Forecasting Spatio-Temporal Time Series with Missing Values Using Graph Neural Networks

Armando Martinez-Ruiz, Pilar Gomez-Gil, Rigoberto Fonseca-Delgado

Reporte Técnico No. CCC-26-001
11 de marzo de 2026

Luis Enrique Erro 1
Sta. Ma. Tonantzintla,
72840, Puebla, México.

Forecasting Spatio-Temporal Time Series with Missing Values Using Graph Neural Networks

Armando Martinez-Ruiz Pilar Gomez-Gil Rigoberto Fonseca-Delgado

Computer Science Department
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro # 1, Santa María Tonantzintla, Puebla, 72840, México
E-mail:
armandomtzruiz@inaoep.mx
pgomez@inaoep.mx
rfonseca@yachaytech.edu.ec

Abstract

Spatio-temporal data underpins a wide range of critical applications, including traffic forecasting, energy management and environmental monitoring. Although deep learning models such as recurrent neural networks (RNNs) and long short-term memory networks (LSTMs) have demonstrated success in modelling temporal dependencies, they often fail to capture the underlying graph structure inherent in many spatio-temporal datasets. Graph Neural Networks (GNNs) and their extensions to Spatial-Temporal Graph Neural Networks (ST-GNNs) offer a more natural framework for modelling both spatial and temporal dynamics. However, existing ST-GNN models are highly sensitive to missing data, since it weakens graph connectivity and undermine the effectiveness of spectral-based regularization techniques, which degrades forecasting performance. This research proposes the development of a learning framework for ST-GNNs that improves robustness and accuracy in missing data environments. The proposed approach aims to integrate training strategies such as spectral based regularization or data imputation methods to address challenges associated with incomplete data. The goal is to design a unified model capable of forecasting while managing missing values leveraging spatial dependencies within the spatio-temporal dataset. The expected contributions include the design of a modular ST-GNN architecture, empirical evaluation of imputation impacts on forecasting and the establishment of reproducible benchmarking tools for fair comparisons with state-of-the-art models.

Keywords: *time series forecasting, graph neural networks, data imputation, spectral based regularization*

1 Introduction

Spatio-temporal data plays a crucial role in a wide range of applications, including traffic flow prediction, energy demand forecasting and environmental monitoring. Advances in deep learning have provided powerful tools to analyze such data. However, in real-world scenarios, it is common for datasets to contain missing values or sensor errors, which significantly hinders the effectiveness of standard forecasting models.

Traditional deep learning models such as recurrent neural networks (RNNs) and long short-term memory networks (LSTMs) [11] have been applied to spatio-temporal tasks. However, these models often ignore the underlying graph structure that characterizes spatio-temporal datasets, limiting their accuracy and generalization capabilities. In contrast, Graph Neural Networks (GNNs) [25] are designed to to operate in graph-structure data, making them a better fit for modelling spatial relationships [20]. Extending this idea, Spatial-Temporal Graph Neural Networks (ST-GNNs) [13] incorporate both spatial and temporal dependencies, enabling them to forecast dynamic processes over evolving networks.

ST-GNNs offer increased flexibility compared to traditional architectures, by leveraging dynamic graph topologies. Despite their effectiveness, ST-GNNs struggle when faced with challenges such as missing data. Missing data weakens the graph connectivity and disrupts spectral-based regularization techniques[3]. These issues limit the capabilities of ST-GNNs, reducing their ability to model long-range spatio-temporal dependencies, which are crucial in real world applications.

To address these limitations, this research propose to design a learning algorithm that incorporates training strategies and imputation methods to enhances the robustness and accuracy of ST-GNNs in missing data environments, to get a unified framework that tackles imputation and forecasting tasks alongside. By leveraging spectral-based regularization and data imputation, the goal is to develop a model that is capable of improving forecasting performance while exploiting the inherent spatial structure of the data in scenarios where data contains missing values.

1.1 Justification

- Spatio-temporal data is essential in real-world applications like traffic, energy and environmental forecasting, yet current models struggle with capturing both spatial and temporal dependencies, especially when data has missing values.
- Traditional deep learning methods often ignore the graph structure or lack robustness in missing data environments, leading to reduced accuracy and generalization.
- Compared to traditional deep learning architectures like convolutional neural networks or recurrent neural networks, graph neural networks are more flexible in handling irregular and sparse data, as they can model dynamic and incomplete topologies by leveraging the graph structure.

1.2 Problem Statement

Spatio-Temporal Graph Neural Networks have shown strong potential for time series forecasting, but their performance degrades significantly under scenarios with missing data due to several challenges:

- **Weakened graph connectivity**: Missing data disrupts the graph structure, limiting the node-to-node message passing.
- **Instability in spectral methods**: Sparse or incomplete graph negatively affects the convergence and performance of spectral based approaches (citar).

These limitations prevent Spatio-Temporal Graph Neural Networks from reliable capturing long-range spatio-temporal dependencies, which are crucial in real world forecasting applications such as traffic flow prediction, energy demand forecasting or environmental monitoring.

1.3 Research Questions

- How can deep learning architectures, such as recurrent neural networks and transformers, be integrated into ST-GNNs to improve forecasting performance under conditions with missing data?
- How can training strategies -such as spectral based regularization or data imputation- enhance the robustness and generalization of ST-GNNs in spatio temporal forecasting tasks that involves missing data?

1.4 Hypothesis

Integrating deep learning architectures within ST-GNNs alongside training strategies (including spectral-based regularization or data imputation) can improve the accuracy, robustness and generalization of ST-GNNs in forecasting tasks when spatio-temporal data that contains missing values.

1.5 Objectives

1.5.1 Main Objective

Develop a learning algorithm for ST-GNNs that enhance robustness and accuracy in time series forecasting under conditions with missing values while leveraging inherent spatial dependencies.

1.5.2 Specific Objectives

- Design training strategies, such as spectral-based regularization or transfer learning, to improve the stability and generalization capabilities of ST-GNNs in forecasting tasks under missing value conditions, achieving performance that matches or exceeds baseline models across standard metrics (RMSE, MAE).
- Explore data imputation techniques as a complementary strategy to enhance the accuracy and robustness of ST-GNNs, particularly when combined with training methods for missing data, and evaluate their impact using standard imputation metrics (MAE, Normalized RMSE).

1.6 Scope and Limitations

1.6.1 Scope

This thesis focuses on developing a unified framework for ST-GNNs under missing data conditions. This work aims to:

- Design and implement a cohesive model that simultaneously handles missing data and forecasting tasks within the ST-GNN architecture.
- Explore training strategies, including spectral-based regularization, to enhance model robustness and generalization under sparse data conditions.
- Evaluate this unified framework across standard real-world spatio-temporal datasets, such as METR-LA, PEMS-BAY, AQI or PM2.5, using standard metrics (RMSE, MAE, NRMSE) to measure the perfomance of the proposed model.

- Provide reproducible benchmarks comparisons for state-of-the-art forecasting and imputation models, offering empirical insights about the impact of imputation in forecasting tasks.

The development of a unified ST-GNNs framework represents a significant contribution to the field by closing the gap between imputation and forecasting, which usually are addressed separately.

1.6.2 Limitations

The research has the following constraints:

- The implementation of ST-GNNs with integrated imputation and complex training strategies can require significant computational resources, which can affect scalability to large graphs.
- The evaluation will be performed on well-known datasets, with the possible case of use of a real-world dataset and synthetic dataset.
- In the proposed framework, the graph structure is assumed to be given as part of the spatio-temporal dataset. Therefore, learning the graph structure will not be addressed within the scope of this project.

1.7 Expected Contributions

1. **Development of a Unified Framework**
   Creation of an integrated ST-GNN based model that simultaneously performs time series forecasting and data imputation under spatio-temporal datasets with missing values, reducing the need for separate pre-processing steps and improving robustness in real-world scenarios.
2. **Empirical Analysis of Imputation Impact on Forecasting Tasks**
   Systematic evaluation of how different imputation strategies affect forecasting accuracy, offering new observations and insights into the design of learning algorithms for incomplete datasets.
3. **Benchmarking and Reproducible Research Tools**
   Establishment of a reproducible benchmarking environment,including open-source code and standardized evaluation protocols, to ensure reproducibility and facilitate fair comparisons with other GNN architectures.

2 Background

The fundamental concepts behind forecasting with spatio temporal graph neural networks is grounded mainly on concepts focused on statistics, graph theory and neural networks. This section contains the theoretical background related to the present project.

