Graph-based Forecasting with Missing Data through Spatiotemporal Downsampling

Ivan Marisca[1]   Cesare Alippi[1 2]   Filippo Maria Bianchi[3 4]

Abstract

Given a set of synchronous time series, each associated with a sensor-point in space and characterized by inter-series relationships, the problem of spatiotemporal forecasting consists of predicting future observations for each point. Spatiotemporal graph neural networks achieve striking results by representing the relationships across time series as a graph. Nonetheless, most existing methods rely on the often unrealistic assumption that inputs are always available and fail to capture hidden spatiotemporal dynamics when part of the data is missing. In this work, we tackle this problem through hierarchical spatiotemporal downsampling. The input time series are progressively coarsened over time and space, obtaining a pool of representations that capture heterogeneous temporal and spatial dynamics. Conditioned on observations and missing data patterns, such representations are combined by an interpretable attention mechanism to generate the forecasts. Our approach outperforms state-of-the-art methods on synthetic and real-world benchmarks under different missing data distributions, particularly in the presence of contiguous blocks of missing values.

1. Introduction

Time-series analysis and forecasting often deal with high-dimensional data acquired by sensor networks (SNs), a broad term for systems that collect (multivariate) measurements over time at different spatial locations. Examples include systems monitoring air quality, where each sensor records air pollutants' concentrations, or traffic, where
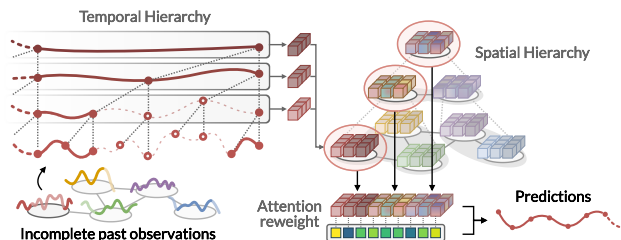

Figure 1. Overview of the proposed framework. The hierarchical design allows us to learn a pool of multi-scale spatiotemporal representations. Conditioned on the data and the missing value pattern, the attention mechanism dynamically combines representation from different scales to compute the predictions.

sensors track vehicles' flow or speed. Usually, data are sampled regularly over time and synchronously across the sensors, which are often characterized by strong correlations and dependencies between each other, i.e., across the *spatial* dimension. For this reason, a prominent deep learning approach is to consider the time series and their relationships as graphs and to process them with architectures that combine graph neural networks (GNNs) (Battaglia et al., 2018; Bronstein et al., 2021) with sequence-processing operators (Hochreiter & Schmidhuber, 1997; Borovykh et al., 2017). These architectures are known as spatiotemporal graph neural networks (STGNNs) (Jin et al., 2023).

A notable limit of most existing STGNNs is the assumption that inputs are complete and regular sequences. However, real-world SNs are prone to failures and faults, resulting eventually in missing values in the collected time series. When missing data occurs randomly and sporadically, the localized processing imposed by the inductive biases in STGNNs acts as an effective regularization, exploiting observations close in time and space to the missing one (Cini et al., 2022). Challenges arise when data are missing in larger and contiguous blocks, with gaps that occur in consecutive time steps and are spatially proximate. In SNs, this might be due to a sensor failure lasting for multiple time lags or problems affecting a whole portion of the network. In such scenarios, reaching valid observations that may be significantly distant in time and space, yet relevant for capturing the underlying dynamics, would require addi-

[1]IDSIA USI-SUPSI, Università della Svizzera italiana [2]Politecnico di Milano [3]Dept. of Mathematics and Statistics, UiT the Arctic University of Norway [4]NORCE, Norwegian Research Centre AS. Correspondence to: Ivan Marisca <ivan.marisca@usi.ch>, Filippo Maria Bianchi <filippo.m.bianchi@uit.no>.

tional layers of processing (Marisca et al., 2022). Nonetheless, deep processing may attenuate faster dynamics, impairing the network's ability to rely on local information, if present (Rusch et al., 2023). The necessity to expand the network's receptive field should be, therefore, contingent on the availability of input data. As such, it is crucial to tailor the processing strategy according to the dynamics in the input data and the specific patterns of missing information.

In this work, we propose a deep learning framework for graph-based forecasting of time series with missing data that computes representations at multiple spatiotemporal scales and weighs them conditioned on the observations and missing data pattern at hand. To this end, we rely on operators that progressively reduce the data granularity across both temporal and spatial dimensions (see Fig. 1). In time, we interleave downsampling within temporal processing layers, handling varying noise levels and isolating specific temporal dynamics. In space, we use graph pooling (Grattarola et al., 2024) to obtain a hierarchy of coarsened graphs that gradually distill the global information necessary for compensating localized gaps in the data. The proposed framework adopts a *time-then-space* (Gao & Ribeiro, 2022) hierarchical design, which efficiently handles representations at multiple scales by increasing the receptive field while limiting the number of parameters and the amount of computation. The hierarchical representations learned by our model are then combined by a soft attention mechanism (Vaswani et al., 2017), whose scores offer a natural interpretability tool to inspect the model's behavior in function of the data.

We compare our approach against state-of-the-art methods in both synthetic and real-world benchmark datasets, showing remarkable improvements in efficiency and performance, particularly with large blocks of missing values. Notably, we introduce two new datasets and an experimental setting specifically designed to reflect typical missing data patterns in the spatiotemporal domain. This contribution addresses a substantial gap in the existing literature by providing a controlled environment for testing the performance and understanding the behavior of complex spatiotemporal models.

2. Preliminaries and Problem Formulation

We represent each sensor in a SN (i.e., a point in space) as a node in the set $\mathcal{V}$, with $|\mathcal{V}| = N$. We model dyadic and possibly asymmetric relationships between sensors with a weighted adjacency matrix $\boldsymbol{A} \in \mathbb{R}_{\geq 0}^{N \times N}$, where each nonzero entry a^{ij} is the nonnegative weight of the directed edge from the i-th to the j-th node. We denote by $\boldsymbol{x}_t^i \in \mathbb{R}^{d_x}$, the d_x-dimensional observation collected by the i-th node at time step t, with $\boldsymbol{X}_t \in \mathbb{R}^{N \times d_x}$ representing all the observations collected synchronously in the SN. We use the notation $\boldsymbol{X}_{t:t+T}$ to indicate the sequence of T observations

in the time interval $[t, t + T)$. We represent node-level exogenous variable with matrix $\boldsymbol{U}_t \in \mathbb{R}^{N \times d_u}$ (e.g., date/time information, external events). To model the presence of missing data, we associate every node observation with a binary mask $\boldsymbol{m}_t^i \in \{0, 1\}^{d_x}$, whose elements are nonzero whenever the corresponding channel in $\boldsymbol{x}_t^i$ is valid. Notably, we do not make any assumption on the missing data distribution and we consider $\boldsymbol{u}_t^i$ always observed, regardless of $\boldsymbol{m}_t^i$. Finally, we use the tuple $\mathcal{G}_t = \langle \boldsymbol{X}_t, \boldsymbol{M}_t, \boldsymbol{U}_t, \boldsymbol{A} \rangle$ to denote all information available at time step t.

Given a *window* $\mathcal{G}_{t-W:t}$ of W past observations, the problem of *spatiotemporal forecasting* consists of predicting an *horizon* of H future observations for each node $i \in \mathcal{V}$:

$$\hat{\boldsymbol{x}}_{t:t+H}^i = f(\mathcal{G}_{t-W:t}). \tag{1}$$

As observations might also be missing in the ground-truth data, to measure forecasting accuracy we average an element-wise loss function ℓ (e.g., absolute or squared error) over only valid values, i.e.,

$$\mathcal{L}_{t:t+H} = \sum_{h=t}^{t+H-1} \sum_{i=1}^{N} \frac{\left\| \boldsymbol{m}_h^i \odot \ell\left(\hat{\boldsymbol{x}}_h^i, \boldsymbol{x}_h^i\right) \right\|_1}{\left\| \boldsymbol{m}_h^i \right\|_1}, \tag{2}$$

where $\odot$ is the Hadamard product.

2.1. Spatiotemporal Message Passing

The cornerstone operator of an STGNN is the spatiotemporal message-passing (STMP) layer (Cini et al., 2023d), which computes nodes' features at the l-th layer as:

$$\boldsymbol{x}_t^{i,l} = \gamma^l \left(\boldsymbol{x}_{\leq t}^{i,l-1}, \underset{j \in \mathcal{N}(i)}{\text{AGGR}} \phi^l \left(\boldsymbol{x}_{\leq t}^{i,l-1}, \boldsymbol{x}_{\leq t}^{j,l-1}, a^{ji} \right) \right) \tag{3}$$

where AGGR is a differentiable, permutation invariant *aggregation* function, e.g., sum or mean, and γ^l and ϕ^l are differentiable *update* and *message* functions, respectively. Whenever γ^l and ϕ^l are such that temporal and spatial processing cannot be factorized in two distinguished operations, the STGNN is said to follow a *time-and-space* (T&S) paradigm (Gao & Ribeiro, 2022; Cini et al., 2023c). In the *time-then-space* (TTS) approach, instead, the input sequences are first encoded in a vector by temporal message-passing (TMP) layers before being propagated on the graph by spatial message-passing (SMP) layers.

We use the term TMP to refer broadly to any deep learning operator enabling the exchange of information along the temporal dimension. Most TMP operators can be categorized as recurrent or convolutional. Recurrent neural networks (RNNs) (Elman, 1990) process sequential data of varying lengths in a recursive fashion, by maintaining a memory of previous inputs:

$$\boldsymbol{x}_t^{i,l} = \gamma^l \left(\boldsymbol{x}_t^{i,l-1}, \phi^l \left(\boldsymbol{x}_t^{i,l-1}, \boldsymbol{x}_{t-1}^{i,l} \right) \right). \tag{4}$$

In modern RNNs, gating mechanisms are used to cope with vanishing (or exploding) gradients that hinder learning long-range dependencies (Hochreiter & Schmidhuber, 1997; Cho et al., 2014). Convolutional TMP operators, instead, learn causal filters conditioned on a sequence of previous observations:

$$\boldsymbol{x}_t^{i,l} = \gamma^l \left(\boldsymbol{x}_t^{i,l-1}, \underset{k>0}{\text{AGGR}} \, \phi^l \left(\boldsymbol{x}_t^{i,l-1}, \boldsymbol{x}_{t-k}^{i,l-1} \right) \right). \quad (5)$$

Temporal convolutional networks (TCNs) (Borovykh et al., 2017; Oord et al., 2016) and attention-based methods (Zhou et al., 2021) follow this approach. An advantage of convolutional TMP is its ability to be executed in parallel along the temporal axis (Eq. 5), offering computational benefits over recurrent TMP (Eq. 4).

Instead, the SMP operator (Gilmer et al., 2017) can be described as

$$\boldsymbol{x}_t^{i,l} = \gamma^l \left(\boldsymbol{x}_t^{i,l-1}, \underset{j \in \mathcal{N}(i)}{\text{AGGR}} \, \phi^l \left(\boldsymbol{x}_t^{i,l-1}, \boldsymbol{x}_t^{j,l-1}, a^{ji} \right) \right). \quad (6)$$

If the messages depend on the receiver node's features $\boldsymbol{x}_t^{i,l-1}$ the SMP operator is called *anisotropic* (Dwivedi et al., 2023). Conversely, if the message function depends only on the source node's features $\boldsymbol{x}_t^{j,l-1}$ and the edge weight a^{ji}, the SMP operator is said to be *isotropic* or *convolutional* (Bronstein et al., 2021).

Notably, the TMP operators (Eq. 4–5) and the SMP operator (Eq. 6) are specific instances of the STMP operator (Eq. 3), underlining the factorization of operations within TTS models compared to the T&S approach. Since TTS models perform SMP – an onerous operation – on a single graph rather than a sequence of graphs, they are more efficient than T&S models. Nonetheless, their uncoupled temporal and spatial processing reduces the flexibility in how information is propagated, compared to the T&S processing (Cini et al., 2023c). In the latter, indeed, it is possible to gradually account for more information while processing the temporal dimension and allow the receptive field to grow with the sequence length.

