Networked Time Series Imputation via Position-aware Graph Enhanced Variational Autoencoders

Dingsu Wang
dingsuw2@illinois.edu
University of Illinois at
Urbana-Champaign
IL, USA

Yuchen Yan
yucheny5@illinois.edu
University of Illinois at
Urbana-Champaign
IL, USA

Ruizhong Qiu
rq5@illinois.edu
University of Illinois at
Urbana-Champaign
IL, USA

Yada Zhu
yzhu@us.ibm.com
IBM Research
NY, USA

Kaiyu Guan
kaiyug@illinois.edu
University of Illinois at
Urbana-Champaign
IL, USA

Andrew J Margenot
margenot@illinois.edu
University of Illinois at
Urbana-Champaign
IL, USA

Hanghang Tong
htong@illinois.edu
University of Illinois at
Urbana-Champaign
IL, USA

ABSTRACT

Multivariate time series (MTS) imputation is a widely studied problem in recent years. Existing methods can be divided into two main groups, including (1) deep recurrent or generative models that primarily focus on time series features, and (2) graph neural networks (GNNs) based models that utilize the topological information from the inherent graph structure of MTS as relational inductive bias for imputation. Nevertheless, these methods either neglect topological information or assume the graph structure is fixed and accurately known. Thus, they fail to fully utilize the graph dynamics for precise imputation in more challenging MTS data such as *networked time series* (*NTS*), where the underlying graph is constantly changing and might have missing edges. In this paper, we propose a novel approach to overcome these limitations. First, we define the problem of imputation over NTS which contains missing values in both node time series features and graph structures. Then, we design a new model named PoGeVon which leverages variational autoencoder (VAE) to predict missing values over both node time series features and graph structures. In particular, we propose a new node position embedding based on random walk with restart (RWR) in the encoder with provable higher expressive power compared with message-passing based graph neural networks (GNNs). We further design a decoder with 3-stage predictions from the perspective of multi-task learning to impute missing values in both time series and graph structures reciprocally. Experiment results demonstrate the effectiveness of our model over baselines.

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Networked time series; imputation; variational autoencoders; random walk with restart; node positional embeddings

1 INTRODUCTION

Multivariate time series (MTS) data are common in many real-world applications, such as stock prediction [13, 69], traffic forecasting [42, 77, 78] and pandemic analysis [30, 51]. However, these data are often incomplete and contain missing values due to reasons such as market close or monitoring sensor/system failure. Predicting the missing values, which is referred to as the MTS imputation task, plays an important role in these real-world applications.

Recently, a large amount of approaches emerge for MTS imputation [16] in the literature. To name a few, BRITS [5] is built upon bidirectional recurrent modules and GAIN [75] is one of the earliest works that use adversarial training for the task. However, many of them ignore the available relational information within the data and thus are less effective to predict missing values compared to those considering both spatial and temporal information. In order to tackle this problem, some recent works utilize GNNs or other similar algorithms to assist the imputation over MTS data. GRIN [11] adopts a bidirectional recurrent model based on message passing neural networks [21]. They perform a one-step propagation of the hidden representations on the graph to capture the spatial dependencies within the MTS data. SPIN [47] is a follow-up method which solves the error accumulation problem of GRIN in highly sparse data. It introduces a new attention mechanism to capture spatial-temporal information through inter-node and intra-node attentions. By stacking several attention blocks, the model simulates a diffusion process and can handle data with high missing rates. Recently, NET^3 [28] generalizes the setting and studies tensor time series data in which the underlying graph contains multiple modes.
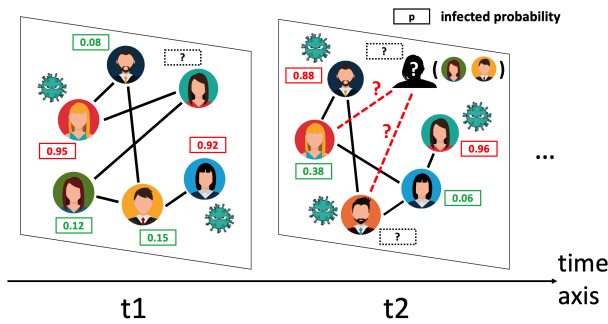
Figure 1: An illustrative example of an interaction network during the COVID-19 pandemic where some patients' infection status might not be available and we have no access to whom these people interact with, which represents a networked time series (NTS) with both missing node features and missing edges.

The authors utilize a tensor graph convolution network (GCNs) and a tensor recurrent neural network (RNNs) to handle the tensor graphs and time series respectively.

Despite the strong empirical performance of these methods on the MTS imputation problem, they rely on the assumption that the underlying graph is fixed and accurately known. However, the graph structure of an MTS may constantly change over time in real-world scenarios. Take epidemiological studies as an example, during the evolution of a pandemic, individuals like human beings or animals may move around and thus the graph that models the spread of disease is dynamic. In literature, such time series data are referred to as *networked time series* (*NTS*)[1] [28]. Given the nature of NTS data, the missing components can occur in both the node features and the graph structures (See an example in Figure 1), which makes NTS imputation an essentially harder problem compared to MTS imputation.

In this paper, we first formally define the problem of NTS imputation. We point out that the key challenges of this problem are twofold: (1) The graph that lies behind time series data is evolving constantly, and contains missing edges. Therefore, algorithms should capture the graph dynamics and at the same time be able to restore the lost structures. (2) The node feature time series also contains missing values, which requires the model to solve a general MTS imputation problem as well. To address these challenges, we formulate NTS imputation as a *multi-task learning* problem and propose a novel model named PoGeVon based on variational autoencoder (VAE) [34]. Our proposed model consists of two parts, including a recurrent encoder with node position embeddings based on random walk with restart (RWR) [59] and a decoder with 3-stage predictions. The *global* and *local* structural information obtained from RWR with respect to a set of anchor nodes provides useful node representations. Moreover, the 3-stage prediction module in the decoder is designed to impute missing features in time series and graph structures reciprocally: the first stage prediction fills the missing values for node features and then is used for the imputation over graph structures during the second stage, in return, the predicted graph structures are used in the third stage for node feature imputation. Finally, we replicate the VAE model in PoGeVon

---

[1]In some research works [38], NTS is also named as *network time series.*

to handle the bidirectional dynamics in the NTS data. The main contributions of this paper can be summarized as:

- **Problem Definition**. To our best knowledge, we are the first to study the joint problem of MTS imputation and graph imputation over networked time series data.
- **Novel Algorithm and Analysis.** We propose a novel imputation model based on VAE, which consists of an encoder with RWR-based node position embeddings, and a decoder with 3-stage predictions. We provide theoretical analysis of the expressive power of our position embeddings compared with message-passing based temporal GNNs, as well as the benefit of multi-task learning approach for NTS imputation problem from the perspective of information bottleneck.
- **Empirical Evaluations.** We demonstrate the effectiveness of our method by outperforming powerful baselines for both MTS imputation and link prediction tasks on various real-world datasets.

The rest of the paper is organized as follows. Section 2 defines the imputation problem over NTS data. Section 3 presents the proposed PoGeVon model. Section 4 shows the experiment results. Related works and conclusions are given in Section 5 and Section 6 respectively.

2 PROBLEM DEFINITION

Table 1 lists main symbols and notations used throughout this paper. Calligraphic letters denote tensors or graphs (e.g., $\mathcal{X}$, $\mathcal{G}$), bold uppercase letters are used for matrices (e.g., $\mathbf{A}$), bold lowercase letters are for vectors (e.g., $\mathbf{v}$). Uppercase letters (e.g., T) are used for scalars, and lowercase letters (e.g., i) are for indices. For matrices, we use $\mathbf{A}[i, j]$ to denote the value at the i-th row and j-th column.

We first present some necessary preliminaries and then introduce the networked time series imputation problem in this section.

DEFINITION 2.1 (**MULTIVARIATE TIME SERIES (MTS)**). *A multivariate time series $\mathcal{X} \in \mathbb{R}^{T \times N \times D}$ is a sequence of observations: $\{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_T\}$, where each observation $\mathbf{X}_t \in \mathbb{R}^{N \times D}$ is a slice of $\mathcal{X}$ at time step t that contains N entities with D features.*

DEFINITION 2.2 (**NETWORKED TIME SERIES (NTS)**). *Networked time series is an extension of multivariate time series, in which a sequence of graphs $\mathcal{G}(\mathcal{A}, \mathcal{X}) = \{G_1, G_2, ..., G_T\}$ is given, and $\mathcal{A}$ models the node interactions as time goes by. Each graph G_t is represented as a weighted adjacency matrix $\mathbf{A}_t \in \mathbb{R}^{N \times N}$ with the node feature matrix $\mathbf{X}_t \in \mathbb{R}^{N \times D}$.*

DEFINITION 2.3 (**MASK TENSOR**). *A binary mask tensor $\mathcal{M}$: $\{\mathbf{M}_1, \mathbf{M}_2, ..., \mathbf{M}_T\} \in \mathbb{R}^{T \times N \times D}$ serves as the indicator of missing values in MTS data, in which the value $\mathbf{M}_t[i, j]$ indicates the availability of each feature j of entity i at time step t: $\mathbf{M}_t[i, j]$ being 0 or 1 indicates the corresponding feature is missing or observed.*

Given the nature of NTS data, its missing data can occur in two parts: (1) missing values in node feature time series, and (2) missing edges in graph structures. The former is similar to missing values in traditional MTS, while the latter is unique in NTS which demonstrates the underlying dynamics of a graph sequence. Therefore, we can also define mask tensor for graph adjacency sequence similar to Definition 2.3. We formally define the partially observed NTS data and NTS imputation problem as follows:

Table 1: Symbols and Notations.