2.1 Time Series

Often, a time series is defined as a sequence of time-ordered observations which describe how a system or a phenomenon evolves over time[3]. Formally, a time series can be defined in terms of a stochastic process [2]. A stochastic process is a family of random variables $\{X_t | t \in T\}$ defined in a probability space $(\Omega, \mathcal{F}, P)$, where T is the index set. A realization of a stochastic process is a function $t \in T \longrightarrow X_t(w)$ for a fixed $w \in \Omega$. Thus, a time series $\{x_t\}_{t \in T}$ is a realization of a given stochastic process $\{X_t | t \in T\}$ where

$\exists \omega \in \Omega$ such that $\forall t \in T$, $x_t = X_t(\omega)$. It is important to mention that if $X_t(\omega) \in \mathbb{R}$, then the time series is univariate, and if $X_t(\omega) \in \mathbb{R}^n$ then the time series is multivariate. When the time series is defined in a discrete domain, then the index set is $T = \mathbb{Z}$; for real time processes, the index set is $T = \mathbb{R}$ or $T = [0, \infty)$. In practice, the time series observed is a finite segment $\{x_t\}_{t \in T_0}$ where $T_0 \subset T$.

2.1.1 Time Series Forecasting

Time series forecasting is centered around predicting future values of the time series given past historical observations [17]. Let $\{x_t\}$ be a given time series. The forecasting process consists on estimating the value of x_{t+1}, written as $\hat{x}_{t+1}$ with the goal of minimizing an error function, usually in terms of $x_{t+1} - \hat{x}_{t+1}$. This task can be divided in two types [3]: *single-step-ahead forecasting*, which consist in predicting single future observations of the time series once at a time, i.e, the target is $y = x_{t+H}$ at time t for some $H \in \mathbb{N}$, which is called prediction horizon of the time series, and *multi-step-ahead forecasting*, with target $y = x_{t+1:t+H}$ where the prediction is an interval of multiple time steps. The time series forecasting models[**?**] usually take the form

$$\hat{y}_{i,t+1} = f(y_{i,t-k}, x_{i,t-k}) + \epsilon_{i,t+1}, \tag{1}$$

where f is a prediction function which is learned from the model and $\epsilon_{i,t+1}$ is a noise term. The work of Masini et al. [14] also mentions that the function f can be defined as the conditional expectation function $f(y_{i,t-k}, x_{i,t-k}) = \mathbb{E}(y_{i,t+h}|x_{i,t})$ for a given horizon $h = 1, 2, ..., H$, i.e., the expected value of $y_{i,t+h}$ given the previous observations $x_{i,t}$.

2.1.2 Time Series Imputation

Conversely, the aim of time series imputation is to estimate and fill missing or incomplete data points within a time series. Let $(\Omega, \mathcal{F}, P)$ be a probability space and $\{X_t | t \in \mathbb{Z}\}$, $X_t = (X_{1,t}, X_{2,t}, ..., X_{n,t})$ an n-variate stochastic process. A missing value mask is a binary vector defined as $M_t = (m_{1,t}, m_{2,t}, ..., m_{n,t})$ where $m_{i,t} = 0$ iff the value $X_{i,t}$ from the time series is missing or $m_{i,t} = 0$ iff $X_{i,t}$ is an observed value from the time series [2]. Given observations from a time series $X_{i,t:t+T}, T \in \mathbb{Z}$, with missing values indicated with the mask $M_{i,t:t+T}$, it can be defined $\tilde{X}_{i,t:t+T}$ as the sequence with no missing values. The goal of time series imputation[**?**] is to find an estimate $\hat{X}_{i,t:t+T}$ which minimizes the following error function:

$$\mathcal{L}(\hat{X}_{i,t:t+T}, \tilde{X}_{i,t:t+T}, M_{i,t:t+T}) = \sum_{\tau=t}^{t+T} \frac{\sum_{i=1}^{n} \bar{m}_{i,\tau} \cdot \ell(\hat{x}_{i,\tau}, \tilde{x}_{i,\tau})}{\sum_{i=1}^{n} \bar{m}_{i,\tau}}, \tag{2}$$

where $\ell(\cdot, \cdot)$ is an element-wise error function and $\bar{m}_{i,\tau}$ is the logical complement of $m_{i,\tau}$.

In this domain, Jin et al.[3] consider two approaches : *in-sample imputation*, which consists in filling missing values on known data, and *out-of-sample imputation*, which refers to the estimation of missing values in unknown data. In the work by Marisca et al. [**?**], they propose two different policies to inject missing data: based on associating to each observation a probability p of being removed:

- **Point missing:** In this setting, each observation from the time series is associated to a probability p of being removed.

- **Block missing:** In this case, the probability p is associated to each time step t, and it represents the probability of injecting missing values for a random number $S \sim \mathcal{U}(a, b), a < b$, of consecutive time steps.

2.1.3 Deep Learning Models for Time Series Analysis

Here we present some examples of statistical and deep learning models commonly used for time series analysis.

- **Auto Regressive Integrated Moving Average (ARIMA):** ARIMA is a model that combines an auto regressive (AR) component with a moving average (MA) for past information, with an integrated differentiation to make the time series stationary [2]. Given the backwards operator $B^k y_t = y_{t-k}$ for a lag k, the ARIMA model with parameters p, q, and d is defined as:

$$\underset{\text{AR}(p)}{(1 - \phi_1 B - \cdots - \phi_p B^p)} \underset{d \text{ differences}}{(1 - B)^d y_t} = \mu + \underset{\text{MA}(q)}{(1 + \theta_1 B + \cdots + \theta_q B^q)} \varepsilon_t, \tag{3}$$

  where ϵ_t is an error term, p is the order of the auto regressive component, q is the order of the moving average component and d is the degree of differentiation.

- **Long Short-Term Memory (LSTM):** Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) which uses three gating mechanisms: an input gate, a hidden gate and an output gate [4]. The input of x_t and the output h_{t-1} from the previous step are combined into the current input vector i_t as follows:

$$i_t = \gamma(W_i x_t + U_i h_{t-1} + b_i) \tag{4}$$

  where $\gamma(\cdot)$ is any sigmoidal function and W_i, U_i, b_i are the parameters. Then, the three gates are computed as follows:

$$g_t = \sigma(W_g x_t + U_g h_{t-1} + b_g), \tag{5}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{6}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{7}$$

  where $\sigma(\cdot)$ is the sigmoid function, g_t, f_t, o_t are the input, forget and output gates respectively.

- **Gated Recurrent Units (GRU):** A GRU uses two mechanisms, an update gate and a reset gate [4]. Both gates depend on x_t and h_{t-1}, which are computed as:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), u_t = \sigma(W_u x_t + U_u h_{t-1} + b_u), \tag{8}$$

  where $\sigma(\cdot)$ is the sigmoid function and the parameters have the same meaning as the LSTM. An intermediate output h'_t is given by the following expression:

$$h'_t = \tanh(W x_t + U(r_t \odot h_{t-1}) + b), \tag{9}$$

  where $\odot$ is the element-wise multiplication. Finally, the output of the GRU layer is given by:

$$h_t = u_t \odot h_{t-1} + (e - u_t) \odot h'_t, \tag{10}$$

  where e is the column vector whose columns are equal to 1.

- **Transformers:** The transformer architecture is based on finding the relationships between various input segments using the dot product [27]. Let $\{x_i\}_{i=1}^n \in \mathbb{R}^d$ be a set of data points. The main component of the transformer architecture is the self-attention operation, which first consists in calculated the normalized correlations between input segments x_i and all others $j = 1, 2, ..., n$:

$$w_{ij} = \text{softmax}(x_i^T x_j) = \frac{e^{x_i^T x_j}}{\sum_k e^{x_i^T x_k}}, \tag{11}$$

where $\sum_{j=1}^{n} w_{ij} = 1$ and $1 \leq i, j \leq n$. Next, for a given input x_i, a weighted sum of all input segments $\{x_i\}_{j=1}^n$ is applied to define z_i as:

$$z_i = \sum_{j=1}^{n} w_{ij} x_j, \forall 1 \leq i \leq n. \tag{12}$$

Furthermore, the self-attention operator in transformers is utilized to build three different matrices Q, K and V, called query, key and value matrices respectively, defined as $Q = W_q x_i, K = W_k x_i$ and $V = W_v x_i$, where $W_q, W_k$and W_v are learnable weight matrices. The output matrix is expressed then as:

$$Z = \textbf{softmax}(\frac{QK^T}{\sqrt{d_k}})V. \tag{13}$$

The transformer architecture is generally composed of two components referred to as encoders and decoders. These two components process the information using the self-attention operator in layers, so every input segment is processed against every other vector in the dataset to get an output from the model.

2.2 Graph Neural Networks

A Graph Neural Network (GNN) is a deep learning model designed to process graph structured data. The first model was proposed by Scarselli et al. [25]. Let $G = (N, E)$ be a graph where N is the set of nodes and E is the set of edges. The set $N(i) = \{j \in N | (n, m) \in E\}$ is called the neighborhood of node i and the set $co(i) = \{(i, j) \in E | j \in N\}$ denotes the set of arcs having node i as the vertex.