2.2. Spatiotemporal Downsampling

Downsampling in temporal data is a common operation, often used to reduce the sample complexity or filter out noisy measurements (Harris, 2022). In classical signal processing, it is implemented by applying a low-pass filter and then keeping only 1-every-k samples, with k being the *downsampling factor* (Strang & Nguyen, 1996). This approach is replicated by strided operations in TMP (Yu & Koltun, 2016; Oord et al., 2016; Chang et al., 2017), which exploit the structural regularity of temporal data. More generally, downsampling a sequence from W_{l-1} to W_l time steps can be conveniently expressed by a *temporal downsampling*

matrix $\boldsymbol{T}_l \in \mathbb{R}^{W_l \times W_{l-1}}$. For example, $\boldsymbol{T}_l = [\boldsymbol{I}_{W_{l-1}}]_{::k}$, i.e., an identity matrix without the rows associated with the decimated time steps, can be applied to keep the samples associated with every k-th time step.

Being non-Euclidean structures, the concept of downsampling for graphs is less straightforward and tied to the procedure for graph coarsening. In the GNNs literature, the latter is known as *graph pooling*. Given a graph $\boldsymbol{A}^{\langle k-1 \rangle} \in \mathbb{R}_{\geq 0}^{N_{k-1} \times N_{k-1}}$ with features $\boldsymbol{X}^{\langle k-1 \rangle} \in \mathbb{R}^{N_{k-1} \times d_x}$ on the nodes, the Select-Reduce-Connect (SRC) framework by Grattarola et al. (2024) expresses a graph pooling operator $\texttt{POOL} : (\boldsymbol{A}^{\langle k-1 \rangle}, \boldsymbol{X}^{\langle k-1 \rangle}) \mapsto (\boldsymbol{A}^{\langle k \rangle}, \boldsymbol{X}^{\langle k \rangle})$ as the combination of three functions:

- $\texttt{SEL} : (\boldsymbol{A}^{\langle k-1 \rangle}, \boldsymbol{X}^{\langle k-1 \rangle}) \mapsto \boldsymbol{S}_k \in \mathbb{R}^{N_k \times N_{k-1}}$, defines how to aggregate the N_{k-1} nodes in the input graph into N_k *supernodes*.

- $\texttt{RED} : (\boldsymbol{X}^{\langle k-1 \rangle}, \boldsymbol{S}_k) \mapsto \boldsymbol{X}^{\langle k \rangle} \in \mathbb{R}^{N_k \times d_x}$, creates the supernode features by combining the features of the nodes assigned to the same supernode. A common way to implement $\texttt{RED}$ is $\boldsymbol{X}^{\langle k \rangle} = \boldsymbol{S}_k \boldsymbol{X}^{\langle k-1 \rangle}$.

- $\texttt{CON} : (\boldsymbol{A}^{\langle k-1 \rangle}, \boldsymbol{S}_k) \mapsto \boldsymbol{A}^{\langle k \rangle} \in \mathbb{R}_{\geq 0}^{N_k \times N_k}$, generates the edges (and, potentially, the edge features) by connecting the supernodes. A typical $\texttt{CON}$ is $\boldsymbol{A}^{\langle k \rangle} = \boldsymbol{S}_k \boldsymbol{A}^{\langle k-1 \rangle} \boldsymbol{S}_k^\top$.

A fourth function is used to *lift*, i.e., upsample, supernode features to the associated nodes in the original graph:

- $\texttt{LFT} : (\boldsymbol{X}^{\langle k \rangle}, \boldsymbol{S}_k) \mapsto \widetilde{\boldsymbol{X}}^{\langle k-1 \rangle} \in \mathbb{R}^{N_{k-1} \times d_x}$, can be implemented as $\widetilde{\boldsymbol{X}}^{\langle k-1 \rangle} = \boldsymbol{S}_k^+ \boldsymbol{X}^{\langle k \rangle}$, where $\boldsymbol{S}_k^+$ is the pseudo-inverse of $\boldsymbol{S}_k$.

As for TMP and SMP, there is a strong analogy also between downsampling in time and space. Notably, the selection matrix $\boldsymbol{S}_k$, hereinafter called *spatial downsampling matrix*, plays the same role as $\boldsymbol{T}_l$. Both matrices, indeed, reduce the input dimensionality conditioned on the underlying structure of the data, which in the temporal domain can be expressed as the path graph connecting the time steps. However, irregularities in arbitrary graphs make it challenging to define concepts like "1-every-k". An interpretation is given by the k-MIS method (Bacciu et al., 2023), which relies on the concept of maximal k-independent sets to keep in the pooled graph the nodes that cover uniformly the k-th power graph. Such a symmetry between the spatial and temporal operators will be pivotal for the design of our architecture.

2.3. Spatiotemporal Missing Data Distributions

Following previous works (Yi et al., 2016; Cini et al., 2022), we categorize missing data patterns according to the conditional distribution $p\left(\boldsymbol{m}_t^i \mid \boldsymbol{M}_{\leq t}\right)$. We call *point missing* the case where the probability $p\left(\boldsymbol{m}_t^i = \boldsymbol{0}\right)$ of a datum being missing at a given node and time step is unconditioned and
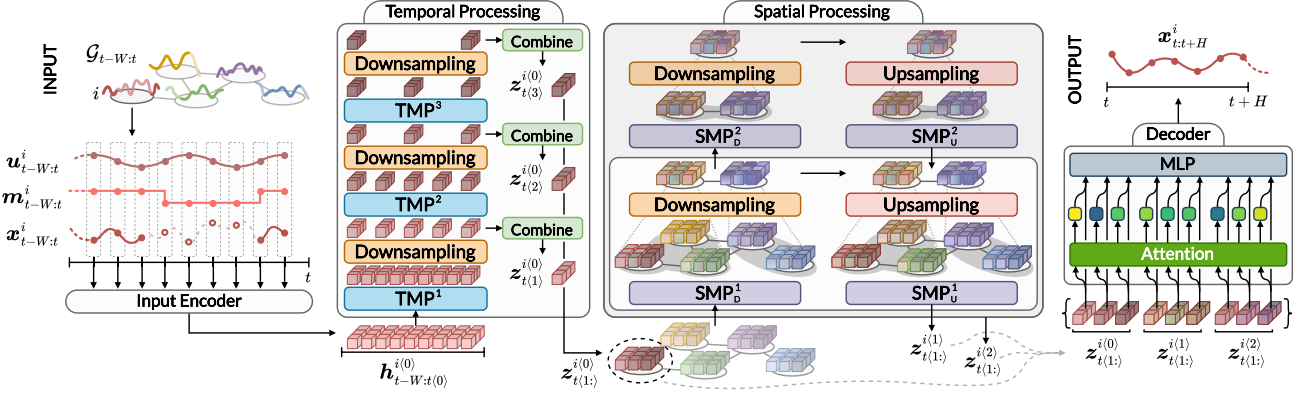
Figure 2. Overview of the proposed architecture. Given input data $\mathcal{G}_{t-W:t}$, all information associated with every i-th node and time step t is encoded in vectors $\boldsymbol{h}_{t\langle 0\rangle}^{i\langle 0\rangle}$, then processed node-wise along the temporal dimension by alternating TMP and downsampling. After each l-th layer, the sequences are combined in a single vector $\boldsymbol{z}_{t\langle l\rangle}^{i\langle 0\rangle}$, which is then processed along the spatial dimension by alternating SMP and pooling. Representations at each k-th pooling layer are then recursively un-pooled up to the initial node level, obtaining $\boldsymbol{z}_{t\langle 1:\rangle}^{i\langle k\rangle}$. Finally, the $L(K+1)$ encodings $\boldsymbol{z}_{t\langle 1:\rangle}^{i\langle 0:\rangle}$ are combined through an attention mechanism and fed to an MLP to obtain the predictions.

constant across nodes and time steps, i.e.,

$$p\left(\boldsymbol{m}_t^i\right) = \mathcal{B}(1-\eta) \quad \forall\, i, t \tag{7}$$

where $1 - \eta$ is the mean of the Bernoulli distribution. This setting is also known as *general* missing (Rubin, 1976), as realizations of the mask have a haphazard pattern. Very often in SNs, instead, $p\left(\boldsymbol{m}_t^i\right)$ does depend on realizations of the missing data distribution at other nodes, time steps, or a combination of them (e.g., due to faults or blackouts). We refer to this setting as *block missing* and decline it differently according to the dimensions of interest. In *temporal block missing*, $p\left(\boldsymbol{m}_t^i\right)$ depends on the realization of the missing data distribution at the previous time step, i.e.,

$$p\left(\boldsymbol{m}_t^i \mid \boldsymbol{m}_{t-1}^i\right) \neq p\left(\boldsymbol{m}_t^i\right). \tag{8}$$

Similarly, in *spatial block missing*, $p\left(\boldsymbol{m}_t^i\right)$ is conditioned on the simultaneous realizations at neighboring nodes, i.e.,

$$p\left(\boldsymbol{m}_t^i \mid \left\{\boldsymbol{m}_t^j\right\}^{j\in\mathcal{N}(i)}\right) \neq p\left(\boldsymbol{m}_t^i\right). \tag{9}$$

The *spatiotemporal block missing* combines Eq. 8–9 as:

$$p\left(\boldsymbol{m}_t^i \mid \boldsymbol{m}_{t-1}^i, \left\{\boldsymbol{m}_t^j\right\}^{j\in\mathcal{N}(i)}\right) \neq p\left(\boldsymbol{m}_t^i\right). \tag{10}$$

Note that in Eq. 7–10 we considered a simplified case with a single channel in the observations, i.e., $d_x = 1$. Instead, in the multivariate case $d_x > 1$, we assume the missing data distribution of each channel to be independent of the others.

3. Proposed Architecture

The dynamics in spatiotemporal data are governed by relationships spanning both the temporal and spatial dimensions.

Operators that exploit the temporal or spatial structure underlying the data during processing, like those in Sec. 2.1, capture such dynamics effectively. Missing values, however, pose a serious challenge in identifying the correct dynamics. In addition, if the distribution of missing values is unknown, it is crucial to adaptively focus on different spatiotemporal scales conditioned on the input data. When a whole block of data is missing at a node (i.e., sensor), it could be beneficial to consider recent observations of the spatial neighbors. Conversely, when data are missing at neighboring nodes, data far back in time might be more informative.

In this section, we introduce Hierarchical Downsampling Time-Then-Space (HD-TTS), an architecture for graph-based forecasting of spatiotemporal data with missing values following arbitrary patterns. To learn spatiotemporal representations at different scales, we rely on hierarchical downsampling, which progressively reduces the size of the input and enables efficient learning of long-range dependencies. In time, HD-TTS learns multiple – yet limited – representations from the input sequences, each at a different temporal scale. Similarly, we process the spatial dimension by propagating messages along a hierarchy of pre-computed coarsened graphs. Notably, our approach combines the advantages of the T&S and TTS paradigms, enabling adaptive expansion of the receptive field while keeping the computational and memory complexity under control.

The key components of HD-TTS are (1) an **input encoder**, (2) a **temporal processing** module, (3) a **spatial processing** module, and (4) an adaptive **decoder**. The first three blocks extract hierarchical representations of the input at different spatial and temporal scales, while the last block reweighs the representations and outputs the predictions. The whole architecture is trained end-to-end to minimize the forecast-

ing error over the prediction horizon (Eq. 2). Notably, our approach does not require missing data imputation as a pre-processing step. The latter might introduce a bias in the data and is generally cumbersome, as it turns the forecasting task into a two-step procedure. Also, domain knowledge is usually required to select the proper imputation technique. In contrast, HD-TTS handles missing data by automatically reweighing the different spatiotemporal dynamics present in the data while learning to solve the downstream task.

Fig. 2 shows an overview of the architecture. In the figure and the following, we use the superscript and subscript indices $\cdot^{\langle k \rangle}_{\langle l \rangle}$ to refer to the k-th spatial and l-th temporal scale, respectively, with $k = l = 0$ corresponding to the input scales.

Input encoder The input encoder combines the information associated with the i-th node at generic time step t, i.e., observations $\boldsymbol{x}^i_t$, exogenous variables $\boldsymbol{u}^i_t$, and mask $\boldsymbol{m}^i_t$ (see Sec. 2), in an embedding vector $\boldsymbol{h}^{i\langle 0 \rangle}_{t\langle 0 \rangle} \in \mathbb{R}^{d_h}$ with a multilayer perceptron (MLP):

$$\boldsymbol{h}^{i\langle 0 \rangle}_{t\langle 0 \rangle} = \mathrm{MLP}\left(\boldsymbol{x}^i_t, \boldsymbol{u}^i_t, \boldsymbol{m}^i_t, \boldsymbol{\theta}^i \right). \qquad (11)$$

Here, d_h is the size of all latent representations learned within our model and $\boldsymbol{\theta}^i$ is a vector of node-specific trainable parameters that facilitates node indetification (Cini et al., 2023d). Missing values are imputed using the last observed value. Obtained representations are then refined by downstream components exploiting temporal and spatial dependencies.