Symbol	Definition
$\mathcal{G}$	sequence of graphs
$\mathcal{A}$	tensor of graph adjacency sequence
$\mathcal{X}$	tensor of multivariate time series
$\mathcal{M}$	mask tensor of $\mathcal{X}$
$\mathcal{R}$	tensor of node position embeddings
G_t	graph at time step t
$\tilde{G}_t$	observed graph at time step t
$\tilde{\mathcal{G}}$	observed sequence of graphs
$\tilde{\mathcal{A}}$	observed tensor of graph adjacency sequence
$\tilde{\mathcal{X}}$	observed multivariate time series
$\mathbf{A}_t$	adjacency matrix at time t
$\mathbf{X}_t$	node feature matrix at time t
$\mathbf{M}_t$	mask matrix at time t
$\mathbf{R}_t$	RWR position matrix at time t
$\mathbf{r}_{t,i}$	RWR position score of node i at time t
$\mathbf{e}_i$	one-hot restart vector with value 1 at index i
$\mathbf{D} = \text{diag}(\mathbf{d})$	diagonal matrix of the degree vector $\mathbf{d}$
$\mathbf{A}^\top$	transpose of $\mathbf{A}$
$\mathbf{Z}$	latent node embedding matrix of VAE
$H(X)$	entropy of random variable X
$I(X;Y)$	mutual information between X and Y
T	length of time series
N	number of nodes
D	number of features
i, j, u, v	indices of nodes
c	restart probability in RWR
z	latent representations of VAE
θ, γ, ϕ	parameters of neural networks
$\| \cdot \|_F$	Frobenius norm
$\odot$	Hadamard product

DEFINITION 2.4 (**PARTIALLY OBSERVED NTS**). *A partially observed NTS:* $\mathcal{G}(\tilde{\mathcal{A}}, \tilde{\mathcal{X}}) = \{\tilde{G}_1, \tilde{G}_2, ..., \tilde{G}_T\}$ *consists of observed graph adjacency tensor* $\tilde{\mathcal{A}}$ *and observed node feature tensor* $\tilde{\mathcal{X}}$. *The value of* $\tilde{\mathbf{A}}_t[i,j]$ *and* $\tilde{\mathbf{X}}_t[i,j]$ *can be observed only if* $\mathbf{M}_t^A[i,j] = 1$ *and* $\mathbf{M}_t^X[i,j] = 1$ *where* $\mathcal{M}^A$ *and* $\mathcal{M}^X$ *are the mask tensors for graph adjacency structure and node features respectively.*

PROBLEM 1 (**NTS IMPUTATION**).

Given: *A partially observed NTS with graph sequence* $\mathcal{G}(\tilde{\mathcal{A}}, \tilde{\mathcal{X}}) = \{\tilde{G}_1, \tilde{G}_2, ..., \tilde{G}_T\}$;

Output: *The predicted graph adjacency tensor* $\mathcal{A}$ *and the tensor* $\mathcal{X}$ *of node feature time series.*

Note. For clarity, we use node features and node time series interchangeably, and same for the graph adjacency imputations and missing edges/links predictions.

3 METHODOLOGY

In this section, we introduce our model named <u>Po</u>sition-aware <u>G</u>raph <u>E</u>nhanced <u>V</u>ariational Aut<u>o</u>e<u>n</u>coders (PoGeVon) in detail. In order to predict the missing values in both the node features and the graph structures, we design a novel *variational autoencoder* (*VAE*), whose detailed architecture is shown in Figure 3. It consists

of an encoder with node position embeddings based on *random walk with restart* (*RWR*), and a decoder with 3-stage predictions. We then replicate the VAE to handle bidirectional dynamics. We start in Subsection 3.1 to discuss the multi-task learning setting of NTS imputation problem, analyze its mutual benefit and implications to the encoder/decoder design. Then, we present the details of the proposed encoder (Subsection 3.2) and decoder (Subsection 3.3), followed by the training objective in Subsection 3.4 as well as the compelxity analysis in Subsection 3.5.
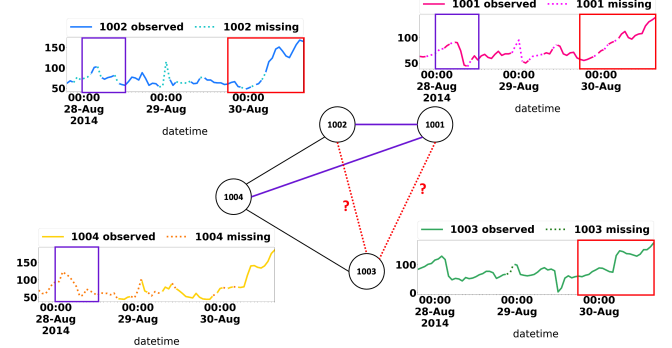

Figure 2: An illustrative example of mutual reinforcing effect between node feature imputation and graph structure imputation, based on 4 monitor stations in AQ36 dataset (See Section 4 for the details of the dataset). Correlation between three time series (1001, 1002, and 1003, indicated by three red boxes) helps impute the missing edges between them (the two red dashed lines). Meanwhile, the edges between 1001, 1002 and 1004 (the two purple lines) helps impute time series/node features by capturing the lagged correlation between them (the three purple boxes). Best viewed in color.

3.1 Multi-Task Learning Framework

Because of the potential mutual benefit of predicting missing node features and edges, it is natural to formulate the NTS imputation as a multi-task learning problem which consists of the *imputation* task for node time series and the *link prediction* task for graph structures. Let us analyze the benefit of modeling NTS imputation as a multi-task learning problem from the perspective of *information bottleneck* in unsupervised representation learning [1, 58], and formulate the objective of NTS imputation as:

$$\max[I(\tilde{\mathcal{A}}, \tilde{\mathcal{X}}; z) - \beta I(z; \tilde{\mathcal{G}}_{t:t+\Delta t})] \quad (1)$$

where z is the latent representation , $I(\cdot; \cdot)$ is the mutual information, $\tilde{\mathcal{G}}_{t:t+\Delta t}$ is the data sample which represents a sliding window of NTS data and β is the Lagrange multiplier. This formulation closely relates to the objective of a β-VAE [1, 24]. Here, the second term $\beta I(z; \tilde{\mathcal{G}}_{t:t+\Delta t})$ in Eq. (1) constraints the amount of identity information of each data sample that can transmit through the latent representation z. In β-VAE, this is upper bounded by minimizing the Kullback–Leibler divergence $\beta \cdot \mathbb{KL}[q_\theta(z|X)||p(z)]$ [3]. The first term $I(\tilde{\mathcal{A}}, \tilde{\mathcal{X}}; z)$ in Eq. (1) represents the reconstruction task of VAE which can be decomposed as [27]:

$$I(\tilde{\mathcal{A}}, \tilde{\mathcal{X}}; z) = I(\tilde{\mathcal{A}}; z) + I(\tilde{\mathcal{X}}; z) - I(\tilde{\mathcal{A}}; \tilde{\mathcal{X}}; z) \quad (2)$$
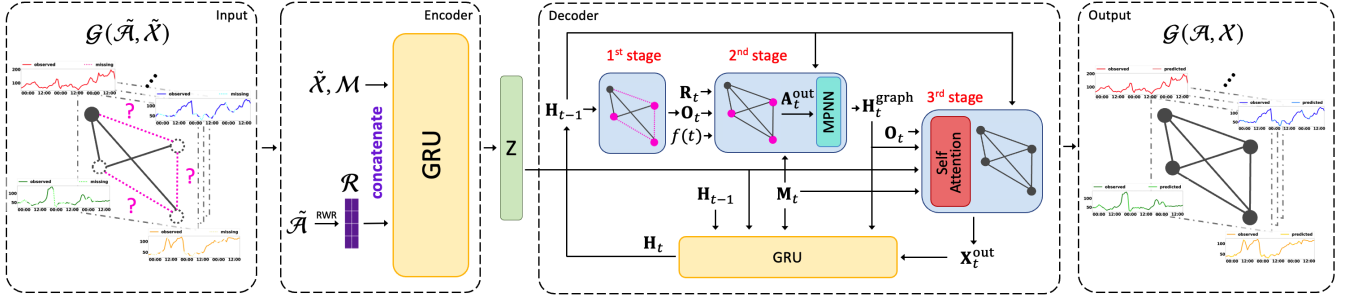
Figure 3: The model architecture of the proposed PoGeVon.

where $I(\tilde{\mathcal{A}}, \tilde{\mathcal{X}}; z)$ represents the mutual information between the partially observed NTS $\tilde{\mathcal{G}}$ (i.e., the joint distribution of $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$) and z, while $I(\tilde{\mathcal{A}}; \tilde{\mathcal{X}}; z)$ is the High-order Mutual Information [27, 48] which measures the shared information among multiple different random variables (i.e., $\tilde{\mathcal{A}}$, $\tilde{\mathcal{X}}$, and z). It is worthy noting that when $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ are independent from each other (even given z), we have:

$$I(\tilde{\mathcal{A}}, \tilde{\mathcal{X}}; z) = H(\tilde{\mathcal{A}}, \tilde{\mathcal{X}}) - H(\tilde{\mathcal{A}}, \tilde{\mathcal{X}}|z)$$
$$= H(\tilde{\mathcal{A}}) + H(\tilde{\mathcal{X}}) - H(\tilde{\mathcal{A}}|z) - H(\tilde{\mathcal{X}}|z) = I(\tilde{\mathcal{A}}; z) + I(\tilde{\mathcal{X}}; z) \quad (3)$$

where $H(\cdot)$ is the entropy. Compared with Eq. (2), it is clear that $I(\tilde{\mathcal{A}}; \tilde{\mathcal{X}}; z)$ now equals to 0. Under such circumstances, i.e., no correlation exists between features of any adjacent node pairs, the objective in Eq. (1) becomes modeling time series features and graph structures independently. However, in reality, this is often not the case. Figure 2 demonstrates an illustrative example from the AQ36 dataset [80] in which NTS imputation problem occurs when monitor stations fail due to system errors and lose data as well as connections with each other. To maximize Eq. (2), we further decompose $I(\tilde{\mathcal{A}}; \tilde{\mathcal{X}}; z)$:

$$I(\tilde{\mathcal{A}}; \tilde{\mathcal{X}}; z) = I(\tilde{\mathcal{A}}; z) - I(\tilde{\mathcal{A}}; z|\tilde{\mathcal{X}}) = I(\tilde{\mathcal{X}}; z) - I(\tilde{\mathcal{X}}; z|\tilde{\mathcal{A}}) \quad (4)$$

where the second equation holds because of symmetry [73]. Combining Eq. (2) and Eq. (4), we can derive the objective term for the decoder as:

$$2 \cdot I(\tilde{\mathcal{A}}, \tilde{\mathcal{X}}; z) = \underbrace{I(\tilde{\mathcal{A}}; z) + I(\tilde{\mathcal{X}}; z)}_{\text{VAE}} + \underbrace{I(\tilde{\mathcal{X}}; z|\tilde{\mathcal{A}}) + I(\tilde{\mathcal{A}}; z|\tilde{\mathcal{X}})}_{\text{Conditional VAE}} \quad (5)$$

where the first two terms can be bounded by the objective for VAE decoder as in [1]. The last two terms represent the objective of conditional VAE (CVAE) since $I(\tilde{\mathcal{X}}; z|\tilde{\mathcal{A}}) = H(\tilde{\mathcal{X}}|\tilde{\mathcal{A}}) - H(\tilde{\mathcal{X}}|\tilde{\mathcal{A}}, z)$. The first term $H(\tilde{\mathcal{X}}|\tilde{\mathcal{A}})$ on the right hand can be dropped because it is independent from our model, and maximizing the second term $-H(\tilde{\mathcal{X}}|\tilde{\mathcal{A}}, z)$ is essentially the same as optimizing the decoder of CVAE with objective $\max p(\tilde{\mathcal{X}}|\tilde{\mathcal{A}})$. Similar analysis applies to $I(\tilde{\mathcal{A}}; z|\tilde{\mathcal{X}})$. Eq. (5) provies an important insight: we can use $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ as the conditions for each other's predictions since imputation over one of them might be instructive for the other.

