Published: 14 July 2025
Online AM: 03 June 2025
Accepted: 15 May 2025
Revised: 03 March 2025
Received: 09 February 2024

TUTORIAL

Graph Deep Learning for Time Series Forecasting

ANDREA CINI, University of Italian Switzerland, Lugano, TI, Switzerland

IVAN MARISCA, University of Italian Switzerland, Lugano, TI, Switzerland

DANIELE ZAMBON, University of Italian Switzerland, Lugano, TI, Switzerland

CESARE ALIPPI, University of Italian Switzerland, Lugano, TI, Switzerland

Graph Deep Learning for Time Series Forecasting

ANDREA CINI, Università della Svizzera italiana, IDSIA, Lugano, Switzerland
IVAN MARISCA, Università della Svizzera italiana, IDSIA, Lugano, Switzerland
DANIELE ZAMBON, Università della Svizzera italiana, IDSIA, Lugano, Switzerland
CESARE ALIPPI, Università della Svizzera italiana, IDSIA, Lugano, Switzerland and Politecnico di Milano, Milan, Italy

Graph deep learning methods have become popular tools to process collections of correlated time series. Unlike traditional multivariate forecasting methods, graph-based predictors leverage pairwise relationships by conditioning forecasts on graphs spanning the time series collection. The conditioning takes the form of architectural inductive biases on the forecasting architecture, resulting in a family of models called spatiotemporal graph neural networks. These biases allow for training global forecasting models on large collections of time series while localizing predictions w.r.t. each element in the set (nodes) by accounting for correlations among them (edges). Recent advances in graph neural networks and deep learning for time series forecasting make the adoption of such processing framework appealing and timely. However, most studies focus on refining existing architectures by exploiting modern deep-learning practices. Conversely, foundational and methodological aspects have not been subject to systematic investigation. To fill this void, this tutorial paper aims to introduce a comprehensive methodological framework formalizing the forecasting problem and providing design principles for graph-based predictors, as well as methods to assess their performance. In addition, together with an overview of the field, we provide design guidelines and best practices, as well as an in-depth discussion of open challenges and future directions.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Neural networks**; **Artificial intelligence;**

Additional Key Words and Phrases: time series forecasting, graph deep learning, graph neural networks

ACM Reference Format:
Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi. 2025. Graph Deep Learning for Time Series Forecasting. *ACM Comput. Surv.* 57, 12, Article 321 (July 2025), 34 pages. https://doi.org/10.1145/3742784

1 Introduction

Shallow and deep neural architectures have been used to forecast time series for decades resulting in stories of both failure [181] and success [95, 153]. One of the key elements enabling most of

This work was partly supported by the Swiss National Science Foundation grants No. 204061 (*HORD GNN: Higher-Order Relations and Dynamics in Graph Neural Networks*) and No. 225351 (*Relational Deep Learning for Reliable Time Series Forecasting at Scale*).
Authors' Contact Information: Andrea Cini, Università della Svizzera italiana, IDSIA, Lugano, Switzerland; e-mail: andrea.cini@usi.ch; Ivan Marisca, Università della Svizzera italiana, IDSIA, Lugano, Switzerland; e-mail: ivan.marisca@usi.ch; Daniele Zambon, Università della Svizzera italiana, IDSIA, Lugano, Switzerland; e-mail: daniele.zambon@usi.ch; Cesare Alippi, Università della Svizzera italiana, IDSIA, Lugano, Switzerland and Politecnico di Milano, Milan, Italy; e-mail: cesare.alippi@usi.ch.

ACM 0360-0300/2025/07-ART321
https://doi.org/10.1145/3742784

the recent achievements in the field is the training of a single global neural network—with shared parameters—on large collections of related time series [12, 143]. Indeed, training a single *global* model allows for scaling the complexity of the architecture given the larger available sample size. Such an approach, however, considers each time series independently from the others and, as a consequence, does not take into account dependencies that might be instrumental for accurate predictions [61]. For example, the large variety of sensors that permeates modern cyber-physical infrastructures (e.g., traffic networks and smart grids) produces sets of time series with inherently rich *spatiotemporal structure* and *spatiotemporal dynamics*. On the one hand, global models appear inadequate in capturing such dependencies across time series, while, on the other, training a single *local* predictor, i.e., modeling the full collection as a large multivariate time series, would negate the benefits brought by sharing the trainable parameters. Furthermore, both approaches would not allow for exploiting any prior information, such as the directionality and sparsity of the dependencies.

The way out, as it often happens with major advancements in both deep learning [71, 96, 145] and time series forecasting [47, 70], is to consider the structure of the data as an inductive bias. Indeed, dependencies can be represented in terms of pairwise relationships among the time series in the collection. The resulting representation is a graph where each time series is associated with a node and functional dependencies among them are represented as edges. The conditioning of the predictor on observations at correlated time series can then take the form of an architectural inductive bias in the processing carried out by the neural architecture. **Graph neural networks (GNNs)** [6, 18], based on the **message-passing (MP)** framework [60], provide the suitable neural operators allowing for sharing parameters in the processing of the time series, while, at the same time, conditioning the predictions w.r.t. observations at neighboring nodes (related time series). The resulting models, operating over both time and space, are known as spatiotemporal graph neural networks (STGNNs) [38, 83, 100, 147]. **STGNNs** implement global and inductive architectures for time series processing by exploiting the MP mechanisms to account for spatial—other than temporal—dynamics, with the term *spatial* referring to dynamics that span the collection across different time series.

Researchers have been proposing a large variety of STGNNs by integrating MP into popular architectures, e.g., by exploiting MP blocks to implement the gates of recurrent cells [36, 100, 118, 147] and to propagate representations in fully convolutional [166, 172] and attention-based architectures [116, 167, 185]. The adoption of the resulting STGNNs has been successful in a wide range of time series processing applications ranging from traffic flow prediction [100, 166, 172] and air quality monitoring [26, 77] to energy analytics [37, 50], financial time series processing [29, 117] and epidemiological data analysis [55, 85]. However, despite the rich literature on architectures and successful applications, the methodological foundations of the field have not been systematically laid out yet. For instance, the categorization of STGNNs as global models and the interplay between globality and locality have been studied in such context only recently [38], regardless of the profound practical implications. We argue that a comprehensive formalization and methodological framework for the design of graph-based deep learning methods in time series forecasting is missing. The goal of this article is, hence, to frame the problem from the proper perspective and propose a framework instrumental to tackling the inherent challenges of the field, ranging from learning the latent graph underlying the observed data to dealing with local effects, missing data, and scalability issues.

Our contributions can be summarised as follows: We

— provide a formalization of the problem settings Section 3 and of time series forecasting given relational side information and the inductive biases associated with the proposed graph-based framework Section 4;

— present guidelines to design effective graph-based forecasting architectures Section 5–7 and to evaluate their performance Section 8;
— identify the challenges inherent to such problem settings and discuss the associated design choices Section 10 addressing, in particular, the problem of dealing with missing data Section 10.1, latent graph learning Section 10.2, the scalability of the resulting models Section 10.3 and learning in inductive settings Section 10.4.

Simulation results Section 9 and a discussion of the related works Section 2 and future directions Section 11 complete the article. We believe that the introduced comprehensive design framework will aid researchers in investigating the foundational aspects of graph deep learning for time series processing. At the same time, the article offers a tutorial to the practitioner, providing the practical and theoretical guidelines needed to apply the introduced methodologies to real-world problems.

2 Related Works

Graph deep learning methods have found widespread application in the processing of temporal data [24, 65, 83, 86, 110, 175]. In this section, we review previous related works that investigate different sub-areas within the field.

Dynamic relational data. The term *temporal graph* (or temporal network) is used to indicate scenarios where nodes, attributes, and edges of a graph are dynamic and are given over time as a sequence of events localized at specific nodes and/or as the interactions among them [86, 110]. A typical reference application is the processing of the dynamic relationships and user profiles that characterize social networks and recommender systems. Kazemi et al. [86] propose an encoder-decoder framework to unify existing representation learning methods for dynamic graphs. Barros et al. [9] compiled a rich survey of methods for embedding dynamic networks, while Skarding et al. [152] focus on GNN approaches to the same problem. Longa et al. [110] introduce a taxonomy of tasks and models in temporal graph processing; Gravina and Bacciu [65], along with a categorization of existing architectures, introduce a benchmark based on a diverse set of available datasets. Huang et al. [74] build an alternative set of benchmarks and datasets with a focus on applications to large-scale temporal graphs. Besides temporal graphs, a large body of literature has been dedicated to the processing of sequences of arbitrary graphs, e.g., without assuming any correspondence between nodes across time steps [175, 178]. Although the settings we deal with in this paper could formally be seen as a sub-area of temporal graph processing, having actual time series associated with each node radically changes the approach to the problem, as well as the available model designs and target applications. Indeed, none of the above-mentioned frameworks explicitly target time series forecasting.

Graph-based time series processing. Spatiotemporal graph neural networks for time series processing have been pioneered in the context of traffic forecasting [100, 172] and the application of graph deep learning methods in traffic analytics have been extremely successful [81, 82, 168]. The analysis of STGNNs in the context of global and local forecasting models has been initiated in [38]. Jin et al. [83] and Chen and Eldardiry [24] carried out an in-depth survey of GNNs architectures for time series forecasting, classification, imputation, and anomaly detection. In contrast, the present paper does not focus on surveying architectures but on providing a methodological framework and a tutorial. More similarly in spirit to our work, Benidis et al. [12] offer a tutorial and a critical discussion of modern practices in deep learning for time series forecasting. Analogously, Bronstein et al. [18] and Bacciu et al. [6] provide frameworks for understanding and developing graph deep learning methods. Finally, outside of deep learning, graph-based methods for time series

processing have been studied in the context of graph signal processing [97, 127, 154] and go under the name of *time-vertex signal processing* methods [62].

3 Problem Settings

This section formalizes the problem settings. In particular, Section 3.1 introduces the reference framework, suitably extended in Section 3.2 to deal with specific scenarios typical of various application domains.

3.1 Reference Problem Settings

We consider collections of correlated time series with side relational information.

Correlated time series. Consider a collection of N, regularly and synchronously sampled, correlated time series; the ith time series of the collection is composed by a sequence of d_x-dimensional vectors $\boldsymbol{x}_t^i \in \mathbb{R}^{d_x}$ observed at each time step t and coming from sensors[1] with d_x channels each. All N time series are assumed to be *homogenous*, i.e., characterized by the same variables (observables)—say the same d_x channels. Matrix $\boldsymbol{X}_t \in \mathbb{R}^{N \times d_x}$ denotes the stacked N observations at time t, while $\boldsymbol{X}_{t:t+T}$ indicates the sequence of observations within time interval $[t, t+T)$; with the shorthand $\boldsymbol{X}_{<t}$ we indicate observations at time steps up to t (excluded). Exogenous variables associated with each time series are denoted by $\boldsymbol{U}_t \in \mathbb{R}^{N \times d_u}$, while static (time-independent) attributes are grouped in matrix $\boldsymbol{V} \in \mathbb{R}^{N \times d_v}$. We consider a setup where observations have been generated by a *time-invariant* spatiotemporal stochastic process such that

$$\boldsymbol{x}_t^i \sim p^i\left(\boldsymbol{x}_t^i | \boldsymbol{X}_{<t}, \boldsymbol{U}_{\leq t}, \boldsymbol{V}\right) \quad \forall i = 1, \dots, N;, \tag{1}$$

in particular, we assume the existence of a predictive causality *à la Granger* [61], i.e., we assume that forecasts for a single time series can benefit—in terms of accuracy—from accounting for the past values of (a subset of) other time series in the collection. We do not assume that the same stochastic process generates all the N time series in the collection. This means that, in general, $p^i \neq p^j$ if $i \neq j$, while the assumption of time invariance remains valid. In the sequel, the term *spatial* refers to the dimension of size N, that spans the time series collection; in the case of physical sensors, the term spatial reflects the fact that each time series might correspond to a different physical location.

Relational information. Relational dependencies among the time series collection can be exploited to inform the downstream processing and allow for, e.g., getting rid of spurious correlations in the observed sequences of data. Pairwise relationships existing among the time series can be encoded by a (possibly dynamic) adjacency matrix $\boldsymbol{A}_t \in \{0, 1\}^{N \times N}$ that accounts for the (possibly asymmetric) dependencies at time step t; optional edge attributes $\boldsymbol{e}_t^{ij} \in \mathbb{R}^{d_e}$ can be associated to each non-zero entry of $\boldsymbol{A}_t$. In particular, we denote the set of attributed edges encoding all the available relational information by $\mathcal{E}_t \doteq \{\langle (i, j), \boldsymbol{e}_t^{ij} \rangle \mid \forall i, j : \boldsymbol{A}_t[i, j] \neq 0\}$. Whenever edge attributes are scalar, i.e., $d_e = 1$, edge set $\mathcal{E}_t$ can be simply represented as a weighted and real-valued adjacency matrix $\boldsymbol{A}_t \in \mathbb{R}^{N \times N}$. Analogously to the homogeneity assumption for observations, edges are assumed to indicate the same type of relational dependencies (e.g., physical proximity) and have the same type of attributes across the collection. We use interchangeably the terms *node* and *sensor* to indicate each of the N entities generating the time series and refer to the node set together with the relational information as *sensor network*. The tuple $\mathcal{G}_t \doteq \langle \boldsymbol{X}_t, \boldsymbol{U}_t, \mathcal{E}_t, \boldsymbol{V} \rangle$ indicates all the available information at time step t. Finally, note that for many applications (e.g., traffic networks) changes in the topology happen slowly over time and the adjacency matrix—as well as edge attributes—can

[1]Here, the term sensor has to be considered in a broad sense, as an entity producing a sequence of observations over time.

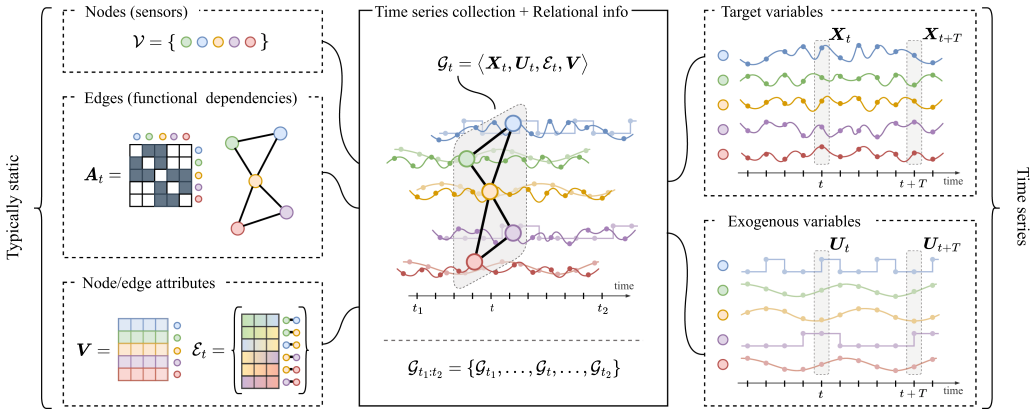

Fig. 1. A collection of synchronous and regularly sampled time series with associated pairwise dependencies.

be considered as fixed within a short window of observations, i.e., $\mathcal{E}_t = \mathcal{E}$ and $\boldsymbol{e}_t^{ij} = \boldsymbol{e}^{ij}$ for all (i, j) pairs. A graphical representation of the problem settings is shown in Figure 1.

Example 1. Consider a sensor network monitoring the speed of vehicles at crossroads. In this case, $\boldsymbol{X}_{1:t}$ refers to past traffic speed measurements sampled at a certain frequency. Exogenous variables $\boldsymbol{U}_t$ account for time-of-the-day and day-of-the-week identifiers, and the current weather conditions. The node-attribute matrix $\boldsymbol{V}$ collects static features related to the sensor's position, e.g., the type of road the sensor is placed in or the number of lanes. A static adjacency matrix $\boldsymbol{A}$ can be obtained by considering each pair of sensors connected by an edge—weighted by the road distance—if and only if a road segment directly connects them. Conversely, road closures and traffic diversions can be accounted for by adopting a dynamic topology $\boldsymbol{A}_t$.

3.2 Extensions to the Reference Settings

This section offers extensions to the reference problem settings by discussing how the framework can be modified to account for peculiarities typical of a wide range of practical applications.

