al., 2021; Zhou et al., 2021). STGNNs usually model the traffic system as a diffusion process (Li et al., 2017) and combine diffusion convolution and sequential models such as GRU and LSTM. (Lei et al., 2019) has presented the novel temporal link prediction model GCN-GAN to tackle the dynamic link prediction problem in network systems. The authors combine GCN, LSTM and GAN to effectively deal with temporal link prediction tasks in weighted dynamic networks. They tested their model with four datasets of different network systems and demonstrated strong performance against other six competitors. In (Bonner et al., 2019), authors introduced the Temporal Neighbourhood Aggregation (TNA) model for temporal link prediction task. Their novel TNA block uses GCN for spatial and GRU for temporal learning before concatenating the outputs to pass into a linear layer for spatio-temporal learning. Additionally, they use Variational Autoencoder for generating the future adjacency matrix. Their model outperformed other popular ones such as GCN-GAN, LSTM and GraphSAGE. Shao et al. (2022) propose a model named FastSTLSG for the same problem but directly apply to MANETs by using chaotic time theory for *slicing* sequence of networks before passing them into a variant of GCN. They generate the future adjacency matrix using Least Squares GAN.

## 3. Methods

Our data consists of timelines of snapshots of graph taken at specific intervals. Timeline here refers to a single rescue attempt carried out by a first responder team, and a snapshot refers to node and edge information of a graph at a specific time. We represent each graph as $G_t = (V_t, E_t, \vec{V}_t, \vec{E}_t, \vec{g}_t)$. Set $V_t$ provides indexing $i$ over $\vec{V}_t$ and $E_t$ provides indexing $k = \{i, j\}$ over $\vec{E}_t$. The remaining $\vec{V}_t = \{\vec{h}_i \mid \vec{h}_i \in \mathbb{R}^{F_{\text{node}}} \ \forall i \in V_t\}$, $\vec{E}_t = \{\vec{e}_k \mid \vec{e}_k \in \mathbb{R}^{F_{\text{edge}}} \ \forall k \in E_t\}$, and $\vec{g}_t \in \mathbb{R}^{F_{\text{global}}}$ with $F_{\text{node}}$, $F_{\text{edge}}$, $F_{\text{global}}$ representing node, edge, and global feature dimensions, are collections of features for nodes, edges, and graph for graph at time $i$, respectively. Our objective therefore is *threefold*:

$$(A_{t+1}, X_{t+1}, g_{t+1}) = f(G_0, G_1, ..., G_t) \quad (4)$$

where $A_{t+1}$ is adjacency and $X_{t+1}$ is node feature matrices, and $g_{t+1}$ is graph feature vector at time $t+1$. Function $f$, which can be seen as composition of many functions, is what we want to learn using samples of sequence $G_0, G_1, ..., G_t$.

### 3.1. GN Block

We make use of the generalized functional model Graph Network (GN) for relational reasoning over our data (Battaglia et al., 2018). As illustrated in Figure 2, GN has 3 self-attention layers to transform each of input node, edge, and graph level features to higher level features. Our attention mechanisms follow the work presented in (Veličković et al.,

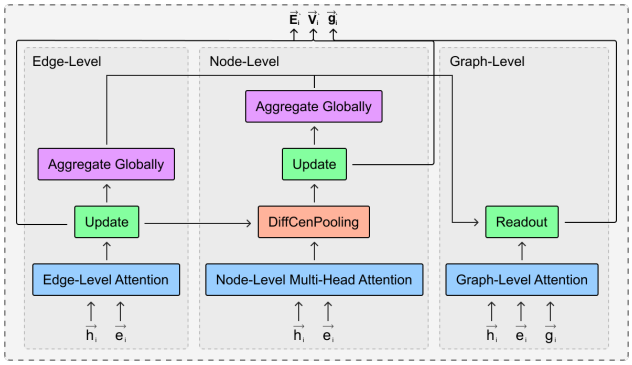


*Figure 3.* Diagram of GN block.