In their fundamental work, each node of a graph is represented as a state $x_i \in \mathbb{R}^s$, which encapsulates all the information of the neighborhood of node i, including the labels of each adjacent node and arc. Then, with a function f_w, usually called transition or aggregation function, the dependence of a node i with its neighborhood can be expressed and, in the same manner, a function g_w usually called output function describes how the information regarding node i will be produced.

If the function f_w is a contraction function with respect to the state, i.e., $\exists \mu \in [0, 1)$ such that $||f_w(x) - f_w(y)|| \leq \mu ||x - y||, \forall x, y$ states, then according to Banach's Fixed Point Theorem [18], if the function f_w is applied iteratively, it will converge to a unique fixed state x'. Then, the GNN model of Scarselli et al.[25] can formally be described by the following system:

$$x_i(t+1) = f_w(l_i, l_{co(i)}, x_{N(i)}(t), l_{N(i)}) \tag{14}$$

$$o_i(t) = g_w(x_i(t), l_i) \tag{15}$$

where l denotes the labels of the nodes and arcs and o_i is the output about the concept that represents node i. This iterative method allows message passing between nodes so they share similarities in their embeddings [19] until a fixed point is reached.

The learning algorithm in GNNs consists on estimating the parameter w given a finite set of graphs $\mathcal{G} = \{G = (N, E) | N \text{ is the set of nodes, } E \text{ is the set of edges}\}$ from the transition and output functions such that a function φ_w approximates the data in the learning data set defined as:

$$\mathcal{L} = \{(G_i, n_{i,j}, t_{i,j}) | G_i = (N_i, E_j) \in \mathcal{G}; n_{i,j} \in N_i, t_{i,j} \in \mathbb{R}^m\}, \tag{16}$$

where $i \in [1, |\mathcal{G}|], q \in [1, |N_i|]$ and $t_{i,j}$ is the desired target associated to the j-th node $n_{i,j} \in N_i$. The learning task can be posed as the minimization of the cost function:

$$e_w = \sum_{m=1}^{|\mathcal{G}|} \sum_{k=1}^{|N_m|} (t_{m,k} - \varphi_w(G_m, n_{m,k}))^2 \tag{17}$$

Furthermore, the learning algorithm is composed by the following steps:

1. The states x_i are iteratively updated by equations (11) and (12) until at time T they approach a fixed state. This convergence is guaranteed based on Banach's fixed point theorem and the definition of the transition functions as contraction maps.
2. The gradient $\frac{\partial e_w(T)}{\partial w}$ is computed based on the backpropagation-through-time algorithm.
3. The weights w are updated according to the gradient computed on the previous step based on the traditional framework of gradient descent.

This is the general description of how a graph neural network is trained. However, the training of spatio-temporal graph neural networks is based on similar principles, and will be explained in Section 2.2.3.

2.2.1 Taxonomy of GNNs

According to Wu et al. [20], GNNs can be classified in the following types:

1. *Recurrent GNNs*: These models are the first that were defined as GNNs [25], and they aim to learn representations with recurrent neural architectures. In these models the node in a graph constantly exchanges information with its neighbors until a stable state is reached.
2. *Convolutional GNNs*: These models generalize the concept of convolution from grid data to graph data. They can be spectral-based, which have a mathematical foundation in graph signal processing using the Laplacian matrix of the graph, or they can be spatial based, where the graph convolutions are based on a node's spatial relations. Unlike recurrent GNNs, convolutional GNNs stack layers of graph convolutions to extract high level node representations.
3. *Graph Autoencoders*: These are unsupervised learning methods that encode node/graph attributes into a latent vector space to reconstruct graph data from the encoded information.
4. *Spatial-Temporal GNNs*: These aim to learn hidden patterns from spatial-temporal graphs, which can be found in domains like traffic speed forecasting, environmental monitoring and human action recognition. The key idea is to consider both spatial and temporal dependencies at the same time, by utilizing spatial and temporal modules based on graph convolution and deep learning architectures like recurrent neural networks respectively.

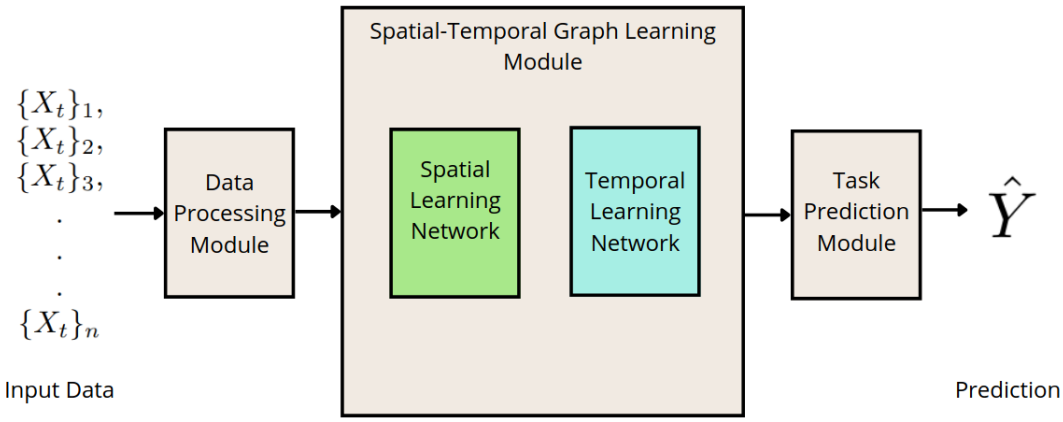


Figure 1: Basic architecture of a Spatio-Temporal Graph Neural Network (ST-GNN) for time series forecasting. The model integrates spatial dependencies through graph convolutional layers and temporal dynamics through recurrent or convolutional modules.

2.2.2 Spatial-Temporal Graph Neural Networks

Spatial-Temporal Graph Neural Networks (ST-GNNs) were originally proposed in the work of Yu et al. [13], they incorporate spatial information by considering the relationship between nodes in the graph and temporal information taking into consideration the evolution of node attributes over time. These models are ideal for time series forecasting and imputation when there is a graph associated to the time series that represent the spatial relations between the nodes.

A typical ST-GNN architecture has three components and can be observed in Figure 1:

1. Data processing module: Constructs the spatio-temporal graph from raw inputs, including defining the graph structure and aligning temporal sequences.

2. Spatio-temporal graph learning module: Learns joint spatial and temporal dependencies from the data. This is typically achieved by combining spatial modeling through GNN variants (e.g., Spectral Graph Convolutional Networks (Spectral GCN), Spatial GNNs, or Graph Attention Networks) with temporal modeling through Recurrent Neural Networks (RNNs) or Temporal Convolutional Networks (TCNs).

3. Task-specific prediction module: Maps the learned spatio-temporal representations to outputs suitable for the downstream task, such as forecasting future values.

The most common model utilized for processing the spatial dependencies is the Spectral GCN [**?**], based on the graph convolution, which takes the graph Fourier transform and the Inverse graph Fourier transform to achieve the transformation from the spatial domain to the spectral domain. Both transforms are defined as follows:

$$\mathcal{F}(x) = U^T x, \tag{18}$$

$$\mathcal{F}^{-1}(x) = Ux, \tag{19}$$

where U is the eigenvector matrix of the normalized Laplacian of the graph. Based on this idea, the graph convolution is defined with the following equation:

$$g \star x = \mathcal{F}^{-1}(\mathcal{F}(g) \odot \mathcal{F}(x)) = U(U^T g \odot U^T x), \tag{20}$$

where $\odot$ is the element-wise multiplication and $U^T g$ represents the filter on the spectral domain.

2.2.3 The Graph Model

The construction of the graph for the Spatial-Temporal GNN can be defined according to several criteria. Jin et al. [12] define the **Spatial-temporal Graph** which characterizes the relationships between the nodes and a certain spatial and temporal range. Given a time series $X_t \in \mathbb{R}^{N \times F}$ where N is the number of nodes and F is the number of features, a spatial-temporal graph is represented as a graph $G_t = (V, E_t, A_t)$ where V is the set of nodes, E_t is the edge set and A is the adjacency matrix $A_t = \{a_{ij}\}$. Both E_t and A_t are assumed to change over time. Then, some of the construction methods to define the graph G_t are the following:

1. **Topology based graphs:** These kind of graphs are constructed based on given topology structures inherent to the problem at hand, like road networks [12]. The adjacency matrix is defined as:

$$a_{ij}^t = \begin{cases} 1, & \text{if } v_i \text{ connects with } v_j \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

2. **Distance based graph:** If there exist a spatial relationship according to a distance between the nodes of the graph, a Gaussian radial basis function can be used to define the distance-based adjacency matrix:

$$a_{ij}^t = \begin{cases} \frac{\exp(-||d_{ij}^t||_2)}{\sigma}, & \text{if } d_{ij}^t < \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

   where d_{ij}^t denotes the distance between node i and node j at time t, ϵ is a defined threshold and σ is a hyperparameter to control the distribution.