To summarize, our analysis reveals that (1) when the features of adjacent nodes are uncorrelated, we can impute the node time series and graph adjacency independently (Eq. (3)); however, (2) in real applications, node features and graph structure are often correlated (e.g., Figure 2), and in such a scenario, there might be a mutual reinforcing effect between node feature imputation and graph adjacency imputation (Eq. (4)). Our analysis also provides

novel and critical clues that can guide the design of the encoder-decoder framework for learning datasets with multi-modality such as NTS. For the encoder, Eq. (5) suggests that the latent representation z (i.e., the output of the encoder) should encode both the graph adjacency information and the node feature information (i.e., the VAE part of Eq. (5)) as well as the mutual interaction between them (i.e., the CVAE part of Eq. (5)). For the decoder, we will present a three-stage prediction method so that the (imputed) graph structures and the (imputed) node features can be used as each other's condition respectively (i.e., the CVAE part of Eq. (5)).

3.2 Encoder

The encoder aims to encode both the structural and the dynamic information of NTS data. Existing message-passing based GNNs typically only capture the *local* information from close neighbors. However, long-distance information between nodes is important in NTS data since the graph is constantly evolving and interactions between nodes can occur at any time step. Therefore, to capture this long-distance *global* information, we propose using position embeddings with random walk with restart (RWR) [19, 59, 71, 72].

3.2.1 RWR-based Position Embeddings. For a graph G_t at time step t, the relative position vector for all nodes w.r.t. one anchor node i is computed by RWR as follows:

$$\mathbf{r}_{t,i} = (1 - c)\hat{\mathbf{A}}_t \mathbf{r}_{t,i} + c\mathbf{e}_i \quad (6)$$

where $\hat{\mathbf{A}}_t = (\mathbf{D}_t^{-1}\mathbf{A}_t)^\top$ is the normalized adjacency matrix of G_t, $\mathbf{e}_i \in \mathbb{R}^N$ is a one-hot vector which only contains nonzero value at position i and c is the restart probability. After reaching the stationary distribution, we concatenate the position scores $\mathbf{r}_{t,i} \in \mathbb{R}^N$ of all the anchor nodes as the final position embeddings $\mathbf{R}_t \in \mathbb{R}^{N \times N}$, where N is the number of nodes.

We next prove the expressive power of RWR-based position embeddings with following proposition and theorem.

PROPOSITION 3.1. *Random walk with restarts (RWR) captures information from close neighbors (local) and long-distance neighbors (global) in graph learning.*

PROOF. See Appendix. □

The benefit of RWR-based position embeddings in temporal graphs is summarized in Theorem 3.2 from the perspective of message-passing based temporal graph networks (TGN) [54], which is a general class of GNNs designed for handling temporal graphs. It contains two main components: *memory* (through RNNs) for capturing the dynamics of each node; *aggregate and update* (through GNNs) for gathering topological information from neighbors.

THEOREM 3.2. *Given a temporal graph $\mathcal{G}$, TGN with RWR-based node position embeddings g_θ has more expressive power than regular TGN f_θ in node representation learning: $\mathbb{D}(g(u), g(v)) \geq \mathbb{D}(f(u), f(v))$ where $\mathbb{D}(\cdot, \cdot)$ measures the expressiveness by counting the distinguishable node pairs (u, v) in $\mathcal{G}$ based on node representations.*

PROOF. See Appendix. □

Finally, to capture the dynamic information in NTS data, we use a 2-layer gated recurrent unit (GRU) [10] as the encoder to model $q_\theta(z|\tilde{\mathcal{X}}, \mathcal{M}, \mathcal{R})$, where z is the latent representation and $\mathcal{R} = \{\mathbf{R}_1, ..., \mathbf{R}_T\}$ is the tensor of node position embeddings. For each $\mathbf{R}_t$, instead of treating all the nodes as anchor nodes, usually only a small subset of anchor nodes $|S| = L$ would be sufficient to distinguish nodes from each other in practice [76]. Masks $\mathcal{M}$ and position embeddings $\mathcal{R}$ are concatenated with the input $\tilde{\mathcal{X}}$ at each time step before feeding into the GRU.

3.3 Decoder

We design the decoder as a GRU with 3-stage predictions. We use $\mathbf{H}_t$ to denote node embedding matrix at time step t and $\mathcal{H}$ to denote node embedding tensor. Based on the analysis in Section 3.1, we model the complementary relation between feature imputation $p_\phi(\tilde{\mathcal{X}}|\tilde{\mathcal{A}}, \mathcal{M}, z)$ and network imputation $p_\gamma(\tilde{\mathcal{A}}|\mathcal{M}, \mathcal{R}, z)$ at different prediction stages in the decoder as follows.

3.3.1 First-stage Feature Prediction.
In the first stage, we use a linear layer to generate an initial prediction of the missing values in the time series:

$$\hat{\mathbf{Y}}_{1,t} = \text{Linear}(\mathbf{H}_{t-1}) \tag{7}$$

where $\mathbf{H}_{t-1}$ is the hidden representation of each node from the previous time step and $\mathbf{H}_0$ is sampled from a normal distribution $N(0, 1/\sqrt{d_h})$ where d_h is the hidden dimension. Similar to [11], we then use a filler operator to replace the missing values in the input $\tilde{\mathbf{X}}_t$ with $\hat{\mathbf{Y}}_{1,t}$ to get the first-stage output $\mathbf{O}_t$:

$$\mathbf{O}_t = \mathbf{M}_t \odot \tilde{\mathbf{X}}_t + (1 - \mathbf{M}_t) \odot \hat{\mathbf{Y}}_{1,t} \tag{8}$$

3.3.2 Second-stage Link Prediction.
Our second-stage prediction imputes the missing weighted edges within graphs. $\mathbf{O}_t$ is used with the mask $\mathbf{M}_t$, the position embedding $\mathbf{R}_t$ and $\mathbf{H}_{t-1}$ to get the embeddings of all nodes at timestep t through a linear layer:

$$\mathbf{U}_t = \text{Linear}(\mathbf{O}_t \| \mathbf{M}_t \| \mathbf{R}_t \| \mathbf{H}_{t-1}) \tag{9}$$

where $\|$ is concatenation. We directly use the hidden states from previous time step $\mathbf{H}_{t-1}$ as the embeddings for those missing nodes since no new features or graph structures of them are available at time step t. In NTS, observations are usually obtained by irregular sampling and the imputation problem over them can occur at any future step in real world problems. Being able to handle such uncertainty and forecasting unseen graph structure/time series data in the future time step are two key characteristics of an NTS imputation model. Therefore, in order to capture the dynamics between different timestamps and enhance the expressiveness of PoGeVon, we also encode the time information with learnable Fourier features based on Bochner's theorem [66, 67], whose properties are summarized in Proposition 3.3, as follows:

$$f(t) = \sqrt{\frac{1}{k}} [\cos(\mathbf{w}_1 t), \sin(\mathbf{w}_1 t), ..., \cos(\mathbf{w}_k t), \sin(\mathbf{w}_k t)] \tag{10}$$

where $\mathbf{w}_1, ..., \mathbf{w}_k$ are learnable parameters.

PROPOSITION 3.3. *Time encoding function $f(t)$ is invariant to time rescaling and generalizes to any future unseen timestamps.*

PROOF. See [32, 67]. □

Then, we concatenate node embeddings with time encodings through broadcasting as the input of a two-layer multi-layer perceptron (MLP) to predict the missing edges:

$$\mathbf{A}_t^{\text{out}} = \text{MLP}(\mathbf{U}_t \| \mathbf{H}_{t-1} \| f(t)) \tag{11}$$

The next step is to enhance node embeddings with updated graph structures. The general class of message-passing neural networks (MPNNs) [21] is used similar to the *aggregate and update* step in TGN to capture the graph topological information, which can be defined as:

$$\mathbf{H}_t^{\text{graph}} = \text{MPNN}(\mathbf{U}_t, \mathbf{A}_t^{\text{out}}) \tag{12}$$

whose detailed design can be found in Appendix.