New nodes, missing observations, and multiple collections. It is often the case that the time frames of the time series in the collection, although synchronous and regularly sampled, do not overlap perfectly, i.e., some time series might become available at a later time and there might be windows with blocks of missing observations. For example, it is typical for the number of installed sensors to grow over time and many applications are affected by the presence of missing data, e.g., associated with readout and/or communication failures which result in transient or permanent faults. These scenarios can be incorporated into the framework by setting N to the total maximum number of time series available, and, whenever needed, padding the time series appropriately to allow for a tabular representation of $\{\boldsymbol{X}_t\}_t$. An auxiliary binary exogenous variable $\boldsymbol{M}_t \in \{0, 1\}^{N \times d_x}$, called *mask*, can be introduced at each time step as $\mathcal{G}_t \doteq \langle \boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{M}_t, \mathcal{E}_t, \boldsymbol{V} \rangle$ to model the availability of observations w.r.t. each node and time step. In particular, we set $\boldsymbol{m}_t^i[k] = 1$ if kth channel in the corresponding observation $\boldsymbol{x}_t^i$ is valid, and $\boldsymbol{m}_t^i[k] = 0$ otherwise. If observations associated with the ith node are completely missing at time step t, the associated mask vector will be null, i.e., $\boldsymbol{m}_t^i = \boldsymbol{0}$. The masked representation simplifies the presentation of concepts and, at the same time, is useful in data reconstruction tasks (see Section 10.1). Finally, if collections from multiple sensor networks are available, the problem can be formalized as learning from M disjoint sets of correlated time series $\mathcal{D} = \{\mathcal{G}^{(1)}_{t_1:t_1+T_1}, \mathcal{G}^{(2)}_{t_2:t_2+T_2}, \ldots, \mathcal{G}^{(M)}_{t_m:t_m+T_m}\}$, potentially without overlapping time frames. In

the latter case, we assume the absence of functional dependencies between time series in different sets and the homogeneity of node features and edge attributes across collections.

Heterogeneous time series and edge attributes. Heterogeneous sets of correlated time series are commonly found in the real world (e.g., consider a set of weather stations equipped with different sensory packages) and result in collections where observations across the time series in the set might correspond to different variables. Luckily, dealing with this setting is relatively straightforward and can be done in several ways. In particular, the masked representation introduced in the above paragraph can be used to pad each time series to the same dimension d_{max} and keep track of the available channels at each node; moreover, the sensor type of each sensor can be encoded in the attribute matrix $\boldsymbol{V}$. If the total number of variables is too large or is expected to change over time, one alternative strategy is to map each observation into a shared homogenous representation (see, e.g., relational models such as [146]). Heterogeneous edge attributes can be dealt with analogously to heterogeneous node features.

4 Graph-based Time Series Forecasting

We address in the following the multi-step-ahead time-series forecasting problem [12], i.e., we are interested in predicting, for each time step t and some forecasting horizon $H \geq 1$, the h step-ahead observations $\boldsymbol{X}_{t+h}$ for all $h \in [0, H)$ given a window of $W \geq 1$ past observations. In particular, we are interested in learning a model $p_{\boldsymbol{\theta}}$ approximating the unknown conditional probability

$$p_{\boldsymbol{\theta}}\left(\boldsymbol{x}^i_{t+h} \middle| \boldsymbol{X}_{t-W:t}, \boldsymbol{U}_{t-W:t+h+1}, \boldsymbol{V}\right) \approx p^i\left(\boldsymbol{x}^i_{t+h} \middle| \boldsymbol{X}_{<t}, \boldsymbol{U}_{\leq t+h}, \boldsymbol{V}\right) \quad \forall h \in [0, H), \forall i = 1, \dots, N, \tag{2}$$

where $\boldsymbol{\theta}$ indicates the learnable parameters of the model which may or may not be specialized w.r.t. the ith time series (see Section 7). Note that not all the exogenous variables $\boldsymbol{U}_{\leq t+h}$ might be available up to time step $t + h$ in practical applications;[2] in such cases, predictions will be conditioned on covariates up to $t - 1$, i.e., $\boldsymbol{U}_{t-W:t}$.

Relational Inductive Biases for Time Series Forecasting. Learning an accurate model $p_{\boldsymbol{\theta}}$ following Equation (2) can become increasingly difficult as the number of time series in the collection grows. Intuitively, the high dimensionality of the problem can lead to spurious correlations among the observed time series, impairing the effectiveness of the learning procedure. One way to address this issue is to embed the available relational information as an inductive bias into the model. In particular, dependencies among the time series can be used to condition the prediction and, as discussed in Section 5, accounted for in the predictor through an architectural bias. The considered family of models can then be written as

$$p_{\boldsymbol{\theta}}\left(\boldsymbol{x}^i_{t+h} \middle| \mathcal{G}_{t-W:t}, \boldsymbol{U}_{t:t+h+1}\right) \approx p^i\left(\boldsymbol{x}^i_{t+h} \middle| \boldsymbol{X}_{<t}, \boldsymbol{U}_{\leq t+h}, \boldsymbol{V}\right) \quad \forall h \in [0, H). \tag{3}$$

Notably, the conditioning on the sequence of attributed graphs $\mathcal{G}_{t-W:t}$ and, in particular, on the relationships encoded in $\mathcal{E}_{t-W:t}$, can localize predictions w.r.t. the neighborhood of each node and is intended to constrain the model to the most plausible ones. In the sequel, we focus on point forecasts, i.e., we limit our analysis to the problem of predicting point estimates rather than modeling a full probability distribution. Under such assumption, we can consider predictive model families $\mathcal{F}(\cdot\,; \boldsymbol{\theta})$ such that

$$\widehat{\boldsymbol{X}}_{t:t+H} = \mathcal{F}\left(\mathcal{G}_{t-W:t}, \boldsymbol{U}_{t:t+h+1}; \boldsymbol{\theta}\right) \quad \text{s.t.} \quad \widehat{\boldsymbol{X}}_{t:t+H} \approx \mathbb{E}_p\left[\boldsymbol{X}_{t:t+H}\right]. \tag{4}$$

[2]Exogenous variables might contain, for example, actual weather conditions (available up to time step t) or estimated values, e.g., weather forecasts, available for future time steps as well (up to time step $t + h$).

Parameters $\boldsymbol{\theta}$ can be learned by minimizing a cost function $\ell(\cdot)$ on a training set, i.e.,

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{T} \sum_{t=1}^{T} \ell\left(\widehat{\boldsymbol{X}}_{t:t+H}, \boldsymbol{X}_{t:t+H}\right), \tag{5}$$

where the cost is, e.g., the *squared error*

$$\ell\left(\widehat{\boldsymbol{X}}_{t:t+H}, \boldsymbol{X}_{t:t+H}\right) = \frac{1}{NH} \sum_{i=1}^{N} \sum_{h=0}^{H-1} \left\| \widehat{\boldsymbol{x}}^{i}_{t+h} - \boldsymbol{x}^{i}_{t+h} \right\|_2^2. \tag{6}$$

The following sections delve into the design of $\mathcal{F}(\cdot\,; \boldsymbol{\theta})$ and graph deep learning methods to embed relational inductive biases [10] into the processing architecture.

5 Spatiotemporal Graph Neural Networks

This section introduces MP operators and their use in deep neural network architectures to process multiple time series; the framework follows Cini et al. [38]. As already discussed, architectures within this framework are usually referred to as STGNNs [100, 172]. STGNNs are global forecasting models where parameters are shared among the target time series; the discussion on this fundamental aspect—already mentioned in the introduction—and on hybrid global-local architectures will be resumed in Section 7. Although the focus of the article is not on providing a taxonomy of the existing architectures, we discuss in this section the design choices available to the practitioner; we refer to Jin et al. [83] for an in-depth survey on the existing architecture across different tasks in time series processing.

5.1 Message-passing Neural Networks

Modern GNNs [6, 18, 145] embed architectural biases into the processing architecture by constraining the propagation of information w.r.t. a notion of neighborhood derived from the adjacency matrix. Most of the commonly used architectures fit into the MP framework [60], which provides a recipe for designing GNN layers; GNNs that fit within the MP framework are usually referred to as *spatial GNNs*, usually in opposition to *spectral GNNs*, which instead operate in the spectral domain[3] [19, 160]. By taking as reference a graph with static node features $\boldsymbol{H}^0 \in \mathbb{R}^{N \times d_h}$ and edge set $\mathcal{E}$, we consider MP neural networks obtained by stacking MP layers that update each ith node representation at each lth layer as

$$\boldsymbol{h}^{i,l+1} = \textsc{Up}^{l}\left(\boldsymbol{h}^{i,l}, \underset{j \in \mathcal{N}(i)}{\textsc{Aggr}}\left\{\textsc{Msg}^{l}\left(\boldsymbol{h}^{i,l}, \boldsymbol{h}^{j,l}, \boldsymbol{e}^{ji}\right)\right\}\right), \tag{7}$$

where $\textsc{Up}^{l}(\cdot)$ and $\textsc{Msg}^{l}(\cdot)$ are, respectively, the update and message functions, e.g., implemented by multilayer perceptrons (MLPs). $\textsc{Aggr}\{\cdot\}$ indicates a generic permutation invariant aggregation function, while $\mathcal{N}(i)$ refers to the set of neighbors of node i, each associated with edge attribute $\boldsymbol{e}^{ji}$. In the following, we use the shorthand $\boldsymbol{H}^{l+1} = \text{MP}^{l}\left(\boldsymbol{H}^{l}, \mathcal{E}\right)$ to indicate a MP step w.r.t. the full node set. MP GNNs are *inductive* models [140], which can process unseen graphs of variable sizes by sharing weights among nodes and localizing representations by aggregating features at neighboring nodes.

By following Dwivedi et al. [49], we call *isotropic* those GNNs where the message function $\textsc{Msg}^{l}$ only depends on the features of the sender node $\boldsymbol{h}^{j,l}$; conversely, we use the term *anistropic* referring to GNNs where $\textsc{Msg}^{l}$ takes both $\boldsymbol{h}^{i,l}$ and $\boldsymbol{h}^{j,l}$ as input. For instance, a standard and

[3]Note that most of the so-called spectral GNNs can be seen as special instances of MP architectures nonetheless.

commonly used isotropic MP layer for weighted graphs (with weighted adjacency matrix $\boldsymbol{A}$) is

$$\boldsymbol{h}^{i,l+1} = \xi\Big(\boldsymbol{W}_1^l \boldsymbol{h}^{i,l} + \underset{j \in \mathcal{N}(i)}{\textsc{Sum}} \Big\{ a^{ji} \boldsymbol{W}_2^l \boldsymbol{h}^{j,l} \Big\}\Big), \tag{8}$$

where $\boldsymbol{W}_1^l$ and $\boldsymbol{W}_2^l$ are matrices of learnable parameters, $a^{ji} = \boldsymbol{A}[j,i]$, and $\xi(\cdot)$ is a generic activation function. Conversely, an example of an anisotropic MP operator, based on [17], is

$$\boldsymbol{m}^{j \to i,l} = \boldsymbol{W}_2^l \xi\Big(\boldsymbol{W}_1^l \Big[\boldsymbol{h}^{i,l} || \boldsymbol{h}^{j,l} || \boldsymbol{e}^{ji}\Big]\Big), \qquad \alpha^{ji,l} = \sigma\Big(\boldsymbol{W}_0^l \boldsymbol{m}^{ji,l}\Big), \tag{9}$$

$$\boldsymbol{h}^{i,l+1} = \xi\Big(\boldsymbol{W}_3^l \boldsymbol{h}^{i,l} + \underset{j \in \mathcal{N}(i)}{\textsc{Sum}} \Big\{ \alpha^{ji,l} \boldsymbol{m}^{j \to i,l} \Big\}\Big), \tag{10}$$

where matrices $\boldsymbol{W}_0^l \in \mathbb{R}^{1 \times d_m}$, $\boldsymbol{W}_1^l$, $\boldsymbol{W}_2^l$ and $\boldsymbol{W}_3^l$ are learnable parameters, $\sigma(\cdot)$ is the sigmoid activation function and $||$ the concatenation operator applied along the feature dimension. Intuitively, isotropic MP operators compute and aggregate messages without taking into account the representations of sender and receiver nodes and rely entirely on the presence of edge weights to weigh the contribution of different neighbors. Conversely, anisotropic schemes allow for learning adaptive aggregation and message-passing schemes aware of the nodes involved in the computation. Popular anisotropic operators exploit multi-head attention mechanisms to learn rich propagations schemes where the information flowing from each neighbor is weighted and aggregated after multiple parallel transformations [157, 158]. Indeed, we point out that selecting the proper MP operator, i.e., choosing the architectural bias for constraining the flow of information, is crucial for obtaining good performance for the problem at hand. In fact, standard isotropic filters are often based on homophily—i.e., the assumption that neighboring nodes behave in a similar way—and can suffer from over-smoothing [141].

5.2 Spatiotemporal message-passing

By following the terminology introduced in [38], STGNNs can be designed by extending MP to aggregate, at each time step, spatiotemporal information from each node's neighborhood; in particular, a **spatiotemporal message-passing (STMP)** block updates representations as

$$\boldsymbol{h}_t^{i,l+1} = \textsc{Up}^l\Big(\boldsymbol{h}_{\leq t}^{i,l}, \underset{j \in \mathcal{N}_t(i)}{\textsc{Aggr}} \Big\{ \textsc{Msg}^l\big(\boldsymbol{h}_{\leq t}^{i,l}, \boldsymbol{h}_{\leq t}^{j,l}, \boldsymbol{e}_{\leq t}^{ji}\big) \Big\}\Big), \tag{11}$$

where $\mathcal{N}_t(i)$ indicates the neighbors of the ith node at time step t (i.e., the nodes associated with incoming edges in $\mathcal{E}_t$). As in the previous case, in the following, the shorthand $\boldsymbol{H}_t^{l+1} = \text{STMP}^l\big(\boldsymbol{H}_{\leq t}^l, \mathcal{E}_{\leq t}\big)$ indicates an STMP step. Blocks of an STMP layer will have to be designed, then, to handle sequences of data. The next section provides recipes for building STGNNs based on different implementations of the STMP blocks and on existing popular STGNN architectures.

6 Forecasting Architectures

We consider forecasting architectures consisting of an encoding step followed by STMP layers and a final readout mapping representations to predictions. As such, models introduced in Equation (4) can be framed as a sequence of three operations performed at each time step:

$$\boldsymbol{h}_{t-1}^{i,0} = \textsc{Encoder}\left(\boldsymbol{x}_{t-1}^i, \boldsymbol{u}_{t-1}^i, \boldsymbol{v}^i\right), \tag{12}$$

$$\boldsymbol{H}_{t-1}^{l+1} = \text{STMP}^l\Big(\boldsymbol{H}_{\leq t-1}^l, \mathcal{E}_{\leq t-1}\Big), \quad l = 0, \ldots, L-1 \tag{13}$$

$$\hat{\boldsymbol{x}}_{t:t+H}^i = \textsc{Decoder}\left(\boldsymbol{h}_{t-1}^{i,L}, \boldsymbol{u}_{t:t+H}^i\right). \tag{14}$$

ENCODER(·) and DECODER(·) indicate generic encoder and readout layers that can be implemented, as an example, as standard fully connected linear layers, or MLPs. Note that both encoder and decoder do not propagate information along time and space. By adopting the terminology of previous works [38, 39, 58], we categorize STGNNs following this scheme in **time-then-space (TTS)**, **space-then-time (STT)**, and **time-and-space (T&S)** models. More specifically, in a TTS model the sequence of representations $\boldsymbol{h}^{i,0}_{<t}$ is processed by a sequence model, such as a **recurrent neural network (RNN)**, before any MP operation along the spatial dimension [58]; STT models are similarly obtained by inverting the order of the two operations. Conversely, in T&S models time and space are processed in a more integrated fashion, e.g., by a recurrent GNN [147] or by spatiotemporal convolutional operators [172].