3. **Similarity based graph:** These graphs typically are constructed based on he similarity of time series. The Pearson correlation coefficient can be used to calculate the similarity between time series, and to define the adjacency matrix as $a_{ij} = \rho_{i,j}$, where $\rho_{i,j}$ is the Pearson correlation coefficient between time series associated to node i and the time series associated to node j.

2.2.4 Learning Algorithm for Spatial-Temporal GNNs

The learning algorithm that will be approached in this project is based on the works of Cini and Marisca [**?**]. The model used by Cini and Marisca relies on a RNN temporal module and a spatial module which utilizes a diffusion convolution model to propagate the information of each node through the underlying graph. It also incorporates an encoder/decoder scheme to represent the spatio-temporal data in a latent space which includes the node embeddings.

To start with the description of the algorithm, let $G = (V, E)$ be a static graph with $|V| = N$ nodes and its weighted adjacency matrix $A \in \mathbb{R}^{N \times N}$. The diffusion matrix row stochastic can be defined as $P = D^{-1}A)$ where $D = diag(A\hat{1})$ and $\hat{1}$ is the vector with all components equal to 1. Let $X_t \in \mathbb{R}^{N \times F}$ be a multivariate time series with $F =$ number of attributes per node. The sampling of this time series can be done with windowing using H and W as the horizon and window respectively. Then, we get the sampling of the time series:

$$X_t = (X_{t-W+1}, X_{t-W+2}, ..., X_t) \in \mathbb{R}^{W \times N \times F} \tag{23}$$

$$Y_t = (X_{t+1}, X_{t+2}, ..., X_{T-H}) \in \mathbb{R}^{H \times N \times F} \tag{24}$$

The training set then is the set of indices $S_{train} = \{t = W, W+1, ..., T-H\}$. The valid mask matrix $M_t \in \{0,1\}^{H \times N \times F}$ represents the valid observations of the target values ($1 =$ valid observation).

The forward computation is as follows:

1. **Linear encoding and node embeddings:** The input data is transformed using a linear encoding to represent the complete information (time series and graph properties) in the same latent space. Formally, $\forall s \in t-W+1, t-W+2, ..., t$ and each node n:

$$Z_{s,n} = W_{enc} X_{s,n} + b_{enc} + E_n, \tag{25}$$

   where $E_n \in \mathbb{R}^d$ is the vector of node embeddings, $W_{enc} \in \mathbb{R}^{F \times d}$ is a learnable weight matrix, b_{enc} is a bias term and $Z_{s,n} \in \mathbb{R}^{N \times d}$.

2. **Temporal module:** The temporal processing is made by evaluating a RNN model (GRU) to the time series of each node:

$$H_n = RNN(Z_{t-W+1,n}, Z_{t-W+2,n}, ..., Z_{t,n}; \theta_{RNN}) \in \mathbb{R}^d. \tag{26}$$

   Then, stacking all the H_n we get $H = [H_1; H_2; ...; H_N] \in \mathbb{R}^{N \times d}$.

3. **Spatial module**: With a kernel of k neighbors on the graph, the diffusion convolution combines "forward" message passing [**?**] and "backwards" message passing with the matrices P and P^T respectively, then it is defined with the following equation:

$$\Phi(H) = \sum_{k=0}^{K} (P^k H \theta_k^{(f)} + (P^T)^k H \theta_k^{(b)}), \tag{27}$$

   with $\theta_k^{(f)}, \theta_k^{(b)}$ learnable parameters. Let $U = \Phi(H) \in \mathbb{R}^{N \times s}$.

4. **Linear decoder:** The information of each node is then projected to each horizon step to reshape it:

$$O_n = U_n W_{dec} + b_{dec} \in \mathbb{R}^{H \cdot F}, \tag{28}$$

$$\hat{Y_t} = reshape(O, H, N, F), \tag{29}$$

   where $W_{dec} \in \mathbb{R}^{d \times (H \cdot F)}$ is a learnable weight matrix and $b_{dec} \in \mathbb{R}^{H \cdot F}$ is the bias term.

The loss function is calculated based on the work of Wu et al. [8], which is called MaskedMAE. The loss function is defined as follows over a batch B:

$$\mathcal{L}(\theta) = \frac{\sum_{t \in B} \sum_{\tau=1}^{H} \sum_{n=1}^{N} \sum_{f=1}^{F} M_{t,\tau,n,f} \cdot |\hat{y}_{t,\tau,n,f} - y_{t,\tau,n,f}|}{\sum_{t \in B} \sum_{\tau,n,f} M_{t,\tau,n,f}}, \tag{30}$$

where θ includes all the learnable parameters $W_{enc}, b_{enc}, \theta_{RNN}, \{\theta_k^{(\cdot)}\}, W_{dec}$ and b_{dec}.

The learning algorithm starts follows a gradient descent approach. For this, the expected loss over the training set is minimized:

$$\min_{\theta} \mathbb{E}_{t \sim S_{train}}[\mathcal{L}(\theta)] \tag{31}$$

Then the algorithm proceeds as follows:

1. Sampling of the sequential batch of windows $\{X_t, Y_t, M_t\}_{t \in B}$.
2. The forward computation is performed to get $\hat{y}_t = FORWARD(X_t, P)$.
3. The loss function $\mathcal{L}(\theta)$ is computed.
4. Computation of the gradient $\nabla_\theta \mathcal{L}$ using backpropagation-through-time in the temporal dimension W through the RNN. Since the graph P is constant, the gradient only affects the parameters θ.
5. The update of the parameters is done with Adam optimizer: $\theta \longleftarrow Adam(\theta, \nabla\mathcal{L}, \gamma)$, where γ is the learning rate.
6. At the end of each epoch, the model goes through the validation step using metrics like MAE or RMSE until the maximum number of epochs is reached or following an early stopping strategy. A checkpoint which saves the information about the parameters of the model is created.
7. Finally, the model is tested using the best checkpoint (the model with the best metric) to assess its performance.

A block diagram of the learning algorithm is presented in Fig. 2. Note that the green blocks is where the contribution of this work is most likely to be implemented, since the temporal and spatial modules are flexible and are compatible with other kind of deep learning models and graph convolutions (through spectral based regularization). Also, the loss function can also be modified and adapted in terms of variables that may be aggregated to the model.

2.2.5 Spectral-Based Regularization

In machine learning, "regularization" refers to the penalization of the error term with the objective of improving the estimation of model parameters [**?**]. Given the loss function $\mathcal{L}(\theta) = l(f_\theta(x), y)$ of a learning model, the regularization can be represented formally as

$$R(\theta) = \mathcal{L}(\theta) - \lambda\varphi(\theta), \tag{32}$$

where $\lambda \in [0, \infty)$ is a hyperparameter that controls the strength of the regularization.

Regularization based on spectral methods encourages the smoothness of node embeddings, meaning hat connected nodes have similar feature representations. This concept is grounded on graph spectral theory , given that the adjacency matrix A encodes the graph structure and the Laplacian matrix L allows the processing of the graph in a frequency domain.