Temporal processing The temporal processing module acts on $\boldsymbol{h}^{i\langle 0 \rangle}_{t-W:t\langle 0 \rangle}$ to generate, for every i-th node, L representations $\{ \boldsymbol{z}^{i\langle 0 \rangle}_{t\langle l \rangle} \}_{l=1,\dots,L}$, each associated with a different temporal scale. In particular, the l-th processing layer takes as input the sequence $\boldsymbol{h}^{i\langle 0 \rangle}_{t-W_{l-1}:t\langle l-1 \rangle}$ and outputs a new sequence of updated encodings decimated by a factor of d. Hence, the output sequence has length $W_l = \left\lceil \frac{W_{l-1}}{d} \right\rceil$, with $W_0 = W$ being the original time series' length. We can write the operations performed by layer l as

$$\boldsymbol{h}^{i\langle 0 \rangle}_{t-W_l:t\langle l \rangle} = \boldsymbol{T}_l \left(\mathrm{TMP}^l \left(\boldsymbol{h}^{i\langle 0 \rangle}_{t-W_{l-1}:t\langle l-1 \rangle} \right) \right), \qquad (12)$$

where TMP^l is the TMP operator defined in Sec. 2.1 acting at temporal scale $l-1$ and $\boldsymbol{T}_l \in \mathbb{R}^{W_l \times W_{l-1}}$ is the temporal downsampling matrix defined in Sec. 2.2. The downsampling operation progressively expands the temporal receptive field, allowing us to capture increasingly slower temporal dynamics in the input.

We then combine the sequences $\boldsymbol{h}^{i\langle 0 \rangle}_{t-W_l:t\langle l \rangle}$ at each layer l into a single representation $\boldsymbol{z}^{i\langle 0 \rangle}_{t\langle l \rangle} \in \mathbb{R}^{d_h}$, i.e.,

$$\boldsymbol{z}^{i\langle 0 \rangle}_{t\langle l \rangle} = \mathrm{COMBINE}\left(\boldsymbol{h}^{i\langle 0 \rangle}_{t-W_l:t\langle l \rangle} \right). \qquad (13)$$
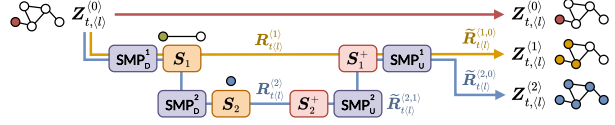

Figure 3. Details of the spatial processing procedure.

A simple implementation is to return the encoding associated with the last time step, i.e., $\boldsymbol{z}^{i\langle 0 \rangle}_{t\langle l \rangle} = \boldsymbol{h}^{i\langle 0 \rangle}_{t-1\langle l \rangle}$.

Spatial processing The set of multi-scale temporal encodings $\{ \boldsymbol{z}^{i\langle 0 \rangle}_{t\langle l \rangle} \}_{l=1,\dots,L}$ is processed spatially with an analogous hierarchical approach. In particular, representations associated with the k-th spatial scale are obtained by applying k times a combination of an SMP layer followed by the graph pooling operation described in (Sec. 2.2):

$$\boldsymbol{R}^{\langle k \rangle}_{t\langle l \rangle} = \boldsymbol{S}_k \left(\mathrm{SMP}^k_{\mathrm{D}} \left(\boldsymbol{R}^{\langle k-1 \rangle}_{t\langle l \rangle}, \boldsymbol{A}^{\langle k-1 \rangle} \right) \right), \qquad (14)$$

where $\boldsymbol{S}_k \in \mathbb{R}^{N_k \times N_{k-1}}$ is the spatial downsampling matrix generated by the pooling's SEL function, $\mathrm{SMP}^k_{\mathrm{D}}$ is the SMP operator applied at k-th layer before downsampling, $\boldsymbol{A}^{\langle 0 \rangle} = \boldsymbol{A}$ and $\boldsymbol{R}^{\langle 0 \rangle}_{t\langle l \rangle} = \boldsymbol{Z}^{\langle 0 \rangle}_{t\langle l \rangle}$. At each spatial resolution level k, the encodings $\boldsymbol{R}^{\langle k \rangle}_{t\langle l \rangle} \in \mathbb{R}^{N_k \times d_h}$ are associated with supernodes rather than the nodes in the original graph. Since the predictions must ultimately be computed for the nodes in the original graph, the coarsened spatial representations must be brought back to the initial spatial resolution. For this reason, we mirror the operations carried out in Eq. 14 and recursively lift supernodes' encodings back to the associated nodes in previous layers. Starting from the coarsened representation $\widetilde{\boldsymbol{R}}^{\langle k,k \rangle}_{t\langle l \rangle} = \boldsymbol{R}^{\langle k \rangle}_{t\langle l \rangle}$, the lifting step from layer j to $j-1$ can be described as

$$\widetilde{\boldsymbol{R}}^{\langle k,j-1 \rangle}_{t\langle l \rangle} = \mathrm{SMP}^j_{\mathrm{U}} \left(\boldsymbol{S}^+_j \left(\widetilde{\boldsymbol{R}}^{\langle k,j \rangle}_{t\langle l \rangle} \right), \boldsymbol{A}^{\langle j-1 \rangle} \right). \qquad (15)$$

After k recursive application of Eq. 15, the representations associated with the k-th spatial scale $\boldsymbol{R}^{\langle k \rangle}_{t\langle l \rangle}$ are propagated to the original graph and we assign $\boldsymbol{Z}^{\langle k \rangle}_{t\langle l \rangle} = \widetilde{\boldsymbol{R}}^{\langle k,0 \rangle}_{t\langle l \rangle}$. The details are depicted in (Fig. 3).

Decoder The $L(K+1)$ representations $\left\{ \boldsymbol{Z}^{\langle k \rangle}_{t\langle l \rangle} \right\}^{k=0,\dots,K}_{l=1,\dots,L}$ obtained in previous steps are associated with different temporal and spatial scales. To condition the importance of a representation to the prediction based on the input dynamics and the missing data pattern, we first compute an adaptive weight $\alpha^{i\langle k \rangle}_{t\langle l \rangle} \in [0,1]$ for each representation as

$$\left\{ \alpha^{i\langle k \rangle}_{t\langle l \rangle} \right\} = \mathrm{softmax} \left\{ \boldsymbol{z}^{i\langle k \rangle}_{t\langle l \rangle} \Theta_\alpha \right\}^{k=0,\dots,K}_{l=1,\dots,L}, \qquad (16)$$

with $\Theta_\alpha \in \mathbb{R}^{d_h \times 1}$ being a matrix of learnable parameters. The weights can be interpreted as soft attention coefficients that select the appropriate spatiotemporal scale for computing the predictions. As we show in Sec. 5.3, they offer a tool to analyze how the model focuses on different scales according to the dynamics and observability of the input.

The representations are then weighed and combined into a single final representation $\tilde{z}_t^i \in \mathbb{R}^{d_h}$ as

$$\tilde{z}_t^i = \sum_{l=1}^{L} \sum_{k=0}^{K} \alpha_{t\langle l \rangle}^{i\langle k \rangle} z_{t\langle l \rangle}^{i\langle k \rangle}. \tag{17}$$

Finally, we use a multi-step MLP decoder to obtain the predictions for each i-th node:

$$\hat{x}_{t:t+H}^i = \text{MLP}\left(\tilde{z}_t^i\right). \tag{18}$$

The pool of multi-scale representations and the adaptive reweighting enable more complex decoding strategies; we discuss them in Appendix C.

3.1. Implementation Details

While the proposed framework is general and different TMP, SMP, and downsampling operators could be chosen, in this section we discuss the specific choices we made for our model. As TMP, we use Gated Recurrent Units (GRUs) (Cho et al., 2014) combined with the standard 1-every-k temporal downsampling, which results in something similar to a dilated RNN (Chang et al., 2017). Referring to Eq. 6, we consider two variants with isotropic and anisotropic functions ϕ to compute the messages. As isotropic SMP, we choose the diffusion-convolutional operator (Atwood & Towsley, 2016), while for the anisotropic variant, we use the operator introduced by Cini et al. (2023d). We use the sum as the aggregation function (AGGR) and we add aggregated messages to the source-node features $x_t^{i,l-1}$ after an affine transformation (update function γ). We rely on the k-MIS pooling method (presented in Sec. 2.2) to obtain spatial downsampling matrices $S_k \in [0, 1]^{N_k \times N_{k-1}}$ that assign each node to exactly one supernode. Notably, k-MIS is expressive, i.e., if two graphs are distinguishable, then the pooled graphs remain distinguishable (Bianchi & Lachi, 2023). For RED, CON and LFT, we use, respectively,

$$X^{\langle k \rangle} = S_k X^{\langle k-1 \rangle},$$
$$A^{\langle k \rangle} = S_k A^{\langle k-1 \rangle} S_k^\top,$$
$$\widetilde{X}^{\langle k-1 \rangle} = S_k^+ X^{\langle k \rangle}.$$

More details on the implementation are in Appendix C.

Analogy with filterbanks The hierarchical processing scheme presented in Sec. 3 is inspired by the design of filterbanks in digital and graph signal processing (Strang &

Nguyen, 1996; Tremblay & Borgnat, 2016; Tremblay et al., 2018). Graph filterbanks often rely on specific node decimation operators that discard approximately half of the node at each application (Shuman et al., 2015). Despite completely removing some nodes, when such techniques are paired with SMP, the original graph signal can be reconstructed without losses from the pooled one. Such an approach has been adapted to the graph pooling framework (Bianchi et al., 2020) and represents an alternative to k-MIS pooling.

3.2. Scalability

Maintaining multiple representations adds a computational burden to the already expensive operations performed by STGNNs for standard spatial and temporal processing. For this reason, we adopted a series of architectural design choices to reduce the computational and memory complexity. As previously discussed, we follow the TTS paradigm (Sec. 2.1), which allows us to perform SMP operations on a single static graph, regardless of the length of the sequence being processed. Additionally, by integrating downsampling operations, the amount of computation reduces progressively with the layers, as the number of time steps and nodes is decimated. Temporal processing, for instance, depends on the dilation factor chosen rather than the number of layers, scaling as $\mathcal{O}\left(NW\left(d/(d-1)\right)\right)$ in contrast to $\mathcal{O}(NWL)$. Notably, this regularization has a positive impact also on the spatial and temporal receptive fields, which grow exponentially faster than standard TTS architectures, ceteris paribus.

The chosen k-MIS pooling operator belongs to the class of sparse and non-trainable pooling methods, which have a low memory footprint and do not require additional computation. A pooling operator is said to be *sparse* if every node is assigned to at most one supernode. Additionally, pooling operators whose selection matrix S_k is learned together with the model's weights are called *trainable* (*non-trainable* otherwise). The use of non-trainable methods allows us to compute the downsampling matrices once in a pre-processing step, making the spatial downsampling computationally comparable to the temporal one. While it can also act as a regularization for large models with high capacity, this lack of flexibility might introduce a too restrictive bias for its strong reliance on graph topology. In Appendix F, we show an empirical example.

4. Related Work

Prominent representatives of STGNNs for forecasting include graph-based RNNs, which integrate SMP within temporal processing (Seo et al., 2018; Li et al., 2018; Zhang et al., 2018; Yu et al., 2019; Bai et al., 2020), and convolutional approaches, which apply convolutions or attention mechanisms across temporal and spatial dimensions (Yu

et al., 2018; Wu et al., 2019; 2022). Remarkably, none of these methods explicitly consider missing data in the input. GRIN (Cini et al., 2022) and other STGNNs (Wu et al., 2021; Marisca et al., 2022) specifically address missing data in time series with relational information but focus on imputation. Forecasting with missing data, instead, is explored by a limited number of works in the context of traffic analytics (Zhong et al., 2021; Wang et al., 2023).

Numerous techniques exist for handling missing data, such as GRU-D (Che et al., 2018) which simultaneously imputes missing values while processing the sequence, along with approaches for multivariate time classification (Shukla & Marlin, 2020), anomaly detection (Bianchi et al., 2019), and forecasting (Tang et al., 2020). Yet, none of these methods explicitly model the spatial dependencies in the data.