3.3.3 Third-stage Feature Prediction.
In the third-stage prediction, we utilize the structural information $\mathbf{H}_t^{\text{graph}}$ to make a fine-grained imputation again over node features time series. Aiming to enhance the semantics of the node representations, we apply a self attention layer [60] to capture cross-node information in our third-stage prediction, which helps to encode richer node interaction information that is not captured in $\mathbf{H}_t^{\text{graph}}$. The latent node representations $\mathbf{Z}$, previous hidden state $\mathbf{H}_{t-1}$, the structural representation $\mathbf{H}_t^{\text{graph}}$ and the first stage output $\mathbf{O}_t$ as well as the masks $\mathbf{M}_t$ are all concatenated and processed by a self attention layer with an MLP to get the final output imputation representations:

$$\mathbf{H}_t^{\text{out}} = \text{MLP}(\text{Attn}(\mathbf{Z} \| \mathbf{H}_{t-1} \| \mathbf{H}_t^{\text{graph}} \| \mathbf{O}_t \| \mathbf{M}_t)) \tag{13}$$

Then a two-layer MLP is used for the third-stage prediction:

$$\hat{\mathbf{Y}}_{2,t} = \text{MLP}(\mathbf{H}_t^{\text{out}} \| \mathbf{H}_{t-1} \| \mathbf{H}_t^{\text{graph}}) \tag{14}$$

A filler operator similar to Eq. (8) is applied to get the imputation output $\mathbf{X}_t^{\text{out}}$ from $\hat{\mathbf{Y}}_{2,t}$. Finally, a single layer GRU is used similar to the *memory* step in TGN to update hidden representations based on the latent node representation $\mathbf{Z}$, the output of second-stage $\mathbf{X}_t^{\text{out}}$, the mask $\mathbf{M}_t$ and the structural representation $\mathbf{H}_t^{\mathcal{A}}$ for each node and move on to the next time step:

$$\mathbf{H}_t = \text{GRU}(\mathbf{Z} \| \mathbf{X}_t^{\text{out}} \| \mathbf{M}_t \| \mathbf{H}_t^{\text{graph}}) \tag{15}$$

3.3.4 Bidirectional Model.
Similar to [11], we extend our VAE model to bidirectional by replicating the architecture to handle both the forward and backward sequences. An MLP is used over the output hidden representations from these two VAEs to produce the final imputation output $\hat{\mathcal{Y}}$:

$$\hat{\mathcal{Y}} = \text{MLP}(\mathcal{H}_f^{\text{out}} \| \mathcal{H}_b^{\text{out}} \| \mathcal{H}_f^{\text{graph}} \| \mathcal{H}_b^{\text{graph}} \| \mathcal{H}_f \| \mathcal{H}_b) \tag{16}$$

where $\mathcal{H}^{\text{out}}$ is the tensor of imputation representations from the final stage prediction, f and b stand for forward and backward directions respectively. Algorithm 1 summarizes the detailed workflow of the proposed PoGeVon.

Algorithm 1 PoGeVon: Position-aware Graph Enhanced Variational Autoencoders

---

Input: A partially observed NTS: $\mathcal{G}(\tilde{\mathcal{A}}, \tilde{\mathcal{X}}) = \{\tilde{G}_1, \tilde{G}_2, ..., \tilde{G}_T\}$.
Output: The predicted tensor $\mathcal{X}$ of node feature time series and the predicted graph adjacency tensor $\mathcal{A}$.
1: Generate node position embeddings $\mathcal{R}$ based on Eq. (6).
2: **for** $e = 1, 2, 3, ...,$ num_epochs **do**
3:    Encode $\tilde{X}_f, \mathcal{M}_f, \mathcal{R}_f$ to get z_f based on Section 3.2.
4:    **for** $t = 1, 2, 3, ..., T$ (forward direction) **do**
5:       Perform first-stage decoding based on Section 3.3.1.
6:       Perform second-stage decoding based on Section 3.3.2.
7:       Perform third-stage decoding based on Section 3.3.3.
8:    **end for**
9:    Encode $\tilde{X}_b, \mathcal{M}_b, \mathcal{R}_b$ to get z_b based on Section 3.2.
10:   **for** $t = T, T - 1, ..., 1$ (backward direction) **do**
11:      Perform first-stage decoding based on Section 3.3.1.
12:      Perform Second-stage decoding based on Section 3.3.2.
13:      Perform third-stage decoding based on Section 3.3.3.
14:   **end for**
15:   Generate final outputs $\hat{\mathcal{Y}}$ based on Eq. (16).
16:   Update parameters θ, γ, ϕ by optimizing the loss in Eq. (19).
17: **end for**
18: Obtain the predicted tensor $\mathcal{X}$ of node feature time series based on Eq. (8) by replacing missing values in $\hat{X}$ with $\hat{\mathcal{Y}}$.
19: Obtain the predicted graph adjacency tensor $\mathcal{A}$ based on Eq. (8) by replacing missing values in $\hat{\mathcal{A}}$ with $\mathcal{A}^{\text{out}}$.
20: **return** the predicted tensor time series $\mathcal{X}$ and the predicted tensor of graph adjacency $\mathcal{A}$.

3.4 Objective and Training

The *Evidence Lower Bound* (*ELBO*) objective function of a vanilla conditional VAE [12, 14] over missing data imputations can be defined as:

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_q[\log p_\phi(\tilde{X}|z, \mathcal{M})] - \\ \mathbb{KL}[q_\theta(z|\tilde{X}, \mathcal{M})||p_\phi(z)] \le \log p_\phi(\tilde{X}|\mathcal{M}) \tag{17}$$

Our goal is to learn a good generative model of both the observed multivariate node feature time series $\tilde{X}$ and the observed graph adjacency $\tilde{\mathcal{A}}$. Thus, we can treat the position embeddings $\mathcal{R}$ as an extra condition in addition to the mask $\mathcal{M}$ similar to [25]. This is because, $\mathcal{M}$ and $\mathcal{R}$ are auxiliary covariates, and are given or can be generated through deterministic functions based on $\tilde{\mathcal{A}}$ and $\tilde{X}$ respectively. Therefore, it is more natural to maximize $\log p(\tilde{X}, \tilde{\mathcal{A}}|\mathcal{M}, \mathcal{R})$ as our objective, which is summarized in the following lemma.

LEMMA 3.4. *Under the condition that $\mathcal{M}$ and $\mathcal{R}$ are jointly independent of the prior $p(z)$: $p(z) = p(z|\mathcal{M}, \mathcal{R})$, the new ELBO objective of the proposed PoGeVon for the NTS imputation problem is:*

$$ELBO^{new}(\theta, \gamma, \phi) = \mathbb{E}_q[\log p_\phi(\tilde{X}|\tilde{\mathcal{A}}, \mathcal{M}, z)] \\ + \mathbb{E}_q[\log p_\gamma(\tilde{\mathcal{A}}|\mathcal{M}, \mathcal{R}, z) - \mathbb{KL}[q_\theta(z|\tilde{X}, \mathcal{M}, \mathcal{R})||p_\phi(z)]] \tag{18}$$

where γ denotes parameters of the link prediction module.

PROOF. The derivation of ELBO$^{\text{new}}$ can be formulated as:

$$\log p(\tilde{X}, \tilde{\mathcal{A}}|\mathcal{M}, \mathcal{R}) = \log \int p(\tilde{X}, \tilde{\mathcal{A}}|\mathcal{M}, \mathcal{R}, z)p(z)dz$$

$$= \log \int p(\tilde{X}|\tilde{\mathcal{A}}, \mathcal{M}, \mathcal{R}, z)p(\tilde{\mathcal{A}}|\mathcal{M}, \mathcal{R}, z)p(z)dz$$

since the node position embedding $\mathcal{R}$ can be generated from the observed graph adjacency $\tilde{\mathcal{A}}$,

$$= \log \int p(\tilde{X}|\tilde{\mathcal{A}}, \mathcal{M}, z)p(\tilde{\mathcal{A}}|\mathcal{M}, \mathcal{R}, z)p(z)\frac{q(z|\tilde{X}, \mathcal{M}, \mathcal{R})}{q(z|\tilde{X}, \mathcal{M}, \mathcal{R})}dz$$

$$= \log \mathbb{E}_q[p(\tilde{X}|\tilde{\mathcal{A}}, \mathcal{M}, z)p(\tilde{\mathcal{A}}|\mathcal{M}, \mathcal{R}, z)\frac{p(z)}{q(z|\tilde{X}, \mathcal{M}, \mathcal{R})}]$$

$$\ge \mathbb{E}_q[\log p(\tilde{X}|\tilde{\mathcal{A}}, \mathcal{M}, z)] + \mathbb{E}_q[\log p(\tilde{\mathcal{A}}|\mathcal{M}, \mathcal{R}, z)]$$

$$- \mathbb{KL}[q(z|\tilde{X}, \mathcal{M}, \mathcal{R})||p(z)] \qquad \square$$

This lemma generalizes the ELBO in Eq. (17) to the multi-task learning setting which ensures the learning objective of the proposed PoGeVon is consistent with our analysis in Section 3.1. That is, ELBO$^{\text{new}}$ corresponds to Eq. (1) by modeling dependencies between the observed node time series $\mathcal{X}$ and observed graph adjacency $\mathcal{A}$ similar to Eq. (5).

We use a similar strategy as in [12, 50] to maximize ELBO$^{\text{new}}$ by training our model over observed data and infer missing ones based on $p(\mathcal{X}|\tilde{\mathcal{X}}) \approx \int p(\mathcal{X}|z)q(z|\tilde{\mathcal{X}})dz$. We (1) use the mean absolute error (MAE) as the error function for the feature imputation and (2) use the Frobenius norm between the predicted adjacency matrices and the observed adjacency matrices as the link prediction loss. The model is trained by minimizing the following loss function which is composed of errors of all three stages:

$$\mathcal{L} = \underbrace{L(\hat{\mathcal{Y}}_{t:t+\Delta t}, \tilde{X}_{t:t+\Delta t}, \mathcal{M}_{t:t+\Delta t}) + \beta \cdot \mathbb{KL}_f + \beta \cdot \mathbb{KL}_b}_{\text{First and third terms in ELBO}^{\text{new}}} \tag{19}$$

$$+ \underbrace{L(O_{f,t:t+\Delta t}, \tilde{X}_{t:t+\Delta t}, \mathcal{M}_{t:t+\Delta t}) + L(O_{b,t:t+\Delta t}, \tilde{X}_{t:t+\Delta t}, \mathcal{M}_{t:t+\Delta t})}_{\text{Error for the 1}^{\text{st}}\text{ stage prediction}}$$

$$+ \underbrace{\gamma \cdot \|\tilde{\mathcal{A}}_{f,t:t+\Delta t} - \mathcal{A}^{\text{out}}_{f,t:t+\Delta t}\|_F + \gamma \cdot \|\tilde{\mathcal{A}}_{b,t:t+\Delta t} - \mathcal{A}^{\text{out}}_{b,t:t+\Delta t}\|_F}_{\text{Error for the 2}^{\text{nd}}\text{ stage prediction (i.e., second term in ELBO}^{\text{new}}\text{)}}$$

$$+ \underbrace{L(\mathcal{X}^{\text{out}}_{f,t:t+\Delta t}, \tilde{X}_{t:t+\Delta t}, \mathcal{M}_{t:t+\Delta t}) + L(\mathcal{X}^{\text{out}}_{b,t:t+\Delta t}, \tilde{X}_{t:t+\Delta t}, \mathcal{M}_{t:t+\Delta t})}_{\text{Error for the 3}^{\text{rd}}\text{ stage prediction}}$$

where β is the weight for KL divergence similar to [24] and γ is the weight for the 2$^{\text{nd}}$ stage prediction. The element wise error function $L(\mathcal{X}^{\text{pred}}, \mathcal{X}^{\text{label}}, \mathcal{M})$ outputs the average error by calculating the inner product between mask tensor $\mathcal{M}$ and $|\mathcal{X}^{\text{label}} - \mathcal{X}^{\text{pred}}|$. The loss $\mathcal{L}$ is optimized through each sample in the dataset which is a sliding window $(t : t + \Delta t)$ of NTS data (i.e., $\tilde{\mathcal{G}}_{t:t+\Delta t}$).

