machine learning & knowledge extraction

MDPI

Article

Missing Data Estimation in Temporal Multilayer Position-Aware Graph Neural Network (TMP-GNN)

Bahareh Najafi *, Saeedeh Parsaeefard and Alberto Leon-Garcia

Department of Electrical and Computer Engineering, University of Toronto, BA4120, 40 St. George Street, Toronto, ON M5S 3G4, Canada; saeideh.fard@utoronto.ca (S.P.); alberto.leongarcia@utoronto.ca (A.L.-G.)
* Correspondence: bahareh.najafi@mail.utoronto.ca

Abstract: GNNs have been proven to perform highly effectively in various node-level, edge-level, and graph-level prediction tasks in several domains. Existing approaches mainly focus on static graphs. However, many graphs change over time and their edge may disappear, or the node/edge attribute may alter from one time to the other. It is essential to consider such evolution in the representation learning of nodes in time-varying graphs. In this paper, we propose a Temporal Multilayer Position-Aware Graph Neural Network (TMP-GNN), a node embedding approach for dynamic graphs that incorporates the interdependence of temporal relations into embedding computation. We evaluate the performance of TMP-GNN on two different representations of temporal multilayered graphs. The performance is assessed against the most popular GNNs on a node-level prediction task. Then, we incorporate TMP-GNN into a deep learning framework to estimate missing data and compare the performance with their corresponding competent GNNs from our former experiment, and a baseline method. Experimental results on four real-world datasets yield up to 58% lower ROC AUC for the pair-wise node classification task, and 96% lower MAE in missing feature estimation, particularly for graphs with a relatively high number of nodes and lower mean degree of connectivity.

Keywords: node embedding; Position-Aware Graph Neural Network; spatio-temporal measurements; temporal multilayer graph; missing data analysis

Citation: Najafi, B.; Parsaeefard, S.; Leon-Garcia, A. Missing Data Estimation in Temporal Multilayer Position-Aware Graph Neural Network (TMP-GNN). *Mach. Learn. Knowl. Extr.* **2022**, *4*, 397–417. https://doi.org/10.3390/make4020017

Academic Editor: Andreas Holzinger

Received: 10 April 2022
Accepted: 27 April 2022
Published: 30 April 2022

1. Introduction

Graph neural networks (GNN) have recently been used as a standard in developing machine learning methods for graphs. These graphs have been formed from sources such as transportation networks [1–4], brain networks [5], social media community networks, etc. The GNN architecture has effectively combined the node/edge features and graph topology to build a distributed representation. The resulting representation can be used to solve node-level, edge-level [6] and graph-level prediction tasks [5]. Several recent works have been applied to represent nodes, edges and graphs in many domains, such as bioinformatics, natural language processing and transportation. In [7], the authors used a hybrid method of a decomposed LSTM and a GCN to predict next period prescription. The authors in [8] proposed a GCN-based Relational Fusion Network (RFN) for speed limit classification and speed estimation in road networks. The study introduced the notion of volatile homophily in the road segments of adjacent regions and enumerated the substantial differences between the road networks and other networks previously learned by GCN. The underlying differences include a smaller number of node attributes, potential abrupt behavioral changes among the neighboring segments and a lower mean node degree. Another study [9] learned the graph representation by applying LSTM autoencoders on graph sequences generated from random walks, and utilized it for the classification task.

The goal of node embedding methods is to identify a vector representation that captures the node location within a broader topological structure of the graph. Most node embeddings learned from GNN architectures focus on single static graphs. These methods

assume that the number/position of nodes as well as their interactions do not change over time. However, in many applications, we tackle spatio-temporal measurements collected over time, wherein the necessity of time-varying graphs emerges. In such graphs, the number of nodes, their connecting edges and the edge weights vary from time to time; thus, dynamic node embedding should be learned accordingly from the graph. Node embedding approaches in such graphs can be mainly categorized as temporal-first [10,11] and structural-first [12,13] according to [14], in which the former and the latter prioritize learning temporal and structural information in their framework first and then feed static topological and dynamic temporal features into their model, respectively. Time-varying graphs can be defined as continuous- or discrete-time dynamic graphs, in which the latter can be represented as a sequence of interdependent time layers. Each layer is a graph building from existing nodes and weighted edges corresponding to a given time. For instance, the study in [5] proposed a graphSAGE-based graph representation learning for time-variant EEG signals to apply brain state classification, where the dynamic graph is modeled as a sequence of individual graphs per one time stamp. Another work [8] combines temporal, spatial and semantic views to predict taxi demands at a future time stamp.

One straightforward way of computing node embedding in discrete time-varying graphs is to use a static GNN-based node embedding for each individual time layer and aggregate the results of corresponding layers through recurrent neural network variants. However, this method would compute an embedding for each layer independently, and consequently ignore the inter-layer correlation. In this paper, we present TMP-GNN, a Temporal Multilayer Position-Aware GNN-based node embedding, which is an extension of its static version, Position-Aware GNN (P-GNN) [15]. The goal of P-GNN is to learn position-aware node embeddings that utilize the local network structure and the global network position of a given node with respect to randomly selected nodes called anchor sets, which enables us to distinguish among isomorphic nodes—nodes that are based in very different parts of the graph but have topologically the same structure. The distinguishing power will improve if the node/edge features are available. The resulting embeddings can be later used to approximately calculate the shortest path distance among the embedded nodes in the graph. The contributions of the paper are summarized as follows:

(1) We learn the short-term temporal dependencies, global position and feature information of the graph jointly through our TMP-GNN embedding component and utilize the derived representation in a missing data estimation framework.

(2) Instead of modeling a dynamic graph via multi-graph, we exploit a supra-adjacency matrix to encode a temporal graph with its intra-layer and inter-layer coupling in a single graph, which facilitates faster learning and a higher area under the receiver operating characteristic curve (ROC AUC) for pair-wise node classification.

(3) We deploy the concept of conditional centrality derived from eigenvector-based centrality to distinguish nodes of higher influence and integrate it in message aggregation across the graph.

(4) We identify a new experimental setting to enhance the training of the multi-graph deployment of dynamic graphs so that it can better trace the behavioral changes of nodes in the graph.

(5) We use hidden states learned from bi-directional GRU (bi-GRU) to learn the long-term temporal dependencies in both forward and backward directions to estimate missing values.

(6) We conduct several experiments using four real-world datasets, with a wide range of node numbers, degrees of connectivity, edge dynamics and areas of study. The results illustrate that TMP-GNN improves the area under the ROC curve significantly for classification tasks as compared to the best baseline, and it reduces the MAE of missing data estimation.

2. Notations and Preliminaries

Before presenting the details of our proposed architecture, we provide some background on temporal multilayer graphs.

2.1. Notation

Figure 1 illustrates a temporal multilayer graph. We can represent the graph as $\mathcal{G}^{(t)} = (\mathcal{V}, \rho^{(\mathcal{E}\times t)}, \mathcal{E}^{(t)}, \tilde{\mathcal{E}})$, where $\mathcal{V}$ are the set of nodes $\{v, \forall v \in \mathcal{V}\}$. Here, $\rho^{(\mathcal{E}\times t)}$ indicate whether a given edge is present at a given time t. $\mathcal{E}^{(t)}$ shows intra-layer weighted edges at time t, $\mathcal{E}^{(t)} = \{e_{uv}^{(t)}, \forall u, v \in \mathcal{V}\}$. $\tilde{\mathcal{E}}$ indicates inter-layer edges $\{(t, t', \tilde{x}_{tt'}) \in \tilde{\varepsilon}\}$, where $\tilde{x}_{tt'}$ is the edge weight between time layers (t, t'). The multilayer graph is also denoted by a sequence of adjacency matrices $\boldsymbol{A^{(t)}} \in \mathbb{R}^{N\times N}$, where $A_{uv}^{(t)}$ denotes the directed edge from node u to v in time layer t, as follows :

$$A_{uv}^{(t)} = \begin{cases} x_{uv}^{(t)}, & \text{if } u, v \text{ are connected at time } t \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

where $x_{uv}^{(t)}$ is the feature vector associated with $e_{uv}^{(t)}$. In scenarios where the node attributes are available, another set is defined as $W^{(t)} = \{w_v^{(t)}, \forall t \in \mathbb{N}_T\}$, where $w_v^{(t)}$ is the feature vector associated with node v in time layer t.

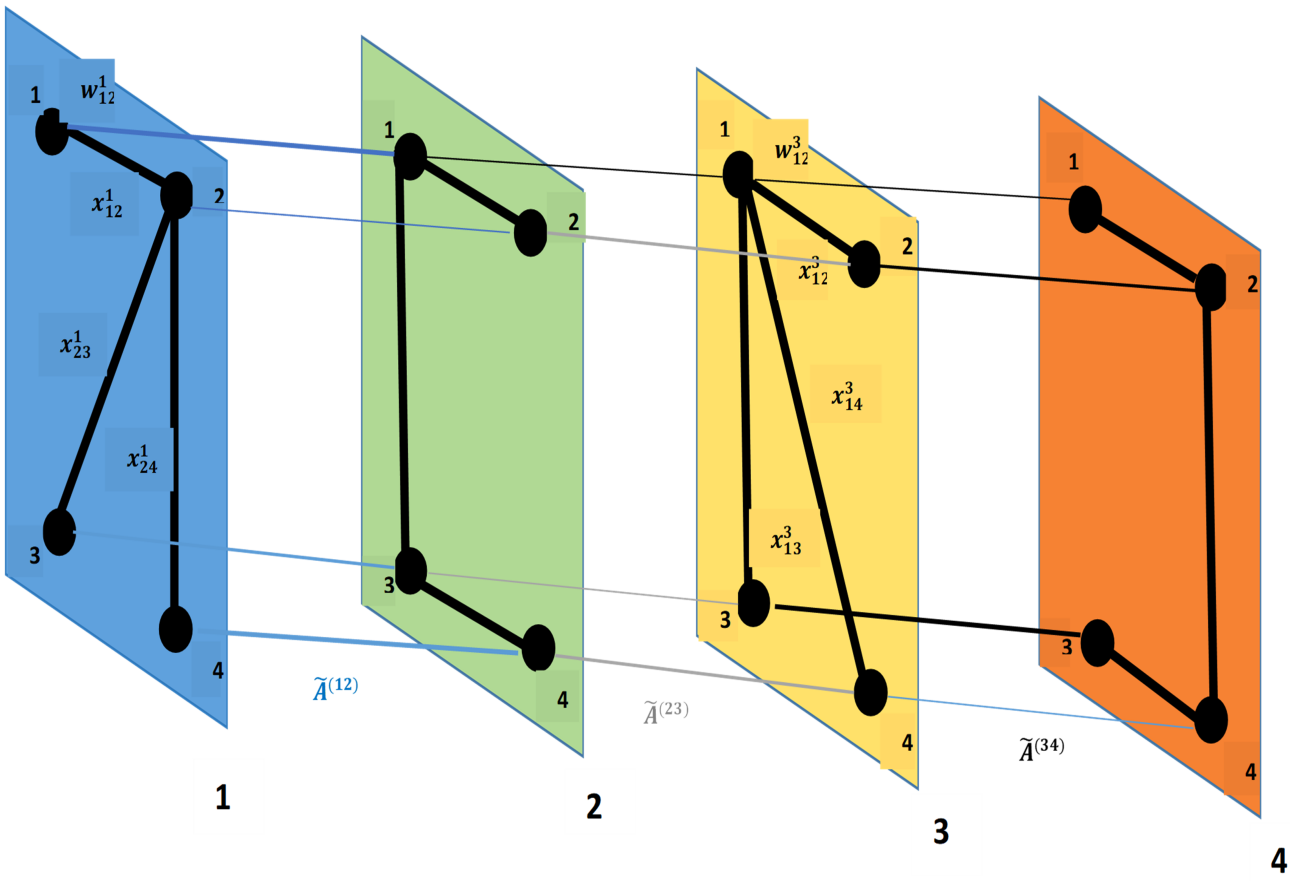


Figure 1. A dynamic multilayer graph.