Time-and-space models. We include in this category any STGNN in which the processing of the temporal and spatial dimensions cannot be factorized in two separate steps. In T&S models, representations at every node and time step are the result of joint temporal and spatial processing as in Equation (13). To the best of our knowledge, the first T&S STGNNs have been proposed by Seo et al. [147], who introduced a popular family of recurrent architectures, hereby denoted as **graph convolutional recurrent neural networks (GCRNNs)**, where standard fully-connected layers in (gated) RNNs are replaced by graph convolutions [44, 91]. As an example, by considering a **gated recurrent unit (GRU)** cell [31] and replacing graph convolutions with generic MP layers, the resulting recurrent model updates representations at each time step t as

$$\boldsymbol{Z}^l_t = \boldsymbol{H}^{l-1}_t, \tag{15}$$

$$\boldsymbol{R}^l_t = \sigma\left(\text{MP}^l_r\left(\left[\boldsymbol{Z}^l_t || \boldsymbol{H}^l_{t-1}\right], \mathcal{E}_t\right)\right), \tag{16}$$

$$\boldsymbol{O}^l_t = \sigma\left(\text{MP}^l_o\left(\left[\boldsymbol{Z}^l_t || \boldsymbol{H}^l_{t-1}\right], \mathcal{E}_t\right)\right), \tag{17}$$

$$\boldsymbol{C}^l_t = \tanh\left(\text{MP}^l_c\left(\left[\boldsymbol{Z}^l_t || \boldsymbol{R}^l_t \odot \boldsymbol{H}^l_{t-1}\right], \mathcal{E}_t\right)\right), \tag{18}$$

$$\boldsymbol{H}^l_t = \boldsymbol{O}^l_t \odot \boldsymbol{H}^l_{t-1} + (1 - \boldsymbol{O}^l_t) \odot \boldsymbol{C}^l_t, \tag{19}$$

with $\odot$ denoting the element-wise (Hadamard) product and $||$ the concatenation operation. Note that we consider, for each gate, a single MP operation at each l-th layer for conciseness' sake (a stack of MP layers is often adopted in practice). Models following a similar approach have found widespread adoption replacing standard RNNs in the context of correlated time series processing [7, 36, 100, 182]. Apart from GCRNNs, an approach to building T&S models consists of integrating a temporal operator directly into the MSG(·) function. Among the others, Wu et al. [167] and Marisca et al. [116] use cross-node attention as a mechanism to propagate information among sequences of observations at neighboring nodes. As an additional example, an analogous model could be obtained by implementing the UP(·) and MSG(·) functions of the STMP layer in Equation (11) as **temporal convolutional networks (TCNs)** [8]:

$$\boldsymbol{h}^{i,l}_{t-W:t} = \text{TCN}^l_1\left(\boldsymbol{h}^{i,l-1}_{t-W:t}, \underset{j \in \mathcal{N}_t(i)}{\text{AGGR}}\left\{\text{TCN}^l_2\left(\boldsymbol{h}^{i,l-1}_{t-W:t}, \boldsymbol{h}^{j,l-1}_{t-W:t}, \boldsymbol{e}^{ji}_{t-W:t}\right)\right\}\right). \tag{20}$$

Note that the operator resulting from the MP processing defined in Equation (20) can be seen as operating on the product graph obtained from spatial and temporal relationships [142]. Finally, a straightforward approach to build T&S architectures is that of stacking blocks of alternating spatial and temporal operators [165, 166, 172], e.g.,

$$\boldsymbol{z}^{i,l}_{t-W:t} = \text{TCN}^l\left(\boldsymbol{h}^{i,l-1}_{t-W:t}\right) \quad \forall i, \qquad \boldsymbol{H}^l_t = \text{MP}^l\left(\boldsymbol{Z}^l_t, \mathcal{E}_t\right) \quad \forall\, t, \tag{21}$$

where $\text{TCN}^l(\cdot)$ indicates a temporal convolutional network layer. The first example of such architecture was introduced in [172]. One of the major drawbacks of T&S models is their time and space complexity which usually scale with the number of nodes and edges in the graph times the number of input time steps, i.e., with $O\big(W\big(N + L|\mathcal{E}_{\max}|\big)\big)$, where $N \ll |\mathcal{E}_{\max}| = \max\{|\mathcal{E}_{t-k}|\}_{k=1}^{W}$ (see Section 10.3).

Time-then-space models. The general recipe for a TTS model consists in (1) encoding time series associated with each node into a vector, obtaining an attributed graph, and (2) propagating the obtained representations throughout the graph with a stack of standard MP layers, i.e.,

$$\boldsymbol{h}_t^{i,1} = \textsc{SeqEnc}\left(\boldsymbol{h}_{\leq t}^{i,0}\right), \tag{22}$$

$$\boldsymbol{H}_t^{l+1} = \text{MP}^l\left(\boldsymbol{H}_t^l, \mathcal{E}_t\right), \qquad \forall\, l = 1, \dots, L-1. \tag{23}$$

The sequence encoder $\textsc{SeqEnc}(\cdot)$ can be implemented by any modern deep learning architecture for sequence modeling such as RNNs, TCNs and attention-based models [98, 157]. Note that this temporal encoder can consist of multiple layers, i.e., it can be a deep network by itself. Since MP is performed only w.r.t. representations corresponding to the last time step, in case of a dynamic topology the edge set used for propagation can be obtained as a function of $\mathcal{E}_{t-W:t}$ rather than simply using $\mathcal{E}_t$, i.e., $\widetilde{\mathcal{E}}_t = \textsc{Aggr}\{\mathcal{E}_{t-W:t}\}$. A possible choice would be to take the union of all the edge sets, which, however, requires further processing in the case of attributed edges [58]. TTS models are relatively uncommon in the literature [34, 37, 58, 144] but are becoming more popular due to their efficiency and scalability compared to T&S alternatives [58], as discussed in Section 10.3. Differently from generic T&S models, in fact, the number of MP operations does not depend on the size of the window W. Indeed, TTS models have a time and space complexity that scales as $O\big(NW + L|\mathcal{E}_t|\big)$, rather than $O\big(W\big(N + L|\mathcal{E}_{\max}|\big)\big)$ of T&S models. However, the two-step encoding might introduce bottlenecks in the propagation of information.

Space-then-time models. STT models can be built by simply inverting the order of Equation (22) and 23, i.e., by using MP layers to process static representations at each time step, then encoded along the temporal axis by a sequence model, i.e.,

$$\boldsymbol{H}_t^{i,l} = \text{MP}^l\left(\boldsymbol{H}_t^{i,l-1}, \mathcal{E}_t\right), \qquad \forall\, l = 1, \dots, L-1 \tag{24}$$

$$\boldsymbol{h}_t^{i,L} = \textsc{SeqEnc}\left(\boldsymbol{h}_{t-W:t}^{i,L-1}\right). \tag{25}$$

The general idea behind STT approaches is to first enrich node observations by accounting for observations at neighboring nodes, and then process obtained sequences with a standard sequence model. Although they have seen some applications [130, 147, 184], STT models do not offer the same computational benefits of TTS models, having the same $O\big(W\big(N + L|\mathcal{E}_{\max}|\big)\big)$ complexity of T&S models. Nonetheless, as in T&S models, dynamic edge sets $\mathcal{E}_{t-W:t}$ can be accounted for by performing MP operations w.r.t. the corresponding edges at each time step. Analogously to TTS models, the factorization of the processing in two steps might introduce bottlenecks.

7 On the Globality and Locality of Spatiotemporal Graph Neural Networks

This section formally defines the concepts of globality and locality in forecasting models and emphasizes that these terms refer to model properties rather than problem settings. As both global and local models can be used to forecast collections of time series, the section discusses the peculiar position of STGNNs within this context. Finally, hybrid global-local STGNN architectures are introduced.

7.1 Global and Local Models

A time series forecasting model is called *global* if its parameters are fitted to a group of time series (either univariate or multivariate); conversely, *local* models are specific to a single (possibly multivariate) time series. In different terms, a global model is trained to make predictions by learning from a set of time series, possibly generated by different stochastic processes, without learning any time-series-specific parameter. Conversely, a local model is obtained by minimizing the forecasting error on observations coming from a single time series. Forecasting multiple time series with a local model requires fitting N models—one for each target time series—or a single (poorly scalable) multivariate model. The advantages of global models have been discussed at length in the time series forecasting literature [12, 80, 120, 143] and are mainly ascribable to the availability of large amounts of data that enable the use of models with a higher capacity w.r.t. single local models. Indeed, as noted by Montero-Manso and Hyndman [120], given a large enough window of observations and model complexity, if a global model is a universal function approximator it could in principle output predictions identical to those of a set of local models individual to each time series. Training a single global model increases the effective sample size available to the learning procedure and, consequently, allows for exploiting models with a higher model complexity preventing overfitting. Finally, being trained on a set of time series, global models can extrapolate to related but unseen time series, i.e., they can be used in inductive learning scenarios where target time series (i.e., the ones to predict) can be potentially different from those in the training set.[4] More formally, following Benidis et al. [12], and considering our problem setting and h-step-ahead forecasts, a node-level local model would approximate the process generating the data as

$$p^i_{\theta^i}\left(\boldsymbol{x}^i_{t+h} \middle| \boldsymbol{x}^i_{t-W:t}, \boldsymbol{u}^i_{t-W:t+h+1}, \boldsymbol{v}^i\right) \approx p^i\left(\boldsymbol{x}^i_{t+h} \middle| \boldsymbol{X}_{<t}, \boldsymbol{U}_{\leq t+h}, \boldsymbol{V}\right) \quad i = 1, \dots, N, \tag{26}$$

where $\boldsymbol{\theta}^i$ indicates the model's parameters fitted on the ith time series. Differently, in a node-level global model, parameters would be shared among time series, i.e,

$$p_{\boldsymbol{\theta}}\left(\boldsymbol{x}^i_{t+h} \middle| \boldsymbol{x}^i_{t-W:t}, \boldsymbol{u}^i_{t-W:t+h+1}, \boldsymbol{v}^i\right) \approx p^i\left(\boldsymbol{x}^i_{t+h} \middle| \boldsymbol{X}_{<t}, \boldsymbol{U}_{\leq t+h}, \boldsymbol{V}\right) \quad i = 1, \dots, N, \tag{27}$$

where parameters $\boldsymbol{\theta}$ can be learned by minimizing the cost function w.r.t. the complete time series collection (see Equation (5)). The main limit of standard global models in Equation (27) is that dependencies among the synchronous time series in the collections are ignored. One option would be to consider models that simply regard the input as a single multivariate time series, i.e., with a local model such that

$$p_{\boldsymbol{\theta}}\left(\boldsymbol{X}_{t+h} \middle| \boldsymbol{X}_{t-W:t}, \boldsymbol{U}_{t-W:t+h+1}, \boldsymbol{V}\right) \approx p\left(\boldsymbol{X}_{t+h} \middle| \boldsymbol{X}_{<t}, \boldsymbol{U}_{\leq t+h}, \boldsymbol{V}\right) \quad i = 1, \dots, N. \tag{28}$$

However, the resulting model would not be able to exploit the advantages that come from the global perspective and would have to deal with the high dimensionality of $\boldsymbol{X}_t$.

Globality and locality in STGNNs. STGNNs presented in Section 4 *are global models* that exploit relational architectural biases to account for related time series, going beyond the limits of the standard global approach. Indeed, by considering the STMP scheme of Equation (11), it is straightforward to see that STMP operators share parameters among the time series in the collection and condition the representations w.r.t. each node's neighborhood to account for spatial dependencies that would be ignored by standard global models. STGNNs are inductive and transferable as they do not rely upon node-specific parameters; such properties make them distinctively different

[4]Such setting is relevant in many practical application domains and also known as the *cold-start* scenario [12]; see Section 10.4.

from the local multivariate approach in Equation (28). Global models of the type implemented by STGNNs are akin to those formalized in Equation (2), i.e.,

$$p_{\theta}\left(\boldsymbol{x}^{i}_{t+h} \middle| \boldsymbol{X}_{t-W:t}, \boldsymbol{U}_{t-W:t+h+1}, \boldsymbol{V}\right) \approx p^{i}\left(\boldsymbol{x}^{i}_{t+h} \middle| \boldsymbol{X}_{<t}, \boldsymbol{U}_{\leq t+h}, \boldsymbol{V}\right) \quad \forall i = 1, \dots, N. \tag{29}$$

Besides resorting to MP operators and the relational inductive biases typical of GNNs, global models of such class can be built by considering other classes of permutation equivariant neural operators acting on sets, such as *deep sets* [174] and *transformers* [157]; comprehensive treatment of such models is out of the scope of the present paper. As discussed in Cini et al. [38], the interplay between global and local aspects plays a major role in the context of graph-based forecasting models. Indeed, although the drawbacks of the local approach are evident, global STGNNs might struggle to account for the peculiarities of each time series in the collection and might require impractically long observation windows and large memory capacity [38, 120]. For example, considering electric load forecasting, the consumption patterns of single residential customers are influenced not only by shared factors, e.g., weather, working hours, and holidays, but also by their individual daily routines, varying among users to different extents. By following [38], we refer to dynamics characterizing individual time series as *local effects*. The remainder of the section discusses how to add specialized local components into otherwise global architectures to strike a balance between the global and local modeling paradigms in the context of graph-based architectures.

7.2 Global-local STGNNs

Combining global graph-based components with local node-level components has the potential for achieving a two-fold objective: (1) exploiting relational dependencies together with side information to learn flexible and efficient graph deep learning models and, at the same time, (2) obtaining accurate predictions specialized for each time series. In particular, introducing local components specific to each time series explicitly accounts for node-level effects that would not be efficiently captured by fully global models [7, 46]. By doing so, the designer accepts a compromise in transferability that often empirically leads to higher forecasting accuracy on the task at hand. In particular, global-local STGNNs model the data-generating process as

$$p^{i}_{\theta, \{\boldsymbol{\omega}^{i}\}}\left(\boldsymbol{x}^{i}_{t+h} \middle| \mathcal{G}_{t-W:t}, \boldsymbol{U}_{t:t+h+1}\right) \approx p^{i}\left(\boldsymbol{x}^{i}_{t+h} \middle| \boldsymbol{X}_{<t}, \boldsymbol{U}_{\leq t+h}, \boldsymbol{V}\right) \quad i = 1, \dots, N, \tag{30}$$

where parameter vector $\boldsymbol{\theta}$ is shared across all nodes, whereas $\{\boldsymbol{\omega}^{i}\}_{i=1}^{N}$ are time-series dependent parameters. The associated point predictor is

$$\widehat{\boldsymbol{X}}_{t:t+H} = \mathcal{F}\left(\mathcal{G}_{t-W:t}, \boldsymbol{U}_{t:t+h+1}; \boldsymbol{\theta}, \{\boldsymbol{\omega}^{i}\}_{i=1}^{N}\right), \tag{31}$$

where $\mathcal{F}(\cdot)$ is shared among all nodes. Predictor $\mathcal{F}(\,\cdot\,)$ could be implemented, for example, as a sum between a global model and a (simpler) local one:

$$\widehat{\boldsymbol{X}}^{(1)}_{t:t+H} = \mathcal{F}_{\mathrm{G}}\left(\mathcal{G}_{t-W:t}; \boldsymbol{\theta}\right), \qquad \hat{\boldsymbol{x}}^{i,(2)}_{t:t+H} = f_{i}\left(\boldsymbol{x}^{i}_{t-W:t}; \boldsymbol{\omega}^{i}\right), \tag{32}$$

$$\hat{\boldsymbol{x}}^{i}_{t:t+H} = \hat{\boldsymbol{x}}^{i,(1)}_{t:t+H} + \hat{\boldsymbol{x}}^{i,(2)}_{t:t+H}, \tag{33}$$

or—with a more integrated approach—by using different weights for each time series at the encoding and/or decoding steps. The latter approach results in using a different encoder and/or decoder for each ith node in the template STGNN architecture Equation (12) 14 to extract representations and, eventually, project them back into the input space:

$$\boldsymbol{h}^{i,0}_{t-1} = \textsc{Encoder}_{i}\left(\boldsymbol{x}^{i}_{t-1}, \boldsymbol{u}^{i}_{t-1}, \boldsymbol{v}^{i}; \boldsymbol{\omega}^{i}_{enc}\right), \tag{34}$$

$$\hat{\boldsymbol{x}}^{i}_{t:t+H} = \textsc{Decoder}_{i}\left(\boldsymbol{h}^{i,L}_{t-1}, \boldsymbol{u}^{i}_{t:t+H}; \boldsymbol{\omega}^{i}_{dec}\right). \tag{35}$$

STMP layers could in principle be modified as well to include specialized operators, e.g., by using a different local update function $\text{UP}_i(\cdot)$ for each node. However, this would be impractical unless subsets of nodes are allowed to share parameters to some extent (e.g., by clustering them [34, 38]). Clearly, specialization compromises the use of such hybrid models in inductive learning settings Section 10.4 and often results in a number of learnable parameters that can be drastically higher compared to fully global models, hence compromising computational scalability as well. Associating each node to a learnable embedding provides a method to amortize the cost of specializing the model and makes transferring the learned model to different node sets easier.

Learnable node embeddings. The presence of static node features $\boldsymbol{V}$ characterizing the time series in the collection might provide node identification mechanisms and, thus, alleviate the need for including specialized components in the architecture. However, such node attributes are either not available or not sufficient in most settings. Resorting to learnable node embeddings, i.e., a table of learnable parameters $\Omega = \boldsymbol{Q} \in \mathbb{R}^{N \times d_v}$, offers a solution and can be interpreted as amortizing the learning of node-level specialized models [38]. More specifically, instead of learning a local model for each time series, embeddings fed into modules of a global STGNN can be learned end-to-end with the forecasting architecture providing a mean to condition representations at each node w.r.t. the peculiarities of each time series.