3.5 Complexity Analysis

The computational complexity of PoGeVon can be analyzed through the following aspects. First, calculating the position embedding $\mathcal{R}$ has the complexity $O(T \cdot \bar{E} \cdot \log \frac{1}{\epsilon})$ [61] where $\bar{E}$ is the average number of edges and ϵ is the absolute error bound for the power iteration of RWR. Second, with a standard bidirectional VAE based on GRU, MPNN increases the complexity by $O(\bar{E})$ with sparse matrix multiplications at each time step. Third, the self-attention used in the third-stage decoder has the complexity $O(N^2)$. There are several ways to reduce the overall time complexity. For example, most of the computations can be parallelized. One computational bottleneck lies in the computation of self-attention. The existing

techniques for efficient attentions [57] can be readily applied in the proposed PoGeVon, such as Linformer [63] which uses low-rank projections to make the cost of the attention mechanism $O(N)$ and Reformer [37] which applies locality sensitive hashing to reduce the complexity of attention to $O(N \cdot \log N)$.

4 EXPERIMENT

We apply the proposed PoGeVon to the networked time series imputation task, and evaluate it in the following aspects:

- $Q1$. How effective is PoGeVon for networked time series imputation?
- $Q2$. To what extent does our method benefit from different components of the model?

4.1 Experimental Setup

4.1.1 Datasets. We evaluate the proposed PoGeVon model on five real-world datasets, and the statistics of all the datasets are listed in Table 2.

Table 2: Statistics of the datasets. Entity numbers of PeMS* datasets refer to the original number of sensors/stations in the corresponding dataset and only part of them are used to build the graphs.

Dataset	# of entity	# of nodes	average # of edges	time length
COVID-19	50	50	1344.75	346
AQ36	36	36	341.57	8759
PeMS-BA	1632	64	675.45	25920
PeMS-LA	2383	64	1095.54	25920
PeMS-SD	674	64	1295.11	25920

- **COVID-19**: A dataset of COVID-19 infection cases and deaths in 50 states in USA from 01/21/2020 to 12/31/2020 [31]. Similar to [30], we choose infection cases of states as the time series data $\mathbf{X}$ and use mobility of people across different states to model the spatial relationship $\mathbf{A}$ between them. Then, we apply a Radial Basis Function (RBF) $f(u, v, t) = \exp(-\frac{||x_t^u - x_t^v||^2}{2\sigma^2})$ [8] to capture the dynamics and generate the graph sequence. Finally, we simulate the missing edges in the NTS imputation problem by masking edges when one of its end nodes contains missing features. Specifically, an edge weight $w_t^{u,v}$ between nodes u and v at time t can be defined as:

$$w_t^{u,v} = \begin{cases} w^{u,v} & \text{if } \mathbf{A}[u, v] \neq 0 \text{ and } f(u, v, t) > k \\ & \text{and } m_t^u = 1, m_t^v = 1. \\ 0 & \text{otherwise.} \end{cases} \quad (20)$$

where k is the positive threshold for graph dynamics and we choose $k = 0.3$ for COVID-19 dataset. We randomly mask out 25% of the node features in this dataset, and split the time axis to 70% for training, 10% for validation and 20% for test respectively.
- **AQ36**: A dataset of AQI values of different air pollutants collected from various monitor stations over 43 cities in China [80]. Following [5, 11], we use the reduced version of

the dataset which contains 36 nodes (**AQ36**) and pick the last four months as the test data. To construct the static graph $G(\mathbf{A}, \mathbf{X})$, we use the thresholded Gaussian kernel from [56] to get the pairwise distances $\mathbf{A}[u, v]$ between stations u and v as the edge weight. The graph sequence is constructed using the similar method as Eq. (20) over normalized time series features and the threshold k is set to 0.8. We use the same mask setting as [74] which simulates the true missing data distribution.
- **PeMS-BA/PeMS-LA/PeMS-SD** Three datasets contain traffic statistics based on the Caltrans Performance Measurement System (PeMS) [6], which cover the freeway system in major areas of California. We collect 5-minute interval traffic flow data from 3 different stations 4, 7 and 11 between 01/01/2022 and 03/31/2022, which represent the traffic information from Bay Area, Los Angeles and San Diego respectively. For each dataset, we pick 64 sensors with the largest feature variance, and use their latitude and longitude values to calculate pairwise distances to build the static graph. We only keep edges with weight within certain threshold, and we use 15 miles for **PeMS-BA**/**PeMS-LA** and 10 miles for **PeMS-SD**. The graph sequence is constructed using the similar method as the AQ36 dataset, and the threshold k is set to 0.8. We use similar masking settings as COVID-19 dataset.

The missing rate of AQ36's time series features is about 13.24%, while for COVID-19 dataset and all the traffic datasets, the time series features have 25% missing values. Based on Eq. (20), the missing rates of edges for AQ36 is 28.06%, for COVID-19 is 43.23%, and for PEMS-BA/PEMS-LA/PEMS-SD are 43.75%/43.74%/43.71% respectively.

To be consistent with the dataset settings in previous works such as GRIN [11], we use the following window length to train the models: (i) 14 for COVID-19 dataset which corresponds to 2 weeks, (ii) 36 for AQ36 dataset which corresponds to 1.5 days and (iii) 24 for all the traffic datasets which corresponds to 2 hours of data.

4.1.2 Baselines. We compare the proposed PoGeVon model with following baselines for the time series imputation task. All the methods are trained with NVIDIA Tesla V100 SXM2 GPU.

(1) **Mean**. Impute with node level feature average along the sequence.
(2) **Matrix Factorization (MF)**. Matrix factorization of the incomplete matrix with rank 10.
(3) **MICE** [64]. Multiple imputation by chained equations. The algorithm fills the missing values iteratively until convergence. We use 10 nearest features and set the maximum iterations to 100.
(4) **BRITS** [5]. BRITS has the similar bidirectional recurrent models as ours for time series imputation. It learns to impute only based on the time series features and does not consider the spatial information of the underlying graphs.
(5) **rGAIN** [11]. A GAN based imputation model which is similar to SSGAN [49]. rGAIN can be regarded as an extension of GAIN [75] with bidirectional encoder and decoder.
(6) **GRIN** [11]. GRIN is a state-of-the-art model for MTS imputation with the relational information from a static and

Table 3: Performance comparison over COVID-19 and AQ36 datasets. Smaller is better.

Models	COVID-19			AQ36		
	MAE	MSE	MRE	MAE	MSE	MRE
Mean	3.081 ±0.000	10.707 ±0.000	0.284 ±0.000	62.299 ±0.000	6525.709 ± 0.000	0.835 ±0.000
MF	0.276 ±0.026	0.165 ±0.025	0.026 ±0.002	39.582 ±0.189	4545.596 ± 61.411	0.531 ±0.002
MICE	0.077 ±0.005	0.013 ±0.002	0.007 ±0.000	38.889 ±0.268	4314.435 ± 20.617	0.521 ±0.003
BRITS	0.386 ±0.006	0.293 ±0.009	0.036 ±0.001	23.393 ±0.802	1276.226 ±102.916	0.314 ±0.011
rGAIN	0.579 ±0.069	0.571 ±0.106	0.055 ±0.006	25.032 ±1.426	1358.134 ±152.361	0.335 ±0.019
GRIN	0.319 ±0.038	0.165 ±0.040	0.029 ±0.004	29.420 ±0.231	2050.726 ± 56.028	0.394 ±0.003
NET^3	0.547 ±0.004	0.682 ±0.006	0.051 ±0.000	34.755 ±0.497	2473.718 ± 37.461	0.466 ±0.007
PoGeVon	**0.007 ±0.001**	**0.000 ±0.000**	**0.001 ±0.000**	**19.494 ±1.101**	**1213.474 ±125.529**	**0.261 ±0.015**

Table 4: Performance comparison over PeMS-BA, PeMS-LA and PeMS-SD datasets. Smaller is better.

Models	PeMS-BA			PeMS-LA			PeMS-SD		
	MAE	MSE	MRE	MAE	MSE	MRE	MAE	MSE	MRE
Mean	192.047 ±0.000	47504.159 ± 0.000	0.474 ±0.000	216.681 ±0.000	62664.657 ± 0.000	0.406 ±0.000	208.192 ±0.000	55780.002 ± 0.000	0.529 ±0.000
MF	57.265 ±1.148	8091.407 ±185.123	0.141 ±0.003	77.339 ±0.699	15202.678 ±156.348	0.145 ±0.001	45.811 ±0.318	6044.345 ± 72.976	0.117 ±0.001
MICE	50.861 ±0.765	6724.148 ±109.829	0.126 ±0.002	64.018 ±1.015	10822.355 ±405.410	0.120 ±0.002	38.978 ±1.036	4771.186 ± 92.335	0.100 ±0.003
BRITS	30.274 ±0.095	2942.411 ± 16.511	0.075 ±0.000	36.921 ±0.133	3681.595 ± 21.635	0.069 ±0.000	21.232 ±0.059	1563.234 ± 28.309	0.054 ±0.000
rGAIN	38.862 ±0.752	3422.914 ± 61.281	0.096 ±0.002	49.611 ±1.083	5533.964 ±234.335	0.093 ±0.002	33.212 ±1.475	2341.466 ± 98.314	0.085 ±0.004
GRIN	30.057 ±1.073	1922.072 ± 74.327	0.074 ±0.003	47.835 ±2.059	4561.512 ±298.533	0.090 ±0.004	41.001 ±1.543	3000.012 ±201.018	0.105 ±0.004
NET^3	35.671 ±0.111	2735.574 ± 6.138	0.009 ±0.000	37.652 ±0.113	3416.784 ± 6.765	0.071 ±0.000	34.111 ±0.184	2487.581 ± 9.798	0.087 ±0.000
PoGeVon	**22.194 ±0.046**	**1248.681 ± 4.297**	**0.055 ±0.000**	**23.905 ±0.245**	**1714.962 ± 31.035**	**0.045 ±0.000**	**18.990 ±0.112**	**951.559 ± 8.264**	**0.048 ±0.000**

accurately known graph, which uses MPNN to build a spatio-temporal recurrent module and solves the problem in a bidirectional way.