The set $\tilde{\mathcal{E}}$ encodes the weight and topology of the coupling of individual instances of the same nodes between pairs of time layers (t, t').

The inter-layer coupling described above is diagonal and uniform [16], which means that the existence of inter-layer edges is restricted between separate instances of the same

nodes from one layer to another. For instance, there is no edge from node v in t to node u in t'. The corresponding layers are uniformly coupled, meaning that the inter-layer edge weight between two layers is identical for all nodes in those layers. Please note that this assumption can be generalized so that different inter-layer edge weights are assigned to different subsets of nodes; that is, $\tilde{x}^{u}_{tt'} \neq \tilde{x}^{v}_{tt'}$.

There are two possible ways to form the coupling between the layers: directed and undirected chain. The former couples the adjacent time layers by neglecting the directionality of time as in the following equation:

$$\tilde{A}^{(tt')}_{uv} = \begin{cases} 1 & \forall u, v \text{ if } u = v, |t' - t| = \delta \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where δ is the sampling rate. The latter not only respects the time direction, but also includes weighted undirected connections between the same nodes of all pairs of time layers as in:

$$\tilde{A}^{(tt')}_{uv} = \begin{cases} 1+\gamma, & \forall u, v \quad \text{if } u = v, |t' - t| = \delta \\ \gamma & \text{otherwise,} \end{cases} \tag{3}$$

where $\tilde{\boldsymbol{A}}$ is a $T \times T$ inter-layer matrix, and γ is the node teleportation probability [17]. In this paper, we have used the former approach for inter-layer coupling. The inter-layer coupling of the same nodes decrease as $|t' - t|$ increases, and we adjust δ according to the data characteristics and inter-sequence correlation in our experiments.

A node embedding function, e.g., SAGE, can be represented as $f\colon \mathcal{V}^{(t)} \rightarrow \mathcal{Z}^{(t)}$, which maps a given node v to d-dimensional vector $\boldsymbol{z}$ in layer t, where $\mathcal{Z}^{(t)} = \{\boldsymbol{z}^{(t)}_{v}, \forall v, t \in \mathbb{N}_{|\mathcal{V}|}, \mathbb{N}_{T}\}, \boldsymbol{z}^{(t)}_{v} \in \mathbb{R}^{d}$. Once the node embedding is computed, the corresponding edge embedding can be calculated as $\boldsymbol{z}^{(t)}_{uv} = g(\boldsymbol{z}^{(t)}_{u}, \boldsymbol{z}^{(t)}_{v})$, where the g function is equal to mean in the case of our study.

2.2. Supracentrality Matrix for Temporal Multilayer Position-Aware Graph

As explained, a temporal graph is represented through a sequence of adjacency matrices, each of which refers to one layer of a dynamic network at a specific point of time. Then, we construct a supracentrality matrix $\mathbb{C}(\omega)$ by linking centrality matrices across time layers through a weighted inter-layer parameter called ω, which is used to adjust the extent of coupling strength among pairs of time layers. The entries of the dominant eigenvector of $\mathbb{C}(\omega)$ show joint centrality, the importance of every node–layer pair (v, t). Additionally, marginal and conditional centralities are defined to represent the relative importance of a node compared to other nodes at time layer t.

The supracentrality framework is mainly focused on eigenvector-based centrality, which is obtained by calculating the centralities as the elements of the dominant eigenvector corresponding to the largest-magnitude eigenvalue of a centrality matrix $\mathcal{C}(\boldsymbol{A})$, which is defined as a function of network adjacency matrix $\boldsymbol{A}$. We have selected a centrality measure from one of the most popular choices to be equal to adjacency matrix ($\mathcal{C}(\boldsymbol{A}) = \boldsymbol{A}$).

$\mathbb{C}(\omega)$, a supracentrality matrix for a dynamic graph, is a group of matrices formed as below:

$$\mathbb{C}(\omega) = \hat{\mathbb{C}} + \omega\hat{\mathbb{A}}, \tag{4}$$

where

$$\hat{\mathbb{C}} = \begin{bmatrix} \boldsymbol{C}^{(1)} & 0 & 0 & \cdots & 0 \\ 0 & \boldsymbol{C}^{(2)} & 0 & \cdots & 0 \\ 0 & 0 & \boldsymbol{C}^{(3)} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \boldsymbol{C}^{(T)} \end{bmatrix},$$

and

$$\hat{\mathbb{A}} = \begin{bmatrix} \tilde{A}^{(11)}\boldsymbol{I} & \tilde{A}^{(12)}\boldsymbol{I} & \tilde{A}^{(13)}\boldsymbol{I} & \cdots & \tilde{A}^{(1T)}\boldsymbol{I} \\ \tilde{A}^{(21)}\boldsymbol{I} & \tilde{A}^{(22)}\boldsymbol{I} & \tilde{A}^{(23)}\boldsymbol{I} & \cdots & \tilde{A}^{(2T)}\boldsymbol{I} \\ \tilde{A}^{(31)}\boldsymbol{I} & \tilde{A}^{(32)}\boldsymbol{I} & \tilde{A}^{(33)}\boldsymbol{I} & \cdots & \tilde{A}^{(3T)}\boldsymbol{I} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \tilde{A}^{(T1)}\boldsymbol{I} & \tilde{A}^{(T2)}\boldsymbol{I} & \tilde{A}^{(T3)}\boldsymbol{I} & \cdots & \tilde{A}^{(TT)}\boldsymbol{I} \end{bmatrix}.$$

Here, $\mathbb{C}(\omega)$ consists of two components in (4): $\hat{\mathbb{C}}$ and $\omega\hat{\mathbb{A}}$. The former (diagonal) component, $Diag[\boldsymbol{C}^{(1)}, \cdots, \boldsymbol{C}^{(T)}]$, represents a set of T weighted centrality matrices of individual layers, and the latter (off-diagonal block), $\hat{\mathbb{A}} = \tilde{\boldsymbol{A}} \otimes \boldsymbol{I}$, encodes the uniform and diagonal coupling with strength parameter ω between the time layers, where $\tilde{\boldsymbol{A}}$ is defined in (2); $\otimes$ is the Kronecker product, and $\boldsymbol{I}$ is a $N \times N$ identity matrix, since we only consider the coupling among the same nodes between consecutive pairs of layers. We study the dominant eigenvector $\mathbb{V}(\omega)$ of $\mathbb{C}(\omega)$ corresponding to the largest eigenvalue $\lambda_{\max}$ as in:

$$\mathbb{C}(\omega)\mathbb{V}(\omega) = \lambda_{\max}(\omega)\mathbb{V}(\omega). \tag{5}$$

The elements in the dominant right eigenvector of $\mathbb{V}(\omega)$ are interpreted as scores that measure the importance of node–layer pairs (v, t). As such, the eigenvalue entity $\mathrm{W}_{v,t}(\omega)$ in the following Equation (6), called the joint centrality, is a d-dimensional vector, reflecting the centrality of node v at layer t, where $\mathbb{V}_{N(t-1)+n}(\omega)$ is the $N(t-1)+n$-th entity of the largest eigenvalue $\mathbb{V}(\omega)$ and n refers to node order v_n, which is omitted here for simplicity.

$$\mathrm{W}_{v,t}(\omega) = \mathbb{V}_{\mathrm{N}(t-1)+n}(\omega). \tag{6}$$

Inspired by probability theory, the authors in [16] defined marginal layer centrality (MLC) and conditional centrality in the following Equations (7) and (8). MLC indicates the average joint centralities over all nodes at time layer t. The conditional centrality of node v shows the importance of the node relative to other nodes at layer t.

$$\mathrm{MLC}_v(\omega) = \sum_{v=1}^{|\mathcal{V}|} \mathrm{W}_{v,t}(\omega), \tag{7}$$

$$CC_v(\omega) = \frac{\mathrm{W}_{v,t}(\omega)}{\mathrm{MLC}_v(\omega)}. \tag{8}$$

Choice of ω for Steady-State Node Ranking

A number of factors should be taken into account while choosing the value of ω. The choice of ω should be appropriate for the targeted application or the research question.

When $\omega \to \infty$, the centrality measures change slowly over time. On the other hand, in the presence of small ω, the $CC_v(\omega)$ fluctuates from one time layer to the other.

We considered ω at a strong coupling regime. If ω is too large (e.g., $\omega > 100$), the $CC_v(\omega)$ becomes stationary, which is equal to the average $CC_v(\omega)$ over all time layers [18]. Moreover, the dominant eigenvalue of $\mathbb{C}(\omega)$ converges to the same values of $\hat{\boldsymbol{A}}$. That is, $\lambda_{\max}(\omega) \to \tilde{\mu}_1$, where $\lambda_{\max}$ and $\tilde{\mu}_1$ are the dominant eigenvalues of $\mathbb{C}(\omega)$ and $\hat{\boldsymbol{A}}$, respectively.

Through the power iteration method of eigenvector measurements [19], we found the minimum value of ω that satisfies the above requirements (Algorithm 1). In the next section, we utilize stationary $CC_v(\omega)$ to distinguish highly important nodes belonging to P-GNN anchor sets, and then exploit it to aggregate the information across the anchor sets.