Pyramidal approaches to process inputs with missing (or noisy) data have been extensively studied in graph (Shuman et al., 2015; Tremblay & Borgnat, 2016) and discrete-time (Strang & Nguyen, 1996) signal processing. The idea of using a hierarchy of either temporal or spatial representations has also been adopted by deep learning models for time series classification (Cui et al., 2016), forecasting (Wang & Jing, 2022; Chen et al., 2023), and forecast reconciliation (Rangapuram et al., 2021; 2023). In particular, Cini et al. (2023a) propose an approach for hierarchical time series forecasting where learnable graph pooling operations are used to cluster the time series.

5. Experiments

In this section, we report the results of the empirical analysis of our approach in different synthetic and real-world settings. We use mean absolute error (MAE) as the figure of merit, averaged over only valid observations (Eq. 2). The code to reproduce the experiments and the instructions to download and pre-process the datasets are available online.[1]

Baselines   We choose as baselines the following state-of-the-art STGNNs: **DCRNN** (Li et al., 2018), a recurrent STGNN using the diffusion-convolutional operator; **Graph WaveNet (GWNet)** (Wu et al., 2019), a spatiotemporal convolutional residual network; **AGCRN** (Bai et al., 2020), an adaptive recurrent STGNN; the four **TTS-IMP**, **TTS-AMP**, **T&S-IMP**, and **T&S-AMP** baselines introduced by Cini et al. (2023d), which are TTS and T&S architectures equipped with isotropic or anisotropic message passing, respectively. Given the similarity of TTS-IMP and TTS-AMP with our model, except for the hierarchical processing, they represent an ablation study about the effectiveness of learning hierarchical representations in TTS architectures. Finally, we consider a **GRU** sharing the parameters across the time series. Since the baselines do not handle missing values

[1] https://github.com/marshka/hdtts

Table 1. Forecasting error (MAE) on GraphMSO with different missing data distributions. **Bold** formatting is used to mark best result in each setting. [†]Models without spatial message passing.

Model	Point (5%)	Block–T	Block–ST
GRU-D[†]	$0.385_{\pm 0.012}$	$0.670_{\pm 0.020}$	$1.081_{\pm 0.003}$
GRU-I[†]	$0.322_{\pm 0.016}$	$0.619_{\pm 0.011}$	$1.064_{\pm 0.003}$
GRIN-P	$0.163_{\pm 0.008}$	$0.392_{\pm 0.031}$	$0.895_{\pm 0.012}$
GRU[†]	$0.346_{\pm 0.027}$	$0.639_{\pm 0.011}$	$1.137_{\pm 0.008}$
DCRNN	$0.291_{\pm 0.277}$	$0.645_{\pm 0.510}$	$1.103_{\pm 0.001}$
AGCRN	$0.067_{\pm 0.004}$	$0.366_{\pm 0.013}$	$1.056_{\pm 0.012}$
GWNet	$0.089_{\pm 0.002}$	$0.340_{\pm 0.001}$	$0.955_{\pm 0.012}$
T&S-IMP	$0.118_{\pm 0.009}$	$0.323_{\pm 0.011}$	$0.935_{\pm 0.005}$
T&S-AMP	$0.063_{\pm 0.003}$	$0.293_{\pm 0.020}$	$0.868_{\pm 0.006}$
TTS-IMP	$0.113_{\pm 0.008}$	$0.271_{\pm 0.007}$	$0.697_{\pm 0.005}$
TTS-AMP	$0.096_{\pm 0.004}$	$0.251_{\pm 0.004}$	$0.669_{\pm 0.013}$
HD-TTS-IMP	$\mathbf{0.058_{\pm 0.004}}$	$\mathbf{0.247_{\pm 0.002}}$	$\mathbf{0.651_{\pm 0.023}}$
HD-TTS-AMP	$0.062_{\pm 0.002}$	$0.261_{\pm 0.009}$	$0.679_{\pm 0.005}$

in input, we impute them node-wise using the last observed value and concatenate the mask to the input as an exogenous variable. In addition, we consider approaches that are specifically designed to process time series with missing data: **GRU-I**, a GRU that imputes missing values with 1-step-ahead predictions during processing; **GRU-D** (Che et al., 2018); **GRIN-P**, the Graph Recurrent Imputation Network (Cini et al., 2022), followed by a predictive decoder. The details about the implementation of the baselines are in Appendix B.

Missing data patterns   Similar to previous works (Yi et al., 2016; Marisca et al., 2022), we mask out observations in the data with missing data patterns that simulate realistic scenarios (see Sec. 2.3). In the **Point** setting, each observation has a constant fixed probability η of being missing (Eq. 7). In the **Block–T** setting, in addition to the random point missing, we simulate sensor faults by letting sensor observations be missing for multiple consecutive steps with probability p_f (Eq. 8). The duration of each fault is independently sampled from the same uniform distribution across sensors. Finally, **Block–ST** builds upon Block–T to reproduce faults affecting a group of sensors in a localized region and is obtained by propagating the fault to all k-hop neighbors with probabilities $\left[\boldsymbol{p}_g\right]_k$ (Eq. 10). In all settings, we let the probabilities be the same for all sensors. Details on the distribution parameters can be found in Appendix B.

5.1. GraphMSO: a New Synthetic Benchmark

We introduce a new synthetic dataset, called **Graph Multiple Superimposed Oscillators (GraphMSO)**, inspired by a popular benchmark in time series forecasting (Bianchi et al., 2017). Given a graph with binary adjacency matrix

Table 2. Forecasting error (MAE) on real-world datasets with different missing data distributions. We use **bold** formatting to mark best results and N/A for runs exceeding resource capacity. [†]Models without spatial message passing.

Model	AQI		EnGRAD		PV-US			
	Original	+ Point	Block–T	Block–ST	Block–T	Block–ST	Batch/s	GPU RAM
GRU-D[†]	$18.26_{\pm0.09}$	$19.23_{\pm0.08}$	$5.29_{\pm0.05}$	$5.41_{\pm0.01}$	$4.04_{\pm0.03}$	$4.27_{\pm0.01}$	$5.07_{\pm0.00}$	9.33 GB
GRU-I[†]	$18.12_{\pm0.03}$	$19.07_{\pm0.01}$	$5.24_{\pm0.04}$	$5.39_{\pm0.00}$	$4.05_{\pm0.02}$	$4.29_{\pm0.02}$	$2.90_{\pm0.00}$	10.28 GB
GRIN-P	$16.85_{\pm0.05}$	$17.59_{\pm0.06}$	$4.91_{\pm0.04}$	$5.05_{\pm0.00}$	$3.62_{\pm0.02}$	$3.85_{\pm0.07}$	$1.52_{\pm0.00}$	17.28 GB
GRU[†]	$18.17_{\pm0.03}$	$19.19_{\pm0.06}$	$5.30_{\pm0.03}$	$5.42_{\pm0.02}$	$3.98_{\pm0.02}$	$4.14_{\pm0.02}$	$11.59_{\pm0.04}$	12.01 GB
DCRNN	$16.99_{\pm0.09}$	$17.51_{\pm0.08}$	$5.14_{\pm0.06}$	$5.33_{\pm0.05}$	$3.54_{\pm0.01}$	$3.76_{\pm0.00}$	$1.36_{\pm0.01}$	19.72 GB
AGCRN	$17.19_{\pm0.06}$	$17.92_{\pm0.05}$	$4.84_{\pm0.01}$	$5.10_{\pm0.06}$	$4.06_{\pm0.01}$	$4.20_{\pm0.04}$	$1.15_{\pm0.01}$	23.40 GB
GWNet	$15.89_{\pm0.04}$	$16.39_{\pm0.14}$	$4.59_{\pm0.04}$	$4.76_{\pm0.03}$	$\mathbf{3.48}_{\pm0.05}$	$3.71_{\pm0.03}$	$2.12_{\pm0.00}$	16.02 GB
T&S-IMP	$16.54_{\pm0.03}$	$17.13_{\pm0.05}$	$4.98_{\pm0.01}$	$5.15_{\pm0.03}$	$3.60_{\pm0.02}$	$3.82_{\pm0.03}$	$2.68_{\pm0.00}$	7.03 GB
T&S-AMP	$16.15_{\pm0.02}$	$16.58_{\pm0.10}$	$4.93_{\pm0.02}$	$5.11_{\pm0.05}$	N/A	N/A	N/A	N/A
TTS-IMP	$16.25_{\pm0.01}$	$16.90_{\pm0.26}$	$4.81_{\pm0.07}$	$5.08_{\pm0.04}$	$3.50_{\pm0.01}$	$3.66_{\pm0.02}$	$18.84_{\pm0.14}$	12.81 GB
TTS-AMP	$15.63_{\pm0.06}$	$16.15_{\pm0.05}$	$4.70_{\pm0.00}$	$4.81_{\pm0.06}$	$\mathbf{3.46}_{\pm0.03}$	$3.65_{\pm0.05}$	$14.26_{\pm0.08}$	12.81 GB
HD-TTS-IMP	$15.50_{\pm0.07}$	$15.94_{\pm0.10}$	$\mathbf{4.48}_{\pm0.01}$	$\mathbf{4.64}_{\pm0.03}$	$3.47_{\pm0.01}$	$3.62_{\pm0.02}$	$7.11_{\pm0.03}$	10.86 GB
HD-TTS-AMP	$\mathbf{15.35}_{\pm0.01}$	$\mathbf{15.76}_{\pm0.07}$	$4.53_{\pm0.03}$	$4.65_{\pm0.04}$	$3.47_{\pm0.02}$	$\mathbf{3.61}_{\pm0.02}$	$6.21_{\pm0.02}$	10.86 GB

$A \in [0, 1]^{N \times N}$, we first assign to all nodes sinusoids with incommensurable frequencies $\bar{x}_t^i = \sin\left(\frac{t}{e^{i/N}}\right)$ and then propagate the signals over the graph as

$$X_{0:T} = \overline{X}_{0:T} + \dot{A}_K^\top \overline{X}_{0:T}, \qquad (19)$$

where $\dot{A}_K$ is a matrix where each column i has only 5 nonzero entries randomly selected from the i-th column of $\sum_{k=1}^{K} A^k$. The resulting time series $X_{0:T}$ are aperiodic signals extremely difficult to predict unless the graph structure is used to recover the initial sinusoids. As such, GraphMSO is a very effective benchmark to test the performance of spatiotemporal models (further details in Appendix A.1). We consider three different settings with increasing difficulty levels: (1) Point missing with $\eta = 0.05$; (2) Block–T missing with $\eta = 0.05$ and $p_f = 0.01$ ($\approx 27\%$ data missing); (3) Block–ST missing with $\eta = 0.05$, $p_f = 0.005$, and $p_g = [\,1\,]$, i.e., all faults being propagated to 1-hop neighbors in $\dot{A}_K$ ($\approx 67\%$ data missing).

Tab. 1 shows the results of the analysis. Notably, HD-TTS outperforms all baselines in all settings. The performance gap is notably pronounced when compared to more complex STGNNs like GWNet and AGCRN. These models, can – in principle – reach all the necessary spatial information through a single SMP step, as they learn a connectivity matrix. Finally, the performance in the Point missing setting suggests that learning dedicated representations for the different spatiotemporal scales may also be beneficial when the missing information can be more easily recovered.

5.2. Real-world Benchmarks

In this experiment, we consider three real-world datasets for time series forecasting. **AQI** (Zheng et al., 2015) contains one year of hourly measurements from 437 air quality

monitoring stations in China. Since 25.67% of the measurements are missing, it is a widely used benchmark for irregular spatiotemporal data analytics. **PV-US** (Hummon et al., 2012) contains one year of solar power production from 5016 simulated photovoltaic farms in the US with a 5-minute sampling rate. We consider the subset of 1081 western farms and aggregate measurements at 20-minute intervals. For the size of the graph, it is used to test the scalability of STGNNs (Cini et al., 2023b). Finally, the second new dataset introduced in this paper, named **EnGRAD**, contains 3 years of 5 historical weather variables sampled hourly at 487 grid points in England. The measurements are provided by open-meteo.com (Zippenfenig, 2023) and licensed under Attribution 4.0 International (CC BY 4.0). This dataset aims to fill a gap in the literature, by providing a spatiotemporal dataset with (1) multiple channels and (2) observations going beyond the slowest seasonality (i.e., one year). Following previous works, we obtain the adjacency matrices with a thresholded Gaussian kernel on the pairwise distances between sensors. For AQI, given the high number of missing values often occurring in blocks, we test the models on both the original data and in a Point setting with an additional $\eta = 0.25$ portion of data masked out. For the other datasets, we consider the Block–T and Block–ST settings. More details about the datasets and the missing data distributions can be found in Appendix A–B.