(7) **NET^3** [28]. NET^3 is a recent work focusing on tensor time series learning and assumes that the tensor graphs are fixed and accurately known.

NTS imputation (i.e., Problem 1) also aims to solve the link prediction problem. We compare the performance of our method with following baselines:

(1) **VGAE** [36]. Vanilla variational graph autoencoder is the first work that brings VAE to graph learning, and has competitive performance on link prediction task over static graphs.

(2) **VGRNN** [23]. Variational graph recurrent neural networks extends VGAE to handle temporal information with the help of RNNs, and is a powerful baseline for the link prediction task on dynamic graphs.

4.1.3 Metrics. We use *mean absolute error* (MAE), *mean squared error* (MSE) and *mean relative error* (MRE) to evaluate the imputation performance of all models over missing features. For the link prediction task, we use the Frobenius norm as the metric since all the edges are weighted. All the experiments are run with 5 different random seeds and the results are presented as mean ± standard deviation (std).

4.2 Time Series Imputation Task

Empirical results from Table 3 and Table 4 demonstrate that the proposed PoGeVon outperforms all the baselines over the time series missing values prediction task in the NTS imputation problem. In particular, PoGeVon achieves more than 10% improvement on all the datasets compared with the best baselines. Especially, PoGeVon has significant improvements over all the baselines over COVID-19 dataset where other neural network based models have even worse

performance than traditional time series imputation methods such as MF and MICE. It is worth noting that, although equipped with modules to handle topological information from graphs, GRIN and NET^3 are less competitive than PoGeVon when the graph is constantly changing and contains missing edges. On the AQ36 dataset and the PeMS-SD dataset, they bear worse performance compared to BRITS and rGAIN, which do not leverage any topological information. PoGeVon outperforms BRITS and rGAIN by at least 12.92% and 10.55% on these two datasets respectively, which further indicates the effectiveness of our method. The main reason PoGeVon fluctuates (with a large std) on AQ36 dataset compared with traffic datasets is that AQ36 has fewer training samples (time steps), which brings more uncertainty for the model and results in larger differences of performances using different random seeds.

4.3 Link Prediction Task

Table 5: Performance comparison of the link prediction task in NTS imputation. Smaller is better.

Models	AQ36	PeMS-BA	PeMS-LA	PeMS-SD
VGAE	134.42 ±0.11	431.94 ±2.31	404.17 ±1.76	399.78 ±1.29
VGRNN	133.92 ±0.29	428.82 ±0.01	402.30 ±0.90	398.81 ±0.01
PoGeVon	**95.42 ±1.80**	**148.44 ±0.31**	**168.05 ±0.31**	**185.86 ±0.15**

VGAE and VGRNN were originally designed for link prediction over unweighted graphs. However, all the graphs are weighted in our NTS imputation settings, and thus, we modify these models correspondingly and apply the same Frobenius loss function we use in PoGeVon to train them. All the results are listed in Table 5. Both baselines have relatively worse performance compared to PoGeVon in all the datasets, and even using RNNs, VGRNN only

gains minor improvement over VGAE. This indicates that both VGAE and VGRNN may not be able to handle the link prediction task over weighted dynamic graphs very well.

4.4 Ablation Studies

Table 6: Ablation study of PoGeVon over AQ36 dataset on time series feature imputation. Smaller is better.

Models	MAE	MSE	MRE
PoGeVon	**19.49** ±1.10	**1213.47** ±125.53	**0.26** ±0.02
change RWR to SPD	21.98 ±1.55	1309.55 ±199.24	0.33 ±0.02
w/o link prediction in 2nd stage	28.71 ±3.38	2130.46 ±417.45	0.38 ±0.05
w/o self-attention in 3rd stage	23.40 ±1.00	1576.06 ±194.45	0.31 ±0.01

To evaluate the effectiveness of different components of our proposed method, we compare PoGeVon with following variants: (1) Replace RWR node position embeddings with the shortest path distance (SPD) based node embeddings by calculating the distance between each node with anchor nodes. (2) Remove the link prediction module in the 2nd stage prediction. (3) Remove the self-attention module in the 3rd stage prediction by replacing it with a linear layer. The results of the ablation study over AQ36 dataset are shown in Table 6. As we can see, the proposed method PoGeVon indeed performs the best which corroborates the necessity of all these components in the model.

5 RELATED WORK

In this section, we review the related works which can be categorized into two groups, including (1) multivariate time series imputation and (2) GNNs with relative position encodings.

Multivariate Time Series Imputation. In addition to traditional methods such as ARIMA [2] and K-Nearest Neighbors (KNN) [7], deep learning models are widely adopted in recent years to solve the MTS imputation problem. BRITS [5] is one of the most representative methods which uses bidirectional RNNs. There also exist a wide range of methods using deep generative models such as generative adversarial nets (GAN) [22] and VAE [34]. GAIN [75] is one of the earliest methods that use GAN to impute missing data, and later [45] applies GAN to the multivariate time series setting based on 2-stage imputation. E^2GAN [46] is an end-to-end GAN and uses the noised compression and reconstruction strategy to generate more reasonable imputed values compared to previous works. SSGAN [49] proposes a novel method based on GAN to handle missing data in partially labeled time series data. VAE is used in GP-VAE [17] to solve the MTS imputation task with Gaussian process as the prior.

Other works handle MTS imputation problem from the perspective of spatial-temporal modeling, which takes the advantage of entities relations from the underlying graph. [4] is the first trial of using matrix factorization algorithm to recover missing values over MTS data with graph structures. More recently, GNNs have been used to capture the topological information in the MTS data.

GRIN [11] proposes a novel bidirectional message passing RNN with a spatial decoder to handle both the spatial and temporal information. SPIN [47] uses sparse spatiotemporal attention to capture inter-node and intra-node information for predicting missing values in MTS. NET^3 [28] generalizes the problem to tensor time series where multiple modes of relation dependencies exist in the time series. It introduces a tensor GCN [35] to handle the tensor graphs and then proposes a tensor RNN to incorporate the temporal dynamics. One common limitation of all these methods is that they either ignore the topological information from graph or assume the graph is fixed and accurately known.

GNNs with Relative Position Encodings. The expressive power of message-passing based GNNs has been proved to be bounded by 1-Weisfeiler-Lehman test (1-WL test) in [68]. Many follow-up works have been done to improve the expressive power of GNNs which go beyond 1-WL test, and position-aware graph neural networks (P-GNNs) [76] is one of them. P-GNNs randomly picks sets of anchor nodes and learn a non-linear distance-weighted aggregation scheme over these anchor sets for each node. This relative position encodings for nodes are proved to be more expressive than regular GNNs. Distance Encoding [39] uses graph-distance measures between nodes as extra features and proves that it can distinguish node sets in most regular graphs in which message-passing based GNNs would fail. [15] proposes a novel module for learnable structural and positional encodings (LSPE) along with GNNs and Transformers [60], which generates more expressive node embeddings. Recently, PEG [62] is introduced for imposing permutation equivariance and stability to position encodings, which uses separate channels for node features and position features. Compared with these existing methods, our proposed RWR-based position embedding could capture more topological information from the entire graph, as our analysis in Section 3.2.1 shows.

6 CONCLUSION

In this paper, we focus on solving networked time series imputation problem, which has two main challenges: (1) the graph is dynamic and missing edges exist, and (2) the node features time series contain missing values. To tackle these challenges, we propose PoGeVon, a novel VAE model utilizing specially designed RWR-based position embeddings in the encoder. For the decoder, we design a 3-stage predictions to impute missing values in both features and structures complementarily. Experiments on a variety of real-world datasets show that PoGeVon consistently outperforms strong baseline methods for the NTS imputation problem.

7 ACKNOWLEDGEMENTS

REFERENCES

[1] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. 2016. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410* (2016).

[2] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control.* John Wiley & Sons.

[3] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. 2018. Understanding disentangling in β-VAE. *arXiv preprint arXiv:1804.03599* (2018).

[4] Yongjie Cai, Hanghang Tong, Wei Fan, and Ping Ji. 2015. Fast mining of a network of coevolving time series. In *Proceedings of the 2015 SIAM International Conference on Data Mining.* SIAM, 298–306.

[5] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. 2018. Brits: Bidirectional recurrent imputation for time series. *Advances in neural information processing systems* 31 (2018).

[6] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. 2001. Freeway performance measurement system: mining loop detector data. *Transportation Research Record* 1748, 1 (2001), 96–102.

[7] Jiahua Chen and Jun Shao. 2000. Nearest neighbor imputation for survey data. *Journal of official statistics* 16, 2 (2000), 113.

[8] Yuzhou Chen, Ignacio Segovia, and Yulia R Gel. 2021. Z-GCNETs: time zigzags at graph convolutional networks for time series forecasting. In *International Conference on Machine Learning.* PMLR, 1684–1694.

[9] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2020. Adaptive universal generalized pagerank graph neural network. *arXiv preprint arXiv:2006.07988* (2020).

[10] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[11] Andrea Cini, Ivan Marisca, and Cesare Alippi. 2021. Filling the g_ap_s: Multivariate time series imputation by graph neural networks. *arXiv preprint arXiv:2108.00298* (2021).