2017; Yang & Li, 2020) with some further additions. We first apply self-attention on each edge $e_i \in \vec{E}_t$ to compute edge attention coefficients $b^e_{ik}$ and $b^n_{ij}$, which are edge $k$'s and node $j$'s influence on edge $i$, respectively.

$$b^e_{ik} = \sigma(\alpha^e([\mathbf{W^e_e}\vec{e_i}, \mathbf{W^e_e}\vec{e_k}])) \tag{5}$$

$$b^n_{ij} = \sigma(\alpha^n([\mathbf{W^e_e}\vec{e_i}, \mathbf{W^e_n}\vec{h_j}])) \tag{6}$$

where $\sigma$ is any non-linearity, $\alpha^e$ is a shared attentional mechanism over edges parameterized by a weight matrix $\mathbf{W^e_e} \in \mathbb{R}^{F_e \times F_e}$, and $\alpha^n$ is a shared attentional mechanism over nodes parameterized by a weight matrix $\mathbf{W^e_n} \in \mathbb{R}^{F_n \times F_n}$. We then normalize the coefficients using softmax function over all choices of $j$

$$\beta^e_{ik} = \frac{\exp(b^e_{ik})}{\sum_{j \in \mathcal{N}^e_i} \exp(b^e_{ij})} \tag{7}$$

$$\beta^n_{ij} = \frac{\exp(b^n_{ij})}{\sum_{k \in \mathcal{N}^n_i} \exp(b^n_{ik})} \tag{8}$$

Here, $\mathcal{N}^e_i = \{j \in L(G) \,|\, j \in \mathcal{N}_i\}$ is the set of edges that are $i$'s node neighbors in line graph representation of $G$ and $\mathcal{N}^n_i$ is the set of nodes that are the endpoints of edge $i$. To get the final embedding per edge, we apply MEAN as our aggregator function and concatenate our results

$$\beta_{\mathcal{N}^e_i} = \sigma(\mathbf{W^e_e}\text{MEAN}(\{\beta^e_{ik}\vec{e_k}, \forall k \in \mathcal{N}^e_i\})) \tag{9}$$

$$\beta_{\mathcal{N}^n_i} = \sigma(\mathbf{W^e_n}\text{MEAN}(\{\beta^n_{ij}\vec{e_j}, \forall j \in \mathcal{N}^n_i\})) \tag{10}$$

$$\vec{e_i} = [\beta_{\mathcal{N}^e_i}, \beta_{\mathcal{N}^n_i}] \tag{11}$$

To keep things simple, we simply concatenate $\beta_{\mathcal{N}^e_i}$ and $\beta_{\mathcal{N}^n_i}$ to get the final hidden embedding of edge $e_i$.

We employ weighted multi-head attention where the final node embedding is based on weighted sum of each head's *opinionatedness* in reference to prediction task, where we measure opinionatedness using variance of the attention weights it assigns to each node. This is similar to "confidence" of a head as described in (Voita et al., 2019), but instead of using maximum weight, we define opinionatedness with variance of weights. We start by calculating normalized attention coefficients $\varkappa^n_{ij}$ and $\varkappa^e_{ik}$ per node $i$, which are node $j$'s and edge $k$'s influence on node $i$, respectively.

$$\varkappa^n_{ij} = \frac{\exp(\sigma(\delta_n([\mathbf{W^n_n}\vec{h_i}, \mathbf{W^n_n}\vec{h_j])))}{\sum_{k \in \mathcal{N}^n_i} \exp(\sigma(\delta_n([\mathbf{W^n_n}\vec{h_i}, \mathbf{W^n_n}\vec{h_k}])))} \tag{12}$$

$$\varkappa^e_{ik} = \frac{\exp(\sigma(\delta_n([\mathbf{W^n_n}\vec{h_i}, \mathbf{W^e_e}\vec{e_k}])))}{\sum_{j \in \mathcal{N}^e_i} \exp(\sigma(\delta_n([\mathbf{W^n_n}\vec{h_i}, \mathbf{W^e_e}\vec{e_j}])))} \tag{13}$$