The Laplacian matrix of a graph $G = (N, E)$ can be defined as follows:

$$L(u, v) = \begin{cases} d_u, & u = v \\ -1, & (u, v) \in E \\ 0, & \text{otherwise} \end{cases} \tag{33}$$

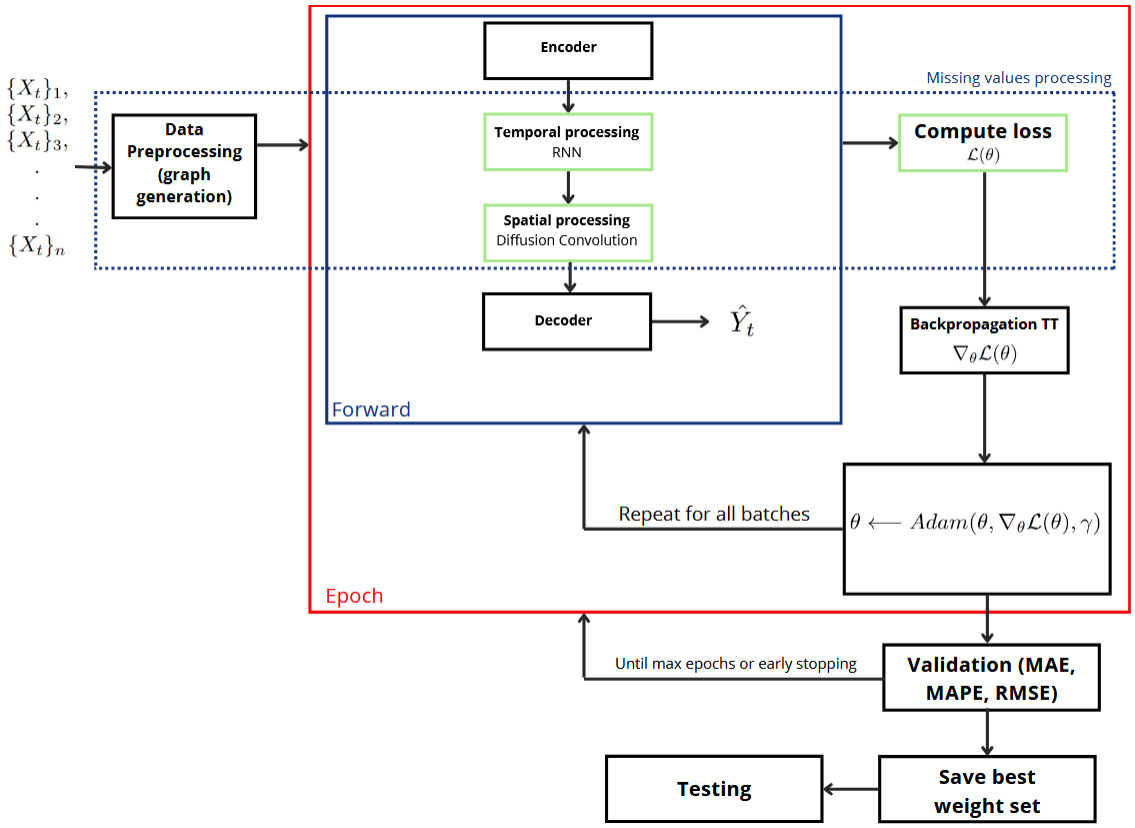


Figure 2: Overview of the training loop for the proposed ST-GNN model. Each sample uses a window of W past steps, encoded linearly with the node embeddings, and then processed temporally by a RNN and spatially by a diffusion convolution; a linear decoder outputs the H-step forecast. After the computation of the gradient $\nabla L(\theta)$, the optimization is made with Adam; the validation is performed at each epoch to save the best checkpoint, to finally report the final metrics obtained from the test set.

where d_u is the degree of node $u \in N$. It can also be written as $\mathcal{L} = D - A$, where $D = \sum_j A_{ij}$.

In GNNs, this kind of regularization can be applied in two sections of its architecture:

1. In the message passing process between the nodes.

$$H^{k+1} = \sigma(AH^kW^k), \tag{34}$$

   where H is the matrix that represents the attributes of the nodes, W is a weight matrix and σ is an activation function.

2. In the loss function.

$$\mathcal{L}_{total} = \mathcal{L}_{task} + \lambda f^T L f, \tag{35}$$

   where $\lambda \in (0, 1]$ is a regularization hyperparameter.

2.3 Temporal and Spatio-Temporal Dependencies

In the scope of the present work, it is imporant to make the distinction how a single observation in the network changes over time and how information is shared across the network structure [26]. These are represented by temporal dependencies and spatio-temporal dependencies.

In the first case, a temporal dependency is given naturally by the definition of time series, since it refers to the correlation that exists between a specific state of a node of the graph (i.e., an observation of a variable in the time series) at current time t and its own previous states $t-1, t-2, ...$ It is assumed that the future value of a node is conditioned by its own history.

On the other hand, the spatio-temporal dependency occurs when there is an interaction when the state of a node i in time step t is influenced by the state of the neighboring nodes j in past time steps. The similarity between neighboring nodes already represents the spatial dependency, thus, the spatio-temporal dependency is represented by the signal that propagates through the graph via message-passing from node to node through time.

3 Related Work

In recent times, although limited [**?**], there has been significant progress in time series forecasting and missing data imputation, through the application of deep learning models [3].

Despite this progress, most of the existing works have approached these two challenges separately. This separation limits robustness of current models and applicability to real world scenarios where imputation and forecasting are related tasks.

This section reviews works for forecasting and imputation, which involves the utilization of a GNN model as their main architecture.

3.1 Works in Time Series Forecasting

The fundamental work of Wang et al. (2020) [21] focuses on the temporal classification/regression problem which consists on building a function that maps a multivariate time series to a discrete class label or real value response. They propose a model called sparse functional multilayer perceptron (SFMLP) for handling

missing values in time series covariates. Simeunovic et al. (2022) [22] propose to take a graph signal processing perspective to model multi-site photovoltaic production time series as signals on a graph to capture their spatio-temporal depencencies. There are two approaches in this work: the graph-convolutional long short term memory (GCLSTM) and the graph convolutional transformer (GCTrafo) models.

Moreover, Oskarsson et al. (2023) [23] proposed a temporal graph neural network model for forecasting of graph structured irregularly observed time series. This approach introduces a time continuous latent state in each node, following a linear ordinary differential equation defined by the output of a Gated Recurrent Unit (GRU). Although research on the topic of forecasting from partial observations is limited [**?**], the most recent work from Kim et al. (2025) [24] proposes a new method for multivariate time series forecasting with missing values, called temporal matrix factorization-based graph neural network (TMF-GNN). The concept of temporal matrix factorization is used to reconstruct partially observed MTS data.

Table 1: Summary of recent state-of-the-art works in time series forecasting.

Year	Authors	Architecture	Baselines	Datasets	Performance
2020	Wang et al.	Functional MLP	LSTM	PBC, C-MAPSS, synthetic data	RMSE = 11.97
2022	Simeunovic et al.	ST-GNN	STAR, STCNN, SVR, ED-LSTM	Synthetic and real dataset	NRMSE = 3.350 NMAE = 7.23
2023	Oskarsson et al.	GNN	GRU-D, Transformer, LG-ODE	PEMS-BAY, METR-LA, USHCN	MSE = 0.071
2025	Kim et al.	TMF-GNN	TMF, TRMF, NoTMF	Google Flu Trends, AQI, PM2.5	$RMSE_{20\%}$ = 0.029 $RMSE_{30\%}$ = 0.035 $RMSE_{40\%}$ = 0.034

3.2 Works related to Imputation

For the imputation task, Ma et al. (2020)[**?**] proposed an imputation methodology named transferred long short-term memory-based iterative estimation (TLSTM-IE) where the imputation occurs in consecutive missing values with large missing rates. The work of Marisca et al. (2022) [**?**] regarding data imputation of time series rely on attention-based architectures to reconstruct missing data points by conditioning the reconstruction only on the available observations on the time series.

Cini et al. (2022) [**?**] proposes the first imputation method using GNNs, with an architecture named GRIN, which reconstructs missing data in the different channels of a multivariate time series by learning spatio-temporal representations through message passing.

Also, the work of Chen et al. (2023) [9] proposes an adaptive graph recurrent network (AGRN) which learns variable and time specific dependencies without domain knowledge, combining a graph convolution module with a spatio-temporal fusion module based on a Gated Recurrent Unit (GRU).

3.3 Discussion

A through review of recent works related to our proposed research reveals that much of the current work in time series forecasting have the assumption of complete datasets. This significantly limits the applicability to real world scenarios where data sparse is common. Furthermore, many baseline models are not based on GNNs, which make the direct and balanced comparisons with other GNN models infrequent.

Table 2: Summary of recent state-of-the-art works in time series imputation.

Year	Authors	Architecture	Baselines	Datasets	Performance
2020	Ma et al.	LSTM, Transfer Learning	ARIMA, SVR, LASSO, RNN	PM2.5	RMSE = 0.0864
2022	Marisca et al.	Attention-based model	GRIN, SAITS, BRITS, rGAIN	PEMS-BAY, METR-LA, AQI	$MAE_{50\%}$ = 0.79 $MAE_{75\%}$ = 0.97 $MAE_{95\%}$ = 1.68
2022	Cini et al.	GNN	BRITS, rGAIN, KNN, MPGRU	PEMS-BAY, METR-LA, AQI, CER-E	MAE_{bm} = 0.41 MAE_{pm} = 0.29
2023	Chen et al.	AGRN	Mean, kNN, BRITS, GRIN	AQI, PEMS-BAY, METR-LA	MAE = 0.67

Regarding imputation tasks, their impact on forecasting performance is not widely analized in depth, despite the inherent connection between imputation precision and forecasting accuracy.