[12] Mark Collier, Alfredo Nazabal, and Christopher KI Williams. 2020. VAEs in the presence of missing data. *arXiv preprint arXiv:2006.05301* (2020).

[13] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *Twenty-fourth international joint conference on artificial intelligence.*

[14] Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* (2016).

[15] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2021. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875* (2021).

[16] Chenguang Fang and Chen Wang. 2020. Time series data imputation: A survey on deep learning approaches. *arXiv preprint arXiv:2011.11347* (2020).

[17] Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. 2020. Gp-vae: Deep probabilistic time series imputation. In *International conference on artificial intelligence and statistics.* PMLR, 1651–1661.

[18] Dongqi Fu, Liri Fang, Ross Maciejewski, Vetle I. Torvik, and Jingrui He. 2022. Meta-Learned Metrics over Multi-Evolution Temporal Graphs. In *KDD 2022.*

[19] Dongqi Fu and Jingrui He. 2021. SDG: A Simplified and Dynamic Graph Neural Network. In *SIGIR 2021.*

[20] Dongqi Fu, Dawei Zhou, and Jingrui He. 2020. Local Motif Clustering on Time-Evolving Graphs. In *KDD 2020.*

[21] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning.* PMLR, 1263–1272.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[23] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. 2019. Variational graph recurrent neural networks. *Advances in neural information processing systems* 32 (2019).

[24] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework. (2016).

[25] Oleg Ivanov, Michael Figurnov, and Dmitry Vetrov. 2018. Variational autoencoder with arbitrary conditioning. *arXiv preprint arXiv:1806.02382* (2018).

[26] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2020. Recurrent Event Network: Autoregressive Structure Inferenceover Temporal Knowledge Graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP).* 6669–6683.

[27] Baoyu Jing, Chanyoung Park, and Hanghang Tong. 2021. Hdmi: High-order deep multiplex infomax. In *Proceedings of the Web Conference 2021.* 2414–2424.

[28] Baoyu Jing, Hanghang Tong, and Yada Zhu. 2021. Network of tensor time series. In *Proceedings of the Web Conference 2021.* 2425–2437.

[29] Baoyu Jing, Si Zhang, Yada Zhu, Bin Peng, Kaiyu Guan, Andrew Margenot, and Hanghang Tong. 2022. Retrieval Based Time Series Forecasting. *arXiv preprint arXiv:2209.13525* (2022).

[30] Amol Kapoor, Xue Ben, Luyang Liu, Bryan Perozzi, Matt Barnes, Martin Blais, and Shawn O'Banion. 2020. Examining covid-19 forecasting using spatio-temporal graph neural networks. *arXiv preprint arXiv:2007.03113* (2020).

[31] Satya Katragadda, Ravi Teja Bhupatiraju, Vijay Raghavan, Ziad Ashkar, and Raju Gottumukkala. 2022. Examining the COVID-19 case growth rate due to visitor vs. local mobility in the United States using machine learning. *Scientific Reports* 12, 1 (2022), 1–12.

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

[33] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[34] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[35] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[36] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[37] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451* (2020).

[38] MI Knight, MA Nunes, and GP Nason. 2016. Modelling, detrending and decorrelation of network time series. *arXiv preprint arXiv:1603.03221* (2016).

[39] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems* 33 (2020), 4465–4478.

[40] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence.*

[41] Xiaohan Li, Mengqi Zhang, Shu Wu, Zheng Liu, Liang Wang, and S Yu Philip. 2020. Dynamic graph collaborative filtering. In *2020 IEEE International Conference on Data Mining (ICDM).* IEEE, 322–331.

[42] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

[43] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. 2021. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems* 34 (2021), 20887–20902.

[44] Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016).

[45] Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, et al. 2018. Multivariate time series imputation with generative adversarial networks. *Advances in neural information processing systems* 31 (2018).

[46] Yonghong Luo, Ying Zhang, Xiangrui Cai, and Xiaojie Yuan. 2019. E2gan: End-to-end generative adversarial network for multivariate time series imputation. In *Proceedings of the 28th international joint conference on artificial intelligence.* AAAI Press, 3094–3100.

[47] Ivan Marisca, Andrea Cini, and Cesare Alippi. 2022. Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations. *arXiv preprint arXiv:2205.13479* (2022).

[48] William McGill. 1954. Multivariate information transmission. *Transactions of the IRE Professional Group on Information Theory* 4, 4 (1954), 93–111.

[49] Xiaoye Miao, Yangyang Wu, Jun Wang, Yunjun Gao, Xudong Mao, and Jianwei Yin. 2021. Generative semi-supervised learning for multivariate time series imputation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 8983–8991.

[50] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. 2020. Handling incomplete heterogeneous data using vaes. *Pattern Recognition* 107 (2020), 107501.

[51] George Panagopoulos, Giannis Nikolentzos, and Michalis Vazirgiannis. 2021. Transfer graph neural networks for pandemic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4838–4845.

[52] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[53] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020.* 259–270.

[54] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637* (2020).

[55] Alex Rubinsteyn and Sergey Feldman. 2016. fancyimpute: An Imputation Library for Python. https://github.com/iskandr/fancyimpute

[56] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains.

IEEE signal processing magazine 30, 3 (2013), 83–98.

[57] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *Comput. Surveys* 55, 6 (2022), 1–28.

[58] Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057* (2000).

[59] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *Sixth international conference on data mining (ICDM'06)*. IEEE, 613–622.

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[61] Hanzhi Wang, Zhewei Wei, Junhao Gan, Sibo Wang, and Zengfeng Huang. 2020. Personalized pagerank to a target node, revisited. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 657–667.

[62] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. 2022. Equivariant and stable positional encoding for more powerful graph neural networks. *arXiv preprint arXiv:2203.00199* (2022).

[63] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768* (2020).

[64] Ian R White, Patrick Royston, and Angela M Wood. 2011. Multiple imputation using chained equations: issues and guidance for practice. *Statistics in medicine* 30, 4 (2011), 377–399.

[65] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 753–763.

[66] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2019. Self-attention with functional time representation learning. *Advances in neural information processing systems* 32 (2019).

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

[68] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[69] Wentao Xu, Weiqing Liu, Lewen Wang, Yingce Xia, Jiang Bian, Jian Yin, and Tie-Yan Liu. 2021. HIST: A Graph-based Framework for Stock Trend Forecasting via Mining Concept-Oriented Shared Information. *arXiv preprint arXiv:2110.13716* (2021).

[70] Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. 2021. Dynamic knowledge graph alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4564–4572.

[71] Yuchen Yan, Si Zhang, and Hanghang Tong. 2021. Bright: A bridging algorithm for network alignment. In *Proceedings of the Web Conference 2021*. 3907–3917.

[72] Yuchen Yan, Qinghai Zhou, Jinning Li, Tarek Abdelzaher, and Hanghang Tong. 2022. Dissecting cross-layer dependency inference on multi-layered interdependent networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2341–2351.

[73] Raymond W Yeung. 1991. A new outlook on Shannon's information measures. *IEEE transactions on information theory* 37, 3 (1991), 466–474.

[74] Xiuwen Yi, Yu Zheng, Junbo Zhang, and Tianrui Li. 2016. ST-MVL: filling missing values in geo-sensory time series data. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*.

[75] Jinsung Yoon, James Jordon, and Mihaela Schaar. 2018. Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning*. PMLR, 5689–5698.

[76] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware graph neural networks. In *International conference on machine learning*. PMLR, 7134–7143.

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

[78] Qi Zhang, Jianlong Chang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2020. Spatio-temporal graph structure learning for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1177–1185.

[79] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. 2022. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082* (2022).

[80] Yu Zheng, Xiuwen Yi, Ming Li, Ruiyuan Li, Zhangqing Shan, Eric Chang, and Tianrui Li. 2015. Forecasting fine-grained air quality based on big data. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2267–2276.

[81] Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. 2021. Graph neural networks with heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11168–11176.

A APPENDIX

In the appendix, we present the additional details of PoGeVon including

- Proofs of Proposition 3.1 and Theorem 3.2 in Section A.1 and Section A.2 respectively.
- Additional details of components in PoGeVon is introduced in Section A.3.
- Reproducibility and parameter settings of baselines and the proposed PoGeVon are listed in Section A.4.
- Additional experiments such as visualization for prediction and sensitivity analysis are given in Section A.5.
- Section A.6 discusses the limitations of our implementations of PoGeVon and propose some potential future works based on NTS imputation.

A.1 Proof Proposition 3.1

We prove Proposition 3.1 by analyzing the properties of RWR.

PROPOSITION. *Random walk with restarts (RWR) captures information from close neighbors (local) and long-distance neighbors (global) in graph learning.*

PROOF. Based on Eq. (6), the closed form solution for RWR can be derived as: $\mathbf{r}_i = c \cdot (\mathbf{I} - (1 - c) \cdot \hat{\mathbf{A}})^{-1} \mathbf{e}_i$, where $\mathbf{I}$ is the identity matrix. We could also solve this equation by power iterations: $(\mathbf{I} - (1 - c) \cdot \hat{\mathbf{A}})^{-1} \approx \sum_{t=0}^{\infty} ((1 - c) \cdot \hat{\mathbf{A}})^t$. First of all, as the power term t goes to infinity, the position embedding $\mathbf{R}$ can indeed capture *global* information of graph. Second, the restart probability c ensures nodes close to anchor nodes have larger values than those farther away, which encodes the *local* information of graph.

Remark. Proposition 3.1 holds with the assumption that the graph is connected. When graph is not connected, with proper choice of a set of anchor nodes that cover all the connected components, RWR is able to *global* information within each components rather than the entire graph. An alternative to generate node position embeddings is using RWR from each node similar to [15] to get the landing probability of a node to itself for multiple steps. However, this usually increases the complexity when having a large graph and still faces similar issues as ours when graph is less connected. □

A.2 Proof of Theorem 3.2

To prove Theorem 3.2, we first introduce the following proposition.