We then apply our weighted multi-head mechanism with mean as our aggregator function, and concatenate $\varkappa^n_{ij}$ and $\varkappa^e_{ik}$ to get the final node embedding

$$\varkappa^*_{\mathcal{N}^e_i} = \sigma\left(\sum_{q=1}^{Q^e} \tilde{Var}(\{\varkappa^e_{ik}, \forall k \in \mathcal{N}^e_i\}) \cdot \mathrm{H}^e_q(i)\right) \tag{14}$$

$$\mathrm{H}^e_q(i) = \left(\frac{1}{|\mathcal{N}^e_i|} \sum_{k \in \mathcal{N}^e_i} \varkappa^e_{ik} \mathbf{W^n_e}\vec{e_k}\right) \tag{15}$$

$$\varkappa^*_{\mathcal{N}^n_i} = \sigma\left(\sum_{q=1}^{Q^n} \tilde{Var}(\{\varkappa^n_{ij}, \forall j \in \mathcal{N}^n_i\}) \cdot \mathrm{H}^n_q(i)\right) \tag{16}$$

$$\mathrm{H}^n_q(i) = \left(\frac{1}{|\mathcal{N}^n_i|} \sum_{j \in \mathcal{N}^n_i} \varkappa^n_{ij} \mathbf{W^n_n}\vec{h_j}\right) \tag{17}$$

$$\vec{h}_i = [\varkappa^*_{\mathcal{N}^e_i}, \varkappa^*_{\mathcal{N}^n_i}] \tag{18}$$

$\tilde{Var}$ is normalized variance of the given set of attention weights, $\mathrm{H}^e_q(i)$ and $\mathrm{H}^n_q(i)$ are HEAD mechanisms that attend to neighboring edges and nodes of node $i$, respectively, and $Q^e$ and $Q^n$ are numbers of heads. Pruning on the number of heads can be done following the work in (Voita et al., 2019).

For graph-level attention, we want to calculate how much weight should be given to each node and edge features while computing graph embedding.

$$\varrho^g_i = \frac{\exp(\sigma(\delta_g([\mathbf{W^g_g}\vec{g}, \mathbf{W^g_n}\vec{h_i}])))}{\sum_{j \in \mathcal{V}} \exp(\sigma(\delta_g([\mathbf{W^g_g}\vec{g}, \mathbf{W^g_n}\vec{h_j}])))} \tag{19}$$

$$\varrho^g_k = \frac{\exp(\sigma(\delta_g([\mathbf{W^g_g}\vec{g}, \mathbf{W^g_e}\vec{e_k}])))}{\sum_{l \in \mathcal{E}} \exp(\sigma(\delta_g([\mathbf{W^g_g}\vec{g}, \mathbf{W^g_e}\vec{e_l}])))} \tag{20}$$

$$\varrho^{g^*}_i = \sigma(\sum_{i \in V} \varrho^g_i \mathbf{W^g_n}\vec{h_i}) \tag{21}$$

$$\varrho^{g^*}_k = \sigma(\sum_{k \in E} \varrho^g_k \mathbf{W^g_e}\vec{e_i}) \tag{22}$$

$$\vec{g} = [\varrho^{g^*}_i, \varrho^{g^*}_k] \tag{23}$$

Next step in our GN block is updating edge embeddings via MLP with one hidden layer.

$$\vec{e_i}' = \mathbf{MLP}_e([\vec{e_i}, \vec{h}_i, \vec{h}_j, \vec{g}]) \tag{24}$$

To aggregate node $i$'s neighborhood information, we use the embedding set of edges that $i$ is an endpoint of. This allows us to capture graph's and neighboring nodes' and edges' features as demonstrated in Eq. (18). Since this aggregation is performed over a set, the function we choose must be permutation-invariant, that is, the output of this function should not change if the order of input changes.