Finally, although the recent work of Kim et al. (2025) [24] is promising, it does not consider spatial dependencies, as it relies on matrix factorization and graph convolutional network-based regularization for handling missing values in time series.

The main distinction between related works and the proposed framework lies in the architectural handling of missing values. Unlike traditional two-stage approaches that treat imputation as an isolated pre-processing task -which may introduce bias before the model can take input- the proposed approach integrates imputation directly into the learning algorithm. By optimizing imputation and forecasting jointly, the model ensures that the filled values will be defined specifically to maximize forecasting accuracy.

4 Research Proposal

This section includes the general steps and guidelines for the completion of this project.

4.1 Methodology

1. **Background and Related Work**
   The project requires a continuous comprehensive review of topics relevant to GNNs and time series forecasting. Initial efforts were focused on studying published works on GNNs and ST-GNNs to develop a solid understanding of the general domain. Subsequently, attention was directed toward the issue of data sparsity in time series and the training strategies that can be implemented in ST-GNNs to address this issue, in order to gain deeper understanding of the specific challenges of this project.

2. **Experimental Design**
   The research design for this project is planned according to the following activities, which may change in the course of the realization of the project:

   - ***Approach.*** The proposed approach is to implement a ST-GNN using the Python library tsl [**?**], which is designed to ease the development of graph neural networks for spatio-temporal data processing. This library is built using PyTorch, PyTorch Geometric and PyTorch Lightning [10], ensuring cohesive framework that spans from data preprocessing to model deployment.

The datasets METR-LA, PEMS-BAY, AQI and USHCN [**?**] will be used for training, testing and validation of the proposed architecture. The properties of each dataset can be found in Table 3.

Dataset	Domain	# Sensors	Missing Data	Typical Tasks
METR-LA	Traffic (speed)	207	Yes (NaNs due to sensor outages)	Forecasting, Imputation
PEMS-BAY	Traffic (speed)	325	Yes (generally sparse/minor gaps)	Forecasting
AQI (Beijing Multi-Site)	Air quality	12	Yes (missing timestamps/values)	Forecasting, Imputation
USHCN v2.5	Climate (monthly)	≈1219	Some gaps/flags; homogenized series	Forecasting, Imputation

Table 3: Summary of METR-LA, PEMS-BAY, AQI, and USHCN v2.5 datasets including missing data and typical tasks.

- ***Evaluation metrics.*** For this project, the following metrics were initially selected to measure the performance of the proposed model, which are also widely used in the existing research.
  - Mean Absolute Error (MAE): The average of the absolute differences between predicted and actual values.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

  - Root Mean Squared Error (RMSE): The square root of the average of squared differences between predicted and actual values.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

  - Mean Absolute Percentage Error (MAPE): It expresses the accuracy as a ratio defined by the following expression:

$$MAPE = 100\frac{1}{n}\sum_{t=1}^{n}|\frac{y_i - \hat{y}_i}{y_i}|$$

- ***Baselines*** Given the present project, we would choose between three kinds of baselines:
  (a) Traditional Time Series models: ARIMA (AutoRegressive Integrated Moving Average) [2] or LSTM (Long Short-Term Memory) [4] for forecasting tasks.
  (b) Deep learning baselines for imputation or forecasting: RNN based models such as GRU (Gated Recurrent Unit)[4] .
  (c) Existing GNN models for spatio-temporal data: STGCN [13], Graph WaveNet [8] or TMF-GNN [24].

  The choice of these baselines will depend on the open-source implementation availability and compatibility with the chosen datasets.

3. **Model Development and Training Strategies** In this part of the research, we will divide the main activities in the following:

(a) ***Model Architecture Development.*** The main goal is to make the architecture modular, allowing flexible integration of imputation tasks and spectral based regularization techniques. This will allow the proposed model to tackle forecasting tasks in environments with missing data by leveraging the proposed ideas regarding training strategies, variants of loss function and even spectral based regularization methods. The flexibility of the spatial and temporal module to choose different models to approach the spatial and temporal processing allow the model to adapt imputation mechanisms for forecasting architectures [**?**].

(b) ***Missing Data.*** Missing values will be treated in two different ways:

- Allow the model to learn directly with sparse data.
- Define a pre-processing imputation to study the effects on ST-GNNs with sparse data.

(c) ***Training Strategies.*** For the training strategies, the following are the techniques that initially will be tested on the proposed model:

- Spectral regularization to apply smoothness over the nodes of the graph or the loss function using Laplacian based constraints.
- Simulation of missing values at different sparsity levels ($10\%, 20\%, 30\%$ and 50%) during training to improve model robustness.
- Combine forecasting loss with regularization loss to define a joint objective loss function.

4. ***Assessment of the Model***

The experimental evaluation will be conducted using well known spatio-temporal datasets (METR-LA, PEMS-BAY, AQI), which are representative benchmarks for different domains, like traffic forecasting and environmental monitoring. To simulate real world levels of missing values, various levels of missingness will be introduced in these datasets. The experimental setup will assess the capabilities of the model regarding forecasting with incomplete data and with preprocessing imputation strategies.

Model performance will be evaluated using standard forecasting metrics which were mentioned on the Research Design, both for forecasting and imputation task. Furthermore, these metrics then will be used for rigorous statistical testing to validate the signficance of performance differences.

Ablation studies will be performed to isolate the contributions of the key ideas like spectral-based regularization, transfer learning or imputation methods to verify the influence of the proposed strategies and the impact on overall model performance.

4.2 Work Plan

The following diagram serves as the activity guide for the execution of this project.

5 Preliminary Results

The goal of the preliminary experiments was to become familiar with the ST-GNN model in forecasting tasks by designing synthetic datasets to evaluate and compare its performance against traditional baseline models. Moreover, the results of this work have been accepted for presentation as a poster at the Mexican International Conference on Artificial Intelligence 2025 [7].

Activity	2024	2025			2026			2027			2028	
	3	1	2	3	1	2	3	1	2	3	1	2
Review of literature	■	■	■	■	■	■	■	■	■	■	■	
Implementing GNNs with custom dataset	■	■										
Experimental trials with synthetic data		■	■									
Writing research article			■	■				■	■	■		
Experimentation on ST-GNNs with missing values				■	■	■						
Development and implementation of training strategies							■	■	■			
Performance assessment									■	■		
Thesis preparation and writing		■	■	■	■	■	■	■	■	■	■	
Submission of the thesis manuscript											■	
Thesis defense presentation											■	■

Figure 3: Proposed timeline of research activities from 2024 to 2028.

5.1 Datasets

The datasets were designed following a sinusoidal function with additive Gaussian noise, which can be represented with Equation 33. Each time series simulates the behavior of a node in a graph over time, with individual patterns designed to introduce varying levels of correlation and complexity.

$$x_i(t) = \sin(\frac{10t}{T} + i) + \epsilon_i(t), \text{for } i \in \{1, 2, 3, 4\}, \tag{36}$$

where $T = 1000$ are the time steps, $x_i(t)$ is the value of the time series associated to node i at time step t and $\epsilon_i(t) \sim \mathcal{N}(0, \sigma^2)$.

An additional time series, $x_0(t)$, was constructed using three different schemas to reflect varying degrees of inter-node dependency:

1. **Sum-based dependency:** Time series x_0 is the sum of the rest of the time series.

$$x_0(t) = \sum_{i=1}^{4} x_i(t). \tag{37}$$

2. **Mean-based dependency:** Time series x_0 is the mean of the rest of the time series.

$$x_0(t) = \frac{1}{4} \sum_{i=1}^{4} x_i(t). \tag{38}$$

3. **Product-based dependency:** Time series x_0 is the multiplication of the rest of the time series.

$$x_0(t) = \prod_{i=1}^{4} x_i(t). \tag{39}$$

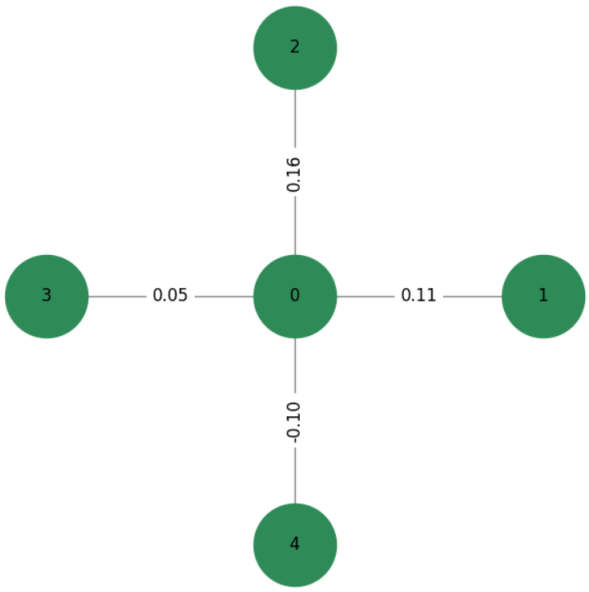


Figure 4: The graph associated to the synthetic data [7]. Each node represents a time series, edges represent statistically significant Pearson correlations between node pairs. Edge labels indicate the correlation coefficients, reflecting the strength of pairwise dependencies.