Algorithm 1 Find the appropriate value of ω

Require: Initialize the value of ω for $\omega \geq 0$ (we start from $\omega = 10$), construct $\boldsymbol{B} = \omega\hat{\mathbb{A}}$, $\boldsymbol{C} = \mathbb{C}(\omega)$
choose a random number for b_k, $c_{k'}$ as the initial value for the largest eigenvector of $\boldsymbol{B}$ and $\boldsymbol{C}$. We set b_k, $c_{k'} = [11 \cdots 1]$ with its length equal to the number of columns of $\boldsymbol{B}$, $\boldsymbol{C}$, which is $N \times T$ in our case.
$b_k \leftarrow \frac{b_k}{||b_k||}$, $c_{k'} \leftarrow \frac{c_{k'}}{||c_{k'}||}$
$\lambda_k \leftarrow b_k^* \boldsymbol{B} b_k$, $\mu_{k'} \leftarrow c_k^* \boldsymbol{C} c_k$
k=0
while $|\lambda_k - \mu_{k'}| < 0.01$ **do**
  while $|\lambda_{k+1} - \lambda_k| < 0.01$ **do**
    $k \leftarrow k + 1$
    $b_k = \frac{\boldsymbol{B} b_k}{||\boldsymbol{B} b_k||}$
    $\lambda_k = b_k^* \boldsymbol{B} b_k$
  end while
  return λ_k, b_k
  while $|\mu_{k'+1} - \mu_{k'}| < 0.01$ **do**
    $k' \leftarrow k' + 1$
    $c_{k'} = \frac{\boldsymbol{C} c_{k'}}{||\boldsymbol{C} c_{k'}||}$
    $\mu_{k'} = c_{k'}^* \boldsymbol{C} c_{k'}$
  end while
  return $\mu_{k'}$, $c_{k'}$
end while
return ω

2.3. TMP-GNN: Multilayer Position-Aware-Based Graph Neural Network

Several GNN methods are distinguished by the way in which they aggregate nodes' information from their neighborhood to form a node representation at each layer k. The approach can be summarized as two functions, AGGREGATE and COMBINE [20], in which the former decides on how the information is aggregated from adjacent nodes, and the latter updates the node representation from the previous layer $k-1$ to layer k. The underlying functions can be combined and jointly represented for some GNNs, as follows:

$$\begin{aligned} \boldsymbol{a}_{v_n}^{(k)} &= \text{AGGREGATE}^{(k)}(\{\boldsymbol{h}_{u_n}^{(k-1)} : u_n \in \text{neighbors}(v_n)\}), \\ \boldsymbol{h}_{v_n}^{(k)} &= \text{COMBINE}^{(k)}(\boldsymbol{h}_{v_n}^{(k-1)}, \boldsymbol{a}_{v_n}^{(k)}), \end{aligned} \tag{9}$$

where $h_v^{(k)}$ is the feature representation vector of node v at iteration k/k-th layer of GNN, and Neighbors(v) is the set of nodes adjacent to v in $\mathcal{G}$.

We use a position-aware GNN, which, instead of aggregating the information from the nearest neighbors, aggregates the positional and feature information of each node with a randomly selected number of nodes called anchor sets. Then, the computed message is aggregated across the anchor sets for each node to incorporate the global topological information of the graph in node embedding. Figure 2 illustrates in detail how the node representation is learned through one P-GNN layer. The goal is to find the best position-aware embedding $\boldsymbol{z}_v^t$ with minimum distortion for a given node v at time layer t.

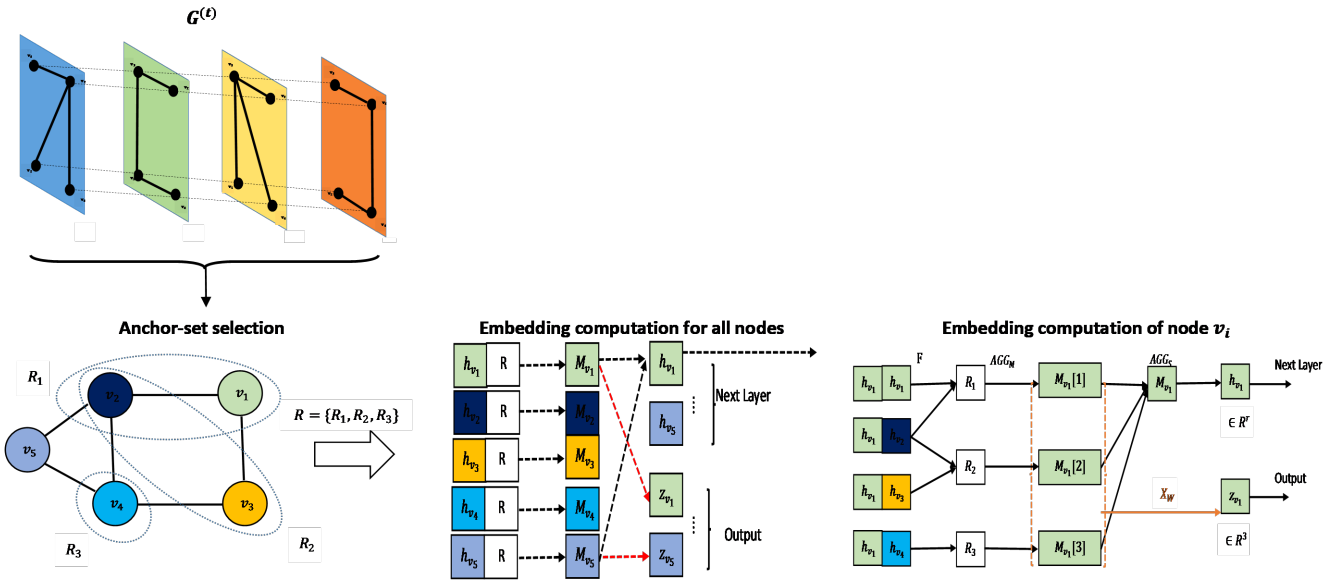


Figure 2. The input to our TMP-GNN layer is a temporal graph $\mathcal{G}^{(t)}$, presented as a supragraph and the corresponding supracentrality matrix $\mathbb{C}(\omega)$. As the first step, anchor sets, a subset of graph nodes with different sizes, are randomly selected according to Bourgain's theorem [21] (Reproduced with written permission) $\mathcal{R} = R_1, R_2, \ldots, R_J$. In order to compute the embedding for v_n, feature information of node v_n and another node from anchor set R_j, is fed into a message computation function $\mathcal{F}$ with output aggregated through AGG_M to produce $\boldsymbol{M}_{v_n}[j], j = 1, 2, .., J$ corresponding to R_j. Then, $\boldsymbol{M}_{v_n}[j]$ is further aggregated through AGG_R to output h_{v_n}, which is passed to the next P-GNN layer. The embedding z_{v_n} at the last layer is generated by applying the nonlinear σ to the inner product of $\boldsymbol{M}_{v_n}$ and weight vector $\boldsymbol{w}$ (You, J., Ying, R. and Leskovec, J., 2019 [15]—modified, permission is obtained).

We have made the following modifications to P-GNN:

- *Generalization of P-GNN to Time-Varying Graphs*:
  We adopt the input of P-GNN as supracentrality matrix $\mathbb{C}(\omega)$, which represents a temporal multilayer graph with $N \times T$ number of nodes. We compute an embedding for all nodes in all time layers, since $e_{uv}/x_{uv}/w_{uv}$ defined in (1) can change from t to t'. The embedding z_v^t will then be aggregated from an RNN-based representation to estimate missing data.
- *Modification of Message Computation Function (F)*:
  In an Intelligent Transportation System (ITS), the average speed of neighboring nodes might correlate more or less at different time layers due to a variety of factors, i.e., different types of residential zone, special events, accidents, etc. Using an attention mechanism while computing an anchor set's message with respect to a given node v can alleviate misinformative messages from the anchor set to influence node v's embedding. Therefore, we use the attention mechanism to learn the relative weights between the feature vector of v and its nearest neighbor from the anchor set. The fact at different degrees could apply to other application domains as well. As such, we modify our message computation function to incorporate the attention mechanism.
  In P-GNN, we have multi-level aggregations, as demonstrated in Figure 2. First, the message of each node v is computed with regard to each anchor set R_j through function F. From each anchor set, only the nodes with an up to two-hop distance to node v are considered for message computation, as shown in the following equation:

$$\begin{aligned} \mathcal{M}_j &= \{F(u, v, \boldsymbol{h}_u^{k-1}, \boldsymbol{h}_v^{k-1}) | u \in R_j, R_j \subset \mathcal{R}, \mathcal{R} \sim \mathcal{V}, \\ d(v, R_j) &= \min_{u \in R_j} d(u, v) \leq 2\}, \\ \boldsymbol{M}_v^k[j] &= \text{MEAN}(\mathcal{M}_j), \end{aligned} \tag{10}$$

where $\boldsymbol{M}_v^k[j]$ indicates the aggregated message of v corresponding to R_j, which is the average of individual outputs of $F(u, v, \boldsymbol{h}_u^{k-1}, \boldsymbol{h}_v^{k-1})$, where F is defined as:

$$F(u, v, \boldsymbol{h}_u^{k-1}, \boldsymbol{h}_v^{k-1}) = \frac{1}{d_{sp}^q(u, v) + 1} \boldsymbol{a}_v, \tag{11}$$

where $d_{sp}^q(u, v)$ is defined as below:

$$d_{sp}^q(u, v) = \begin{cases} d_{sp}(u, v) & \text{if } d_{sp}(u, v) \leq q, \\ \infty, & \text{otherwise,} \end{cases} \tag{12}$$

where $d_{sp}(u, v)$ is the shortest path between a pair of nodes (u, v), and we write $\boldsymbol{a}_v = w_c(\text{CONCAT}(\boldsymbol{c}_v, \boldsymbol{h}_v))$, where $\boldsymbol{c}_v$ is denoted as $\boldsymbol{c}_v = \sum_{u'} \boldsymbol{\alpha}_{vu'} \boldsymbol{h}_{u'}$ and $\boldsymbol{\alpha}_{vu'}$ is computed as follows:

$$\boldsymbol{\alpha}_{vu'} = \frac{\exp\left(\text{score}(\boldsymbol{h}_v^{k-1}, \boldsymbol{h}_{u'}^{k-1})\right)}{\sum_{u'} \exp\left(\text{score}(\boldsymbol{h}_v^{k-1}, \boldsymbol{h}_{u'}^{k-1})\right)}, \tag{13}$$

where the score is calculated as:

$$\text{score}(\boldsymbol{h}_v^{k-1}, \boldsymbol{h}_{u'}^{k-1}) = V^T(tanh(W_1 \boldsymbol{h}_v^{k-1} + W_2 \boldsymbol{h}_{u'}^{k-1})),$$

where V, W_1, W_2 and w_c are trainable weights. $\boldsymbol{c}_v$ and $\boldsymbol{a}_v$ are the context vector and attention coefficient, respectively. This is inspired by [22].

- *Modification of AGG_R*:
In P-GNN, $\boldsymbol{M}_v[j]$, associated with anchor set j, is averaged across the anchor sets to generate $\boldsymbol{h}_v$. We choose to differentiate nodes based on their conditional centrality in stationary status $(CC_v(\omega)|_{\omega \to \infty})$, as higher conditional eigenvector centrality indicates a higher influence of a given node v and its surrounding nodes compared to the ones with lower eigenvector centrality. Additionally, corresponding informative anchor sets contain at least 2-hop neighbor(s) of node v, wherein the ones with higher $CC_v(\omega)$ deserve to have higher weights for aggregation. As such, we substitute AGG_R by the weighted mean of $\boldsymbol{M}_v[j]$, where the weights are proportionate to stationary $CC_v(\omega)$, as follows:

$$\boldsymbol{h}_v^{(k)} = \frac{1}{J} \sum_j r_j \boldsymbol{M}_v[j], \forall j \in [1, J] \tag{14}$$

where r_j is calculated as:

$$\begin{aligned} r_j &= \frac{\sum_{u' \in R_j} CC_{u'_j}(\omega)}{\sum_{j'=1}^{J} \sum_{u \in R_{j'}} CC_{u_{j'}}(\omega)} |_{\omega \to \infty}, \\ & d(v, u') \leq 2, \end{aligned} \tag{15}$$

where j is the number of anchor sets. Calculation of the conditional node centrality in stationary condition is implemented separately from the main algorithm, and the result has been used in AGG_R to aggregate the computed messages across the anchor sets.