However, an important property of our network is that its topology is partially determined by the distances and signal-to-noise ratios between each node. The probability that an edge will exist between a pair of nodes is influenced by the distance and signal-to-noise ratio between them since this is a communication network of radio devices. Therefore, while aggregating edge features per node, we can form a canonical ordering using their distance and signal-to-noise ratio, which will allow us to use permutation-invariant functions. Specifically, while aggregating edges, we may want to put more emphasis on some than others like we did in previous step using self-attention mechanism. But this time, we make use of the fact that, given two edges $e_i$ and $e_k$, the probability that the one with smaller distance and larger signal-to-noise ratio will be present in the next time step is higher than the one with larger distance and smaller signal-to-noise ratio. Therefore, we perform aggregation to compute final node embedding in the following way

$$\vec{h}_i = \lambda_i \cdot \mathring{n}_i^e \tag{25}$$

where $\lambda_i \in \mathbb{R}^{|\mathcal{N}_i^e|}$ is the weights vector and $\mathring{n}_i^e$ is ordered sequence of embeddings of node $i$'s neighboring edges. We can calculate the weights vector in the following way. We start with a vector $v_{ij} = |d_{ij},\ snr_{ij}|$ representing distance and signal-to-noise ratio between nodes $i$ and $j$. We first scale $d_{ij}$ and $snr_{ij}$ using MinMaxScaler so that they are within the same range of $[0, 1]$. Noting that a good *connection* is determined by small distance and large signal-to-noise ratio, we need to apply a function $f$ to distance so that for small values of $d_{ij}$ we get large $f(x)$ which makes it so that both distance and signal-to-noise ratio *point to* in the same direction.

$$invert(x) = 1 - \frac{1}{1 + (\frac{1}{x} - 1)^k} \tag{26}$$

where k is a parameter for steepness. For our purposes, we chose $k = 1.7$ so that $invert(x) : [0, 1] \to [0, 1]$ has a nice curvature that doesn't reward or punish values close to boundaries of $[0, 1]$ too harshly. Letting $d_{ij}^{'} =$ MinMaxScaler$(d_{ij})$ and $snr_{ij}^{'} =$ MinMaxScaler$(snr_{ij})$, we get

$$\eta_i^j = a \cdot invert(d_{ij}^{'}) + b \cdot snr_{ij}^{'} \tag{27}$$

$$\lambda_i^j = \frac{\eta_i^j}{\sum_{r \in \mathcal{N}_i^e} \eta_i^r} \tag{28}$$

where $a, b \in [0, 1]$ are used to signify importance of distance and signal-to-noise ratio and keep the total value within $[0, 1]$ range. We use $0.5$ for both for simplicity. Eq. (22) is to make sure the weights are relative to each other and that $\sum_{j \in \mathcal{N}_i^e} \lambda_i^j = 1$. We proceed by updating the aggregated node embeddings.

$$\vec{h}_i^* = \mathbf{MLP}_n([\vec{e}_i^{'}, \vec{h}_i, \vec{g}]) \tag{29}$$

To compute global graph embedding, we apply aggregation over all node and edge embeddings to pass into our readout function along with $\vec{g}$. We have several ways to do this: (a) we can separately pool nodes' and edges' embeddings and concatenate them with $\vec{g}$ before passing them to our readout function, as shown in (Hamrick et al., 2018), or (b) we can apply a similar canonical ordering to nodes or edges, train a neural function on the ordered sequence, and then perform (a). We choose to go with (b) since it allows us to exploit some important properties about our graph that we discussed previously. The method we are using is a variant of Janossy pooling described in (Murphy et al., 2018). However, instead of using all permutations of the node set as our space, we first order nodes according to their diffusion centrality defined as $DC(\mathbf{g}; q, T) := \mathbf{H}(\mathbf{g}; q, T) \cdot \mathbf{1} = (\sum_{t=1}^{T} (q\mathbf{g})^t) \cdot \mathbf{1}$ where $DC(\mathbf{g}; q, T)_i$ is the expected total number of times that message originating from i is received by any of the nodes of the network during a T-period time interval (Banerjee et al., 2014; 2013), and then for each node $i \in V$, we let $\pi_i$ be sequence of nodes $j \in V$ ordered by $\sum_{(k,l) \in \psi_{ij}} \eta_l^k$, where $\psi_{ij}$ is the edge pair sequence of shortest path between $i$ and $j$, and $\Pi$ the set of all such sequences. Then, we globally aggregate node embeddings as

$$\vec{V}^{'} = \mathbf{MEAN}(\{\mathbf{GRU}([\pi_i]) \mid \pi_i \in \Pi\}) \tag{30}$$