Tab. 2 shows the models' performance with different missing data patterns. Our approach ranks among the best-performing methods in all considered settings. Improvements are more evident with blocks of missing data spanning both time and space (Block–ST). Here, HD-TTS outperforms by a larger margin more complex STGNNs and both IMP and AMP variants w.r.t. the respective TTS model. This result underlines the advantage of combining hierar-
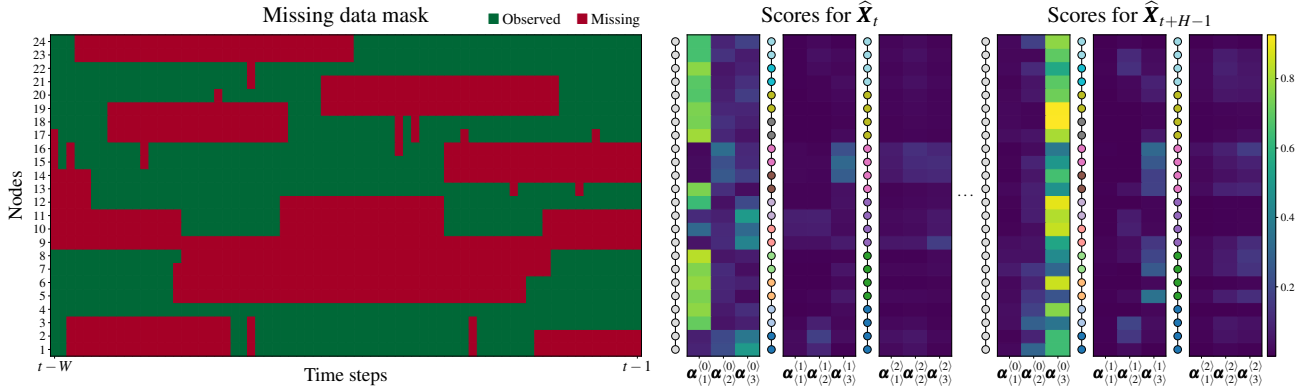
Figure 4. Decoder weights in GraphMSO (Block–ST). The graph used to produce this plot is the undirected line graph shown close to the scores associated with the first spatial resolution $k = 0$. Node colors in the graphs associated with higher spatial scales show how nodes are clustered in supernodes by $\boldsymbol{S}_k$ at each scale k.

chical representations from multiple temporal and spatial scales when the amount of missing data is substantial. The results on AQI, a dataset with a high ratio of missing observations, prove that HD-TTS is impactful in practical applications where missingness in data is a real concern. Finally, results on training speed and memory utilization on PV-US highlight the scalability advantages of our approach compared to state-of-the-art STGNNs, which also produce higher forecasting errors. In Appendix E, we report additional results for EngRAD in a setting where some variables are considered exogenous rather than inputs and targets.

5.3. Interpretability of Decoder Weights

In Fig. 4, we show the missing data mask $\boldsymbol{M}_{t-W:t}$ associated with the input and the attention scores computed by the decoder. The scores can be used to inspect which spatial and temporal scales the model focuses on when the pattern of missing data and the dynamics in the input change. For readability, we use an undirected line graph such that adjacent nodes are in consecutive positions in the figure. We consider the Block–ST setting and use the original adjacency matrix $\boldsymbol{A}$ to propagate both the signal and the faults in Eq. 19. We use a multi-step decoder to compute different scores for the different forecasting steps (see Appendix C); the figure shows only the scores associated with the first and last time step of the forecasts, i.e., $\widehat{\boldsymbol{X}}_t$ and $\widehat{\boldsymbol{X}}_{t+H-1}$. The scores computed for a time step are grouped into $K + 1$ blocks, with $K = 2$, each containing the scores of $L = 3$ temporal scales associated with a given spatial scale. As an example, the leftmost block contains the scores $\left[\boldsymbol{\alpha}_{\langle 1 \rangle}^{\langle 0 \rangle}, \boldsymbol{\alpha}_{\langle 2 \rangle}^{\langle 0 \rangle}, \boldsymbol{\alpha}_{\langle 3 \rangle}^{\langle 0 \rangle}\right]$ associated with the first spatial scale, i.e., the original graph.

It is possible to observe that, to predict the first step $\widehat{\boldsymbol{X}}_t$, the model focuses on the first temporal scale if the most recent data is not missing. Instead, when data are missing at a given node i, more weight is given to higher levels in

the spatial hierarchy. This indicates that the information is retrieved from the corresponding temporal scale of the i-th node's neighbors. As expected, representations associated with slower dynamics – in both time and space – become more relevant when forecasting values farther in time. For example, we see that to predict $\widehat{\boldsymbol{X}}_{t+H-1}$ the model focuses more on coarser temporal scales. Also, note that the scores vary across nodes subject to the same or similar missing data patterns. Indeed, the most relevant spatiotemporal scales also depend on the specific dynamics of the time series associated with each node. In Appendix D, we show additional analyses by considering a scenario where we keep the missing data patterns but assign each node the same signal.

6. Conclusions

We presented a novel framework for graph-based forecasting with missing data based on hierarchical spatiotemporal downsampling. Our model learns representations at different spatiotemporal scales, which are then combined by an interpretable attention mechanism to generate the forecasts. Thanks to this design, it can dynamically adapt the receptive field and handle different types of missing data patterns in a scalable fashion. Empirical comparisons with state-of-the-art methods on both synthetic and real-world datasets showed notable performance improvements, both in terms of forecasting accuracy and computational efficiency. Improvements are even more evident in the challenging settings with blocks of missing values in space and time, where other STGNNs models struggle.

The framework we proposed is very general and flexible. In this work, we investigated a few options to implement the temporal and spatial components. However, different combinations and more powerful operators are likely to deliver further performance improvements.

Acknowledgments

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Atwood, J. and Towsley, D. Diffusion-convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

Bacciu, D., Conte, A., and Landolfi, F. Generalizing downsampling from regular data to graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6718–6727, 2023.

Bai, L., Yao, L., Li, C., Wang, X., and Wang, C. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems*, 33:17804–17815, 2020.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Bianchi, F. M. and Lachi, V. The expressive power of pooling in graph neural networks. *Advances in neural information processing systems*, 36, 2023.

Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., and Jenssen, R. *Recurrent neural networks for short-term load forecasting: an overview and comparative analysis*. Springer Briefs in Computer Science, 2017.

Bianchi, F. M., Livi, L., Mikalsen, K. Ø., Kampffmeyer, M., and Jenssen, R. Learning representations of multivariate time series with missing data. *Pattern Recognition*, 96: 106973, 2019.

Bianchi, F. M., Grattarola, D., Livi, L., and Alippi, C. Hierarchical representation learning in graph neural networks with node decimation pooling. *IEEE Transaction on Neural Networks and Learning Systems*, 2020.

Borovykh, A., Bohte, S., and Oosterlee, C. W. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Chang, S., Zhang, Y., Han, W., Yu, M., Guo, X., Tan, W., Cui, X., Witbrock, M., Hasegawa-Johnson, M. A., and Huang, T. S. Dilated recurrent neural networks. *Advances in neural information processing systems*, 30, 2017.

Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

Chen, L., Chen, D., Shang, Z., Wu, B., Zheng, C., Wen, B., and Zhang, W. Multi-scale adaptive graph neural network for multivariate time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.

Cini, A. and Marisca, I. Torch Spatiotemporal, 3 2022. URL https://github.com/TorchSpatiotemporal/tsl.

Cini, A., Marisca, I., and Alippi, C. Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks. In *International Conference on Learning Representations*, January 2022.

Cini, A., Mandic, D., and Alippi, C. Graph-based time series clustering for end-to-end hierarchical forecasting. *arXiv preprint arXiv:2305.19183*, 2023a.

Cini, A., Marisca, I., Bianchi, F. M., and Alippi, C. Scalable spatiotemporal graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6): 7218–7226, June 2023b.

Cini, A., Marisca, I., Zambon, D., and Alippi, C. Graph deep learning for time series forecasting. *arXiv preprint arXiv:2310.15978*, 2023c.

Cini, A., Marisca, I., Zambon, D., and Alippi, C. Taming local effects in graph-based spatiotemporal forecasting. In *Advances in Neural Information Processing Systems*,

volume 36, pp. 55375–55393. Curran Associates, Inc., 2023d.

Cui, Z., Chen, W., and Chen, Y. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*, 2016.

Dwivedi, V. P., Joshi, C. K., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43), 2023.

Elman, J. L. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Falcon, W. and The PyTorch Lightning team. PyTorch Lightning, 3 2019. URL https://github.com/PyTorchLightning/pytorch-lightning.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Gao, J. and Ribeiro, B. On the equivalence between temporal and static equivariant graph representations. In *International Conference on Machine Learning*, pp. 7052–7076. PMLR, 2022.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Grattarola, D., Zambon, D., Bianchi, F. M., and Alippi, C. Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 35(2):2708–2718, 2024.

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

Harris, F. J. *Multirate signal processing for communication systems*. CRC Press, 2022.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hummon, M., Ibanez, E., Brinkman, G., and Lew, D. Sub-hour solar data for power system modeling from static spatial variability analysis. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2012.

Jin, M., Koh, H. Y., Wen, Q., Zambon, D., Alippi, C., Webb, G. I., King, I., and Pan, S. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *arXiv preprint arXiv:2307.03759*, 2023.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Marisca, I., Cini, A., and Alippi, C. Learning to reconstruct missing data from spatiotemporal graphs with sparse observations. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32069–32082. Curran Associates, Inc., 2022.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Rangapuram, S. S., Werner, L. D., Benidis, K., Mercado, P., Gasthaus, J., and Januschowski, T. End-to-end learning of coherent probabilistic forecasts for hierarchical time series. In *International Conference on Machine Learning*, pp. 8832–8843. PMLR, 2021.

Rangapuram, S. S., Kapoor, S., Nirwan, R. S., Mercado, P., Januschowski, T., Wang, Y., and Bohlke-Schneider, M. Coherent probabilistic forecasting of temporal hierarchies. In *International Conference on Artificial Intelligence and Statistics*, pp. 9362–9376. PMLR, 2023.

Rubin, D. B. Inference and Missing Data. *Biometrika*, 63 (3):581–592, 1976. ISSN 0006-3444.

Rusch, T. K., Bronstein, M. M., and Mishra, S. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

Seo, Y., Defferrard, M., Vandergheynst, P., and Bresson, X. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, pp. 362–373. Springer, 2018.

Shukla, S. N. and Marlin, B. Multi-time attention networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2020.

Shuman, D. I., Faraji, M. J., and Vandergheynst, P. A multi-scale pyramid transform for graph signals. *IEEE Transactions on Signal Processing*, 64(8):2119–2134, 2015.

Strang, G. and Nguyen, T. *Wavelets and filter banks*. SIAM, 1996.

Tang, X., Yao, H., Sun, Y., Aggarwal, C., Mitra, P., and Wang, S. Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5956–5963, 2020.

Tremblay, N. and Borgnat, P. Subgraph-based filterbanks for graph signals. *IEEE Transactions on Signal Processing*, 64(15):3827–3840, 2016.

Tremblay, N., Gonçalves, P., and Borgnat, P. Design of graph filters and filterbanks. In *Cooperative and Graph Signal Processing*, pp. 299–324. Elsevier, 2018.

Van Rossum, G. and Drake, F. L. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, A., Ye, Y., Song, X., Zhang, S., and James, J. Traffic prediction with missing data: A multi-task learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 24(4):4189–4202, 2023.

Wang, Y. and Jing, C. Spatiotemporal graph convolutional network for multi-scale traffic forecasting. *ISPRS International Journal of Geo-Information*, 11(2):102, 2022.

Wu, Y., Zhuang, D., Labbe, A., and Sun, L. Inductive Graph Neural Networks for Spatiotemporal Kriging. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4478–4485, 2021.

Wu, Z., Pan, S., Long, G., Jiang, J., and Zhang, C. Graph WaveNet for deep spatial-temporal graph modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 1907–1913. Association for the Advancement of Artificial Intelligence (AAAI), 2019.

Wu, Z., Zheng, D., Pan, S., Gan, Q., Long, G., and Karypis, G. Traversenet: Unifying space and time in message passing for traffic forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Yadan, O. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL https://github.com/facebookresearch/hydra.