The template model can be updated to account for the learned embeddings by changing the encoder and decoder as

$$\boldsymbol{h}_{t-1}^{i,0} = \textsc{Encoder}\left(\boldsymbol{x}_{t-1}^{i}, \boldsymbol{u}_{t-1}^{i}, \boldsymbol{v}^{i}, \boldsymbol{q}^{i}\right), \tag{36}$$

$$\hat{\boldsymbol{x}}_{t:t+H}^{i} = \textsc{Decoder}\left(\boldsymbol{h}_{t-1}^{i,L}, \boldsymbol{u}_{t:t+H}^{i}, \boldsymbol{q}^{i}\right), \tag{37}$$

which can be seen as amortized versions of the encoder and decoder in Equations 34–35. The encoding scheme of Equation (36) also facilitates the propagation of relevant information by identifying nodes before message passing. STMP layers can be updated as well to process, e.g., as an additional input of the message function, the embeddings of source and target nodes. Besides conditioning encoding and decoding steps, many of the popular STGNN architectures use node embeddings within the processing, often as positional encodings [116, 149] or to learn a factorized weighted adjacency matrix [7, 165, 166] (see Section 10.2). Such hybrid approaches result in impressive empirical results [38], noticeably improving the forecasting accuracy of fully global models, and have become predominant in transductive settings, i.e., when the node set is fixed. In summary, globality and locality play a pivotal role in deep learning for time series forecasting and are aspects to consider when designing graph-based predictors. Indeed, while in practical application hybrid global-local models often outperform global architectures, trade-offs in flexibility must be taken into account.

8 Assessing the Quality of Predictive Models

The quality of forecasting models is primarily assessed in terms of their performance at task. The best model (among many) is, then, usually selected as the one with *significantly* better performance than the others [45]. The squared error $\mathbb{E}\left[\|\boldsymbol{r}_t^i\|_2^2\right]$ is a commonly used performance metric and is given by the 2-norm[5] of the prediction residual

$$\boldsymbol{r}_t^i \doteq \boldsymbol{x}_{t:t+H}^{i} - \hat{\boldsymbol{x}}_{t:t+H}^{i} \in \mathbb{R}^{H \times d_x}, \tag{38}$$

that is, the difference between the observed target data $\boldsymbol{x}_{t:t+H}^{i}$ at the ith node and the associated prediction $\hat{\boldsymbol{x}}_{t:t+H}^{i}$ of model $\mathcal{F}$ made at time step $t-1$. We stress that the residual $\boldsymbol{r}_t^i$ should be

[5] As $\boldsymbol{X}_{t:t+H}$ is a $H \times N \times d_x$ tensor, norm $\|\boldsymbol{X}_{t:t+H}\|_2^2$ is intended here as $\sum_{i=1}^{N}\sum_{h=1}^{H}\|\boldsymbol{x}_{t+h}^{i}\|_2^2$.

interpreted as a (Hd_x)-dimensional vector, rather than a time series of length H, with each component associated with predictions made at time step $t-1$; for instance, considering one-step-ahead forecasts, $H = 1$ and $\boldsymbol{r}_t^i$ refers to d_x-dimensional forecast of $\boldsymbol{x}_t^i$. Given a test sequence, the mean squared error

$$\mathrm{MSE}(\mathcal{F}) \doteq \frac{1}{TN}\sum_{t=1}^{T}\sum_{i=1}^{N}\|\boldsymbol{r}_t^i\|_2^2, \tag{39}$$

is an estimate of $\mathbb{E}\left[\|\boldsymbol{r}_t^i\|_2^2\right]$. Other common performance metrics are the **root mean squared error (RMSE)** (computed as $\sqrt{\mathrm{MSE}(\mathcal{F})}$), the **mean absolute error (MAE)** considering the 1-norm instead of the 2-norm, the **mean absolute percentage error (MAPE)** weighing each residual norm $\|\boldsymbol{r}_t^i\|_1$ by the observed target $\|\boldsymbol{x}_{t:t+H}^i\|_1$, and the **mean relative error (MRE)** normalizing the sum absolute errors by sum of the observed values. Indeed, the best-performing model can be different depending on the considered figure of merit.

Limiting the evaluation to computing a set of performance metrics suffers from one important drawback. Model performance does not provide any information about the possible room for improvement and does not allow for assessing whether or not the model can be considered optimal. This limitation emerges in all real-world scenarios, as the optimal achievable performance is unknown. A possible way out—where graph-based processing can be pivotal, as discussed below—consists in assessing the presence of correlations among residuals to complement the performance-based evaluation. The underlying principle is that correlated residuals indicate the presence of structural information not captured by the model [99], thus suggesting that the predictions can be improved. Several statistical hypothesis tests to detect the presence of residual dependencies have been conceived over the years [16, 48, 109] and are mainly referred to as *randomness* tests or *whiteness* tests; these terms are reminiscent of white noise, i.e., a stochastic process displaying no correlations w.r.t. different points in time and space thus being completely random. Given residuals $\{\boldsymbol{r}_t^i\}$, such tests typically compute a statistic $C(\{\boldsymbol{r}_t^i\})$ in the form of a weighted sum of pairwise scalar statistics $\kappa(\boldsymbol{r}_t^i, \boldsymbol{r}_s^j)$ between residuals $\boldsymbol{r}_t^i, \boldsymbol{r}_s^j$. Whenever the absolute value of test statistic $C(\{\boldsymbol{r}_t^i\})$ is larger than a threshold γ, correlation is considered to be significant, that is

$$\text{If } \left|C\left(\{\boldsymbol{r}_t^i\}\right)\right| > \gamma \implies \text{reject hypothesis of uncorrelated residuals.} \tag{40}$$

While whiteness tests do not quantify the margin for model improvement, they overcome the limits of performance-based analyses by providing a global assessment independently from specific performance metrics and baselines. The following discusses how spatiotemporal relational structure can help in performing such a correlation analysis in collections of correlated time series.

Testing for residual spatiotemporal correlations. Most previous works focused on analyzing correlations along the temporal dimension [15, 72, 101] or among different time series (sensors) [40, 121]. The joint analysis of spatiotemporal correlation however is more challenging, as studying the correlation between all possible residual pairs scales quadratically with both N and T. In such a setting, the relational side information associated with the time series—the graphs defined by $\{\mathcal{E}_t\}$ —has enabled spatiotemporal correlation analyses that scale to large time series collections. In particular, Zambon and Alippi [176] introduce the AZ-whiteness test by designing the test statistic

$$C(\{\boldsymbol{r}_t^i\}) = \sum_t \sum_{(i,j)\in\mathcal{E}_t} \alpha_t^{ij}\,\kappa(\boldsymbol{r}_t^i, \boldsymbol{r}_t^j) + \sum_t\sum_i \beta_t^i\,\kappa(\boldsymbol{r}_t^i, \boldsymbol{r}_{t+1}^i), \tag{41}$$

composed of a sum over *spatial* edges in $\{\mathcal{E}_t\}$ and a sum over *temporally* consecutive residuals. Scalars α_t^{ij}, β_t^i account for edge weights (e.g., the strength of the relations between time series) and trade off the weight given to spatial and temporal correlations, respectively. The scalar statistic

$\kappa(\cdot, \cdot)$ is defined as

$$\kappa(\boldsymbol{r}_t^i, \boldsymbol{r}_t^j) = \text{sgn}\left(\left\langle \boldsymbol{r}_t^i, \boldsymbol{r}_s^j \right\rangle\right), \tag{42}$$

with $\text{sgn}(\cdot)$ being the sign function[6] and $\langle \cdot, \cdot \rangle$ the scalar product. Under mild assumptions, the test statistic in Equation (41) has been proven to be asymptotically distributed as a standard Gaussian distribution [176] so that γ can be easily selected to meet desired confidence levels. The AZ-whiteness test has two main advantages. First, the test is scalable as it confines the correlation analysis only to residual pairs that are more likely to be correlated, i.e., those close in time or space (e.g., $(\boldsymbol{r}_t^i, \boldsymbol{r}_{t+1}^i)$ and $(\boldsymbol{r}_t^i, \boldsymbol{r}_t^j)$ with $(i, j) \in \mathcal{E}_t$ as shown in Equation (41)). As a result, the computation of test statistic $C(\{\boldsymbol{r}_t^i\})$ scales linearly in the number of edges and time steps. Second, thanks to the use of function $\text{sgn}(\cdot)$ in Equation (42), the test is distribution-free, which enables its application to real-world scenarios where the distribution of the residuals is typically unknown.

Besides global assessments of the overall model quality, the analysis of residual patterns localized in space and/or time provides valuable insights for discovering issues related to, e.g., faulty sensors, non-stationary dynamics, or dependencies that the model could not properly learn [40]. Anselin [4] pioneered research in this direction, although focusing on spatial data for geographical analysis. Zambon and Alippi [177], instead, provide a set of procedures, based on the AZ-whiteness statistics, to identify space-time regions associated with significant residual correlations or inaccurate forecasts.

9 Practical Examples and Experiments

Before analyzing challenges and future directions, this section complements the discussion carried out so far with numerical simulations on benchmark datasets from relevant application domains and synthetic data. The objective here is to show the impact of the transition from standard global and local deep learning predictors to graph-based architectures when forecasting collections of correlated time series. We follow the same experimental settings of Cini et al. [38].

Baselines. As a case study, we consider *recurrent* architectures. In particular, starting from standard RNNs, implemented as GRUs [33], we compare the performance of a single global RNN sharing parameters across the collections against a set of local models and against the multivariate approach. In particular, we consider the following baselines.

RNN: a global node-level GRU conditioning predictions only on the history of the target as in Equation (27). This model does not take spatial dependencies into account.

FC-RNN: a GRU taking as input all of the time series concatenated along the spatial dimension as if they were a single multivariate sequence. This model lacks flexibility and does not exploit prior relational information.

LocalRNNs: a set of local GRUs. Each GRU is specialized on a specific time series and no parameter is shared. Similarly to the global node-level model, spatial dependencies are ignored.

Then, for what concerns graph-based architectures, we consider both TTS and T&S recurrent architectures. Specifically, we build TTS models by stacking MP layers after a RNN encoder and take GCRNNs as reference T&S architectures. For both architectures, we implement variants with both isotropic and anisotropic message-passing. In particular, we compare the following model architectures.

RNN+IMP: a global TTS model composed by a GRU followed by a stack of isotropic MP layers. The MP operator is defined as in Equation (8).

[6]The sign function is such that $\text{sgn}(a)$ is equal to -1, 0 or 1 if a is positive, null or negative, respectively.

RNN+AMP: a global TTS model composed by a GRU followed by anisotropic MP layers. The MP operator is defined as in Equations 9–10.

GCRNN-IMP: a global T&S gated GCRNN with isotropic MP. The recurrent cell implementation follows Equations 16–19, the MP operator is set up as in Equation (8).

GCRNN-AMP: a global T&S gated GCRNN with anisotropic MP. The recurrent cell implementation follows Equations 16–19, the MP operator is set up as in Equations 9–10.

All the considered architectures follow the schema defined in Equations 12 14, and the different variants are obtained by changing the implementation of the STMP block. We stress again that all the global models share the same parameters across the time series in the collection. Finally, we also consider global-local variants of the above global models (RNN included) by adding node embeddings as inputs to the encoder and/or decoder, as in Equation (36) and Equation (37).

9.1 Synthetic Data

In this experiment, we show the models' performance in a controlled environment. We train the models on the task of one-step-ahead prediction. We use the MAE as the figure of merit and report the AZ-whiteness statistics.

System model. We consider the variation of GPVAR [176] provided by Cini et al. [38] as the data-generating process. Data are generated by the recursive application, starting from noise, of a polynomial graph filter [78] (with parameters shared across time series) and an autoregressive filter (with parameters specific to each time series). Formally, the underlying system model is specified by

$$\begin{aligned} \boldsymbol{H}_t &= \sum_{l=1}^{L} \sum_{q=1}^{Q} \Theta_{q,l} \boldsymbol{A}^{l-1} \boldsymbol{X}_{t-q}, \\ \boldsymbol{X}_{t+1} &= \boldsymbol{a} \odot \tanh\left(\boldsymbol{H}_t\right) + \boldsymbol{b} \odot \tanh\left(\boldsymbol{X}_{t-1}\right) + \eta_t, \end{aligned} \tag{43}$$

where $\boldsymbol{\Theta} \in \mathbb{R}^{Q \times L}$, $\boldsymbol{a} \in \mathbb{R}^{N}$, $\boldsymbol{b} \in \mathbb{R}^{N}$ and $\eta_t \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{I})$. As in [38], we consider two variants of the dataset, according to the initialization of $\boldsymbol{a}$ and $\boldsymbol{b}$. In **GPVAR-L** we set $\boldsymbol{a}$ and $\boldsymbol{b}$ by sampling them from a uniform distribution as $\boldsymbol{a}, \boldsymbol{b} \sim \mathcal{U}(-2, 2)$ to inject local effects into the process. In **GPVAR-G**, instead, we fix $\boldsymbol{a} = \boldsymbol{b} = \mathbf{0.5}$ to remove any local effect. A detailed description of the experimental setting is reported in Appendix B.1.

Results. Table 1 reports the models' forecasting performance in terms of MAE and three values of the AZ-whiteness test statistic [176] accounting for temporal, spatial, and spatiotemporal correlations.[7] The performance of the optimal model is obtained analytically by considering the variance of the noise η_t in Equation (43). As expected, models that do not exploit spatial dependencies (RNN, FC-RNN, and LocalRNNs) struggle in both datasets, displaying large residual spatial correlation, as shown by the spatial and spatiotemporal statistics. Note that spatial correlations are more difficult to detect in GPVAR-L, due to the presence of the local dynamics determined by random vectors $\boldsymbol{a}, \boldsymbol{b}$ in Equation (43), which is reflected in the values of the spatiotemporal statistic as well, as it balances the temporal and spatial components. Graph-based methods, instead, achieve performance close to the theoretical optimum in GPVAR-G, with the test statistics close to zero. For what concerns GPVAR-L, global models (including, the graph-based methods) struggle to account for the introduced local effects. Conversely, global-local graph-based methods achieve good results in both benchmarks.

[7] Statistics for temporal (spatial) correlations are obtained by setting all weights β_t^i (α_t^{ij}) to 0.

Table 1. One-step-ahead Forecasting Error (MAE) of on GPVAR (5 Runs)

	MODELS	GPVAR-G MAE	GPVAR-G AZ-test Time	GPVAR-G AZ-test T+S	GPVAR-G AZ-test Space	GPVAR-L MAE	GPVAR-L AZ-test Time	GPVAR-L AZ-test T+S	GPVAR-L AZ-test Space
	RNN	.3999$_{\pm.0000}$	-3.0$_{\pm1.3}$	35.7$_{\pm1.0}$	53.5$_{\pm0.5}$	.5441$_{\pm.0002}$	10.8$_{\pm2.6}$	0.5$_{\pm1.9}$	-10.1$_{\pm0.3}$
	↪ + Emb.	.3991$_{\pm.0000}$	-2.6$_{\pm1.4}$	34.7$_{\pm1.2}$	51.7$_{\pm1.1}$	.4611$_{\pm.0003}$	6.1$_{\pm1.4}$	-1.1$_{\pm1.1}$	-7.7$_{\pm0.8}$
	FC-RNN	.4388$_{\pm.0027}$	261.0$_{\pm1.4}$	252.2$_{\pm6.3}$	95.6$_{\pm8.6}$	.5948$_{\pm.0102}$	108.4$_{\pm8.1}$	73.6$_{\pm6.5}$	-4.4$_{\pm2.3}$
	LocalRNNs	.4047$_{\pm.0001}$	7.0$_{\pm3.7}$	43.4$_{\pm4.2}$	54.4$_{\pm2.3}$	.4610$_{\pm.0003}$	3.2$_{\pm1.1}$	-2.3$_{\pm1.1}$	-6.5$_{\pm1.1}$
TTS	RNN+IMP	.3193$_{\pm.0000}$	0.9$_{\pm0.0}$	0.5$_{\pm0.7}$	-0.3$_{\pm0.1}$	.3808$_{\pm.0031}$	13.8$_{\pm2.2}$	7.9$_{\pm1.6}$	-2.6$_{\pm0.9}$
TTS	↪ + Emb.	.3194$_{\pm.0000}$	2.8$_{\pm2.3}$	1.8$_{\pm1.7}$	-0.2$_{\pm0.2}$	.3197$_{\pm.0001}$	1.4$_{\pm1.0}$	1.0$_{\pm0.9}$	-0.0$_{\pm0.3}$
TTS	RNN+AMP	.3193$_{\pm.0000}$	1.2$_{\pm1.6}$	0.8$_{\pm1.1}$	-0.1$_{\pm0.1}$	.3639$_{\pm.0032}$	13.1$_{\pm2.6}$	7.5$_{\pm2.4}$	-2.5$_{\pm1.0}$
TTS	↪ + Emb.	.3194$_{\pm.0000}$	1.4$_{\pm3.6}$	0.8$_{\pm2.5}$	-0.2$_{\pm0.1}$	.3199$_{\pm.0001}$	1.8$_{\pm0.7}$	1.0$_{\pm0.6}$	-0.3$_{\pm0.3}$
T&S	GCRNN-IMP	.3194$_{\pm.0000}$	1.9$_{\pm0.4}$	1.2$_{\pm0.4}$	-0.3$_{\pm0.2}$	.3714$_{\pm.0070}$	15.2$_{\pm2.9}$	9.0$_{\pm1.6}$	-2.5$_{\pm1.5}$
T&S	↪ + Emb.	.3196$_{\pm.0000}$	0.8$_{\pm3.0}$	0.4$_{\pm2.1}$	-0.3$_{\pm0.2}$	.3204$_{\pm.0001}$	2.4$_{\pm0.9}$	1.8$_{\pm0.7}$	0.1$_{\pm0.2}$
T&S	GCRNN-AMP	.3195$_{\pm.0000}$	2.6$_{\pm2.0}$	1.7$_{\pm1.4}$	-0.3$_{\pm0.2}$	.3518$_{\pm.0013}$	10.5$_{\pm2.5}$	5.7$_{\pm1.9}$	-2.4$_{\pm0.6}$
T&S	↪ + Emb.	.3197$_{\pm.0000}$	1.7$_{\pm2.6}$	1.2$_{\pm1.9}$	-0.0$_{\pm0.2}$	.3204$_{\pm.0002}$	1.8$_{\pm0.6}$	0.9$_{\pm0.4}$	-0.4$_{\pm0.5}$
	Optimal model	.3192	—	—	—	.3192	—	—	—

9.2 Benchmarks

This sequel of the section provides an assessment of the introduced baselines on datasets coming from real-world applications to show the performance of the discussed methodologies.