This schema introduces an explicit dependency structure, considering linear and non linear dependencies, allowing us to test the model's ability to capture node relationships and exploit spatial correlations in the time series.

The graph structure was defined based on the Pearson correlation coefficient between each pair of time series; the weight of each edge represents statistically significant correlations (see Fig. 4). This synthetic setup provides complete control over spatial and temporal attributes, which enables detailed analysis of the robustness and generalization of the model under different scenarios.

5.2 Spatio-Temporal GNN Model

The spatio-temporal forecasting model used for these experiments is based on the architecture proposed by Li et al. [**?**], which integrates temporal encoding via Gated Recurrent Units (GRUs) and spatial modeling using Spectral Graph Convolutional Networks (Spectral GCNs) informed by diffusion processes.

The model processes inputs sequences in two stages:

1. **Temporal encoding:** For each node, a GRU processes its historical time series to extract latent temporal features.

2. **Graph-based spatial encoding:** The node embeddings are passed through a spectral graph convolution layer, which performs message passing across the graph based on the connectivity defined by the similarity matrix.

This approach enables the model to jointly learn temporal dynamics and spatial correlations encoded in the graph structure. Training is performed end-to-end, using a mean squared error loss between the predicted and ground-truth future values.

5.3 Baseline Models and Metrics

To contextualize the performance of the ST-GNN model, we implemented two widely-used forecasting baselines:

- **ARIMA**: A classical autoregressive integrated moving average model [**?**], fit independently to each series.
- **LSTM**: A deep learning model based on Long Short-Term Memory units, which models temporal dependencies but do not consider spatial dependencies [3].

All models are evaluated on a multi-step-ahead forecasting task, where the objective is to predict future values of node 0 given historical data from all nodes. The dataset is split into training (70%), validation (10%), and test (20%) subsets. Each model is trained for 100 epochs with a forecasting horizon of $H = 12$.

We use two standard regression metrics to evaluate forecasting accuracy: RMSE and MAE; these metrics quantify both average and squared prediction errors across different models and scenarios.

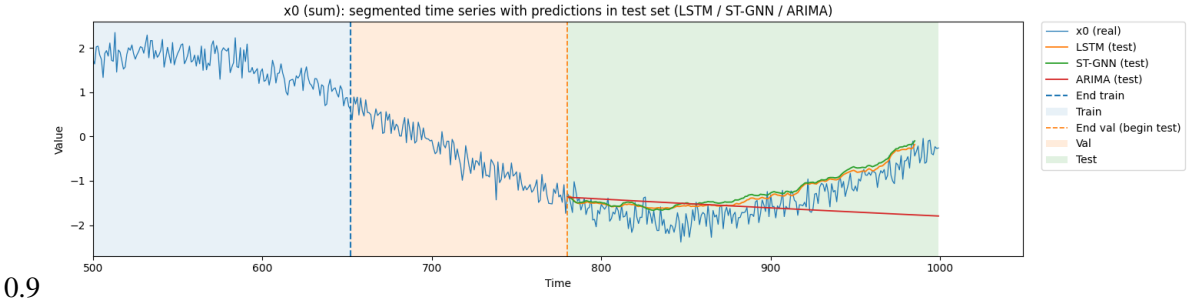
5.4 Experimental Setup

The experiment was repeated for each dependency setting (sum, mean and product setting). The models were trained for 100 epochs each; then, we calculated the metrics RMSE and MAE using the test set.

Table 4 summarizes the key hyper-parameters used for training the LSTM and ST-GNN models. Both models were trained using the same learning rate, batch size, and number of epochs to ensure a fair comparison. While the LSTM uses a larger hidden size (64 vs. 32), the ST-GNN compensates by incorporating node embeddings and a graph convolutional layer with a kernel size of 2, which means that each graph convolution will consider the information of the next two neighbors of a given node. This architectural design allows ST-GNN to model spatial relationships between nodes in addition to temporal dependencies. Additionally, the ST-GNN had fewer trainable parameters (12.1k) compared to the LSTM (17.9k), resulting in faster training times. Figure 3 illustrates the predicted versus true values of the target time series (Node 0, which is dependent of the rest of the nodes) across the three dependency settings.

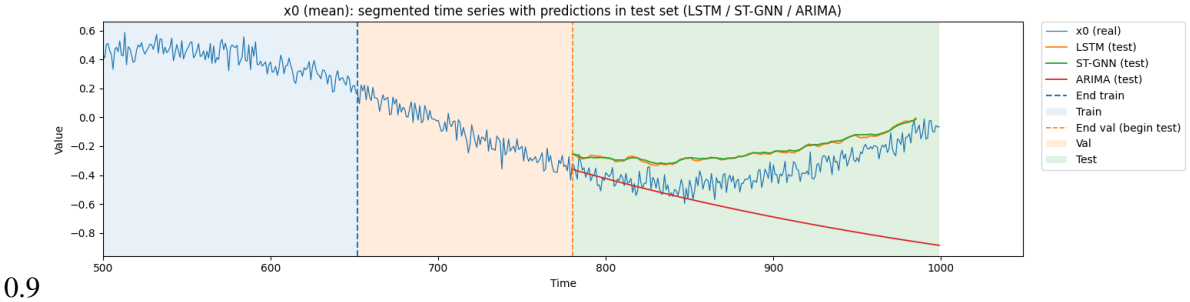
Table 4: Summary of hyperparameters used for the LSTM and ST-GNN models. Both models were trained under identical learning conditions to ensure a fair comparison.

Hyperparameter	**LSTM**	**ST-GNN**
Input size	1	1
Hidden size	64	32
Horizon	12	12
RNN layers	1 (LSTM)	1 (GRU)
GNN kernel size	–	2
Node embeddings	–	Yes
Learning rate	0.001	0.001
Batch size	32	32
Max epochs	100	100



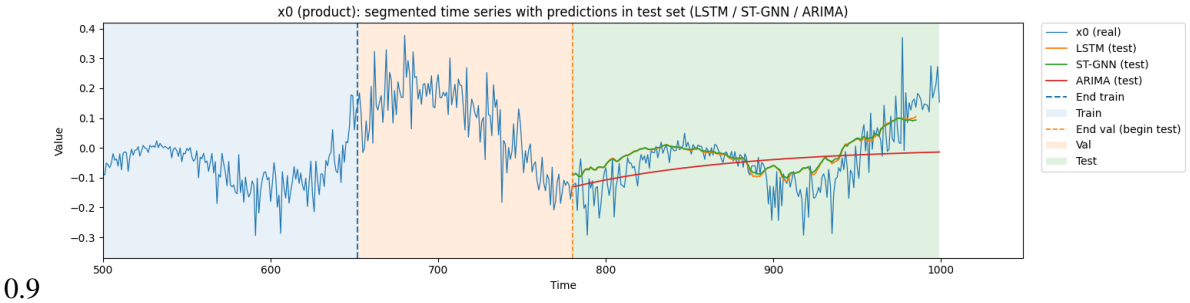
0.9

Figure 5: Sum-based time series.



0.9

Figure 6: Mean-based time series.



0.9

Figure 7: Product-based time series.

Figure 8: Forecasting results for the dependent time series under three synthetic dependency settings: (a) sum-based, (b) mean-based, and (c) product-based [7]. The plots illustrate the ground truth versus model predictions.

5.5 Discussion

Table 5 summarizes the forecasting accuracy across both synthetic settings. The reported values for the deep learning models (LSTM and ST-GNN) represent the mean and standard deviation computed over 10 independent training runs. For the ARIMA model, the forecast was computed independently for each of the five time series, and the final metrics is the average of the previous results, followed by calculating the overall standard deviation.

The results showed that the ST-GNN outperforms the ARIMA model by a large margin in all synthetic scenarios, which highlights the limitations of traditional statistical methods when handling multivariate and spatially dependent time series.

The comparison between ST-GNN and LSTM revealed the following findings:

- Mean-based setting: LSTM achieved slightly lower RMSE than ST-GNN, suggesting that in scenarios with weak spatial dependencies, temporal models should be enough for the forecasting task at hand.
- Sum-based setting: ST-GNN significantly outperformed LSTM, indicating that spatial modeling is crucial when the target series aggregates multiple sources.
- Product-based setting: Both deep learning models performed comparably, showing that they could capture nonlinear dependencies. The benefit of spatial modeling was less pronounced in this case.