Anchor sets are selected randomly based on Bourgain's theorem; they come in different sizes that distribute exponentially. $\boldsymbol{M}$ is computed for node v from anchor set R_j, if R_j hits v, which means $d(v, R_j) \leq 1$. Large anchor sets have a higher probability of hitting v, but are less informative of the positional information of the node, as v hits at least one of many nodes in the anchor sets. On the other hand, small anchor sets have a lower chance of hitting v; however, they provide positional information with high certainty [15]. In terms of message communication, each node in a P-GNN layer shares a message with $\mathcal{O}(n \log^2 n)$ number of anchor sets for a graph with n nodes. Suppose that we have a temporal graph with $n' = (|\mathcal{V}| \times T)$ and average anchor set size of m; the total communicated messages will be $\mathcal{O}(mn' \log^2 n')$. This is because each node in TMP-GNN only communicates with nodes that are up to two-hop distant within each anchor set R_j, so that m is limited ($1 \leq m << n'$).

At the end of the last P-GNN layer, the node embedding $\boldsymbol{z}_v$ is calculated after applying a nonlinear σ to an inner product of a weight vector $\boldsymbol{w}$ and $\boldsymbol{M}_v$ as in $\boldsymbol{z}_v = \sigma(\boldsymbol{w}\boldsymbol{M}_v)$. The output embedding of for all nodes at all layers t can be reformed into a supraembedding matrix $\mathbb{Z}$ as follows:

$$\mathbb{Z} = \begin{bmatrix} z^1_{v_1} & z^2_{v_1} & z^3_{v_1} & \cdots & z^T_{v_1} \\ z^1_{v_2} & z^2_{v_2} & z^3_{v_2} & \cdots & z^T_{v_2} \\ z^1_{v_3} & z^2_{v_3} & z^3_{v_3} & \cdots & z^T_{v_3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ z^1_{v_{|\mathcal{V}|}} & z^2_{v_{|\mathcal{V}|}} & z^3_{v_{|\mathcal{V}|}} & \cdots & z^T_{v_{|\mathcal{V}|}} \end{bmatrix}, \tag{16}$$

where $\boldsymbol{z}^t_v$ is the embedding vector of node v at time t. In some scenarios, edge embedding is needed, rather than node embedding. We estimate $e_(uv)$ embedding by averaging the embedding of the ending nodes as:

$$\boldsymbol{z}_{uv} = \frac{1}{2}(\boldsymbol{z}_u + \boldsymbol{z}_v) \tag{17}$$

We utilize the result of TMP-GNN embedding for missing data estimation. Before proceeding to our proposed architecture, we briefly review the other components of our framework for missing data.

2.4. Bi-Directional Recurrent Neural Network (Bi-GRU)

We use bi-directional GRU as a component in our proposed architecture to estimate missing points in the temporal multilayer graph $\mathcal{G}$. The Bi-GRU [23] (Figure 3) captures the temporal correlation within a time layer in both forward and backward directions. The input to Bi-GRU is a triplet array $\boldsymbol{Y}_e$, $\boldsymbol{M}_e$ and $\boldsymbol{\Delta}_e$ that is produced from the graph feature set, $\mathbb{X} = \{x^1_{11}, x^2_{11}, \ldots, x^t_{uv}, \ldots, x^T_{uv}, \ldots, x^T_{u'v'}\}$, where x^t_{uv} is the feature associated with e_{uv} at time layer t. We sometimes refer to x_{uv} as x_e interchangeably, where e could be any edge in the graph. We also assume that the graph nodes are constant. However, the number of edges and their corresponding features may change from one layer to another. We set x^t_{uv} to 0, if e_{uv} does not exist at time t ($\rho^{(e_{uv} \times t)} = 0$). We also choose to randomly remove missing points and set it to 0 in $\boldsymbol{Y}_e$. $\boldsymbol{M}_e$ is called the mask array containing 0 s and 1 s, which indicates the coordinates of missing and observed points, respectively, corresponding to $\boldsymbol{Y}_e$. Additionally, each element in $\boldsymbol{\Delta}_e$ illustrates the time difference between the current and the last layer at which the measurement is recorded. $\boldsymbol{\Delta}_e$ is defined to handle the different sampling rates associated with data heterogeneity from different sources [24,25]. Each edge contains D streams of features. We use x_{e_d} to represent stream d of the feature associated with e, where e is an edge connecting a pair of nodes in the graph $\mathcal{G}$.

Our goal is to find the best estimate $\hat{x}^t_{e_d}$ with minimum RMSE for a particular missing point through solving the following optimization problem and finding the function f as defined below.

$$\min_{\mathrm{f}} \boldsymbol{E}_{\mathcal{F}}\left[\sum_{t=1}^{T}\sum_{d=1}^{\mathrm{D}}(1-m^t_{e_d})\mathcal{L}(\mathcal{X}^t_{e_d},\widehat{\mathcal{X}}^t_{e_d})\right], \tag{18}$$

where the loss function is defined as $\mathcal{L}(\mathcal{X}^t_{e_d},\widehat{\mathcal{X}}^t_{e_d}) = (\{x^t_{e_d} - f^t_{e_d}(\mathcal{X},\mathcal{T}))^2, \mathcal{X} = \{x^t_{e_d}, \forall e, \forall d, \forall t\}$, $\mathcal{T} = \{1, 2, \ldots, T\}$. $\boldsymbol{E}$; f and $\mathcal{F}$ indicate expected values for the desired estimator function and unknown probability distribution that the records are sampled from, respectively. $\mathcal{X}^t_{e_d}$ shows stream d of all edges' features at time t, and m_{e_d} represents one element of the mask array $\boldsymbol{M}_e$ defined above.

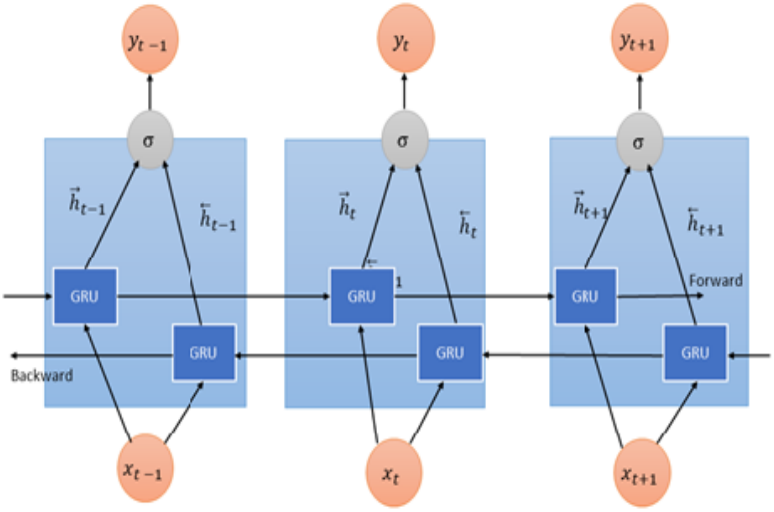


Figure 3. Bi-directional gated recurrent unit.

2.5. E-TMP-GNN: Extended TMP-GNN for Missing Data Estimation

Figures 4 and 5 illustrate our proposed hybrid architecture of TMP-GNN and Bi-GRU. We propose two architectures: E-TMP-GNN I and E-TMP-GNN II. In E-TMP-GNN I, the input is supracentrality matrix $\mathbb{C}(\omega)$, which encodes the intra-layer and inter-layer edge weights of multilayer graph $\mathcal{G}$. The TMP-GNN's output $\mathbb{Z}$ encodes the topological structure and positional and feature information across the nodes within a layer, and among the same nodes between consecutive layers. Thereafter, the resulting d-dimensional embedding $\mathbb{Z}$ is used to calculate edge embedding as in (16), which is then passed to the input of Bi-GRU to merge with existing edge features and form the below array.

$$\mathbb{X}_{\text{TMP-PGNN}} = \{x^{(t)}_{e_{d_i}}, z^{(t)}_{e_{d_i}} \forall i \in \mathbb{N}_{\mathrm{D}}, \forall e \in \mathcal{E}^{(t)}, \rho^{(e\times t)} = 1\} \tag{19}$$

The Bi-GRU is an RNN variant wherein the output of the previous layer is a part of the current layer's input. This characteristic allows the information to propagate step by step [23] and enables the Bi-GRU to capture long-term temporal correlations of edges in two directions, which is advantageous for missing data imputation. We use one layer of Bi-GRU and form a triplet array of $\boldsymbol{M}_e$, $\boldsymbol{Y}_e$ and $\boldsymbol{\Delta}_e$, as explained earlier in this section. An initial estimation, $\tilde{x}^t_{e_d}$, is made by applying a nonlinear activation function to the last hidden states of Bi-GRU, $h^{(t)}_{hybrid_{e_d}}$, as below:

$$\tilde{x}^t_{e_d} = \sigma(U_d[\overrightarrow{h}^{(t)}_{hybrid_{e_d}}; \overleftarrow{h}^{(t)}_{hybrid_{e_d}}] + g_d), \tag{20}$$

where the two arrows indicate the forward and backward direction of Bi-GRU, respectively. U_d and g_d are weight and bias vectors. In order to capture the correlation among sequences of $x_{e_{d_i}}|_{i=1}^{\mathrm{D}}$, a fully connected layer with a hidden dimension equal to the number of feature streams D is placed afterward. The output of this layer $\hat{x}_{e_d}^t$ is the final estimate of $x_{e_d}^t$.

$$\hat{\boldsymbol{x}}^{(t)} = \sigma(\boldsymbol{W}\boldsymbol{h}^{(t)} + \beta_0), \tag{21}$$

where $\boldsymbol{h}^{(t)}$ is defined as follows:

$$\boldsymbol{h}^{(t)} = \phi(U\boldsymbol{x}^{(t)} + V\boldsymbol{y}^{(t)} + \gamma_0), \tag{22}$$

where $\boldsymbol{y}^{(t)}$ is denoted as $[\boldsymbol{x}^{(t)}, \boldsymbol{m}^{(t)}]$.

The Bi-GRU and FC components apply interpolation and imputation, respectively, as they capture the dynamics within a time layer ($\{\boldsymbol{x}_{e_d}^t, \forall t \in \mathbb{N}_T\}$), and across the streams of features ($\{\{x, z^{T*}\}_{e_d}^{(t)}, d = \{1, 2, \cdots, D\}\}$) simultaneously. The combination of Bi-GRU and FC, named Multi-directional RNN, is inspired by [24] (see Figure 6).