Datasets. Following Cini et al. [38], we consider benchmarks coming from traffic forecasting, energy analytics and air quality monitoring domain. In particular, we use the following datasets.

METR-LA & PEMS-BAY METR-LA and PEMS-BAY, introduced by Li et al. [100], are two popular traffic forecasting datasets consisting of measurements from loop detectors in the Los Angeles County Highway and San Francisco Bay Area, respectively [23].

CER-E The CER-E dataset [41] consists of energy consumption readings, aggregated into 30-minutes intervals, from 485 smart meters monitoring small and medium-sized enterprises.

AQI The AQI dataset [187] collects hourly measurements of pollutant PM2.5 from 437 air quality monitoring stations in China, spread across different cities.

EngRAD Introduced by Marisca et al. [115], the EngRAD dataset contains three years of 5 hourly-sampled weather variables generated by ECMWF **Integrated Forecasting System** (**IFS**). for 487 grid points in England. For our experiment, we use shortwave radiation as the target variable $\boldsymbol{x}_t^i$ and the remaining four variables as exogenous $\boldsymbol{u}_t^i$.

We use the same data splits and preprocessing of previous works [38, 166]. In particular, the adjacency matrices for the traffic and air quality monitoring datasets are obtained by applying a thresholded Gaussian kernel [151] on the pairwise geographic distances among sensors; for CER-E, following previous works [36], the graph connectivity is derived from the correntropy [105] among time series. We refer to Appendix B.1 for more details.

Results. Table 2 shows the results of the empirical evaluation of the reference models on the selected datasets. Graph-based architectures outperform standard local and global predictors in all considered scenarios; the performance gap is particularly wide when considering fully global models. As one might expect, local models perform and scale poorly. This is particularly evident in EngRAD, where data are generated from similar processes. In this scenario, local components

Table 2. Forecasting Results on 4 Benchmark Datasets (5 Runs)

	METR-LA		PEMS-BAY		CER-E		AQI		EngRAD	
	MAE	MRE	MAE	MRE	MAE	MRE	MAE	MRE	MAE	MRE
Local models										
FC-RNN	**3.56**$_{\pm.03}$	**6.16**$_{\pm.04}$	2.32$_{\pm.01}$	3.72$_{\pm.02}$	713.01$_{\pm8.27}$	33.75$_{\pm.39}$	18.24$_{\pm.07}$	28.45$_{\pm.11}$	**55.37**$_{\pm1.57}$	**23.15**$_{\pm.66}$
LocalRNNs	3.69$_{\pm.00}$	6.38$_{\pm.01}$	**1.91**$_{\pm.00}$	**3.06**$_{\pm.00}$	**508.95**$_{\pm1.48}$	**24.09**$_{\pm.07}$	**14.75**$_{\pm.02}$	**23.02**$_{\pm.03}$	58.80$_{\pm0.31}$	24.59$_{\pm.13}$
Global models										
RNN	3.54$_{\pm.00}$	6.13$_{\pm.00}$	1.77$_{\pm.00}$	2.84$_{\pm.00}$	456.98$_{\pm0.61}$	21.63$_{\pm.03}$	14.02$_{\pm.04}$	21.87$_{\pm.07}$	47.41$_{\pm0.57}$	19.82$_{\pm.24}$
RNN+IMP	3.34$_{\pm.01}$	5.79$_{\pm.01}$	1.72$_{\pm.00}$	2.75$_{\pm.01}$	439.13$_{\pm0.51}$	20.79$_{\pm.02}$	12.74$_{\pm.02}$	19.88$_{\pm.04}$	**44.48**$_{\pm0.24}$	**18.60**$_{\pm.10}$
RNN+AMP	**3.24**$_{\pm.01}$	**5.61**$_{\pm.01}$	1.66$_{\pm.00}$	**2.65**$_{\pm.01}$	**431.33**$_{\pm0.68}$	**20.42**$_{\pm.03}$	**12.30**$_{\pm.02}$	**19.20**$_{\pm.03}$	**44.62**$_{\pm0.35}$	**18.66**$_{\pm.15}$
GCRNN-IMP	3.35$_{\pm.01}$	5.80$_{\pm.01}$	1.70$_{\pm.01}$	2.73$_{\pm.01}$	443.85$_{\pm0.99}$	21.01$_{\pm.05}$	12.87$_{\pm.02}$	20.08$_{\pm.04}$	45.55$_{\pm0.33}$	19.05$_{\pm.14}$
GCRNN-AMP	**3.22**$_{\pm.02}$	**5.58**$_{\pm.03}$	**1.65**$_{\pm.00}$	**2.64**$_{\pm.00}$	456.72$_{\pm3.91}$	21.62$_{\pm.19}$	**12.29**$_{\pm.02}$	**19.18**$_{\pm.04}$	**43.93**$_{\pm0.55}$	**18.37**$_{\pm0.23}$
Global-local models (with embeddings)										
RNN	3.15$_{\pm.03}$	5.45$_{\pm.05}$	1.59$_{\pm.00}$	2.54$_{\pm.00}$	421.50$_{\pm1.78}$	19.95$_{\pm.08}$	13.73$_{\pm.04}$	21.42$_{\pm.06}$	46.83$_{\pm0.19}$	19.58$_{\pm.08}$
RNN+IMP	**3.08**$_{\pm.01}$	**5.33**$_{\pm.03}$	**1.58**$_{\pm.00}$	**2.53**$_{\pm.00}$	**412.44**$_{\pm2.80}$	**19.52**$_{\pm.13}$	12.33$_{\pm.02}$	19.24$_{\pm.03}$	43.96$_{\pm0.42}$	18.38$_{\pm.17}$
RNN+AMP	**3.06**$_{\pm.01}$	**5.29**$_{\pm.02}$	**1.58**$_{\pm.01}$	**2.54**$_{\pm.01}$	**412.95**$_{\pm1.28}$	**19.55**$_{\pm.06}$	**12.15**$_{\pm.02}$	**18.96**$_{\pm.03}$	43.70$_{\pm0.33}$	18.27$_{\pm.14}$
GCRNN-IMP	3.10$_{\pm.01}$	5.36$_{\pm.02}$	1.59$_{\pm.00}$	2.55$_{\pm.00}$	417.71$_{\pm1.28}$	19.77$_{\pm.06}$	12.48$_{\pm.03}$	19.47$_{\pm.04}$	44.90$_{\pm0.33}$	18.77$_{\pm.14}$
GCRNN-AMP	**3.07**$_{\pm.02}$	**5.31**$_{\pm.04}$	1.59$_{\pm.00}$	**2.54**$_{\pm.01}$	416.74$_{\pm1.57}$	19.73$_{\pm.07}$	**12.17**$_{\pm.05}$	**18.98**$_{\pm.08}$	**43.14**$_{\pm0.19}$	**18.04**$_{\pm.08}$

Best model performance within each group (local, global, global-local) reported in **bold**.

bring limited benefits compared to a fully global architecture. In every other case, hybrid global-local models obtain markedly better performance than fully global baselines. However, it should be noted that such models lack flexibility in inductive settings as discussed in Section 10.4. Moreover, anisotropic message-passing schemes outperform their isotropic counterparts in most scenarios, while TTS architectures perform on par or better than T&S models. Finally note that, although the above results are significant, they do not necessarily generalize to all datasets and TTS/T&S architectures.

10 Challenges

This section identifies and discusses the main challenges that the practitioner would typically have to deal with in processing collections of time series with graph-based forecasting methods.

10.1 Dealing with Missing Data

The time series in the collection, i.e., $\boldsymbol{X}_{t:t+T}$, may be affected by missing values, as pointed out in Section 3.2. Phenomena that result in incomplete observations include (among others) irregular sampling procedures, acquisition and communication errors, and hardware and software faults. Moreover, it is often the case that the time frames of the time series in the collection do not overlap perfectly, e.g., sensors might be installed at different points in time. This section provides guidelines on dealing with incomplete observations in settings where dependencies among the time series in the collection can be exploited for reconstruction. In particular, we discuss how the graph-based methodologies presented in the article provide useful tools to tackle the problem. For a complete treatment of these aspects, we refer to Cini et al. [36] and Marisca et al. [116].

Graph Deep Learning for Time Series Imputation. Although incomplete, we assume all the available time series to be synchronous and regularly sampled and consider the masked representation introduced in Section 3.2. In particular, we pair each $\mathcal{G}_t$ with a binary mask $\boldsymbol{M}_t$ to indicate the missing observations. To simplify the presentation, we do not consider partial observability at the level of the single sensor, i.e., given an observation vector $\boldsymbol{x}_t^i$, either all the channels are observed

or none is available, i.e., $\boldsymbol{m}_t^i \in \{\mathbf{0}, \mathbf{1}\}$. However, no further assumption is made about the missing data distribution. Clearly, the gaps in the observed data must be accounted for while processing the data. A common approach consists of reconstructing missing observations before carrying out the downstream processing, by exploiting some imputation model. Besides standard statistical methods, deep learning approaches have become a popular alternative [20, 103, 171]. In particular, graph deep learning offers the tools to exploit dependencies among time series in this context as well [28, 36, 104, 116, 125, 159]. Indeed, STGNNs have been successfully applied to multivariate time series imputation in the presence of relational side information, with attention-based methods gaining traction by solving error-compounding issues typical of recurrent architectures [116]. Cini et al. [36] formalize the reconstruction problem in the context of graph-based representations and provide a bidirectional GCRNN—paired with an additional spatial decoder—that reconstructs the missing observations by exploiting both spatial and temporal dependencies. Indeed, the spatial decoder designed in [36] offers an example of how relational inductive biases can be exploited for data reconstruction. In particular, representations w.r.t. each $\boldsymbol{x}_t^i$ vector can be obtained through STMP by masking out unavailable past observations; representations can then be used for reconstruction by aggregating values observed at neighboring nodes, i.e., as

$$\boldsymbol{Z}_t = \text{STMP}\left(\boldsymbol{H}_{<t} \odot \boldsymbol{M}_{<t}, \mathcal{E}_{<t}\right), \tag{44}$$

$$\widehat{\boldsymbol{x}}_t^i = \textsc{Dec}\left(\boldsymbol{z}_t^i, \underset{j \in \mathcal{N}(i)\setminus\{i\}}{\textsc{Aggr}}\left\{\textsc{Msg}(\boldsymbol{z}_t^j, \boldsymbol{x}_t^j \odot \boldsymbol{m}_t^j)\right\}\right), \tag{45}$$

where $\widehat{\boldsymbol{x}}_t^i$ denotes the reconstructed signal and Decoder a generic readout layer. The reconstruction can be conditioned on both past and future values by exploiting, e.g., a bidirectional architecture. Clearly, many possible designs are possible and research on the topic is increasingly active (see [83]).

Forecasting from Partial Observations. A different and more direct approach to the problem is to avoid the reconstruction step and to consider forecasting architecture that can directly deal with irregular observations. Although research on the topic of graph-based methods in this context is limited [188], many of the mechanisms used in imputation models to process the incomplete observations can potentially be adapted to build forecasting architectures (see, e.g., [116]). The main advantage of such adaptations is that the resulting model, trained end-to-end, could be used to jointly impute missing observations and forecast future values. Other methods, instead, tackle this problem in the context of continuous-time modeling, we discuss them in Section 11.

10.2 Latent Graph Learning

STGNNs rely on propagating representations through the spatial connections encoded in the graph that comes with the time series collection. The available relational information, however, can be inaccurate or inadequate for modeling the relevant dependencies. For instance, in neurobiology, the physical proximity of brain regions does not always explain the observed dynamics [63, 156]. In other cases, relational information might be completely missing. Nonetheless, relational architectural biases can be exploited by learning a graph end-to-end with the forecasting architecture [39, 90, 148, 165, 166]; in some sense, graph learning can be seen as a regularization of attention-based architectures [157], where, rather than relying on attention scores between each pair of nodes, the learned graph is used to route information only between certain nodes, thus providing localized node representations typical of graph-based processing. Learning discrete representations [122] while keeping computations sparse is indeed a key challenge for graph learning, with a large impact on the scalability of the resulting forecasting architecture [39]. The following paragraphs provide a critical overview of the most common approaches.