Yi, X., Zheng, Y., Zhang, J., and Li, T. ST-MVL: Filling missing values in geo-sensory time series data. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pp. 2704–2710, New York, New York, USA, July 2016. AAAI Press. ISBN 978-1-57735-770-4.

Yu, B., Yin, H., and Zhu, Z. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3634–3640, 2018.

Yu, B., Yin, H., and Zhu, Z. St-unet: A spatio-temporal u-network for graph-structured time series modeling. *arXiv preprint arXiv:1903.05631*, 2019.

Yu, F. and Koltun, V. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations*, 2016.

Zhang, J., Shi, X., Xie, J., Ma, H., King, I., and Yeung, D. Y. GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. In *34th Conference on Uncertainty in Artificial Intelligence*, 2018.

Zheng, Y., Yi, X., Li, M., Li, R., Shan, Z., Chang, E., and Li, T. Forecasting fine-grained air quality based on big data. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 2267–2276, 2015.

Zhong, W., Suo, Q., Jia, X., Zhang, A., and Su, L. Heterogeneous spatio-temporal graph convolution network for traffic forecasting with missing values. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pp. 707–717. IEEE, 2021.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Zippenfenig, P. Open-meteo.com weather api, 2023. URL https://open-meteo.com/.

Appendix

In this appendix, we provide further details about our architecture and additional material about the experimental evaluation.

A. Datasets

Table 3. Statistics of the datasets and considered sliding-window parameters.

Datasets	Type	Nodes	Edges	Time steps	Sampling Rate	Channels	Window	Horizon
GraphMSO	Directed	10,000	100	300	N/A	1	72	36
AQI	Undirected	437	2,730	8,760	1 hour	1	24	6
EngRAD	Undirected	487	2,297	26,304	1 hour	5	24	6
PV-US	Undirected	1081	5,280	26,283	20 minutes	1	72	6

In this section, we report the details of the used datasets, with a particular focus on the two introduced in this paper: GraphMSO and EngRAD. Notably, all datasets used in our study are publicly available.

A.1. GraphMSO

Fig. 5 illustrates the procedure for generating the GraphMSO dataset. After creating a graph with a given topology, each node is assigned a sinusoid characterized by a frequency that is incommensurable with the ones at the other nodes. As such, summing one or more sinusoids results in a signal that is aperiodic and, thus, very difficult to predict. In addition, by aggregating from neighbors randomly chosen at different hops, predicting the signal becomes an even more challenging task. To obtain such a signal, we combine each sinusoid with the sinusoids of the graph neighbors according to the propagation scheme described in Sec. 5.1. For illustration purposes, in Fig. 5 we show a graph with a ring topology and propagate the sinusoids using the first-order adjacency matrix.
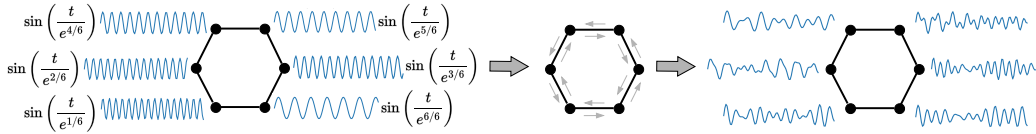

*Figure 5.* Schematic depiction of the generation of the GraphMSO dataset.

The details of the specific GraphMSO dataset used in the paper are the following. To build the adjacency matrix, we generated a random graph where each node has exactly 3 incoming edges. The number of nodes in the graph is $N = 100$ and the length of each time series is $10,000$ time steps.
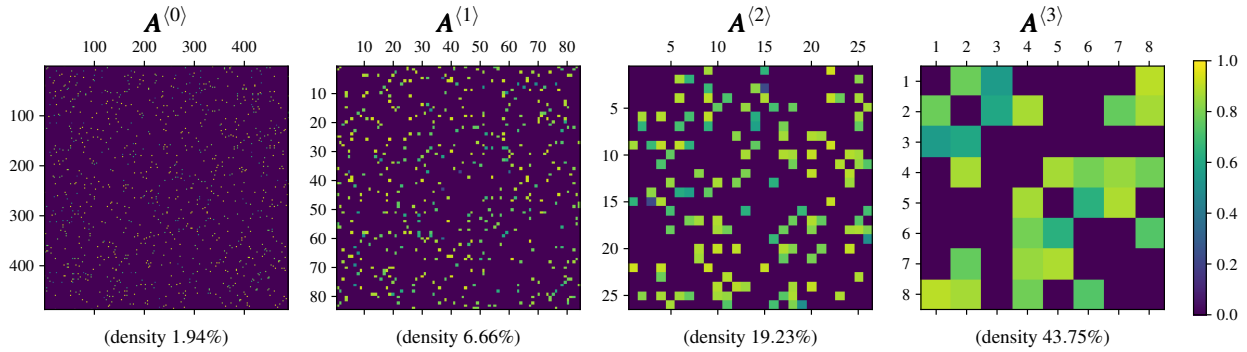
A.2. EngRAD

The EngRAD dataset contains measurements of 5 different weather variables collected at 487 grid points in England from 2018 to 2020. Data has been provided by open-meteo.com[2] (Zippenfenig, 2023) and licensed under Attribution 4.0 International (CC BY 4.0). The numerical weather prediction model used to generate the data is ECMWF IFS, which has a spatial resolution of 9 km. The grid points are located in correspondence to cities. For each point, we provide basic information such as geographic coordinates, elevation, closest city, and the county to which it belongs. The physical variables collected are (1) air temperature at 2 meters above ground (°C); (2) relative humidity at 2 meters above ground (%); (3) summation of total precipitation (rain, showers, snow) during the preceding hour (mm); (4) total cloud cover (%); (5) global horizontal irradiance (W/m^2). The three complete years of observations allow for a comprehensive capture of phenomena associated with yearly seasonalities, a feature rarely found in existing benchmarks on spatiotemporal data, which are often too short.

In Fig. 6, we show a geographic map (6a) and corresponding adjacency matrix (6b) of the EngRAD graph ($\boldsymbol{A}^{\langle 0 \rangle}$) and the coarsened graphs ($\boldsymbol{A}^{\langle 1:4 \rangle}$) obtained with k-MIS pooling. As can be seen from the plots, the structural information of the original graph is preserved throughout the pooling operations.

---

[2]https://open-meteo.com/
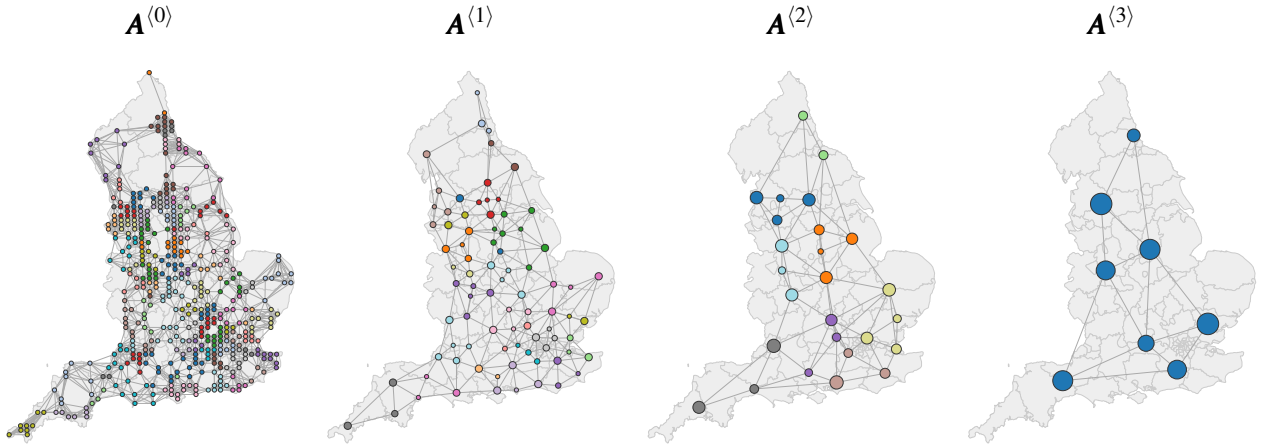
(a) Plot of the graphs in geographic space. Nodes with the same color in a graph are assigned to the same supernode in the corresponding pooled graph (due to the limited amount of colors in the palette, the same color might be used in different clusters). The size of a supernode is proportional to its cardinality and its spatial coordinates are obtained by averaging the coordinates of the associated nodes with the RED operation.

(b) Adjacency matrices at different spatial scales.

Figure 6. Original and pooled graphs in EngRAD obtained by recursive application of k-MIS pooling.

A.3. Other Datasets

The two additional datasets considered in the experimental evaluation are AQI and PV-US. Introduced by Zheng et al. (2015), **AQI** contains hourly measurements of pollutant PM 2.5 from 437 air quality monitoring stations located in 43 cities in China. Since 25.67% of the measurements are missing, it is often used to evaluate imputation models (Yi et al., 2016; Cini et al., 2022; Marisca et al., 2022). The **PV-US** dataset (Hummon et al., 2012) contains simulated energy production by 5016 photovoltaic farms in the US over all the year 2006. In the original datasets, samples are generated every 5 minute. We use the data pre-processed by Cini et al. (2023b) and mask zero values associated with night hours (i.e., with no irradiance). We consider only the subset containing 1081 farms in the western zones and aggregate observations at 20-minute intervals by taking their mean.

A.4. Data Pre-processing

Following previous works (Li et al., 2018; Cini et al., 2023b), to obtain the adjacency matrix in real-world datasets, we first compute a weighted, dense matrix containing the pairwise haversine distances between the geographic coordinates of the sensors. Then, we sparsify this matrix by applying a Gaussian kernel on the distances and zeroing out values under a threshold $\tau = 0.1$. For PV-US and EngRAD, we additionally cap the maximum number of neighbors for each node by keeping the 8 with the highest weight (i.e., shorter distance). To make the resulting graph undirected, we mirror edges that have a unique direction.

As these datasets contain communities of nodes very close in space and far from other communities, we insert additional edges to reduce the total number of connected components. In particular, an edge with minimum weight τ is added between the node in a disconnected community closest to the closest node in any other community. We repeat this step until the graph is connected or no edges can be added (e.g. if two connected components are very far apart).

For all datasets, we use as exogenous variables the encoding of the time of the day and the day of the year with two sinusoidal functions. For AQI, we additionally include a one-hot encoding of the day of the week, as the presence of air pollutants may be influenced by humans' routines.

We split the datasets into windows of W time steps, and train the models to predict the next H observations. Values for W and H in each dataset are in Tab. 3. For GraphMSO and PV-US, we divide the obtained windows sequentially into $70\%/10\%/20\%$ splits for training, validation, and testing, respectively. For AQI, we use as the test set the months of March, June, September, and December, as done by Yi et al. (2016). For EngRAD, containing 3 years of data, we use the year 2020 for testing and one week per month in the year 2019 for validation. We use all the samples that do not overlap with the validation and test sets to train the models.

For AQI and GraphMSO, we transform the data to have zero mean and unitary variance, while we rescale them to the $[0, 1]$ interval in EngRAD and PV-US. The statistics used to transform the data are computed using only valid values in the training set.

B. Experimental Setting

Software & Hardware   All the code used for the experiments has been developed with Python (Van Rossum & Drake, 2009) and relies on the following open-source libraries: PyTorch (Paszke et al., 2019); PyTorch Geometric (Fey & Lenssen, 2019); Torch Spatiotemporal (Cini & Marisca, 2022); PyTorch Lightning (Falcon & The PyTorch Lightning team, 2019); Hydra (Yadan, 2019); Numpy (Harris et al., 2020); Scikit-learn (Pedregosa et al., 2011).

Experiments were run on different workstations with varying models of processors and GPUs. However, we used the same workstation for measuring time and memory requirements in PV-US, whose results are reported in Tab. 2. More details are provided later in this section.

Baselines   In the following, we report the hyperparameters used in the experiment for the considered baselines. Whenever possible, we relied on code provided by the authors or available within open-source libraries to implement the baselines. To ensure a fair comparison, we tried to keep the number of trainable parameters of our model and each baseline in the same range.