We choose to use GRU for our function since the ordering is based on the propagation of information spreading out from node $i$. Based on experiments, this could be changed to RNN or LSTM. We proceed by globally aggregating the edge embeddings and pass our results to readout function. A similar ordering could be found for edges using *current-flow betweenness centrality*. However, to keep things simple, we decided to use MEAN function over edge embedding set $\vec{E}$.

$$\vec{E}^{'} = \mathbf{MEAN}(\vec{E}) \tag{31}$$

$$\vec{G} = \text{READOUT}([\vec{E}^{'}, \vec{V}^{'}, \vec{g}]) \tag{32}$$

where $\vec{E}$ and $\vec{V}$ are embedding sets of nodes and edges, respectively. For READOUT function, we can use both standard and neural functions. Recent work by Buterez et al. (2022) demonstrated that neural readouts performed better in majority of the datasets they used with competitive results

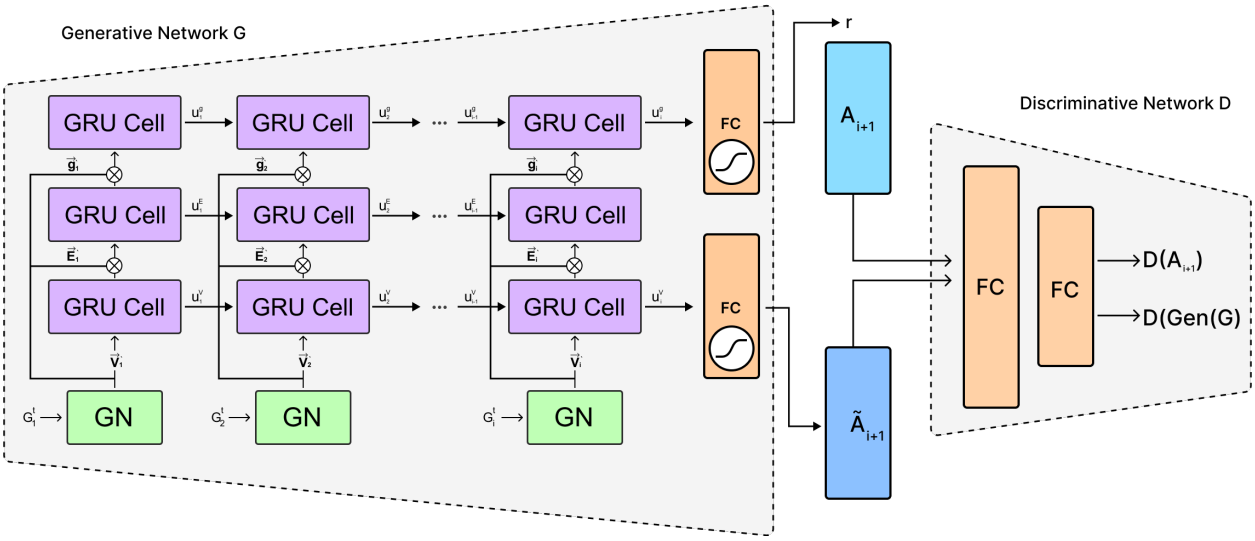


*Figure 4.* Full network architecture for predicting next state of mesh network.

in the rest without any hyper-parameter tuning or cross-validation. With this in mind, we choose to use a multilayer perceptron in our model as well, though, depending on its performance, we can easily switch to a standard function. Our GN block will output the following for each graph snapshot

$$GN_{G_i^t} = (\vec{E}, \vec{V}, \vec{g}) \tag{33}$$

### 3.2. Temporal Processing

## References


Ahmed, N. M. and Chen, L. An efficient algorithm for link prediction in temporal uncertain social networks. *Information Sciences*, 331:120–136, 2016.

Banerjee, A., G a, A., Duflo, E., and Jackson, M. The diffusion of microfinance. *Science (New York, N.Y.)*, 341: 1236498, 07 2013. doi: 10.1126/science.1236498.