PROPOSITION A.1. *For any random variables A, B, and C, the following inequality of the mutual information $I(\cdot; \cdot)$ holds [53]:*

$$I(A, B; C) \geq I(A; C) \quad (21)$$

PROOF. Based on the chain rule of mutual information, we have:

$$I(A, B; C) = H(A, B) - H(A, B|C)$$
$$= H(A) + H(B|A) - H(A|C) - H(B|A, C)$$
$$= I(A; C) + I(B; C|A)$$

where $H(A)$ is the marginal entropy, $H(A|C)$ is the conditional entropy and $H(A, B)$ is the joint entropy. Since the mutual information $I(B; C|A) \geq 0$, we can conclude that $I(A, B; C) \geq I(A; C)$. □

Now we can prove Theorem 3.2 as:

THEOREM. *Given a temporal graph $\mathcal{G}$, TGN with RWR-based node position embeddings g_θ has more expressive power than regular TGN f_θ in node representation learning: $\mathbb{D}(g(u), g(v)) \geq \mathbb{D}(f(u), f(v))$*

where $\mathbb{D}(\cdot, \cdot)$ measures the expressiveness by counting the distinguishable node pairs (u, v) in $\mathcal{G}$ based on node representations.

PROOF. It is natural to see that g_θ has at least same expressive power as f_θ since we add additional information with the positional embeddings for each node. By setting all the parameters of g_θ that handle such positional embeddings to zero, we will have a regular TGN model same as f_θ.

To prove that why the additional information brought by node position embeddings is useful for node representation learning, we provide following analysis with the help of Proposition A.1. Regular TGN only encodes topologically local information within q-hop neighbors and q usually is a small number because of the over-smoothing problem [40], we denote the random variable for f_θ's node representations as X_{local}. Based on Proposition 3.1, we know that the random variable for g_θ's node representations $X_{local+global}$ follows the joint distribution of both local and global topological information. The objective of a node representation learning task over graph $\mathcal{G}$ can be denoted as $\max I(X; Y)$ where Y is the random variable follows label distributions [76]. This derivation can be obtained from Information Bottleneck we discussed in Section 3.1 without the constraints term. Therefore, based on Proposition A.1, we have $I(X_{local+global}; Y) \geq I(X_{local}; Y)$ which denotes that g_θ has more expressive power than f_θ. □

A.3 Additional Details over PoGeVon

The message-passing neural network (MPNN) used in PoGeVon is defined as:

$$\text{MPNN}(F_u, F_m, \mathbf{d}_{t,i}, \mathbf{A}) = F_u(\mathbf{d}_{t,i}, \sum_{j \in \mathcal{N}(i)} F_m(\mathbf{h}_{t,i}, \mathbf{d}_{t,j}, e_{i,j})) \quad (22)$$

where F_u and F_m are update and message functions with learnable parameters, $\mathbf{d}_{t,i}$ is the node representation for node i at time step t, $e_{i,j}$ is the edge weight between node i and j, and $\mathcal{N}(i)$ represents node i's neighbors. The two-layer MPNNs with skip connection used in PoGeVon can be defined as:

$$\mathbf{H}_{1,t} = \text{MPNN}(F_u^1, F_m^1, \mathbf{U}_t, \mathbf{A}_t^{\text{out}})$$
$$\mathbf{H}_{2,t} = \text{MPNN}(F_u^2, F_m^2, \mathbf{H}_{t,1}, \mathbf{A}_t^{\text{out}}) \quad (23)$$
$$\mathbf{H}_t^{\text{graph}} = \mathbf{H}_{1,t} \oplus \mathbf{H}_{2,t}$$

where $\oplus$ is the element-wise addition.

A.4 Reproducibility

We introduce the detailed parameter settings of our models as well as baselines in this subsection. For PoGeVon, the restart probability c of RWR for position embeddings is set to the commonly used 0.15, and we picked the number of anchor nodes as $L = \log_2(N)$. We set the hidden dimension to be 64, β to be 0.2 and γ to be 0.01.

All the experiments are based on codes from a open source library[2] [55] and those provided by corresponding authors. We modify their implementations for the NTS imputation problem and the details of parameter settings are listed as follows: for the parameters for each baseline model in the time series feature imputation, we refer to previous works for their settings [5, 11]. For BRITS[3], we use the same hidden dimension as [5] for AQ36 dataset, and for the

---

[2]https://github.com/iskandr/fancyimpute
[3]https://github.com/caow13/BRITS

traffic datasets, the hidden size is set to 256 which is aligned with the setting used in PeMS data in [11]. For rGAIN, we follow the exact same setting as in [11, 47] in which we use 64 as the hidden size for AQ36 dataset and 256 for traffic datasets. For GRIN[4], we use the same hidden size for AQ36 as [11], while using 80 for the traffic datasets since they are much larger than AQ36. As for the hidden dimension of NET[3] [5], we use 128 for AQ36 dataset and 256 for traffic datasets. For COVID-19 dataset, we use the same parameters settings of traffic datasets for all the models .

For VGAE and VGRNN in the link prediction task in NTS imputation, we use hidden size 256/128 for the AQ36 and 320/150 for the traffic datasets respectively.

We train all the models using PyTorch [52] with Adam optimizer [33], learning rates are set to be 0.001/0.01 for time series feature imputation and link prediction baselines respectively with cosine annealing scheduler [44] to adjust the values dynamically. The batch sizes are all set to 32 and we use validation dataset for early stopping. The codes of PoGeVon will be released upon acceptance.

A.5 Additional Experiments

A.5.1 Visualization of Prediction. The prediction results of different baselines over PeMS-LA with 50% missing rates over test data can be found in Figure 4. It is clear that PoGeVon can achieve better
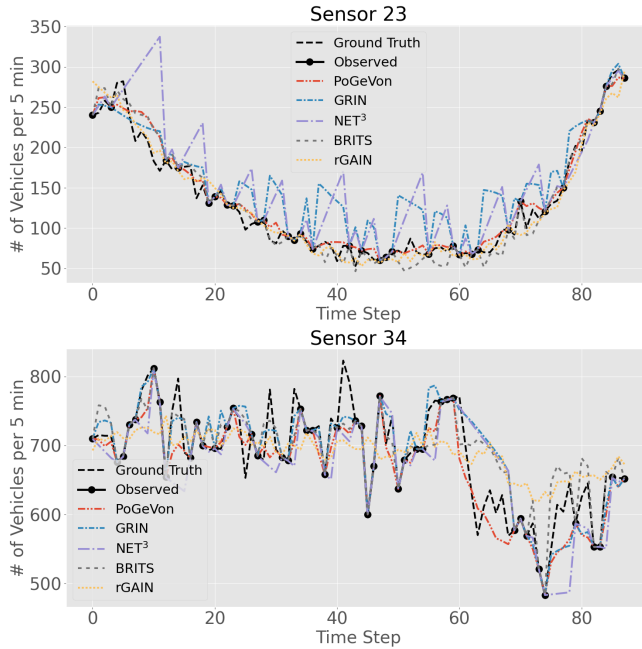

Figure 4: Different models' predictions of traffic flow in sensor 23 and 34 from PeMS-LA dataset. Best viewed in color.

predictions results compared with other baselines. In particular, GRIN and NET[3] sometimes suffer a lot from fluctuations due to the missing edges in the NTS data, which result in poor performance compared to our proposed PoGeVon. Besides, we can see that PoGeVon can have finer predictions compared with all the baselines when there exist abrupt change of data values which also

---

[4]https://github.com/Graph-Machine-Learning-Group/grin
[5]https://github.com/baoyujing/NET3

has high missing rates (e.g., data from time step 60 to 65 in the figure of prediction of sensor 34).

A.5.2 Sensitivity Analysis. We conduct sensitivity analysis to study the effect brought by increasing the masking rates. We consider the following mask rates: 15%, 25%, 35%, 45%. In order to keep a reasonable edge missing rate, for each edge with either end node being masked, they have 70% of chance being masked instead of using the setting from Eq. (20). The results are shown in Figure 5, in which
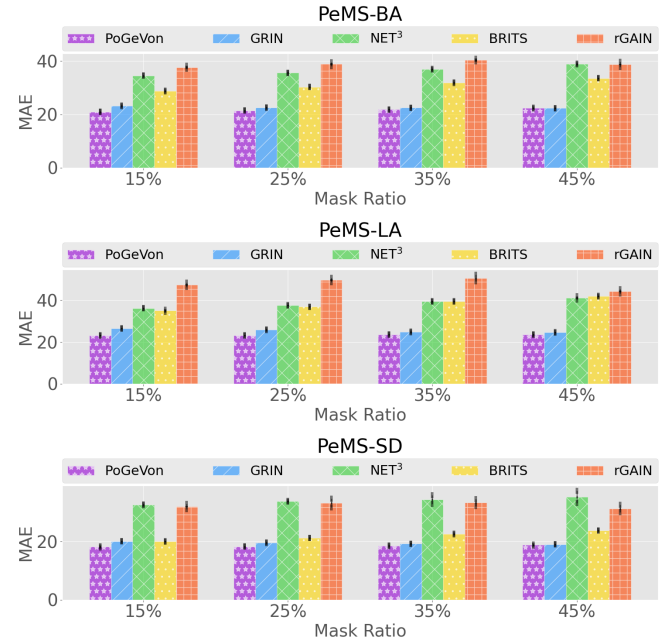

Figure 5: Sensitivity analysis for time series imputation with different masking rates on the traffic dataset. Lower is better. Best viewed in color.

the error bar demonstrates the standard deviation of MAE over 5 runs with different random seeds. The proposed PoGeVon consistently outperforms all the baselines in these settings which further demonstrates the effectiveness and robustness of our method.

A.6 Limitations and Future Works

One limitation of the proposed PoGeVon model lies in its quadratic complexity $O(N^2)$ due to the self-attention module. As we have discussed in Section 3.5, we can reduce this complexity to either $O(N)$ by Linformer [63] or $O(N \log N)$ by Reformer [37]. Another limitation lies in the potential negative transfer effect, which might happen when negative correlation exists between the time series of adjacent node pairs. Under such circumstances, directly applying multi-task learning framework in PoGeVon might hurt the performance of NTS imputation. A possible solution is to resort to GNNs designed for graphs with heterophily [9, 43, 79, 81] in the decoder of the proposed PoGeVon. There are several interesting aspects that are worth future study, including (1) generalizing the proposed PoGeVon for detecting anomalies on NTS, forecasting NTS [29, 65] and assisting temporal graphs analysis [18, 20]; (2) applying it to temporal knowledge graph completion [26] and alignment [70] as well as dynamic recommendations [41].