Directly learning an adjacency matrix. Most STGNNs rely on learning an adjacency matrix $\widetilde{\boldsymbol{A}}$ as a function of a matrix of edge scores $\boldsymbol{\Phi} \in \mathbb{R}^{N \times N}$ as

$$\widetilde{\boldsymbol{A}} = \xi(\boldsymbol{\Phi}) \quad \text{with learnable parameters} \quad \boldsymbol{\Phi}, \tag{46}$$

where $\xi(\cdot)$ indicates a nonlinear activation function that can be used to induce sparsity in the resulting adjacency matrix $\widetilde{\boldsymbol{A}}$, e.g., by thresholding the scores s.t. $[\boldsymbol{A}]_{ij} = 1$ if $\phi_{ij} > \varepsilon$ and 0 otherwise. The cost of parametrizing the full score matrix $\boldsymbol{\Phi}$ can be amortized by factorizing it as

$$\widetilde{\boldsymbol{A}} = \xi(\boldsymbol{\Phi}) \quad \text{with} \quad \boldsymbol{\Phi} = \boldsymbol{Z}_{src}\boldsymbol{Z}_{tgt}^{\top}, \tag{47}$$

where $\boldsymbol{Z}_{src}, \boldsymbol{Z}_{tgt} \in \mathbb{R}^{N \times d_z}$ are node embeddings obtained, e.g., as a function of the available data or as tables of learnable parameters. Such factorization approach has been pioneered in the context of STGNNs by the Graph WaveNet architecture [166], where $\xi(\cdot)$ is implemented by a ReLU followed by a row-wise softmax activation and node embeddings are learnable parameters. Several other methods have followed this direction [7, 107, 126] which is quite flexible; indeed, making the embeddings dependent on the input window can easily allow for modeling dynamic relationships [90], e.g., as

$$\widetilde{\boldsymbol{A}}_t = \xi(\boldsymbol{\Phi}_t) \quad \text{with} \quad \boldsymbol{\Phi}_t = (\boldsymbol{Z}_t\boldsymbol{W}_{src})(\boldsymbol{Z}_t\boldsymbol{W}_{tgt})^{\top}, \tag{48}$$

$$\boldsymbol{z}_t^i = \textsc{SeqEnc}\left(\boldsymbol{x}_{t-W:t}^i, \boldsymbol{u}_{t-W:t}^i, \boldsymbol{v}^i\right), \tag{49}$$

where $\textsc{SeqEnc}(\cdot)$ indicates a generic sequence encoder (e.g., an RNN) and $\boldsymbol{W}_{src}, \boldsymbol{W}_{tgt} \in \mathbb{R}^{d_z \times d}$ are learnable weight matrices. The drawback of such methods is that they often result in a dense $\widetilde{\boldsymbol{A}}$ matrix which makes any subsequent MP operation scale with $O(N^2)$ rather than with $O(|\mathcal{E}|)$. MTGNN [165] and GDN [46] sparsify the learned factorized adjacency by selecting, for each node, the K edges associated with the largest weights, which, however, results in sparse gradients. More recently, Zhang et al. [183] proposed a different approach based on the idea of sparsifying the learned graph by thresholding the average of learned attention scores. Finally, a general approach to learning dynamic edge scores is to compute them directly as a function of source and target node representations $\boldsymbol{z}_t^i$ and $\boldsymbol{z}_t^j$ [39], e.g,

$$\widetilde{\boldsymbol{A}}_t = \xi(\boldsymbol{\Phi}_t) \quad \text{with} \quad \boldsymbol{\Phi}_t[i,j] = \text{MLP}(\boldsymbol{z}_t^i, \boldsymbol{z}_t^j). \tag{50}$$

Learning graph distributions. A different approach to the graph learning problem consists of adopting a probabilistic perspective. Instead of directly learning a graph, probabilistic methods learn a probability distribution over graphs $p_{\Phi}(\boldsymbol{A})$ such that graphs sampled from p_{Φ} maximize the performance at task. The probabilistic approach allows for the embedding of priors directly into p_{Φ}, enabling the learning of sparse graphs as realizations of a discrete probability distribution. For example, one can consider graph distributions p_{Φ} such that

$$\widetilde{\boldsymbol{A}}_t \sim p_{\Phi_t}(\boldsymbol{A}_t), \tag{51}$$

where p_{Φ_t} is parameterized by edge scores $\boldsymbol{\Phi}_t$ obtained adopting any of the parameterizations discussed in the previous paragraph. The graph distribution can be, e.g., implemented by considering a Bernoulli variable associated with each edge or by considering more complex distributions such as top-K samplers (see, e.g., [1, 39, 87, 132]). Among probabilistic methods, NRI [90] introduces a latent variable model for predicting the interactions of physical objects by learning the discrete and dynamic edge attributes of a fully connected graph. GTS [148] simplifies the NRI module by considering only binary relationships and integrates the graph inference module in a recurrent STGNN [100]. To learn p_{Φ}, Both NRI and GTS exploit path-wise gradient estimators based on the

categorical *Gumbel trick* [79, 113]; as such, they rely on continuous relaxations of discrete distributions and suffer from the same computational setbacks of previously discussed methods. Recently, Cini et al. [39] propose variance-reduced score-based estimators that allow for sparse MP operations with $O(|\mathcal{E}|)$ computational complexity.

Outside of applications to time series forecasting, Franceschi et al. [53] propose a bi-level optimization routine to learn graphs based on a straight-through estimator [11]. Kazi et al. [87] uses the Gumbel-Top-K trick [93] to sample a K-**nearest neighbors** (K-**NN**) graph and learn edge scores by using a heuristic for increasing the likelihood of sampling edges contributing to correct classifications. Wren et al. [162] learn DAGs end-to-end by exploiting implicit maximum likelihood estimation [123]. In summary, the graph learning problem is inherently complex due to challenges related to both computational complexity as well as the learning of discrete representations with neural networks. We refer to Niculae et al. [122] and Mohamed et al. [119] for an in-depth discussion on methods and estimators for learning latent (discrete) structures in machine learning.

10.3 Computational Scalability

In the problems considered so far, scalability concerns can emerge from both the number of time series in the collection as well as their length. Indeed, data span both the temporal and the spatial dimensions. In real-world applications, e.g., smart transportation systems in large cities, dealing with thousands of time series acquired at high sampling rates over long periods of time is rather common [37, 106]. This results in a large amount of data that needs to be processed at once to account for long-range spatiotemporal dependencies across the time series in the collection. When designing and/or implementing an STGNN, the scalability issue, then, must be taken into account. As mentioned in Section 6, a generic T&S model performs L stacked MP operations for each time step resulting in a time and space complexity scaling with $O\big(W(N + L|\mathcal{E}_{\max}|)\big)$, or $O\big(WL|\mathcal{E}_{\max}|\big)$ assuming $N \ll |\mathcal{E}_{\max}|$. STT models are characterized by an analogous computation complexity, as the decoupled processing generally does not bring any advantage in this direction. Conversely, TTS models, by encoding the time series ahead of any MP operation, scale with $O\big(WN + L|\mathcal{E}_t|\big)$, which, again assuming $N \ll |\mathcal{E}_t|$, is a notable improvement. However, even models following this paradigm can struggle whenever either N, W, or $|\mathcal{E}|$ are large, and appropriate computational resources can quickly become unaffordable. This issue is particularly relevant at training time when processing batches of such high-dimensional data concurrently is needed to fit STGNNs' parameters on the available data. In the following, we discuss available methods to scale existing approaches to extremely large sensor networks, highlighting the shortcomings and advantages of the different approaches.

Graph subsampling and clustering. An often viable solution is to subsample the data fed to the model. In particular, the computational burden can be reduced at training time by extracting subgraphs from the full-time series collection [30, 68, 180] by, e.g., considering the Kth order neighborhood of a subset of nodes. Such approaches have been exploited, mostly adapted from methods developed in the context of static graph processing, and have indeed been applied to scale graph-based time series forecasting to large networks [57, 137, 165]. Subsampling methods, then, allow for capping the number of nodes/edges to be processed for each sample based on the available computational resources. The drawback of these approaches is that such a subsampling might break long-range spatiotemporal dependencies (n.b., data are not i.i.d.) and result in a noisy learning signal [37], i.e., high-variance gradient estimates. Similar arguments can be made w.r.t. small batch sizes and short input windows. As an alternative, other approaches reduce the computational complexity of processing the full graph by relying on graph clustering and pooling [13, 64] to operate

on hierarchical representations of the graph [34, 173], but still require to process the full graph at the input and output layers.

Precomputing spatiotemporal encodings. Finally, a successful and popular approach to scale GNNs to large graphs is to precompute a representation for each node ahead of training and then process the data as if they were i.i.d. (e.g., see [54]). Such an approach has been extended to spatiotemporal data in [37] by exploiting randomized deep echo state networks [14, 56] and powers of a graph shift operator to extract, in an unsupervised fashion, spatiotemporal representations w.r.t. each time step and node before performing any training. The obtained representations can then be sampled uniformly across both time and space to efficiently train a decoder for mapping them to predictions [37]. The advantage of the preprocessing approach is that it makes the computational cost of a training step independent from both the length of the sequence and the number of nodes and edges by delegating the propagation of representation through time and space to the training-free encoding step. This encoding can be carried out only once before any training epoch, without being limited, e.g., by GPU memory availability. Clearly, although empirical performance matches the state-of-the-art in relevant benchmarks [37], the downside is that separating encoding and decoding can be less effective in certain scenarios and more reliant on hyperparameter selection compared to end-to-end approaches.

10.4 Inductive Learning

As previously mentioned, global STGNNs can make predictions for never-seen-before node sets, and handle graphs of different sizes and variable topology. In practice, graph-based predictors can be used for zero-shot transfer and inductive learning and can easily handle new time series being added to the collection which, for example, corresponds to the real-world scenario of new sensors being added to a network over time. The flexibility of these models has several applications in time series processing besides forecasting, e.g., as models for performing spatiotemporal kriging [155] or virtual sensing [36, 164, 186], where inductive STGNNs can be used to perform graph-based spatial interpolation. However, performance in the inductive setting can quickly degrade as soon as the target time series exhibit dynamics that deviate from those observed in the training examples [38]. Furthermore, including node-specific local components in the forecasting architecture—which as we discussed can be critical for accurate predictions—makes such STGNNs unable to perform inductive inferences. Luckily, as discussed in the following, adapting such models by exploiting a small number of observations can enable transfer.

Transfer learning. STGNNs can be adjusted to account for other sets of time series (with different dynamics) by fine-tuning on the available data a subset of the forecasting architectures weights [38] or exploiting other transfer learning strategies [114], e.g., based on ideas from meta learning [129]. For what concerns global-local architectures, the use of node embeddings can amortize the cost of the transfer learning by limiting the fine-tuning of the model to fitting a new set of embeddings for the nodes in the target set while freezing the shared weights [38]. Furthermore, node embeddings can be regularized to facilitate transfer by structuring the latent space [38] or by forcing new node embeddings to be close to those learned from the initial training set [170].

11 Future Directions

Besides the challenges identified in the paper that are indeed still open and the subject of extensive research, we can identify several promising research directions for future works to delve into.

Graph state-space models. Graph-based processing has been recently exploited to design state-space models [47, 135] based on a graph-structured state representations [179]. The resulting *graph*

state-space models learn state graphs that can be disjoint from the input, i.e., can have a number of nodes that is larger or smaller than the number of input time series and a different associated topology. Ad-hoc Kalman filtering techniques have been introduced and have led to promising empirical results [3]. The resulting framework encompasses several existing architectures (e.g., [7, 147]) while enabling more advanced designs that, however, have not been fully explored yet.

Spatial and temporal hierarchies. The spatiotemporal structure of the data allows for processing observations and making predictions at different scales, both in time [5] and in space [76]. This idea has been indeed exploited in deep learning methods for hierarchical time series forecasting [21, 69, 134, 136, 189]. As briefly mentioned in Section 10.3, several STGNN can take advantage of graph pooling to operate at different levels of resolution [66, 173]. In particular, hierarchical time series forecasting and end-to-end graph-based time series clustering have been recently integrated within the same forecasting framework [34]. Marisca et al. [115] used hierarchal representation as a means to deal with missing data and sparse observations. Future works can investigate methodologies to process spatiotemporal time series in an integrated hierarchical fashion across both time and space while taking the coherency of the forecasts into account.

Continuous space-time models. Modeling dynamics in the continuum, whether in the spatial or temporal dimensions, with differential equations has become popular in deep learning [22, 94, 111]. This approach is particularly convenient to operate in scenarios involving irregularly sampled data [150], where both training and inference can be performed w.r.t. arbitrary points in time and space [27, 88]. Indeed, continuous space-time models find applications in multi-scale analysis and simulation of physical systems [133]. Rubanova et al. [139] pioneered research in this direction showcasing the potential of such techniques in time-series applications. Graph-based approaches have been recently adopting an analogous approach to spatiotemporal data [32, 51, 75, 108], dynamic topologies [112], and graph learning procedures [84]. Future works should address the design of sound and unifying frameworks for neural continuous space-time modeling; in particular, models that process space and time in an integrated fashion should be further explored.

Probabilistic forecasting. While we focused on point predictions, deep learning methods have been widely applied to probabilistic forecasting [42, 59, 135, 143, 161]. One can in principle exploit (most of) such methodologies to make STGNNs output probabilistic predictions. As an example, quantile regression [92] allows for obtaining uncertainty estimates by simply using an appropriate loss function and adding an output for each quantile of interest [161]. In this regard, Wu et al. [163] carry out a study of several standard uncertainty estimation techniques in the context of spatiotemporal forecasting. However, while specialized probabilistic STGNN architectures exist (e.g, [25, 128]), as similarly recognized in [83], future works should further explore the use of relational inductive biases to obtain calibrated probabilistic forecasts and uncertainty estimates.

Open benchmarks and models. As mentioned in Section 2, graph deep learning for time series forecasting has been historically pioneered in the context of traffic forecasting and, as a consequence, most of the publicly available benchmarks come from this domain (e.g., see [67, 100]). Within this context, Liu et al. [106] released one of the largest benchmarks available, consisting of a large collection of traffic speed measurements recorded by the California Department of Transportation.[8] The CER-E dataset [41] has been widely used as a load forecasting benchmark [36, 38, 116]. Cini et al. [37] introduced large-scale benchmarks for forecasting photovoltaic production and energy consumption, with the latter being based on CER-E. Additional weather and photovoltaic forecasting datasets were introduced by Marisca et al. [115] and De Felice et al. [43].

[8] https://pems.dot.ca.gov/

Other commonly used benchmarks come from air quality monitoring applications [187]. However, a large and heterogeneous benchmark for correlated time series forecasting is currently missing, as well as a shared benchmarking procedure and software infrastructure. Although standardized implementations of popular architectures are becoming available [35, 102], the field would benefit from shared evaluation platforms and benchmarks, e.g., following similar trends in (temporal) graph learning [65, 73, 74].

12 Conclusions

We introduced a comprehensive methodological framework for time series forecasting with graph deep learning methods. We formalized the problem setup, characterizing settings common to several application domains. We then introduced different spatiotemporal graph neural network architectures; we discussed their properties, advantages, and drawbacks with particular attention to the impact of global and local components. We then discussed ad-hoc techniques for model evaluation and performance assessment. We identified the challenges inherent to the field and the possible strategies to address them. Finally, we provided an outlook on future research directions.

This article offers a foundation for future research to build on, as well as a tutorial for practitioners. The result is a set of graph deep learning methods aimed at enriching modern time series forecasting practices and targeting practical, high-impact, real-world applications.

References

[1] Kareem Ahmed, Zhe Zeng, Mathias Niepert, and Guy Van den Broeck. 2023. SIMPLE: A gradient estimator for k-subset sampling. In *Proceedings of the 11th International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=GPJVuyX4p_h

[2] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Türkmen, and Yuyang Wang. 2020. GluonTS: Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research* 21, 116 (2020), 1–6. Retrieved from http://jmlr.org/papers/v21/19-820.html

[3] Cesare Alippi and Daniele Zambon. 2023. Graph Kalman Filters. DOI: 10.48550/ARXIV.2303.12021

[4] Luc Anselin. 1995. Local indicators of spatial association—LISA. *Geographical Analysis* 27, 2 (1995), 93–115. DOI: https://doi.org/10.1111/j.1538-4632.1995.tb00338.x

[5] George Athanasopoulos, Rob J Hyndman, Nikolaos Kourentzes, and Fotios Petropoulos. 2017. Forecasting with temporal hierarchies. *European Journal of Operational Research* 262, 1 (2017), 60–74.

[6] Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. 2020. A gentle introduction to deep learning for graphs. *Neural Networks* 129 (2020), 203–221.

[7] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive graph convolutional recurrent network for traffic forecasting. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.). Vol. 33, Curran Associates, Inc., 17804–17815.

[8] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[9] Claudio DT Barros, Matheus RF Mendonça, Alex B Vieira, and Artur Ziviani. 2021. A survey on embedding dynamic graphs. *ACM Computing Surveys (CSUR)* 55, 1 (2021), 1–37.

[10] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, Razvan Pascanu. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).

[11] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

[12] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, François-Xavier Aubet, Laurent Callot, and Tim Januschowski. 2022. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys* 55, 6, Article 121 (dec 2022), 36 pages. DOI: https://doi.org/10.1145/3533382

[13] Filippo Maria Bianchi and Veronica Lachi. 2023. The expressive power of pooling in Graph Neural Networks. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.). Vol. 36, Curran Associates, Inc., 71603–71618.

[14] Filippo Maria Bianchi, Simone Scardapane, Sigurd Løkse, and Robert Jenssen. 2020. Reservoir computing approaches for representation and classification of multivariate time series. *IEEE Transactions on Neural Networks and Learning Systems* 32, 5 (2020), 2169–2179.

[15] Arup Bose and Walid Hachem. 2020. A whiteness test based on the spectral measure of large non-hermitian random matrices. In *Proceedings of the ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 8768–8771.

[16] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons.

[17] Xavier Bresson and Thomas Laurent. 2017. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553* (2017).

[18] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. 2021. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478* (2021).

[19] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and deep locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014*.

[20] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. 2018. BRITS: Bidirectional recurrent imputation for time series. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Vol. 31, Curran Associates, Inc.

[21] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. 2023. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 6989–6997.

[22] Ben Chamberlain, James Rowbottom, Maria I. Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. 2021. GRAND: Graph neural diffusion. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 1407–1418.

[23] 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.

[24] Hongjie Chen and Hoda Eldardiry. 2024. Graph time-series modeling in deep learning: A survey. *ACM Transactions on Knowledge Discovery from Data* 18, 5, Article 119 (2024), 35 pages. DOI: https://doi.org/10.1145/3638534

[25] Hongjie Chen, Ryan A Rossi, Kanak Mahadik, Sungchul Kim, and Hoda Eldardiry. 2021. Graph deep factors for forecasting with applications to cloud resource allocation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 106–116.

[26] Ling Chen, Jiahui Xu, Binqing Wu, and Jianlong Huang. 2023. Group-aware graph neural network for nationwide city air quality forecasting. *ACM Transactions on Knowledge Discovery from Data* 18, 3 (2023), 1–20.