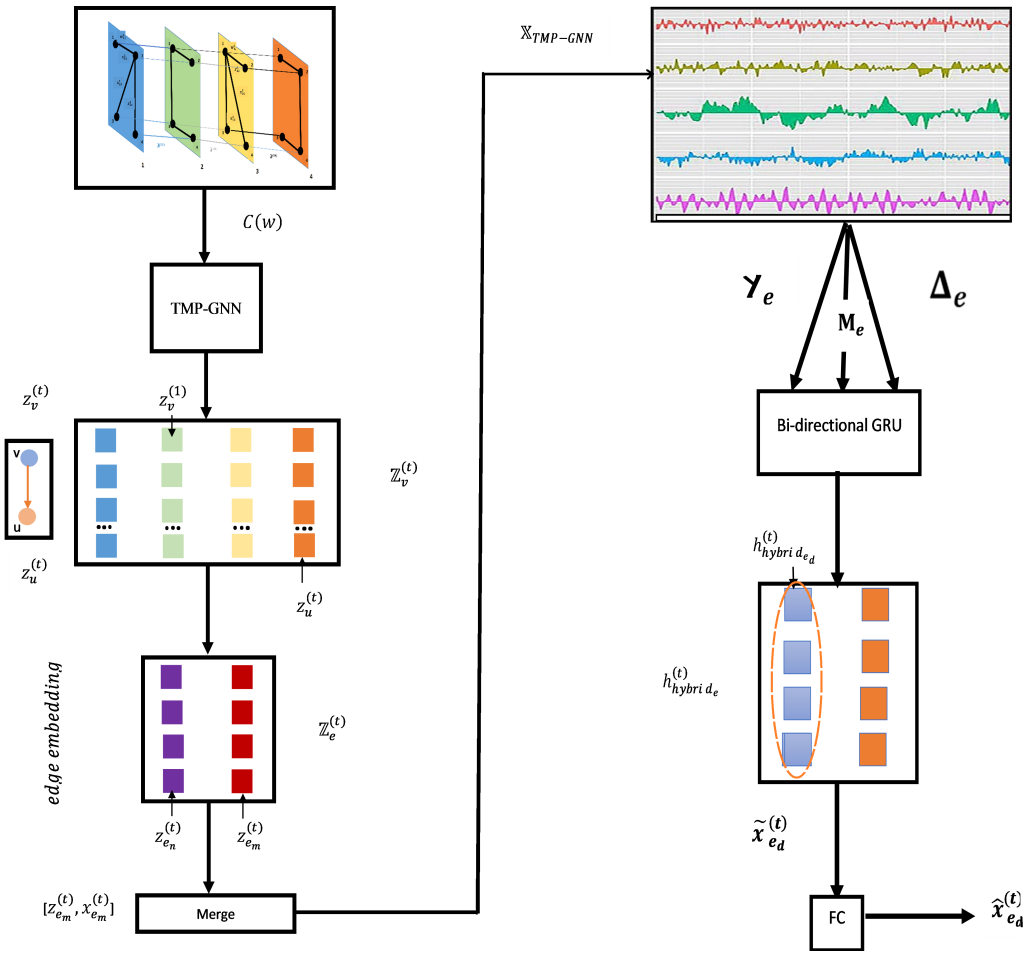


Figure 4. E-TMP-GNN I.

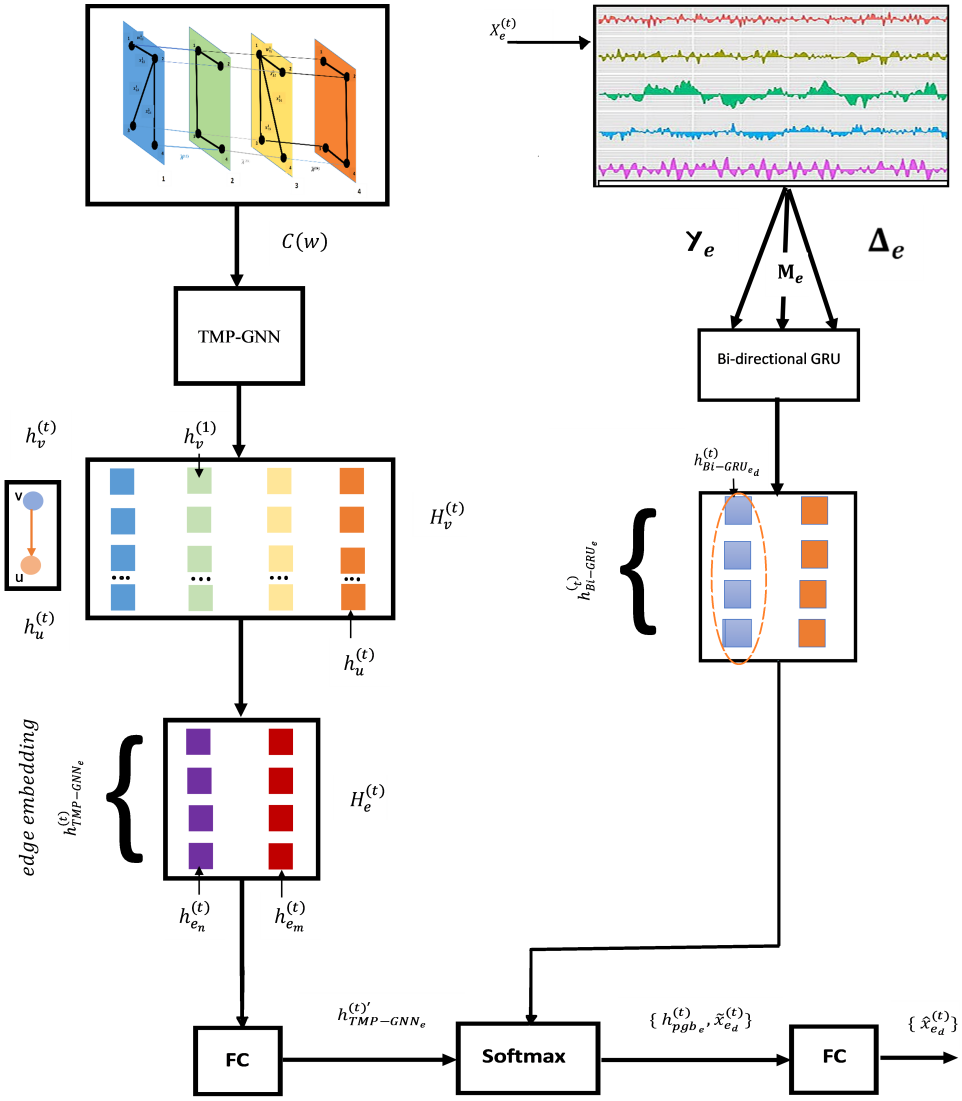


Figure 5. E-TMP-GNN II.

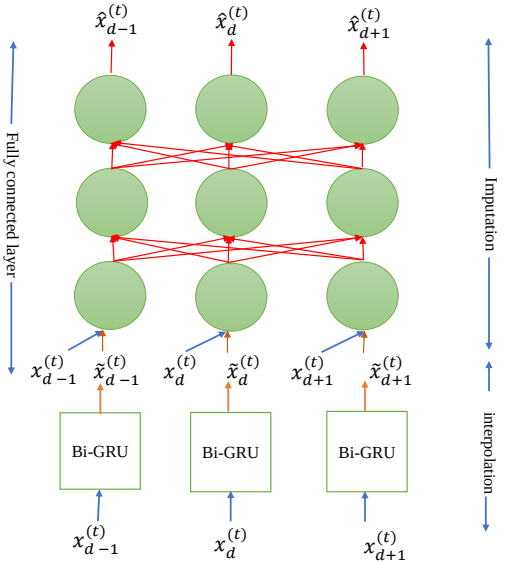


Figure 6. M-RNN architecture in time domain based on the theory in [24].

Our second proposed architecture is illustrated in Figure 4. As can be seen, the two pipelines have distinct inputs. The input to TMP-GNN is the temporal multilayer network represented as suppracentrality matrix $\mathbb{C}(\omega)$, and we feed Bi-GRU through $\mathbb{X}$. The last $h^{(t)}_{\text{TMP}-\text{GNN}_e}$ output from TMP-GNN is then passed through a fully connected layer to reduce the representation dimension as in the following equation:

$$\boldsymbol{h'}^{(t)}_{\text{TMP}-\text{GNN}_e} = \sigma(\boldsymbol{w}_p \boldsymbol{h}^{(t)}_{\text{TMP}-\text{GNN}_e} + \boldsymbol{b}_p), \tag{23}$$

where $\boldsymbol{w}_p$ and $\boldsymbol{b}_p$ are weight and bias vectors, respectively. The output is then merged by $\boldsymbol{h}^{(t)}_{\text{Bi}-\text{GRU}_e} = [\overrightarrow{\boldsymbol{h}}^{(t)}_{\text{Bi}-\text{GRU}_e}; \overleftarrow{\boldsymbol{h}}^{(t)}_{\text{Bi}-\text{GRU}_e}]$ through softmax attention, $\boldsymbol{h}_{\text{pgb}_e}$ is generated as follows:

$$\boldsymbol{w'}_p, \boldsymbol{w'}_b = \text{Softmax}(\boldsymbol{w}_p, \boldsymbol{w}_b), \tag{24}$$

$$\boldsymbol{h}^{(t)}_{\text{pgb}_e} = \boldsymbol{w'}_p \boldsymbol{h'}^{(t)}_{\text{TMP}-\text{GNN}_e} + \boldsymbol{w'}_b \boldsymbol{h}^{(t)}_{\text{Bi}-\text{GRU}_e} \tag{25}$$

and an initial estimate of the missing value is made as:

$$\tilde{x}^{(t)}_{e_d} = \sigma(Q^d \boldsymbol{h}_{\text{pgb}_{e_d}} + \alpha^d_0) \tag{26}$$

A fully connected neural network is used at the end to finalize the prediction of missing values as follows:

$$\hat{x}^{(t)}_{e_d} = \sigma(\boldsymbol{w}_1 \boldsymbol{h}^d_{\text{pgb}_e} + \gamma_1) \tag{27}$$

The E-TMP-GNN I aims at extracting additional features out of the embedding that is yielded from TMP-GNN, and use it to further enrich the edge feature sets. We demonstrate in Section 3 that the added streams provide a significant improvement in MAE compared to the baseline. TMP-GNN II, however, aims at reducing the number of feature streams by implementing a softmax layer that obtains $\boldsymbol{h}^{(t)}_{\text{Bi}-\text{GRU}_e}$ and $\boldsymbol{h}^{(t)}_{\text{TMP}-\text{GNN}_e}$ as inputs, since $\boldsymbol{h}^{(t)}_{\text{TMP}-\text{GNN}_e}$ is expected to contain valuable information about the temporal correlation among the same nodes of consecutive time layers.