- **DCRNN** We used the same parameters of the original paper (Li et al., 2018), with an embedding size of 64 and a $K = 2$ spatial receptive field. Following previous works (Cini et al., 2023b), we used an MLP as the decoder with one hidden layer with 128 units.
- **GWNet** We used the same parameters reported in the original paper (Wu et al., 2019), except for those controlling the receptive field. Being GWNet a convolutional architecture, this was done to ensure that the receptive field covers the whole input sequence. In particular, we used 8 layers with temporal kernel size and dilation of 3 when the input window is 24 and 7 layers with dilation and kernel size of 5 when the input window is 72.
- **AGCRN** We used the same hyperparameters reported in the original paper (Bai et al., 2020).
- **GRIN-P** To implement this baseline, we used the unidirectional layer available in the Torch Spatiotemporal library[3] followed by an MLP as the decoder. We kept the same hyperparameters of the original paper (Cini et al., 2022).

For the GRUs, namely **GRU**, **GRU-D**, and **GRU-I**, we used a size of $d_h = 64$ for the hidden dimension, $L = 2$ recurrent layers and one hidden layer with 128 units for the MLP used as decoder. For the four **TTS-IMP**, **TTS-AMP**, **T&S-IMP**, and **T&S-AMP** baselines, we used the same hyperparameters reported in the paper (Cini et al., 2023d) and increased the SMP layers to $K = 4$ to obtain a wider receptive field.

Training setting   We used AdamW (Loshchilov & Hutter, 2019) as the optimizer with an initial learning rate of 0.001. We used the `ReduceLROnPlateau` scheduler of Pytorch that reduces the learning rate by a factor of 0.5 if no improvements are noticed after 10 epochs. We trained all models for 200 epochs of 300 randomly drawn mini-batches of 32 examples and

---

[3]https://github.com/TorchSpatiotemporal/tsl

Table 4. Parameters used to generate the missing data mask in the different settings.

| Parameter | GraphMSO | | | AQI | EngRAD / PV-US | |
	Point	Block–T	Block–ST	Point	Block–T	Block–ST
η	0.05	0.05	0.05	0.25	0.05	0.075
p_f	0	0.01	0.005	0	0.01	0.003
$s_{\min}$	–	8	8	–	4	6
$s_{\max}$	–	48	48	–	12	21
$\boldsymbol{p}_g$	0	0	$\begin{bmatrix} 1 \end{bmatrix}$	0	0	$\begin{bmatrix} 0.33 & 0.15 & 0.05 \end{bmatrix}$

stopped the training if the MAE computed on the validation set did not decrease after 30 epochs. We then used the weights of the best-performing model for evaluation on the test set. For the baselines that jointly perform imputation and forecasting, we used the imputation error as an auxiliary loss.

Simulated missing data patterns To evaluate the models' performance in function of the missing data pattern in input, we design a strategy to inject missing values in a controlled fashion. Given a sequence of observations of length T, we generate the missing data mask $\boldsymbol{M}_{t:t+T}$ according to three parameters:

η: The parameter of the Bernoulli distribution associated with the probability of an observation being missing.

p_f: The parameter of the Bernoulli distribution associated with the probability of a sensor fault (consecutive missing values).

   $s_{\min}, s_{\max}$: The boundaries of the uniform distribution $\mathcal{U}(s_{\min}, s_{\max})$ from which the length of the fault s is sampled.

$\boldsymbol{p}_g$: The vector of the Bernoulli distributions' parameters associated with the probability of a fault to be propagated at neighboring nodes. The k-th element of the vector is associated with the k-th neighborhood order.

All parameters are constant across nodes and time steps, meaning that the process generating the missing data is stationary. For each dataset, we fix the seed used to generate the mask, ensuring that all models observe the same data in all runs. In the case of multivariate sensor observations, such as in EngRAD, the mask is sampled independently across the channels, possibly resulting in partially available observations. We report the parameters used to generate the mask in the different settings in Tab. 4. Note that in GraphMSO even if we have $\boldsymbol{p}_g = [1]$, the propagation is done with the matrix $\dot{\boldsymbol{A}}_K$ defined in Sec. 5.1, which contains neighbors sampled from all orders up to K. To give an idea of how the missing data distributions look like, in Fig. 7 we report the missing data masks used in GraphMSO for the three settings specified in Tab. 4.
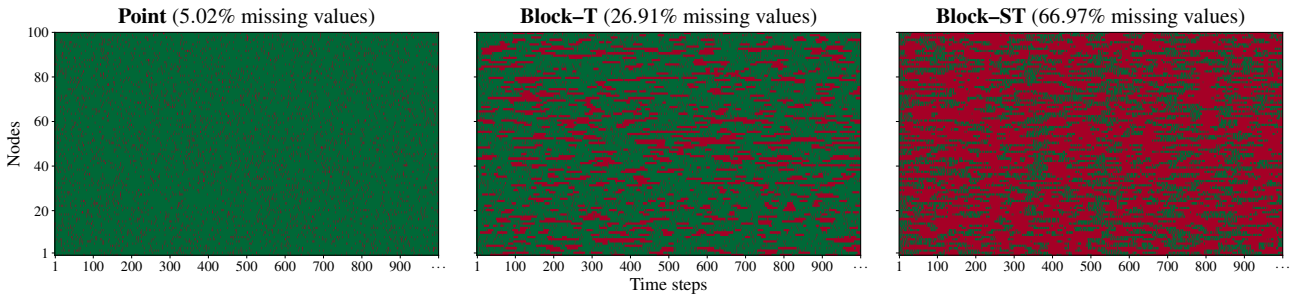

Figure 7. From left to right, the masks used in GraphMSO. Green and red colors denote valid and missing observations, respectively.

During training, the generated mask is used to remove observations in both the input and ground-truth data. Conversely, when testing the models only the input is masked out and the forecasting error metrics are computed on all valid observations (i.e., not missing in the original dataset).

Time and memory requirements In Tab. 2 of the paper, we report the time required for a single model update (expressed in batches per second) and the corresponding GPU memory usage for each method under consideration. To ensure a fair assessment, we record the time elapsed between the initiation of the inference step and the completion of the weights' update

for a total of 50 mini-batches of 16 samples each, excluding the initial 5 and final 5 measurements to mitigate potential overhead impacts. To measure the GPU memory reserved by each model, we relied on the NVIDIA System Management Interface[4], which provides near real-time GPU usage monitoring. We ran all experiments sequentially on a workstation running Ubuntu 20.04.6 LTS and equipped with one AMD Ryzen 9 5900X 12-core processor, 128GB 3200MHz DDR4 RAM, and two NVIDIA RTX A6000 GPU with 48 GB GDDR6 RAM.

C. Architecture Details

In this section, we provide additional details about the implementation of our model.

Spatial message passing   We consider two different SMP operators, belonging to the isotropic and anisotropic categories, respectively. As isotropic SMP, we use the diffusion-convolutional operator (Atwood & Towsley, 2016), that computes the messages for each p-hop neighborhood as

$$\phi^l(\boldsymbol{x}_t^{j,l-1}, \tilde{a}_p^{ji}) = \tilde{a}_p^{ji} \boldsymbol{x}_t^{j,l-1} \Theta_p^l \tag{20}$$

where ϕ^l is the message function (Sec. 2.1), $\tilde{a}_p^{ji}$ is the weight of the edge between nodes j and i in $\boldsymbol{A}^p$ normalized by the sum of the incoming edges' weight. For directed graphs, we further compute messages from p-hop neighbors in $\boldsymbol{A}^\top$ with different parameters $\Theta_p'^l$, following previous works (Li et al., 2018).

For the anisotropic variant, we rely on the anisotropic SMP layer introduced by Cini et al. (2023d), that computes the messages for each p-hop neighborhood as

$$\boldsymbol{m}_t^{ji,l-1} = \xi\Big(\big[\boldsymbol{x}_t^{i,l-1} \,\|\, \boldsymbol{x}_t^{j,l-1} \,\|\, a_{ji}\big] \Theta_1^l\Big) \Theta_2^l, \tag{21}$$

$$\tilde{a}_t^{ji,l-1} = \sigma\Big(\boldsymbol{m}_t^{ji,l-1} \Theta_3^l\Big), \tag{22}$$

$$\phi^l(\boldsymbol{m}_t^{ji,l-1}, \tilde{a}_t^{ji,l-1}) = \tilde{a}_t^{ji,l-1} \boldsymbol{m}_t^{ji,l-1} \tag{23}$$

where Θ_1^l, Θ_2^l, and Θ_3^l are matrices of learnable parameters, $\|$ the concatenation operation, and σ and ξ the sigmoid and ELU activation functions, respectively.
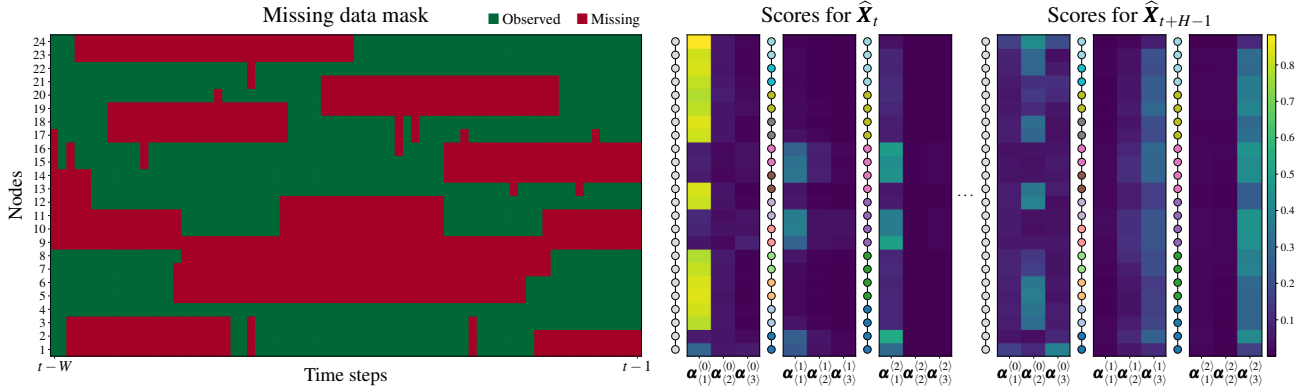
These two strategies are only used to implement SMP during the downsampling stage, i.e., SMP_D. To reduce the total number of parameters and operations, after upsampling we only perform a propagation step along the graph, i.e., we implement $\text{SMP}_\text{U}(\boldsymbol{X}, \boldsymbol{A}) = \boldsymbol{A}^\top \boldsymbol{X}$.

k**-MIS pooling** k-MIS pooling (Bacciu et al., 2023) is a graph-based equivalent of evenly-spaced coarsening mechanisms for regular data. k-MIS pooling provides theoretical guarantees for distortion bounds on path lengths and the ability to preserve key topological properties in the coarsened graphs. The method utilizes maximal k-independent sets to find a set of nodes that are approximately equally spaced in the original graph. The selected nodes are then used as vertices of the reduced graph. k-MIS pooling first identifies the supernodes with the centroids of the maximal k-independent set of a graph. The centroids are selected with a greedy approach based on a ranking vector π applied to the vertex features to speed up computation. Depending on how the ranking π is computed, k-MIS pooling declines as a trainable or non-trainable pooling operator. In the former case, ranking is computed as $\pi = \boldsymbol{X}^{\langle k-1 \rangle} \mathbf{p}^\top$ where the projector $\mathbf{p}$ is the output of a trainable function. The non-trainable version of k-MIS pooling, instead, relies on a pre-defined ranking function that does not depend on trainable parameters or node features. The latter is the setting adopted in this work, as we let the ranking be constant across the node, i.e., $\pi = \mathbf{1}$. In addition, we convert directed graphs to undirected, as the method acts on undirected graphs.