[27] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 31. Curran Associates, Inc.

[28] Yakun Chen, Zihao Li, Chao Yang, Xianzhi Wang, Guodong Long, and Guandong Xu. 2022. Adaptive graph recurrent network for multivariate time series imputation. In *Proceedings of the International Conference on Neural Information Processing*. Springer, 64–73.

[29] Yingmei Chen, Zhongyu Wei, and Xuanjing Huang. 2018. Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1655–1658.

[30] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 257–266.

[31] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, Doha, Qatar, 103–111. DOI: 10.3115/v1/W14-4012

[32] Jeongwhan Choi, Hwangyong Choi, Jeehyun Hwang, and Noseong Park. 2022. Graph neural controlled differential equations for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 6367–6374.

[33] 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).

[34] Andrea Cini, Danilo Mandic, and Cesare Alippi. 2024. Graph-based time series clustering for end-to-end hierarchical forecasting. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*, Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (Eds.). PMLR, 8985–8999.

[35] Andrea Cini and Ivan Marisca. 2022. Torch Spatiotemporal. (3 2022). Retrieved from https://github.com/TorchSpatiotemporal/tsl

[36] Andrea Cini, Ivan Marisca, and Cesare Alippi. 2022. Filling the G_ap_s: Multivariate time series imputation by graph neural networks. In *Proceedings of the International Conference on Learning Representations*.

[37] Andrea Cini, Ivan Marisca, Filippo Maria Bianchi, and Cesare Alippi. 2023. Scalable spatiotemporal graph neural networks. In*Proceedings of the 37th AAAI Conference on Artificial Intelligence* (2023).

[38] Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi. 2023. Taming local effects in graph-based spatiotemporal forecasting. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.). Vol. 36, Curran Associates, Inc., 55375–55393.

[39] Andrea Cini, Daniele Zambon, and Cesare Alippi. 2023. Sparse graph learning from spatiotemporal time series. *Journal of Machine Learning Research* 24, 242 (2023), 1–36.

[40] Andrew D. Cliff and Keith Ord. 1970. Spatial autocorrelation: A review of existing and new measures with applications. *Economic Geography* 46 (1970), 269–292. DOI: https://doi.org/10.2307/143144

[41] Commission for Energy Regulation. 2016. CER smart metering project - electricity customer behaviour trial, 2009-2010 [dataset]. *Irish Social Science Data Archive. SN: 0012-00* (2016). Retrieved from https://www.ucd.ie/issda/data/commissionforenergyregulationcer

[42] Emmanuel de Bézenac, Syama Sundar Rangapuram, Konstantinos Benidis, Michael Bohlke-Schneider, Richard Kurle, Lorenzo Stella, Hilaf Hasson, Patrick Gallinari, and Tim Januschowski. 2020. Normalizing kalman filters for multivariate time series analysis. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.). Vol. 33, Curran Associates, Inc., 2995–3007. https://proceedings.neurips.cc/paper_files/paper/2020/file/1f47cef5e38c952f94c5d61726027439-Paper.pdf

[43] Giovanni De Felice, Andrea Cini, Daniele Zambon, Vladimir Gusev, and Cesare Alippi. 2024. Graph-based virtual sensing from sparse and partial multivariate observations. In *Proceedings of the International Conference on Learning Representations*.

[44] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.). Vol. 29, Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf

[45] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7, 1 (2006), 1–30.

[46] Ailin Deng and Bryan Hooi. 2021. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4027–4035.

[47] James Durbin and Siem Jan Koopman. 2012. *Time Series Analysis by State Space Methods*. Oxford university press.

[48] James Durbin and Geoffrey S Watson. 1950. Testing for serial correlation in least squares regression: I. *Biometrika* 37, 3/4 (1950), 409–428.

[49] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2023. Benchmarking graph neural networks. *Journal of Machine Learning Research* 24, 43 (2023), 1–48. Retrieved from http://jmlr.org/papers/v24/22-0567.html

[50] Simone Eandi, Andrea Cini, Slobodan Lukovic, and Cesare Alippi. 2022. Spatio-temporal graph neural networks for aggregate load forecasting. In *Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[51] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. 2021. Spatial-temporal graph ODE networks for traffic flow forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21)*. Association for Computing Machinery, New York, NY, USA, 364–373. DOI: https://doi.org/10.1145/3447548.3467430

[52] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch geometric. In *ICLRWorkshop on Representation Learning on Graphs and Manifolds*.

[53] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1972–1982.

[54] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. 2020. Sign: Scalable inception graph neural networks. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*.

[55] Cornelius Fritz, Emilio Dorigatti, and David Rügamer. 2022. Combining graph neural networks and spatio-temporal disease models to improve the prediction of weekly COVID-19 cases in Germany. *Scientific Reports* 12, 1 (2022), 3930.

[56] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. 2017. Deep reservoir computing: A critical experimental analysis. *Neurocomputing* 268 (2017), 87–99. DOI: 10.1016/j.neucom.2016.12.089

[57] Ankit Gandhi, Sivaramakrishnan Kaveri, Vineet Chaoji, and Aakanksha. 2021. Spatio-temporal multi-graph networks for demand forecasting in online marketplaces. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 187–203.

[58] Jianfei Gao and Bruno Ribeiro. 2022. On the equivalence between temporal and static equivariant graph representations. In *Proceedings of the International Conference on Machine Learning*. PMLR, 7052–7076.

[59] Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. 2019. Probabilistic forecasting with spline quantile function RNNs. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 1901–1910.

[60] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1263–1272.

[61] Clive WJ Granger. 1969. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society* 37, 3 (1969), 424–438.

[62] Francesco Grassi, Andreas Loukas, Nathanaël Perraudin, and Benjamin Ricaud. 2017. A time-vertex signal processing framework: Scalable processing and meaningful representations for time-series on graphs. *IEEE Transactions on Signal Processing* 66, 3 (2017), 817–829.

[63] Daniele Grattarola, Lorenzo Livi, Cesare Alippi, Richard Wennberg, and Taufik A Valiante. 2022. Seizure localisation with attention-based graph neural networks. *Expert Systems with Applications* 203 (2022), 117330. DOI: 10.1016/j.eswa.2022.117330

[64] Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi. 2024. Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 35, 2 (Feb. 2024), 2708–2718. DOI: https://doi.org/10.1109/TNNLS.2022.3190922

[65] Alessio Gravina and Davide Bacciu. 2024. Deep Learning for Dynamic Graphs: Models and Benchmarks. *IEEE Transactions on Neural Networks and Learning Systems* 35, 9 (2024), 11788–11801. DOI: 10.1109/TNNLS.2024.3379735

[66] Kan Guo, Yongli Hu, Yanfeng Sun, Sean Qian, Junbin Gao, and Baocai Yin. 2021. Hierarchical graph convolution network for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 151–159.

[67] Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. 2022. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering* 34, 11 (2022), 5415–5428. DOI: 10.1109/TKDE.2021.3056502

[68] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Vol. 30, Curran Associates, Inc.

[69] Xing Han, Sambarta Dasgupta, and Joydeep Ghosh. 2021. Simultaneously reconciled quantile forecasting of hierarchically related time series. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*. PMLR, 190–198.

[70] Andrew C. Harvey. 1990. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

[71] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[72] JRM Hosking. 1981. Equivalent forms of the multivariate portmanteau statistic. *Journal of the Royal Statistical Society: Series B (Methodological)* 43, 2 (1981), 261–262.

[73] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.). Vol. 33, Curran Associates, Inc., 22118–22133. https://proceedings.neurips.cc/paper_files/paper/2020/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf

[74] Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. 2023. Temporal graph benchmark for machine learning on temporal graphs. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.). Vol. 36, Curran Associates, Inc., 2056–2073.

[75] Zijie Huang, Yizhou Sun, and Wei Wang. 2021. Coupled graph ODE for learning interacting system dynamics. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21)*. Association for Computing Machinery, New York, NY, USA, 705–715. DOI: https://doi.org/10.1145/3447548.3467385

[76] Rob J Hyndman, Roman A Ahmed, George Athanasopoulos, and Han Lin Shang. 2011. Optimal combination forecasts for hierarchical time series. *Computational Statistics & Data Analysis* 55, 9 (2011), 2579–2589.

[77] Ditsuhi Iskandaryan, Francisco Ramos, and Sergio Trilles. 2023. Graph neural network for air quality prediction: A case study in madrid. *IEEE Access* 11 (2023), 2729–2742. DOI: 10.1109/ACCESS.2023.3234214

[78] Elvin Isufi, Andreas Loukas, Nathanael Perraudin, and Geert Leus. 2019. Forecasting time series with VARMA recursions on graphs. *IEEE Transactions on Signal Processing* 67, 18 (2019), 4870–4885.

[79] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *Proceedings of the International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=rkE3y85ee

[80] Tim Januschowski, Jan Gasthaus, Yuyang Wang, David Salinas, Valentin Flunkert, Michael Bohlke-Schneider, and Laurent Callot. 2020. Criteria for classifying forecasting methods. *International Journal of Forecasting* 36, 1 (2020), 167–177.

[81] Weiwei Jiang and Jiayun Luo. 2022. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications* 207 (2022), 117921. DOI: 10.1016/j.eswa.2022.117921

[82] Guangyin Jin, Yuxuan Liang, Yuchen Fang, Zezhi Shao, Jincai Huang, Junbo Zhang, and Yu Zheng. 2023. Spatio-temporal graph neural networks for predictive learning in urban computing: A survey. *IEEE Transactions on Knowledge and Data Engineering* 36, 10 (2023), 5388–5408.

[83] Ming Jin, Huan Yee Koh, Qingsong Wen, Daniele Zambon, Cesare Alippi, Geoffrey I. Webb, Irwin King, and Shirui Pan. 2024. 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), 10466–10485. DOI: https://doi.org/10.1109/TPAMI.2024.3443141

[84] Ming Jin, Yu Zheng, Yuan-Fang Li, Siheng Chen, Bin Yang, and Shirui Pan. 2023. Multivariate time series forecasting with dynamic graph neural ODEs. *IEEE Transactions on Knowledge and Data Engineering* 35, 9 (2023), 9168–9180. DOI: 10.1109/TKDE.2022.3221989

[85] 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).

[86] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2020. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research* 21, 70 (2020), 1–73.

[87] Anees Kazi, Luca Cosmo, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M. Bronstein. 2023. Differentiable graph module (DGM) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 2 (2023), 1606–1617. DOI: 10.1109/TPAMI.2022.3170249

[88] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. 2020. Neural controlled differential equations for irregular time series. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 6696–6707.

[89] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

[90] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. 2018. Neural relational inference for interacting systems. In *Proceedings of the International Conference on Machine Learning*. PMLR, 2688–2697.

[91] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[92] Roger Koenker and Kevin F Hallock. 2001. Quantile regression. *Journal of Economic Perspectives* 15, 4 (2001), 143–156.

[93] Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *Proceedings of the International Conference on Machine Learning*. PMLR, 3499–3508.

[94] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2023. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research* 24, 89 (2023), 1–97.

[95] Manuel Kunz, Stefan Birr, Mones Raslan, Lei Ma, Zhen Li, Adele Gouttes, Mateusz Koren, Tofigh Naghibi, Johannes Stephan, Mariia Bulycheva, Matthias Grzeschik, Armin Kekic, Michael Narodovitch, Kashif Rasul, Julian Sieber, and Tim Januschowski. 2023. Deep learning based forecasting: A case study from the online fashion industry. *arXiv preprint arXiv:2305.14406* (2023).

[96] Yann LeCun and Yoshua Bengio. 1998. *Convolutional Networks for Images, Speech, and Time Series*. MIT Press, Cambridge, MA, USA, 255–258.

[97] Geert Leus, Antonio G Marques, José MF Moura, Antonio Ortega, and David I Shuman. 2023. Graph signal processing: History, development, impact, and outlook. *IEEE Signal Processing Magazine* 40, 4 (2023), 49–60.

[98] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/6775a0635c302542da2c32aa19d86be0-Paper.pdf

[99] Wai Keung Li. 2003. *Diagnostic Checks in Time Series*. CRC Press.

[100] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *Proceedings of the International Conference on Learning Representations*.

[101] Zeng Li, Clifford Lam, Jianfeng Yao, and Qiwei Yao. 2019. On testing for high-dimensional white noise. *The Annals of Statistics* 47, 6 (2019), 3382–3412.

[102] Yubo Liang, Zezhi Shao, Fei Wang, Zhao Zhang, Tao Sun, and Yongjun Xu. 2022. BasicTS: An open source fair multivariate time series prediction benchmark. In *Proceedings of the International Symposium on Benchmarking, Measuring and Optimization*. Springer, 87–101.

[103] Zachary C Lipton, David Kale, and Randall Wetzel. 2016. Directly modeling missing data in sequences with RNNs: Improved classification of clinical time series. In *Proceedings of the 1st Machine Learning for Healthcare Conference (Proceedings of Machine Learning Research)*, Finale Doshi-Velez, Jim Fackler, David Kale, Byron Wallace, and Jenna Wiens (Eds.), Vol. 56. PMLR, Northeastern University, Boston, MA, USA, 253–270.

[104] Mingzhe Liu, Han Huang, Hao Feng, Leilei Sun, Bowen Du, and Yanjie Fu. 2023. Pristi: A conditional diffusion framework for spatiotemporal imputation. In *Proceedings of the 2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 1927–1939.

[105] Weifeng Liu, Puskal P Pokharel, and Jose C Principe. 2007. Correntropy: Properties and applications in non-Gaussian signal processing. *IEEE Transactions on Signal Processing* 55, 11 (2007), 5286–5298.

[106] Xu Liu, Yutong Xia, Yuxuan Liang, Junfeng Hu, Yiwei Wang, Lei Bai, Chao Huang, Zhenguang Liu, Bryan Hooi, and Roger Zimmermann. 2023. LargeST: A benchmark dataset for large-scale traffic forecasting. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.). Vol. 36, Curran Associates, Inc., 75354–75371.

[107] Yijing Liu, Qinxian Liu, Jian-Wei Zhang, Haozhe Feng, Zhongwei Wang, Zihan Zhou, and Wei Chen. 2022. Multivariate time-series forecasting with temporal polynomial graph neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). Retrieved from https://openreview.net/forum?id=pMumil2EJh

[108] Zibo Liu, Parshin Shojaee, and Chandan K. Reddy. 2023. Graph-based multi-ODE neural networks for spatio-temporal traffic forecasting. *Transactions on Machine Learning Research* (2023). Retrieved from https://openreview.net/forum?id=Oq5XKRVYpQ

[109] Greta M Ljung and George EP Box. 1978. On a measure of lack of fit in time series models. *Biometrika* 65, 2 (1978), 297–303.

[110] Antonio Longa, Veronica Lachi, Gabriele Santin, Monica Bianchini, Bruno Lepri, Pietro Lio, Franco Scarselli, and Andrea Passerini. 2023. Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities. *Transactions on Machine Learning Research* (2023). Retrieved from https://openreview.net/forum?id=pHCdMat0gI

[111] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. 2021. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence* 3, 3 (March 2021), 218–229. DOI: https://doi.org/10.1038/s42256-021-00302-5

[112] Xiao Luo, Jingyang Yuan, Zijie Huang, Huiyu Jiang, Yifang Qin, Wei Ju, Ming Zhang, and Yizhou Sun. 2023. HOPE: High-order graph ODE for modeling interacting dynamics. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 23124–23139.

[113] C Maddison, A Mnih, and Y Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations*.

[114] Tanwi Mallick, Prasanna Balaprakash, Eric Rask, and Jane Macfarlane. 2021. Transfer learning with graph neural networks for short-term highway traffic forecasting. In *Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 10367–10374.

[115] Ivan Marisca, Cesare Alippi, and Filippo Maria Bianchi. 2024. Graph-based forecasting with missing data through spatiotemporal downsampling. In *Proceedings of the International Conference on Machine Learning*. PMLR, 34846–34865.

[116] Ivan Marisca, Andrea Cini, and Cesare Alippi. 2022. Learning to reconstruct missing data from spatiotemporal graphs with sparse observations. In *Proceedings of the Advances in Neural Information Processing Systems*.

[117] Daiki Matsunaga, Toyotaro Suzumura, and Toshihiro Takahashi. 2019. Exploring graph neural networks for stock market predictions with rolling window analysis. *arXiv preprint arXiv:1909.10660* (2019).

[118] Alessio Micheli and Domenico Tortorella. 2022. Discrete-time dynamic graph echo state networks. *Neurocomputing* 496 (2022), 85–95. DOI: 10.1016/j.neucom.2022.05.001

[119] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. 2020. Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research* 21, 132 (2020), 1–62.