3. Performance Evaluation

We divide this section to three parts. First, we review our real-world datasets and their associated characteristics in depth. Then, we discuss two potential inputs to our proposed TMP-GNN pipeline and their impact on node embedding performance. Thereafter, we evaluate the performance of E-TMP-GNN I and E-TMP-GNN II on missing data estimation. We implemented E-TMP-GNN I and E-TMP-GNN II in python using Networkx (https://github.com/networkx, accessed on 9 April 2022), PyTorch (https://github.com/pytorch, accessed on 9 April 2022) and Tensorflow (https://github.com/tensorflow/tensorflo, accessed on 9 April 2022). We built in and added various components to the P-GNN implementation [15].

3.1. Datasets

We run our experiment on four datasets: TomTom, COVID-19 Mobility (Mobility) [26], PhD Exchange (PhD), and the Seattle Inductive Loop Detector (Loop Detector).

TomTom Dataset: This dataset [27] contains space mean speed and average travel time during peak congestion for hundreds of approximately one-km-long road segments across the Greater Toronto Area (GTA). The measurements are collected on a one-minute basis during congestion, where the average space mean speed is less than 70% of the vehicular speed during free-flow conditions. We select a three-hour interval from 4:30 p.m. to 7:30 p.m. on Thursday, 8 September 2016, which is expected to have a high proportion of collected records. During this interval, we select road segments with a significant number of mutual timestamps, where measurements for a sufficient number of the segments are conducted. From the point of view of the multilayer network, we select 10 layers of the GTA road

network of 1867 nodes and 985 edges, where each node and edge is the start/ending point of a road segment and a road segment, respectively. We define each edge feature as the ratio of $\frac{\text{space mean speed}}{\text{free-flow speed}}$ and $\frac{\text{free-flow travel time}}{\text{travel time}}$ that is in range of [0, 1]. To reduce the correlation between the consecutive records and better assess the performance of pair-wise node classification in TMP-GNN and missing data estimation, the records are downsampled by a rate of $\frac{8}{1}$ ($\delta = \frac{1}{8}$). The edge weights are also normalized so that the features are comparable during the learning process. Moreover, the weighted adjacency matrix is created using the highly accurate geolocations of the road segments in the GTA. Once the node embedding is calculated through TMP-GNN, the edge embedding is estimated by averaging the embedding of the corresponding nodes that the edge is connected to as in (17). The resulting embedding will then be used in missing data estimation.

PhD Dataset: This case of study utilizes a network that represents the exchange of PhD graduates between universities in the fields of mathematical science. The features we use in our experiments include the graduation year of the students who obtained a doctoral degree, his/her official academic advisor(s) and the degree-granting university. The aim of former studies was to estimate the flow of doctorate exchanges between universities, individuals who graduate from university a and then are hired at the university b. We consider the years 1950–2010, which result in $T = 61$ time layers, and there exists a set of $N = 231$ connected universities during this time span in the united states. In order to build the graph, directed intra-layer edges are created to represent a doctoral degree obtained by a graduate from university a at year t, who is later hired as a professor at university b, and at least advised one student there. In our case, the edge weight indicates the number of graduate doctorates from university a in year t who was later hired at university b as a faculty member. The edge direction opposes the flow of PhD graduates moving between universities $(b \rightarrow a)$ [16,28].

Loop Detector Dataset: Our third case of study is the data collected by inductive loop detectors on freeways in the Seattle area [3,29]. The dataset contains temporal sequences of the time mean speed of the freeway system. The speed information at a desired reference point (milepost) is averaged from multiple loop detectors on the main lanes of the same direction at the specific reference point. Measurements are recorded every 5 minutes. The adjacency matrix calculated in [29] is used to convert the data into an undirected graph where an edge indicates that a pair of reference points are connected, without specifying the direction of connectivity. We have downsampled the dataset by a rate of $\frac{2}{1}$ ($\delta = \frac{1}{2}$) and selected 11 time layers of the graph.

Mobility Dataset: Our last case of study is the human mobility flow dataset in the U.S. during the COVID-19 epidemic [26]. To build the graph, a directed intra-layer edge is constructed to represent the flow of people between pairs of states. Each edge indicates the ratio of visits to the population count among the states. We focused on January 1st–22nd 2020, which led to 22 time layers.

Table 1 demonstrates the characteristics associated with the graphs built from the 4 selected datasets.

Table 1. Characteristics of the study datasets.

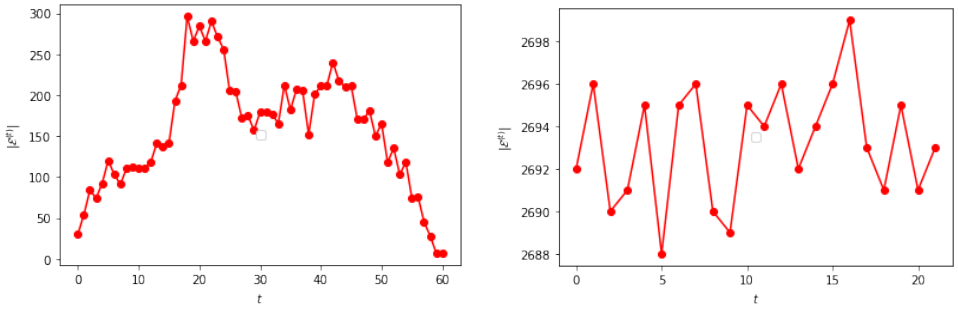
Characteristics	TomTom	Loop Detector	PhD	Mobility
$\lvert\mathcal{V}\rvert$	1867	323	231	72
static : $\lvert\mathcal{E}\rvert$, dynamic *: $\sum_{t=1}^{T} \mathcal{E}^{(t)}$	985	1001	10,365 *	2692
$\lvert\mathcal{V}\rvert \times T$	18,670	3553	14,091	1584
Largest Connected Component (LCC)	50	323	13,847	1144
No. of Isolated Nodes	18,620	3230	244	440

* indicates the graph with a dynamic number of edges over time.

Among all, TomTom and Mobility have the highest and the lowest number of nodes, respectively. From an edge density perspective, we have ordered the graph from the most connected to the least connected as follows: Mobility, Loop Detector, PhD and TomTom.

From a topological point of view, TomTom and Loop Detector are static graphs. That is, $\mathcal{E}^{(t)} = \mathcal{E}^{(t')}, t \neq t'$ and the change is in the value of $x_{uv}^{(t)}/w_v^{(t)}$ only. However, the PhD and Mobility graphs change throughout the time layers in terms of $|\mathcal{V}^{(t)}|, x_{uv}^{(t)}$ and $w_v^{(t)}$. Among the two, PhD has the highest dynamic rate, and turns into the sparsest graph in multiple time layers. Figure 7 demonstrates $|\mathcal{E}^{(t)}|$ per layer t for these graphs.

The third row in the table shows the number of nodes in the temporal multilayered graph, explained in Section 2.2, which is equal to $(|\mathcal{V}| \times T)$. TomTom and PhD have the largest numbers of nodes in the temporal graph given the number of available time layers. The fourth row indicates the size of the largest connected component (LCC) in https://www.overleaf.com/project/6251d321650dd619f849d51c (accessed on 9 April 2022) for the dynamic graph. The parameter is considered as an important topological invariant of the graph and can be computed using a variant of depth-first search for directed and undirected graphs [30]. Here, graphs can be ordered according to ratio of $\frac{\text{LCC}}{|\mathcal{V}|}$ in descending order as follows: Loop Detector, PhD, Mobility, and TomTom. The entire Loop detector undirected graph is considered as one connected component; TomTom as a directed graph has multiple weakly connected components, with each component as small as 2% of the total $|\mathcal{V}|$. We calculate the number of isolated nodes as the nodes that are not connected to the largest component by this equation: no. of isolated nodes $= N \times T -$ LCC, which could be another indicator of nodes with lower node degrees and, thus, sparser graphs. TomTom has the largest number of isolated nodes, as expected.



(**a**) PhD dataset (**b**) Mobility dataset

Figure 7. Number of edges per time layer.

3.2. Potential Inputs to TMP-GNN: Single Graph vs. Multiple Graphs

There are several ways in which we can feed P-GNN:

- *Supracentrality Graph* $\mathbb{C}(\omega)$: In this case, we represent the centrality of a temporal multilayer graph via a supracentrality matrix that is analogous to a single graph with $N \times T$ number of nodes. The main advantage of this approach is to illustrate the graph with all weighted intra-layer and inter-layer edges. However, the matrix size might be relatively large in the presence of a large number of T. In the next section, we demonstrate that this approach outperforms the one with a multiple-graph input.
- *Multiple Graphs*: In this approach, a sequence of centrality/adjacency matrices associated with each time layer, $\{\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \cdots, \boldsymbol{A}^{(T)}\}$, is fed into P-GNN. That is, the multilayer graph is treated as individual instances of a single graph without consideration of inter-layer coupling.
- The input to P-GNN can be a single graph with the entry of its corresponding adjacency matrix as follows:

$$a_{uv} = \begin{cases} [x_{uv}^{(1)}, x_{uv}^{(2)}, \cdots, x_{uv}^{(T)}] & : \text{if } u, v \text{ are connected,} \\ 0 & : \text{otherwise.} \end{cases} \tag{28}$$

In this approach, we treat the multilayer graph as a single graph where each element of the adjacency matrix is a time series of features associated with an e_{uv}. In order to

capture the temporal characteristics of a v/e_{uv}, a recurrent neural network is required before applying function F in Figure 2, which leads to high computational complexity. We have not implemented this method.

Training: There are two different ways to train the time-varying graph to compute node embedding. In the case of a single graph, graph nodes are split into training, evaluation, and testing, with 80%, 10%, and 10% respectively. The classification accuracy on the test set is recorded once the best performance on the evaluation is reached. As for the multiple graphs, there are two techniques to train the graph: first, 80% of the graphs can be used for training and the remaining for evaluation and the test set. Second, 80% of the nodes of all available graphs are used for training, and 10% of the remaining nodes in all time layers of graphs are used for testing, with an equal number for the validation set. Although both techniques can be utilized for time-invariant multiple graphs, e.g., Protein [15,31], we exploit the second choice for our multilayer graph task (Figure 8). It enables us to capture change trajectories throughout the time layers, whereas the first approach splits the graphs without considering the time ordering, and also the behavioral changes of any of the nodes in 20% of the graphs are not taken into account. To the best of our knowledge, this is the first time that such an experimental setting is used for node embedding in multiple graphs.