Banerjee, A., Chandrasekhar, A. G., Duflo, E., and Jackson, M. O. Gossip: Identifying central individuals in a social network. Working Paper 20422, National Bureau of Economic Research, August 2014. URL http://www.nber.org/papers/w20422.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V. F., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gülçehre, Ç., Song, H. F., Ballard, A. J., Gilmer, J., Dahl, G. E., Vaswani, A., Allen, K. R., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M. M., Vinyals, O., Li, Y., and Pascanu, R. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL http://arxiv.org/abs/1806.01261.

Bonner, S., Abarghouei, A. A., Jackson, P. T. G., Brennan, J., Kureshi, I., Theodoropoulos, G., McGough, A. S., and Obara, B. Temporal neighbourhood aggregation: Predicting future links in temporal graphs via recurrent variational graph convolutions. *CoRR*, abs/1908.08402, 2019. URL http://arxiv.org/abs/1908.08402.

Buterez, D., Janet, J. P., Kiddle, S. J., Oglic, D., and Liò, P. Graph neural networks with adaptive readouts, 2022. URL https://arxiv.org/abs/2211.04952.

Dunlavy, D. M., Kolda, T. G., and Acar, E. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(2):1–27, 2011.

Erdogmus, D., Rende, D., Principe, J. C., and Wong, T. F. Nonlinear channel equalization using multilayer perceptrons with information-theoretic criterion. In *Neural Networks for Signal Processing XI: Proceedings of the 2001 IEEE Signal Processing Society Workshop (IEEE Cat. No. 01TH8584)*, pp. 443–451. IEEE, 2001.

Gao, H. and Ji, S. Graph u-nets. In *international conference on machine learning*, pp. 2083–2092. PMLR, 2019.

Grimaldi, S., Mahmood, A., and Gidlund, M. An svm-based method for classification of external interference in industrial wireless sensor and actuator networks. *Journal of Sensor and Actuator Networks*, 6(2):9, 2017.

Hamrick, J. B., Allen, K. R., Bapst, V., Zhu, T., McKee, K. R., Tenenbaum, J. B., and Battaglia, P. W. Relational

inductive bias for physical construction in humans and machines. *CoRR*, abs/1806.01203, 2018. URL http://arxiv.org/abs/1806.01203.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907.

Kumar, P., Saratchandran, P., and Sundararajan, N. Communication channel equalisation using minimal radial basis function neural networks. In *Neural Networks for Signal Processing VIII. Proceedings of the 1998 IEEE Signal Processing Society Workshop (Cat. No.98TH8378)*, pp. 477–485, 1998. doi: 10.1109/NNSP.1998.710678.

Lee, J. B., Rossi, R., and Kong, X. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1666–1674, 2018.

Lei, K., Qin, M., Bai, B., Zhang, G., and Yang, M. GCN-GAN: A non-linear temporal link prediction model for weighted dynamic networks. *CoRR*, abs/1901.09165, 2019. URL http://arxiv.org/abs/1901.09165.

Li, X., He, H., and Yao, Y.-D. Reinforcement learning based adaptive rate control for delay-constrained communications over fading channels. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7. IEEE, 2010.

Li, X., Du, N., Li, H., Li, K., Gao, J., and Zhang, A. A deep learning approach to link prediction in dynamic networks. In *Proceedings of the 2014 SIAM International conference on data mining*, pp. 289–297. SIAM, 2014.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. Graph convolutional recurrent neural network: Data-driven traffic forecasting. *CoRR*, abs/1707.01926, 2017. URL http://arxiv.org/abs/1707.01926.

Liu, J., Jiang, X., Nishiyama, H., and Kato, N. Reliability assessment for wireless mesh networks under probabilistic region failure model. *IEEE Transactions on Vehicular Technology*, 60(5):2253–2264, 2011.

Long, Y., Wu, M., Liu, Y., Fang, Y., Kwoh, C. K., Chen, J., Luo, J., and Li, X. Pre-training graph neural networks for link prediction in biomedical networks. *Bioinformatics*, 38(8):2254–2262, 2022.

Luomala, J. and Hakala, I. Effects of temperature and humidity on radio signal strength in outdoor wireless sensor networks. In *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 1247–1255. IEEE, 2015.