[120] Pablo Montero-Manso and Rob J Hyndman. 2021. Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting* 37, 4 (2021), 1632–1653.

[121] P. A. P. Moran. 1950. Notes on continuous stochastic phenomena. *Biometrika* 37, 1/2 (1950), 17–23. DOI: https://doi.org/10.2307/2332142

[122] Vlad Niculae, Caio Corro, Nikita Nangia, Tsvetomila Mihaylova, and André F. T. Martins. 2025. Discrete latent structure in neural networks. *Foundations and TrendsR in Signal Processing* 19, 2 (2025), 99–211. DOI: 10.1561/2000000134

[123] Mathias Niepert, Pasquale Minervini, and Luca Franceschi. 2021. Implicit MLE: Backpropagating through discrete exponential family distributions. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.). Vol. 34, Curran Associates, Inc., 14567–14579. https://proceedings.neurips.cc/paper_files/paper/2021/file/7a430339c10c642c4b2251756fd1b484-Paper.pdf

[124] Kin G. Olivares, Cristian Challú, Federico Garza, Max Mergenthaler Canseco, and Artur Dubrawski. 2022. NeuralForecast: User friendly state-of-the-art neural forecasting models. PyCon Salt Lake City, Utah, US 2022. (2022). Retrieved from https://github.com/Nixtla/neuralforecast

[125] Shayegan Omidshafiei, Daniel Hennes, Marta Garnelo, Zhe Wang, Adria Recasens, Eugene Tarassov, Yi Yang, Romuald Elie, Jerome T Connor, Paul Muller, Natalie Mackraz, Kris Cao, Pol Moreno, Pablo Sprechmann, Demis Hassabis, Ian Graham, William Spearman, Nicolas Heess, and Karl Tuyls. 2022. Multiagent off-screen behavior prediction in football. *Scientific Reports* 12, 1 (2022), 8638.

[126] Boris N. Oreshkin, Arezou Amini, Lucy Coyle, and Mark J. Coates. 2021. FC-GAGA: Fully connected gated graph architecture for spatio-temporal traffic forecasting. In *Proceedings of the AAAI*.

[127] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. 2018. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE* 106, 5 (2018), 808–828.

[128] Soumyasundar Pal, Liheng Ma, Yingxue Zhang, and Mark Coates. 2021. RNN with particle flow for probabilistic spatio-temporal forecasting. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Vol. 139. PMLR, 8336–8348.

[129] 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.

[130] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5363–5370.

[131] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of the Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035.

[132] Max Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison. 2020. Gradient estimation with stochastic softmax tricks. *Advances in Neural Information Processing Systems* 33 (2020), 5691–5704.

[133] M. Raissi, P. Perdikaris, and G. E. Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* 378 (Feb. 2019), 686–707. DOI: https://doi.org/10.1016/j.jcp.2018.10.045

[134] Syama Sundar Rangapuram, Shubham Kapoor, Rajbir Singh Nirwan, Pedro Mercado, Tim Januschowski, Yuyang Wang, and Michael Bohlke-Schneider. 2023. Coherent probabilistic forecasting of temporal hierarchies. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*. PMLR, 9362–9376.

[135] Syama Sundar Rangapuram, Matthias W. Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. 2018. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Vol. 31, Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdf

[136] Syama Sundar Rangapuram, Lucien D Werner, Konstantinos Benidis, Pedro Mercado, Jan Gasthaus, and Tim Januschowski. 2021. End-to-end learning of coherent probabilistic forecasts for hierarchical time series. In *Proceedings of the International Conference on Machine Learning*. PMLR, 8832–8843.

[137] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*.

[138] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, , Guzman Lopez, Nicolas Collignon, and Rik Sarkar. 2021. PyTorch geometric temporal: Spatiotemporal signal processing with neural machine learning models. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 4564–4573.

[139] Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. 2019. Latent ordinary differential equations for irregularly-sampled time series. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.

[140] Luana Ruiz, Luiz Chamon, and Alejandro Ribeiro. 2020. Graphon neural networks and the transferability of graph neural networks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.). Vol. 33, Curran Associates, Inc., 1702–1712. https://proceedings.neurips.cc/paper_files/paper/2020/file/12bcd658ef0a540cabc36cdf2b1046fd-Paper.pdf

[141] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. 2023. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993* (2023).
[142] Mohammad Sabbaqi and Elvin Isufi. 2023. Graph-time convolutional neural networks: Architecture and theoretical analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 12 (2023), 14625–14638.
[143] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.
[144] Victor Garcia Satorras, Syama Sundar Rangapuram, and Tim Januschowski. 2022. Multivariate time series forecasting with latent graph inference. *arXiv preprint arXiv:2203.03423* (2022)
[145] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2008), 61–80.
[146] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of the European Semantic Web Conference*. Springer, 593–607.
[147] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *Proceedings of the International Conference on Neural Information Processing*. Springer, 362–373.
[148] Chao Shang and Jie Chen. 2021. Discrete graph structure learning for forecasting multiple time series. In *Proceedings of the International Conference on Learning Representations*.
[149] Zezhi Shao, Zhao Zhang, Fei Wang, Wei Wei, and Yongjun Xu. 2022. Spatial-temporal identity: A simple yet effective baseline for multivariate time series forecasting. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4454–4458.
[150] Satya Narayan Shukla and Benjamin M Marlin. 2020. A survey on principles, models and methods for learning from irregularly sampled time series. *arXiv preprint arXiv:2012.00168* (2020).
[151] 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.
[152] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. 2021. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access* 9 (2021), 79143–79168. DOI: 10.1109/ACCESS.2021.3082932
[153] Slawek Smyl. 2020. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting* 36, 1 (2020), 75–85.
[154] Ljubiša Stanković, Danilo Mandic, Miloš Daković, Miloš Brajović, Bruno Scalzo, Shengxi Li, and Anthony G Constantinides. 2020. Data analytics on graphs part II: Signals on graphs. *Foundations and Trends® in Machine Learning* 13, 2-3 (2020), 158–331. DOI: 10.1561/2200000078-2
[155] Michael L Stein. 1999. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media.
[156] Pieter Van Mierlo, Margarita Papadopoulou, Evelien Carrette, Paul Boon, Stefaan Vandenberghe, Kristl Vonck, and Daniele Marinazzo. 2014. Functional brain connectivity from EEG in epilepsy: Seizure prediction and epileptogenic focus localization. *Progress in Neurobiology* 121 (2014), 19–35.
[157] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*. 5998–6008.
[158] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the International Conference on Learning Representations*.
[159] Dingsu Wang, Yuchen Yan, Ruizhong Qiu, Yada Zhu, Kaiyu Guan, Andrew Margenot, and Hanghang Tong. 2023. Networked time series imputation via position-aware graph enhanced variational autoencoders. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2256–2268.
[160] Xiyuan Wang and Muhan Zhang. 2022. How powerful are spectral graph neural networks. In *Proceedings of the International Conference on Machine Learning*. PMLR, 23341–23362.
[161] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. 2017. A multi-horizon quantile recurrent forecaster. arXiv:1711.11053. Retrieved from https://arxiv.org/abs/1711.11053 (2017).
[162] Andrew J Wren, Pasquale Minervini, Luca Franceschi, and Valentina Zantedeschi. 2022. Learning discrete directed acyclic graphs via backpropagation. In *Proceedings of the NeurIPS 2022 Workshop on Neuro Causal and Symbolic AI (nCSI)*.
[163] Dongxia Wu, Liyao Gao, Matteo Chinazzi, Xinyue Xiong, Alessandro Vespignani, Yi-An Ma, and Rose Yu. 2021. Quantifying uncertainty in deep spatiotemporal forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1841–1851.
[164] Yuankai Wu, Dingyi Zhuang, Aurelie Labbe, and Lijun Sun. 2021. Inductive graph neural networks for spatiotemporal kriging. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4478–4485.

[165] 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.

[166] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 1907–1913.

[167] Zonghan Wu, Da Zheng, Shirui Pan, Quan Gan, Guodong Long, and George Karypis. 2024. TraverseNet: Unifying space and time in message passing for traffic forecasting. *IEEE Transactions on Neural Networks and Learning Systems* 35, 2 (2024), 2003–2013. DOI: 10.1109/TNNLS.2022.3186103

[168] Jiexia Ye, Juanjuan Zhao, Kejiang Ye, and Chengzhong Xu. 2020. How to build a graph-based deep learning architecture in traffic domain: A survey. *IEEE Transactions on Intelligent Transportation Systems* 23, 5 (2020), 3904–3924.

[169] 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*.

[170] Xueyan Yin, Feifan Li, Yanming Shen, Heng Qi, and Baocai Yin. 2022. NodeTrans: A graph transfer learning approach for traffic prediction. *arXiv preprint arXiv:2207.01301* (2022).

[171] Jinsung Yoon, James Jordon, and Mihaela Schaar. 2018. Gain: Missing data imputation using generative adversarial nets. In *Proceedings of the International Conference on Machine Learning*. PMLR, 5689–5698.

[172] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3634–3640.

[173] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2019. ST-UNet: A spatio-temporal U-network for graph-structured time series modeling. *arXiv preprint arXiv:1903.05631* (2019).

[174] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R. Salakhutdinov, and Alexander J. Smola. 2017. Deep Sets. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Vol. 30, Curran Associates, Inc.

[175] Daniele Zambon. 2022. *Anomaly and Change Detection in Sequences of Graphs*. Ph.D. Dissertation. Università della Svizzera italiana.

[176] Daniele Zambon and Cesare Alippi. 2022. AZ-whiteness test: A test for signal uncorrelation on spatio-temporal graphs. In *Proceedings of the Advances in Neural Information Processing Systems*.

[177] Daniele Zambon and Cesare Alippi. 2023. Assessment of Spatio-Temporal Predictors in the Presence of Missing and Heterogeneous Data. arXiv:2302.01701 [cs, stat]. DOI: 10.48550/arXiv.2302.01701

[178] Daniele Zambon, Cesare Alippi, and Lorenzo Livi. 2018. Concept drift and anomaly detection in graph streams. *IEEE Transactions on Neural Networks and Learning Systems* 29, 11 (Nov. 2018), 5592–5605. DOI: https://doi.org/10.1109/TNNLS.2018.2804443

[179] Daniele Zambon, Andrea Cini, Lorenzo Livi, and Cesare Alippi. 2023. Graph State-Space Models. DOI: 10.48550/ARXIV.2301.01741

[180] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2020. GraphSAINT: Graph sampling based inductive learning method. In *Proceedings of the International Conference on Learning Representations*.

[181] Guoqiang Zhang, B Eddy Patuwo, and Michael Y Hu. 1998. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting* 14, 1 (1998), 35–62.

[182] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit Yan Yeung. 2018. GaAN: Gated attention networks for learning on large and spatiotemporal graphs. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*.

[183] Xiang Zhang, Marko Zeman, Theodoros Tsiligkaridis, and Marinka Zitnik. 2022. Graph-guided network for irregularly sampled multivariate time series. In *Proceedings of the International Conference on Learning Representations, ICLR*.

[184] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. 2019. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems* 21, 9 (2019), 3848–3858.

[185] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. GMAN: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1234–1241.

[186] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, Jianzhong Qi, Chaochao Chen, and Longbiao Chen. 2023. INCREASE: Inductive graph representation learning for spatio-temporal kriging. In *Proceedings of the ACM Web Conference 2023*. 673–683.

[187] 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.

[188] Weida Zhong, Qiuling Suo, Xiaowei Jia, Aidong Zhang, and Lu Su. 2021. Heterogeneous spatio-temporal graph convolution network for traffic forecasting with missing values. In *Proceedings of the 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 707–717.

[189] Fan Zhou, Chen Pan, Lintao Ma, Yu Liu, Shiyu Wang, James Zhang, Xinxin Zhu, Xuanwei Hu, Yunhua Hu, Yangfei Zheng, Lei Lei, and Yun Hu. 2023. SLOTH: Structured learning and task-based optimization for time series forecasting on hierarchies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11417–11425.

Appendix

A Software

PyTorch Geometric (PyG) [52] is the most widely used library for developing graph neural networks. As the name suggests, PyG is based on PyTorch [131] and offers utilities to process temporal relational data as well. Specialized libraries that implement models from the temporal graph learning literature exist [138]. The PyTorch ecosystem has several options for what concern deep learning for time series forecasting such as GluonTS [2], PyTorch Forecasting,[9] Neural Forecast [124] and BasicTS [102]. Considering the settings discussed in the paper, Torch Spatiotemporal [35] focuses on graph deep learning models for processing time series collections, offering several utilities to accelerate research and prototyping.

B Experimental Setting

As mentioned in the article, the setup of the computational experiments closely follows Cini et al. [38] for most of the considered scenarios. We report the pre-processing steps here for completeness.

B.1 Datasets

Table 3 and the following provide relevant additional information on each dataset.

Table 3. Statistics of Datasets Used in the Experiments

Datasets	Type	Time steps	Nodes	Edges	Rate
GPVAR-G	Undirected	30,000	120	199	N/A
GPVAR-L	Undirected	30,000	120	199	N/A
METR-LA	Directed	34,272	207	1515	5 minutes
PEMS-BAY	Directed	52,128	325	2369	5 minutes
CER-E	Directed	25,728	485	4365	30 minutes
AQI	Undirected	8,760	437	2699	1 hour

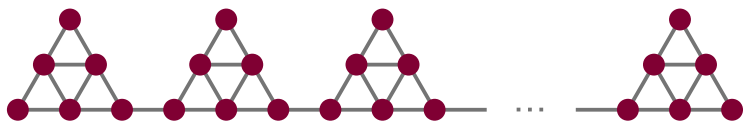

Fig. 2. GPVAR community graph. We used a graph with 20 communities resulting in a network with 120 nodes.

[9] https://github.com/jdb78/pytorch-forecasting

GPVAR. As already mentioned for the GPVAR datasets we follow the same setting reported in [38]. In particular, the parameters of the spatiotemporal process are set as

$$\Theta = \begin{bmatrix} 2.5 & -2.0 & -0.5 \\ 1.0 & 3.0 & 0.0 \end{bmatrix}, \qquad \boldsymbol{a}, \boldsymbol{b} \sim \mathcal{U}(-2, 2),$$
$$\eta \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\sigma^2)), \quad \sigma = 0.4.$$

The graph topology used to generate the data is the community graph shown in Figure 2. In particular, we considered a network with 120 nodes with 20 communities and then added self-loops by setting the diagonal of the corresponding adjacency matrix to 1.

Benchmarks. We normalize the target variable to have zero mean and unit variance on the training set. The adjacency matrix for each dataset is obtained as discussed in Section 9 by following previous works [36, 67, 100, 116]. We use as exogenous variables sinusoidal functions encoding the time of the day and, for each dataset excluding EngRAD, one-hot encodings of the day of the week. We split datasets into windows of W time steps, and train the models to predict the next H observations. Unless otherwise stated, the obtained windows are sequentially split into 70%/10%/20% partitions for training, validation, and testing, respectively. In the following, we report detailed information for experiments on each dataset.

METR-LA & PEMS-BAY Window and horizon length are set as $W = 12$ and $H = 12$. For METR-LA, given a large number of missing values, we add as an additional exogenous variable the binary mask introduced in Section 3.2.

CER-E Window and horizon length are set as $W = 48$ and $H = 6$.

AQI Window and horizon length are set as $W = 24$ and $H = 3$. We use the same data splits of previous works for training and evaluation Cini et al. [36], Yi et al. [169].

EngRAD Window and horizon length are set as $W = 24$ and $H = 6$. We use the other weather variables from the dataset, along with sinusoidal encodings of the time of the year, as additional exogenous inputs. We normalize temperature values to have zero mean and unit variance. We do not compute loss and metrics on time steps with zero radiance and follow the protocol of previous work Marisca et al. [115] to obtain training/validation/testing folds.

Hyperparameters. The reference TTS architectures are implemented by a single-layer GRU followed by 2 message-passing layers. GCRNNs have a single layer as well. For the benchmark datasets, the number of neurons in each layer is set to 64 and the embedding size to 32 for all the reference architectures and the RNN baselines. We train with early stopping for a maximum of 200 epochs with the Adam optimizer [89] and a learning rate of 0.003 divided by four every 50 epochs. In each training epoch, we randomly sample without replacement 300 batches of size 64 from the training set. We reduce both batch size and size of the hidden representations to 32 when the model exceeds the available GPU memory capacity (approximately 24 GB). For GPVAR, we use 16 and 8 as hidden and embedding sizes, respectively, and 128 as the batch size. We set 0.01 as the initial learning rate, which is halved every 50 epochs.

Received 9 February 2024; revised 3 March 2025; accepted 15 May 2025