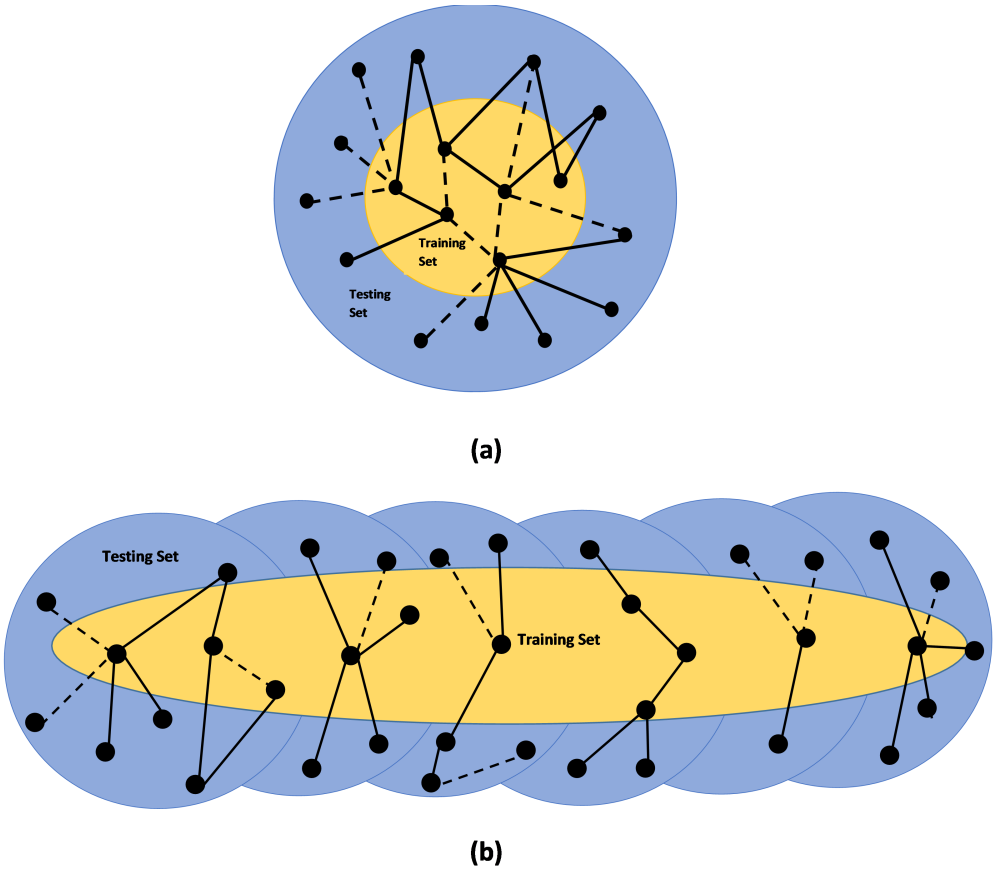


Figure 8. Train–test split for temporal graph as (**a**) single graph, (**b**) a sequence of single-layered graph.

Inductive vs. Transductive Learning: There are two different settings to learn the node embedding: inductive learning and transductive learning. In the latter setting, the graph is trained with fixed node ordering. In this technique, the model needs to be retrained when some additional nodes are added, the ordering has changed or a new graph is given. Moreover, the node attributes are augmented as one-hot identifiers that could restrict the generalization ability of the model [32]. The authors in [15] assess the transductive learning performance of P-GNN on a link prediction task, in terms of whether a pair of nodes are connected. On the other hand, in inductive learning, the learned

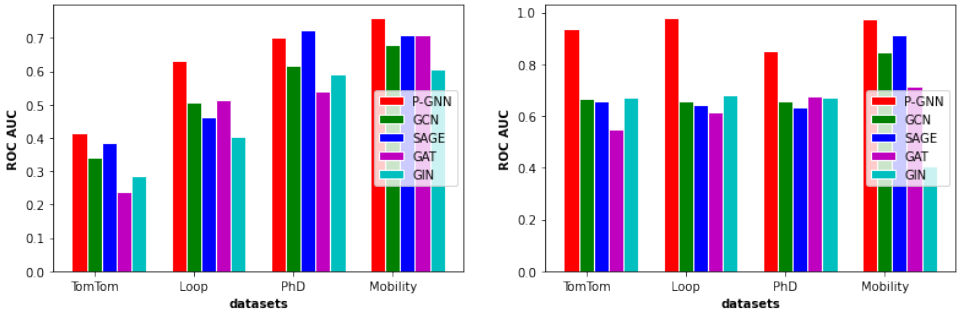
positional information can be transferred to an unseen graph. In particular, instead of augmenting the one-hot identifiers of node attributes, an order-invariant scalar could be assigned to nodes whenever node attributes are not available. We choose to train TMP-GNN in an inductive learning setting and utilize the pair-wise node classification task, as in [15], to demonstrate the performance of TMP-GNN embedding.

As discussed in (2), our goal is to compute the embedding for time-varying graphs with potential changes in edge existence $\rho^{(\mathcal{E}\times t)}$ or intra-layer edge weights represented in $\boldsymbol{A}^{(t)}$. Therefore, pair-wise node classification would be a more challenging task than link prediction, since the node class is determined according to the status of its surrounding edges and/or the status of other isomorphic nodes that are dynamic in such graphs. When the learning task requires node positional information too, only using structure-aware embedding is not sufficient.

Figure 9 illustrates the ROC AUC of TMP-GNN for pair-wise node classification given the second and first type of input, the sequences of individual graphs with each graph associated with one layer and a single supragraph, respectively (Section 3.2). All datasets demonstrate significantly better performance in the presence of a single-graph input versus a multi-graph input. It is highlighted that a single graph has better represented the chronological property of a multilayer temporal graph. The difference in performance improvement achieved by TMP-GNN is significantly larger in the case of a single supragraph for the same reason. In the case of TomTom, Loop Detector and PhD, the other four GNNs perform relatively the same. As for Mobility, all five GNN-based approaches perform well, with relatively negligible differences in classification accuracy. This could be due to the lower number of nodes plus lower dynamics in the number of edges per time layer $|\mathcal{E}^{(t)}|$, as shown in Figure 2, and stronger connectivity within time layers in comparison to other case studies that are confined to the expressive power of TMP-GNN, being less distinct from others. TMP-GNN demonstrates the largest difference in AUC improvement for TomTom from multi-graph to a single supragraph. As noted in Section 3.1, it has the highest number of nodes, with over 5 times more than PhD, the second largest graph, in conjunction with the lowest connectivity degree, and the largest number of isolated nodes, which further underlines the discriminating power of TMP-GNN. It is worth mentioning that the TomTom temporal graph, with a relatively larger number of nodes, requires higher communication message passing in the TMP-GNN layer, as described in Section 2.3, and thus higher computational effort compared to the other three datasets.

For three out of four case studies in multi-graph implementation (Figure 9a), the best performer is TMP-GNN; the second most competent is SAGE. The reason could be the similarity in message computation function F, where the feature information of two nodes $(\boldsymbol{h}_v, \boldsymbol{h}_u)$ or the attended message from u towards node v are concatenated as in (11).

Figure 10 reveals the impact of the availability of the edge feature on ROC AUC for all five GNN-based models for the Loop Detector dataset. From a topological point of view, Loop Detector is an undirected static graph. When edge features are not available, all GNNs perform relatively the same. When taking the time-varying edge attributes $x_{uv}^{(t)}$ into account, it turns into a dynamic graph wherein the node class is also dependent on time-varying attributes, and TMP-GNN demonstrates its superiority over the other widely used structure-aware embedding. This is where the power of TMP-GNN is highlighted for time-varying graphs.



(**a**) Temporal multi-graph input (**b**) Single supragraph

Figure 9. Pair-wise node classification accuracy of TMP-GNN vs. GNNs.

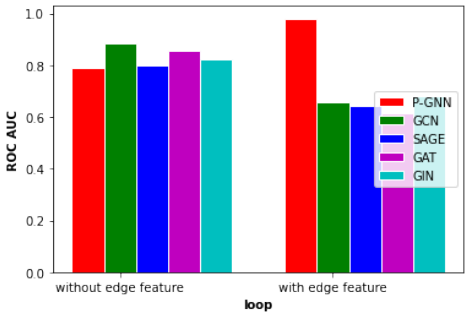


Figure 10. Pair-wise node classification accuracy of TMP-GNN vs. GNNs for Loop Detector dataset with and without edge feature.

Table 2 shows the percentage of ROC AUC improvement achieved by TMP-PGNN for multi-graph and single-supragraph implementations, as well as the decrease in MAE achieved by E-TMP-GNN I and II compared to the baselines for all four datasets. We define $\hat{\eta}_{\mathrm{y}}^{\mathrm{metric}}$ in the following equation as a measure to calculate the improvement:

$$\hat{\eta}_{\mathrm{y}}^{\mathrm{metric}} = \frac{\left|\eta_{\mathrm{y}}^{\mathrm{metric}} - \eta_{\mathrm{y'}}^{\mathrm{metric}}\right|}{\eta_{\mathrm{y'}}^{\mathrm{metric}}} \times 100, \tag{29}$$

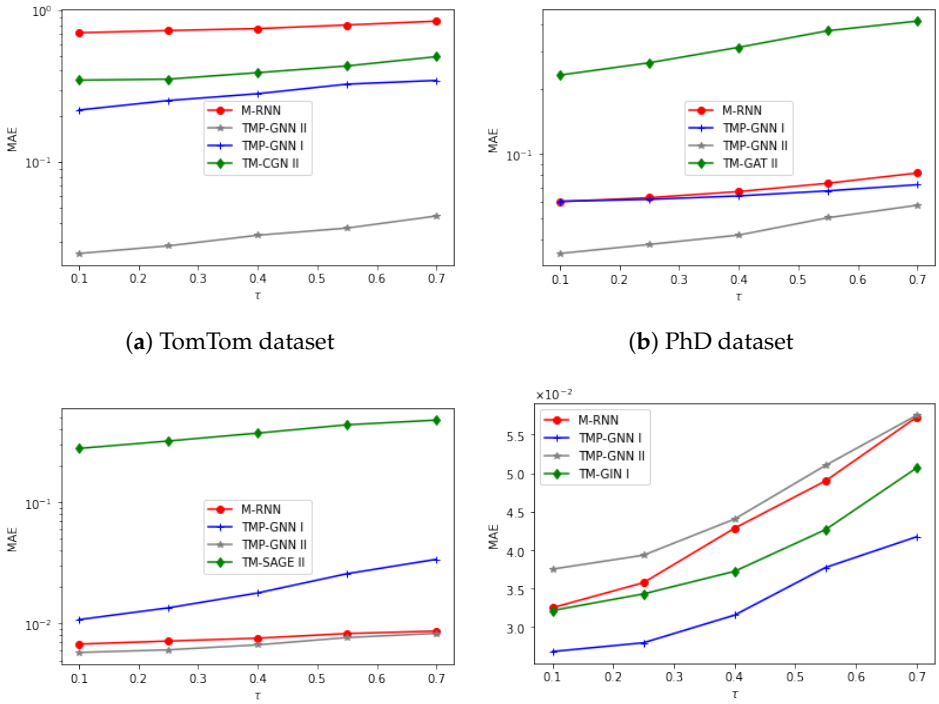
where η notes the performance, metric indicates ROC AUC or MAE according to the underlying experiment, and $|*|$ is the absolute value. y represents TMP-GNN, E-TMP-GNN I, or E-TMP-GNN II, and y' is the compared baseline method. It is noted that 0 indicates zero or no improvement.

Table 2. Performance Comparison of E-TMP-GNN I, E-TMP-GNN II vs. counterpart methods.