To assess that the observed performance gaps were statistically significant, we conducted a Welch's independent-sample t test on the ten repeated runs of the deep-learning models and the five ARIMA models. For the RMSE in the mean-based scenario, ST-GNN vs LSTM yielded $t = 61.4$ ($df \approx 15.7,\ p < 10^{-18}$); ST-GNN vs ARIMA yielded $t = 36$ ($df \approx 4.2,\ p < 10^{-4}$). Similar results were obtained for the MAE score and the sum-based scenario, confirming that the improvements are statistically significant in a linear dependency scenario.

In the multiplicative setting, ST-GNN and LSTM achieved statistically similar performance ($t \approx 17.9,\ df \approx 18,\ p > 0.05$), suggesting both models can learn the non linear dependency. In contrast, ARIMA performed significantly worse ($t \approx -4.54,\ df \approx 9,\ p < 0.01$); including a multiplicative dependency revealed that both ST-GNN and LSTM were able to achieve low error, while ARIMA failed, highlighting the importance of nonlinear modeling capacity. It is important to mention that in this setting, the benefit of explicit spatial modeling in ST-GNN was not as pronounced as in the sum-based scenario.

Note that the ten runs for each deep learning model differ only in random initializations; therefore the independence assumption is approximated. The ARIMA results were averaged over five series, whose forecast errors could be correlated. These factors may affect slightly the t-statistics, so the reported p-values must be interpreted considering these observations.

Interestingly, while the LSTM performs slightly better than the ST-GNN in the mean-based setting, the ST-GNN achieves better results in the sum-based setting. This suggests that the ST-GNN has better performance when the target node has higher variability due to the mean-based dependency and stronger spatial correlation with other nodes.

The results obtained provide insights about incorporating graph-based spatial structure into forecasting models, which can enhance the performance in structured time series domains.

6 Final Remarks

Spatio-temporal forecasting with missing data remains an open challenge, especially in real world applications like traffic flow forecasting, environmental monitoring and energy consumption. Current models

Table 5: Forecasting performance of ARIMA, LSTM, and ST-GNN models (mean ± std) on three synthetic scenarios where Node 0 is the mean, sum or product of other time series.

Setting	Model	RMSE	MAE
Mean	ST-GNN	0.098 ± 0.0003	0.076 ± 0.0003
	ARIMA	1.073 ± 0.640	1.038 ± 0.650
	LSTM	0.091 ± 0.0002	0.091 ± 0.002
Sum	ST-GNN	0.131 ± 0.0005	0.098 ± 0.0005
	ARIMA	1.230 ± 0.450	1.190 ± 0.480
	LSTM	0.117 ± 0.001	0.116 ± 0.001
Product	ST-GNN	0.095 ± 0.001	0.074 ± 0.001
	ARIMA	1.053 ± 0.667	1.017 ± 0.679
	LSTM	0.087 ± 0.001	0.087 ± 0.001

based on statistical approaches or deep learning methods normally rely on complete datasets or do not consider the impact of imputation in forecasting tasks.

Therefore, this research aims to improve model robustness and accuracy in missing data scenarios by exploring training strategies that incorporate spectral regularization, transfer learning and data imputation.

As expected contributions, this research aims to propose a unified ST-GNN framework for forecasting sparse spatio-temporal time series, to get empirical insights into the effect of imputation on forecasting performance, and benchmark comparison against state-of-the-art models with clearly documented and reproducible evaluation tools.

References

[1] Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The Graph Neural Network Model. IEEE Transactions on Neural Networks. 20-1 (2009) 61 – 80.

[2] Brockwell, P., Davis, R.: Time Series: Theory and Methods. Springer. (2013).

[3] Jin, M.: A Survey on Graph Neural Networks for Time Series: Forecasting, Classification, Imputation and Anomaly Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 46-12 (2024) 10466-85

[4] Casolaro, A., Capone, V., Iannuzzo, G., Camastra, F.: Deep learning for time series forecasting: Advances and open problems. Information (Basel). 14-11 (2023) 598.

[5] Cini, A., Marisca, I., Zambon, D., Alippi, C.: Graph deep learning for time series forecasting. Association for Computing Machinery (ACM) 57-12 (2025) 1 – 34.

[6] Cini, A., Marisca, I., Alippi, C.: Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks. International Conference on Learning Representations. (2021) https://api.semanticscholar.org/CorpusID:246705934.

[7] Martinez-Ruiz, A., Gomez-Gil, P., Fonseca-Delgado, R.: An analysis of Spatio-Temporal Graph Neural Networks based on synthetic time series with known structural dependencies. Advances in Soft Computing. MICAI 2025 Posters Track. 2712 (2025) https://doi.org/10.1007/978-3-032-08704-1_24.

[8] Wu, Z., Pan, S., Long, G., Jiang, J., Zhang, C.: Graph WaveNet for deep spatial-temporal graph modeling. Proceedings of the 28th International Joing Conference on Artificial Intelligence. AAAI Press (2019) 1907 – 1913.

[9] Chen, Y., Li, Z., Yang, C., Wang, X., Long, G., Xu, G.: Adaptive graph recurrent network for multivariate time series imputation. Communications in computer and information science. Springer Nature Singapore. (2023) 64 – 73.

[10] PyG Documentation 2014; pytorch_geometric documentation — pytorch-geometric.readthedocs.io. `https://pytorch-geometric.readthedocs.io/en/latest/`. [Accessed 20-06-2025]

[11] Mienye, I., Swart, T., Obaido, G.: Recurrent neural networks: A comprehensive review of architectures, variants, and applications. Information (Basel), MDPI AG. 15-9 (2024) 517.

[12] Jin, G., Liang, Y., Fang, Y., Shao, Z., Huang, J., Zhang, J., Zheng, Y.: Spatio-Temporal Graph Neural Networks for Predictive Learning in Urban Computing: A Survey. IEEE Transactions on Knowledge and Data Engineering. 36-10 (2024) 5388 – 5408.

[13] Yu, B., Yin, H., Zhu, Z.: Spatio-Temporal Graph Convolutional Networks: A deep learning framework for traffic forecasting. International Joint Conferences on Artificial Intelligence Organization. (2018).

[14] Masini, R., Medeiros, M., Mendes, E.: Machine learning advances for time series forecasting. J. Econ. Surv., Wiley. 37-1 (2023) 76 – 111.

[15] Lim, B., Zohren, S.: Time-series forecasting with deep learning: a survey. Philos. Trans. A Math. Phys. Eng. Sci. The Royal Society. 379-2194 (2021).

[16] Little, R., Rubin, D.: Statistical Analysis with Missing Data. Wiley series in probability and statistics. (2020).

[17] Torres, J., Hadjout, D., Sebaa, A., Martinez-Alvarez, F., Troncoso, A.: Deep Learning for Time Series Forecasting: A survey. Big Data. (2021) 3 – 21.

[18] Rudin, W.: Principles of Mathematical Analysis. McGraw Hill. 20-1 (1976).

[19] Hamilton, W., Ying, R., Leskovec, J.: Representation Learning on Graphs: Methods and Applications. IEEE Data Eng. Bull. 40 (2017) 52–74.

[20] Wu, Z.: A Comprehensive Survey on Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems. 32-1 (2021).

[21] Wang, Q.: Deep Time Series Models for Scarce Data". Neurocomputing. 456 (2021) 504 – 518.

[22] J. Simeunović, B. Schubnel, P. -J. Alet, R. E. Carrillo: Spatio-Temporal Graph Neural Networks for Multi-Site PV Power Forecasting. IEEE Transactions on Sustainable Energy. 13-2 (2022) 1210 – 1220.

[23] Oskarsson, J., Sid'en, P., Lindsten, F.: Temporal Graph Neural Networks for Irregular Data. IEEE Transactions on Neural Networks. 20-1 (2009) 61 – 80.

[24] Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: TMF-GNN: Temporal matrix factorization-based graph neural network for multivariate time series forecasting with missing values. Expert Systems with Applications. 275 (2025).

[25] Ma, J., Cheng, J., Ding, Y., Lin, C., Jiang, F., Wang, M., Zhai, C.: Transfer learning for long-interval consecutive missing values imputation without external features in air pollution time series. Advanced Engineering Informatics. 44 (2020).

[26] Wikle, C., Zammit-Mangion, A.,: Statistical deep learning for spatial and spatiotemporal data. Annual Review of Statistics and Its Application. 10 - 1 (2023)

[27] Ahmed, S., Nielsen, I. E., Tripathi, A., Siddiqui, S., Ramachandran, R. P., Rasool, G. (2023). Transformers in time-series analysis: A tutorial. Circuits, Systems, and Signal Processing, 42(12), 7433–7466