Lymberopoulos, D., Lindsey, Q., and Savvides, A. An empirical characterization of radio signal strength variability in 3-d ieee 802.15. 4 networks using monopole antennas. In *European Workshop on Wireless Sensor Networks*, pp. 326–341. Springer, 2006.

Min, S., Gao, Z., Peng, J., Wang, L., Qin, K., and Fang, B. Stgsn—a spatial–temporal graph neural network framework for time-evolving social networks. *Knowledge-Based Systems*, 214:106746, 2021.

Murphy, R. L., Srinivasan, B., Rao, V. A., and Ribeiro, B. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *CoRR*, abs/1811.01900, 2018. URL http://arxiv.org/abs/1811.01900.

Peng, H., Wang, H., Du, B., Bhuiyan, M. Z. A., Ma, H., Liu, J., Wang, L., Yang, Z., Du, L., Wang, S., et al. Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting. *Information Sciences*, 521:277–290, 2020.

Raffelsberger, C. and Hellwagner, H. Evaluation of manet routing protocols in a realistic emergency response scenario. In *Proceedings of the 10th International Workshop on Intelligent Solutions in Embedded Systems*, pp. 88–92. IEEE, 2012.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Schmidt, M., Block, D., and Meier, U. Wireless interference identification with convolutional neural networks. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pp. 180–185. IEEE, 2017.

Shao, H., Wang, L., Liu, H., and Zhu, R. A link prediction method for manets based on fast spatio-temporal feature extraction and lsgans. *Scientific Reports*, 12(1):16896, 2022. doi: 10.1038/s41598-022-20981-3. URL https://doi.org/10.1038/s41598-022-20981-3.

Singer, U., Guy, I., and Radinsky, K. Node embedding over temporal graphs. *CoRR*, abs/1903.08889, 2019. URL http://arxiv.org/abs/1903.08889.

Singer, U., Roitman, H., Guy, I., and Radinsky, K. tbdfs: Temporal graph neural network leveraging dfs, 2022. URL https://arxiv.org/abs/2206.05692.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. Analyzing multi-head self-attention: Specialized heads

do the heavy lifting, the rest can be pruned. *CoRR*, abs/1905.09418, 2019. URL http://arxiv.org/abs/1905.09418.

Wang, X., Ma, Y., Wang, Y., Jin, W., Wang, X., Tang, J., Jia, C., and Yu, J. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference 2020*, pp. 1082–1092, 2020.

Whitehouse, K., Karlof, C., and Culler, D. A practical evaluation of radio signal strength for ranging-based localization. *ACM SIGMOBILE Mobile Computing and Communications Review*, 11(1):41–52, 2007.

Xu, D., Ruan, C., Körpeoglu, E., Kumar, S., and Achan, K. Inductive representation learning on temporal graphs. *CoRR*, abs/2002.07962, 2020. URL https://arxiv.org/abs/2002.07962.

Yang, Y. and Li, D. Nenn: Incorporate node and edge features in graph neural networks. In *Asian Conference on Machine Learning*, 2020.

Yu, W., Aggarwal, C. C., and Wang, W. Temporally factorized network modeling for evolutionary network analysis. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pp. 455–464, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346757. doi: 10.1145/3018661.3018669. URL https://doi.org/10.1145/3018661.3018669.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. J. Deep sets. *CoRR*, abs/1703.06114, 2017. URL http://arxiv.org/abs/1703.06114.

Zhang, L. and Zhang, X. Mimo channel estimation and equalization using three-layer neural networks with feedback. *Tsinghua Science Technology*, 12(6):658–662, 2007. ISSN 1007-0214. doi: https://doi.org/10.1016/S1007-0214(07)70171-2. URL https://www.sciencedirect.com/science/article/pii/S1007021407701712.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Zhou, H., Ren, D., Xia, H., Fan, M., Yang, X., and Huang, H. Ast-gnn: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction. *Neurocomputing*, 445:298–308, 2021.

Zhu, L., Guo, D., Yin, J., Ver Steeg, G., and Galstyan, A. Scalable temporal latent space inference for link prediction in dynamic social networks. *IEEE Transactions on Knowledge and Data Engineering*, 28(10):2765–2777, 2016.