Estimate	TomTom	Loop Detector	PhD	Mobility
$\hat{\eta}_{\text{Multi-graph}}^{\text{ROC AUC}}(\%)$	7–42	0–22	18–35	6–20
$\hat{\eta}_{\text{Single supragraph}}^{\text{ROC AUC}}(\%)$	28–41	20–25	30–37	6–58
$\hat{\eta}_{\text{E-TMP-GNN I}}^{\text{MAE}}(\%)$	59–69	0–12	17–27	0
$\hat{\eta}_{\text{E-TMP-GNN II}}^{\text{MAE}}(\%)$	94–96	29–96	0	0–15

In Figure 11, we evaluate the performance of our proposed architectures, TMP-GNN I and TMP-GNN II, on the estimation of missing data, and compare it to an RNN variant architecture named M-RNN (Figure 6) and the second best classifier demonstrated in Figure 9b, which is CGN, GAT, GIN, and SAGE for the TomTom, PhD, Loop, and Mobility datasets, respectively. We use Mean Absolute Error (MAE) as a measure of estimation

accuracy and choose to remove missing values on a temporal dimension. The points are chosen according to uniform distribution in conjunction with a determined missing threshold τ.



(**a**) TomTom dataset (**b**) PhD dataset

(**c**) Mobility dataset (**d**) Loop dataset

Figure 11. Performance evaluation of E-TMP-GNN I and II on missing data estimation vs. M-RNN and counterpart GNN.

All case studies experience a higher MAE as τ increases. Moreover, either TMP-GNN I or TMP-GNN II outperform both M-RNN and the counterpart GNN for each case. This is due to the high expressive power of additional features, learning through TMP-GNN representation. The underlying time-varying features proved to be significantly more efficient for missing data estimation than the ones derived from the second-best node classifier in the previous experiment, since it takes position information and inter-layer coupling into account simultaneously; the latter makes the architecture more powerful than the ones utilizing the RNN variant for capturing long-term temporal dependencies. TomTom has the highest MAE compared to others, followed by PhD, Loop and Mobility, respectively. This follows the same order as their corresponding multi-layer graphs when ranked according to their degree of connectivity; the sparser the graph is, the larger the estimation error would be. In addition, the percentage of improvement is larger for TomTom and PhD compared to the other two, which means that the proposed method is more effective for graphs with weaker connectivity.

For three out of four datasets (TomTom, Loop, Mobility), E-TMP-GNN II outperforms E-TMP-GNN I. In the case of the PhD dataset, E-TMP-GNN I is the best performer. The nature of being dynamic is more highlighted in the case of PhD and Mobility, since there is a significant change in the number of edges per time point (Figure 9), in addition to the variation in edge weights existing in the two other datasets. In TMP-GNN I, the time-aware node embedding generated by TMP-GNN will play a more effective role once added to the existing features of the dataset so that it is fed into the interpolation component implemented by Bi-GRU, whereas E-TMP-GNN II reduces the dimension of TMP-GNN's time-aware embedding, and then combines it with Bi-GRU's generated representation in an attentive way. In summary, dimension reduction and the softmax function might reduce the effectiveness of the time-aware embedding, later affecting the level of MAE

decrease in missing data estimation. The more time-varying the dataset is, the more efficient time-aware representation is needed to better estimate missing data points.

4. Conclusions

We propose TMP-GNN, a generalized version of P-GNN node embedding, for temporal multilayer graphs. It takes short-term temporal correlations, as well as feature and positional information, into account. We also incorporate the notion of eigenvector-based centrality to distinguish nodes with a higher influence on their neighbors. Then, we extend the design to E-TMP-GNN I and II to tackle the missing data challenge in dynamic networks with multiple streams of node/edge features. Conducted experiments on four real-world datasets with diverse characteristics demonstrate significant improvements in node-level classification and missing data estimation tasks. It would be interesting to consider formulating the missing data problem directly into an edge representation task for future work.

Author Contributions: Conceptualization, B.N., S.P. and A.L.-G.; methodology, B.N., S.P. and A.L.-G.; software, B.N. and S.P.; validation, B.N., S.P. and A.L.-G.; formal analysis, B.N. and S.P.; investigation, B.N., S.P. and A.L.-G.; resources, A.L.-G.; data curation, B.N. and A.L.-G.; writing—original draft preparation, B.N., S.P. and A.L.-G.; writing—review and editing, B.N., S.P. and A.L.-G.; visualization, B.N., S.P. and A.L.-G.; supervision, A.L.-G.; project administration, B.N. and A.L.-G.; funding acquisition, A.L.-G. All authors have read and agreed to the published version of the manuscript.

Funding: This work and its Article Processing Charge (APC) were funded by A.L.-G.'s University of Toronto operating grant (2020–2022).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Not applicable.

Acknowledgments: The authors are grateful to the journal reviewers for their constructive comments and suggestions on the first draft of the manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GNN	Graph Neural Networks
TMP	Temporal Multilayer Position-Aware
ROC	Receiver Operating Characteristic
ROC AUC	Area Under Receiver Operating Characteristic Curve
ITS	Intelligent Transportation System
GRU	Gated Recurrent Unit
MAE	Mean Absolute Error
MLC	Marginal Layer Centrality
RNN	Recurrent Neural Network
GTA	Greater Toronto Area
LCC	Largest Connected Component
CGN	Convolutional Graph Neural Network
GAT	Graph Attention Network
GIN	Graph Isomorphism Network
SAGE	Sample And Aggregate

References

1. Lv, Y.; Duan, Y.; Kang, W.; Li, Z.; Wang, F.Y. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intell. Transp. Syst.* **2014**, *16*, 865–873. [CrossRef]
2. Lin, Y.; Dai, X.; Li, L.; Wang, F. Pattern sensitive prediction of traffic flow based on generative adversarial framework. *IEEE Trans. Intell. Transp. Syst.* **2018**, *20*, 2395–2400. [CrossRef]
3. Cui, Z.; Henrickson, K.; Ke, R.; Wang, Y. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 4883–4894. [CrossRef]
4. Yu, H.; Wu, Z.; Wang, S.; Wang, Y.; Ma, X. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *Sens. Multidiscip. Digit. Publ. Inst.* **2017**, *17*, 1501. [CrossRef]
5. Ghoroghchian, N.; Groppe, D.M.; Genov, R.; Valiante, T.A.; Draper, S.C. Node-Centric Graph Learning From Data for Brain State Identification. *IEEE Trans. Signal Inf. Process. Netw.* **2020**, *6*, 120–132. [CrossRef]
6. Zhang, M.; Chen, Y. Link prediction based on graph neural networks. *arXiv* **2018**, arXiv:1802.09691.
7. Liu, S.; Li, T.; Ding, H.; Tang, B.; Wang, X.; Chen, Q.; Yan, J.; Zhou, Y. A hybrid method of recurrent neural network and graph neural network for next-period prescription prediction. *Int. J. Mach. Learn. Cybern.* **2020**, *11*, 2849–2856. [CrossRef] [PubMed]
8. Yao, H.; Wu, F.; Ke, J.; Tang, X.; Jia, Y.; Lu, S.; Gong, P.; Ye, J.; Li, Z. Deep multi-view spatial-temporal network for taxi demand prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
9. Taheri, A.; Gimpel, K.; Berger-Wolf, T. Learning graph representations with recurrent neural network autoencoders. In Proceedings of the KDD'18 Deep Learning Day, London, UK, 19–23 August 2018.
10. Kumar, S.; Zhang, X.; Leskovec, J. Predicting dynamic embedding trajectory in temporal interaction networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1269–1278.
11. Xue, H.; Yang, L.; Jiang, W.; Wei, Y.; Hu, Y.; Lin, Y. Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal rnn. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Wuerzburg, Germany, 16–20 September 2019; pp. 282–298.
12. Xu, D.; Cheng, W.; Luo, D.; Gu, Y.; Liu, X.; Ni, J.; Zong, B.; Chen, H.; Zhang, X. Adaptive neural network for node classification in dynamic networks. In Proceedings of the 2019 IEEE International Conference on Data Mining (ICDM), Beijing, China, 8–11 November 2019; pp. 1402–1407.
13. Xu, D.; Cheng, W.; Luo, D.; Liu, X.; Zhang, X. Spatio-Temporal Attentive RNN for Node Classification in Temporal Attributed Graphs. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), Macao, China, 10–16 August 2019; pp. 3947–3953.
14. Xie, Y.; Li, C.; Yu, B.; Zhang, C.; Tang, Z. A survey on dynamic network embedding. *arXiv* **2020**, arXiv:2006.08093.
15. You, J.; Ying, R.; Leskovec, J. Position-aware graph neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 7134–7143.
16. Taylor, D.; Myers, S.A.; Clauset, A.; Porter, M.A.; Mucha, P.J. Eigenvector-based centrality measures for temporal networks. *Multiscale Model. Simul.* **2017**, *15*, 537–574. [CrossRef] [PubMed]
17. Gleich, D.F. PageRank beyond the (WEB). *SIAM Rev.* **2015**, *57*, 321–363. [CrossRef]
18. Shai, S.; Stanley, N.; Granell, C.; Taylor, D.; Mucha, P.J. Case studies in network community detection. In *The Oxford Handbook of Social Networks*; Oxford University Press: Oxford, UK, 2017.
19. Ipsen, I.; Wills, R.M. Analysis and computation of google's pagerank. In Proceedings of the 7th IMACS International Symposium on Iterative Methods in Scientific Computing, Fields Institute, Toronto, ON, Canada, 5–8 May 2005; Volume 5.
20. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How powerful are graph neural networks? *arXiv* **2018**, arXiv:1810.00826.
21. Bourgain, J. On Lipschitz embedding of finite metric spaces in Hilbert space. *Isr. J. Math.* **1985**, *52*, 46–52. [CrossRef]
22. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
23. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef]
24. Yoon, J.; Zame, W.R.; van der Schaar, M. Estimating missing data in temporal data streams using multi-directional recurrent neural networks. *IEEE Trans. Biomed. Eng.* **2018**, *66*, 1477–1490. [CrossRef] [PubMed]
25. Najafi, B.; Parsaeefard, S.; Leon-Garcia, A. Estimation of Missing Data in Intelligent Transportation System. In Proceedings of the 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall), Virtual, 18 November–16 December 2020; pp. 1–6.
26. Kang, Y.; Gao, S.; Liang, Y.; Li, M.; Rao, J.; Kruse, J. Multiscale dynamic human mobility flow dataset in the US during the COVID-19 epidemic. *Sci. Data* **2020**, *7*, 1–13. [CrossRef]
27. TomTom Road Analytics. Available online: Https://www.tomtom.com/products/road-traffic-data-analytics/ (accessed on 1 January 2022).
28. Burris, V. The academic caste system: Prestige hierarchies in PhD exchange networks. *Am. Sociol. Rev.* **2004**, *69*, 239–264. [CrossRef]
29. Cui, Z.; Ke, R.; Wang, Y. Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. *arXiv* **2018**, arXiv:1801.02143.
30. Tarjan, R. Depth-first search and linear graph algorithms. *SIAM J. Comput.* **1972**, *1*, 146–160. [CrossRef]
31. Adhikari, B. DEEPCON: Protein contact prediction using dilated convolutional neural networks with dropout. *Bioinformatics* **2020**, *36*, 470–477. [CrossRef]
32. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.