Multi-step decoder   In some applications, the different dynamics captured by the multi-scale representations might be more or less relevant to the prediction depending on the distance of the target time step from the last observations. The structured multi-scale encodings, paired with the adaptive weighting mechanism, allow us to compute H different score sets $\left\{\alpha_{t+h\langle l \rangle}^{i\langle k \rangle}\right\}_{h=0,\dots,H-1}$, and, hence, H aggregated representations $\left\{\tilde{\boldsymbol{z}}_{t+h}^i\right\}_{h=0,\dots,H-1}$, per node. The predictions are then obtained, $\forall\, h \in [0, H)$, as

$$\hat{\boldsymbol{x}}_{t+h}^i = \text{MLP}\left(\tilde{\boldsymbol{z}}_{t+h}^i\right). \tag{24}$$

---

[4] https://developer.nvidia.com/nvidia-system-management-interface

(a) GraphMSO dataset with the same time series associated with each node.
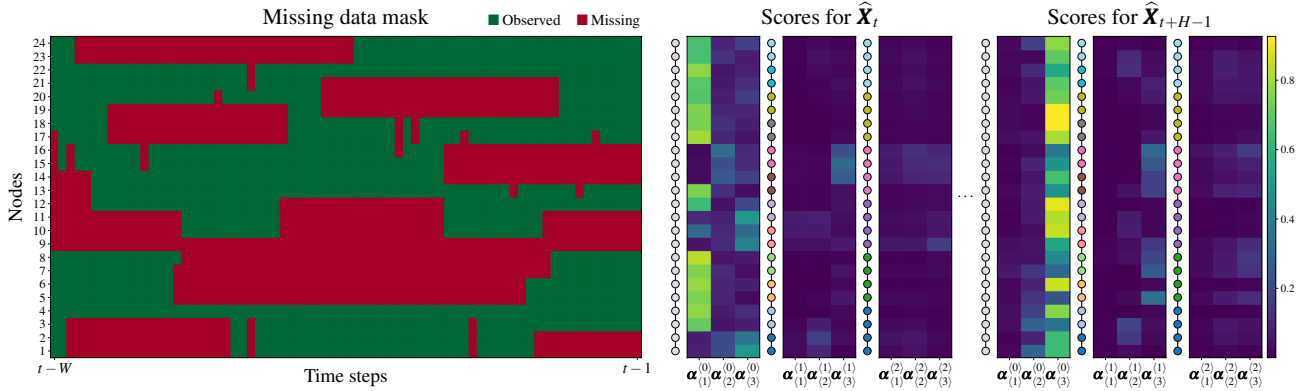
(b) GraphMSO dataset with the original data-generating process (different time series across nodes).

Figure 8. Decoder weights in GraphMSO (Block–ST) with the same missing data pattern but different node signals as input. The graph used to produce this plot is the undirected line graph shown close to the scores associated with the first spatial resolution $k = 0$. Node colors in the graphs associated with higher spatial scales show how nodes are clustered in supernodes by $\boldsymbol{S}_k$ at each scale k.

We will use this implementation in Appendix D to inspect the differences in the scores computed for the first and last time steps in the forecasting horizon.

Hyperparameters We use an embedding size of $d_h = 64$ for all hidden representations $\boldsymbol{h}_t^i$ and $\boldsymbol{z}_t^i$. In the input encoder, we concatenate the input with node embeddings $\boldsymbol{\theta}^i$ of size 32 and apply an affine transformation. For temporal processing, we use $L = 4$ layers with a downsampling factor $d = 3$. For spatial processing, we use $K = 3$ pooling layers. The decoder is an MLP with 2 hidden layers with 128 units. We use ELU as the activation function within the architecture. In the real-world datasets, we compute different attention weights for each forecasting step, as discussed in the previous section.

D. Interpretability of Decoder Weights

In this appendix, we extend the analysis on the decoder weights reported in Sec. 5.3 including a variant of GraphMSO with the same time series associated with each node. Indeed, the scores are also influenced by the actual dynamics in the input time series, which can make it harder to see the relationship between the missing data patterns and the scores. In Fig. 8a, we show the scores computed by the decode in this setting; we include the original plot of Fig. 4 as Fig. 8b for a better comparison. In both settings, we can observe that missing data affecting a given spatiotemporal scale in a node results in a lower weight given to the corresponding representations. This is even more evident in the variant with the same time series in each node, where the contrast between high and low scores is more pronounced.

Table 5. Forecasting error (MAE) on EngRAD with different target variables and missing data distributions. We use **bold** formatting to mark best results. [†]Models without spatial message passing.

Model	ALL VARIABLES		TEMP. & RAD.	
	Block–T	**Block–ST**	**Block–T**	**Block–ST**
GRU-D[†]	$5.77_{\pm 0.03}$	$6.01_{\pm 0.06}$	$16.63_{\pm 0.12}$	$16.82_{\pm 0.12}$
GRU-I[†]	$5.79_{\pm 0.01}$	$5.95_{\pm 0.01}$	$16.66_{\pm 0.07}$	$16.80_{\pm 0.04}$
GRIN-P	$5.41_{\pm 0.06}$	$5.61_{\pm 0.03}$	$15.59_{\pm 0.14}$	$15.80_{\pm 0.11}$
GRU[†]	$5.84_{\pm 0.00}$	$6.03_{\pm 0.04}$	$16.77_{\pm 0.08}$	$16.97_{\pm 0.10}$
DCRNN	$5.63_{\pm 0.03}$	$5.90_{\pm 0.04}$	$16.27_{\pm 0.18}$	$16.52_{\pm 0.18}$
AGCRN	$5.26_{\pm 0.01}$	$5.53_{\pm 0.02}$	$15.31_{\pm 0.04}$	$15.71_{\pm 0.09}$
GWNet	$5.05_{\pm 0.08}$	$5.22_{\pm 0.07}$	$14.49_{\pm 0.15}$	$14.87_{\pm 0.07}$
T&S-IMP	$5.47_{\pm 0.02}$	$5.67_{\pm 0.05}$	$15.85_{\pm 0.13}$	$16.13_{\pm 0.16}$
T&S-AMP	$5.37_{\pm 0.03}$	$5.58_{\pm 0.09}$	$15.82_{\pm 0.17}$	$15.86_{\pm 0.10}$
TTS-IMP	$5.21_{\pm 0.04}$	$5.46_{\pm 0.07}$	$15.32_{\pm 0.18}$	$15.19_{\pm 0.23}$
TTS-AMP	$5.10_{\pm 0.03}$	$5.31_{\pm 0.04}$	$14.89_{\pm 0.19}$	$15.02_{\pm 0.07}$
HD-TTS-IMP	$\mathbf{4.93_{\pm 0.02}}$	$\mathbf{5.12_{\pm 0.03}}$	$\mathbf{14.34_{\pm 0.09}}$	$\mathbf{14.41_{\pm 0.07}}$
HD-TTS-AMP	$\mathbf{4.93_{\pm 0.03}}$	$\mathbf{5.10_{\pm 0.01}}$	$\mathbf{14.22_{\pm 0.08}}$	$\mathbf{14.46_{\pm 0.14}}$

E. EngRAD Additional Results

In this appendix, we consider two additional settings for the EngRAD dataset. The first setting is similar to the one considered in Sec. 5.2, but we do not compute loss and metrics on the prediction of the global horizontal irradiance variable during night hours when it is null. In the second setting, we consider as target variables X_t only air temperature and irradiance (masked during night hours), while we use the remaining weather variables as exogenous U_t. This scenario differs from the one considered in previous experiments as the exogenous variables here vary across nodes rather than being the same temporal encodings. Tab. 5 shows the experiment results. We can see that the results are consistent with Tab. 2 and both HD-TTS variants outperform all the baselines in every considered setting.

F. Traffic Forecasting

In Sec. 3.2, we pointed out at potential issues of using non-trainable pooling operators when the input graph is noisy or does not reflect the actual correlations and dependencies across the time series. In this experiment, we show this phenomenon in two popular traffic forecasting benchmarks, namely, **METR-LA** and **PEMS-BAY**. These datasets contain traffic speed records measured every 5 minutes by 207 and 325 sensors located over highways in Los Angeles and the Bay Area, respectively. In both datasets, the adjacency matrix is directed and, as discussed in Appendix C, it must be transformed to undirected to obtain the coarsened graphs when using the k-MIS pooling method.

As it is evident from Fig. 9, the graph topology does not reflect the underlying structure of the road network. Distant nodes are assigned by the pooling operator to the same supernode, causing significant discrepancies between the topology of the coarsened graphs and the underlying data. Indeed, points at a significant driving distance end up being connected in the graph. Additionally, some nodes are disconnected from the rest of the graph (5 in METR-LA and 12 in PEMS-BAY). These issues affect the computation of the coarsened graphs and, consequently, the performance of HD-TTS, which relies on spatially aggregated representations to capture long-range spatial dynamics. Tab. 6 shows the performance of the models. While outperforming the baselines in some settings, in some other cases flatter models (like TTS-IMP) are more effective.

Table 6. Forecasting error on traffic benchmarks with different block missing settings. **Bold** formatting is used to mark best result in each setting. [†]Models without spatial message passing.

| Model	METR-LA				PEMS-BAY			
	Block–T		Block–ST		Block–T		Block–ST	
	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
GRU-D[†]	$3.91_{\pm0.03}$	$62.03_{\pm1.06}$	$4.17_{\pm0.04}$	$70.11_{\pm1.09}$	$2.10_{\pm0.01}$	$24.19_{\pm0.17}$	$2.19_{\pm0.01}$	$26.08_{\pm0.29}$
GRU-I[†]	$4.13_{\pm0.00}$	$69.47_{\pm0.24}$	$4.49_{\pm0.01}$	$80.43_{\pm0.53}$	$2.16_{\pm0.01}$	$25.54_{\pm0.05}$	$2.25_{\pm0.00}$	$27.89_{\pm0.01}$
GRIN-P	$3.48_{\pm0.07}$	$49.59_{\pm1.51}$	$3.67_{\pm0.06}$	$55.14_{\pm0.78}$	$1.93_{\pm0.03}$	$18.33_{\pm0.61}$	$2.00_{\pm0.04}$	$19.60_{\pm0.51}$
GRU[†]	$4.01_{\pm0.01}$	$67.26_{\pm0.34}$	$4.31_{\pm0.01}$	$76.29_{\pm0.54}$	$2.08_{\pm0.00}$	$24.29_{\pm0.08}$	$2.17_{\pm0.00}$	$26.40_{\pm0.17}$
DCRNN	$3.48_{\pm0.01}$	$47.21_{\pm0.48}$	$3.78_{\pm0.03}$	$56.49_{\pm1.39}$	$1.86_{\pm0.00}$	$17.66_{\pm0.07}$	$1.95_{\pm0.00}$	$19.39_{\pm0.14}$
AGCRN	$3.38_{\pm0.00}$	$47.80_{\pm0.74}$	$3.53_{\pm0.01}$	$52.89_{\pm0.77}$	$1.80_{\pm0.01}$	$16.56_{\pm0.33}$	$1.85_{\pm0.01}$	$17.41_{\pm0.15}$
GWNet	$3.21_{\pm0.01}$	$\mathbf{42.17_{\pm0.47}}$	$3.43_{\pm0.03}$	$49.23_{\pm1.66}$	$1.71_{\pm0.01}$	$15.30_{\pm0.21}$	$1.78_{\pm0.00}$	$16.69_{\pm0.08}$
T&S-IMP	$3.28_{\pm0.01}$	$45.61_{\pm0.39}$	$3.45_{\pm0.01}$	$51.63_{\pm0.70}$	$1.74_{\pm0.00}$	$15.81_{\pm0.08}$	$1.81_{\pm0.01}$	$17.24_{\pm0.05}$
T&S-AMP	$3.24_{\pm0.01}$	$44.67_{\pm0.45}$	$3.45_{\pm0.01}$	$52.32_{\pm0.82}$	$1.74_{\pm0.00}$	$15.94_{\pm0.16}$	$1.81_{\pm0.00}$	$17.34_{\pm0.07}$
TTS-IMP	$\mathbf{3.16_{\pm0.02}}$	$42.36_{\pm0.96}$	$\mathbf{3.36_{\pm0.02}}$	$\mathbf{49.01_{\pm0.58}}$	$\mathbf{1.70_{\pm0.00}}$	$14.94_{\pm0.17}$	$1.77_{\pm0.01}$	$16.34_{\pm0.15}$
TTS-AMP	$3.19_{\pm0.02}$	$43.40_{\pm1.17}$	$3.40_{\pm0.02}$	$50.89_{\pm0.79}$	$1.74_{\pm0.00}$	$15.76_{\pm0.13}$	$1.80_{\pm0.01}$	$17.18_{\pm0.21}$
HD-TTS-IMP	$3.23_{\pm0.01}$	$44.11_{\pm0.55}$	$3.40_{\pm0.01}$	$50.18_{\pm0.39}$	$\mathbf{1.70_{\pm0.01}}$	$15.01_{\pm0.30}$	$\mathbf{1.76_{\pm0.01}}$	$16.17_{\pm0.25}$
HD-TTS-AMP	$3.22_{\pm0.01}$	$43.41_{\pm0.36}$	$3.39_{\pm0.02}$	$49.62_{\pm0.62}$	$\mathbf{1.70_{\pm0.01}}$	$\mathbf{14.91_{\pm0.21}}$	$1.77_{\pm0.00}$	$\mathbf{16.08_{\pm0.12}}$
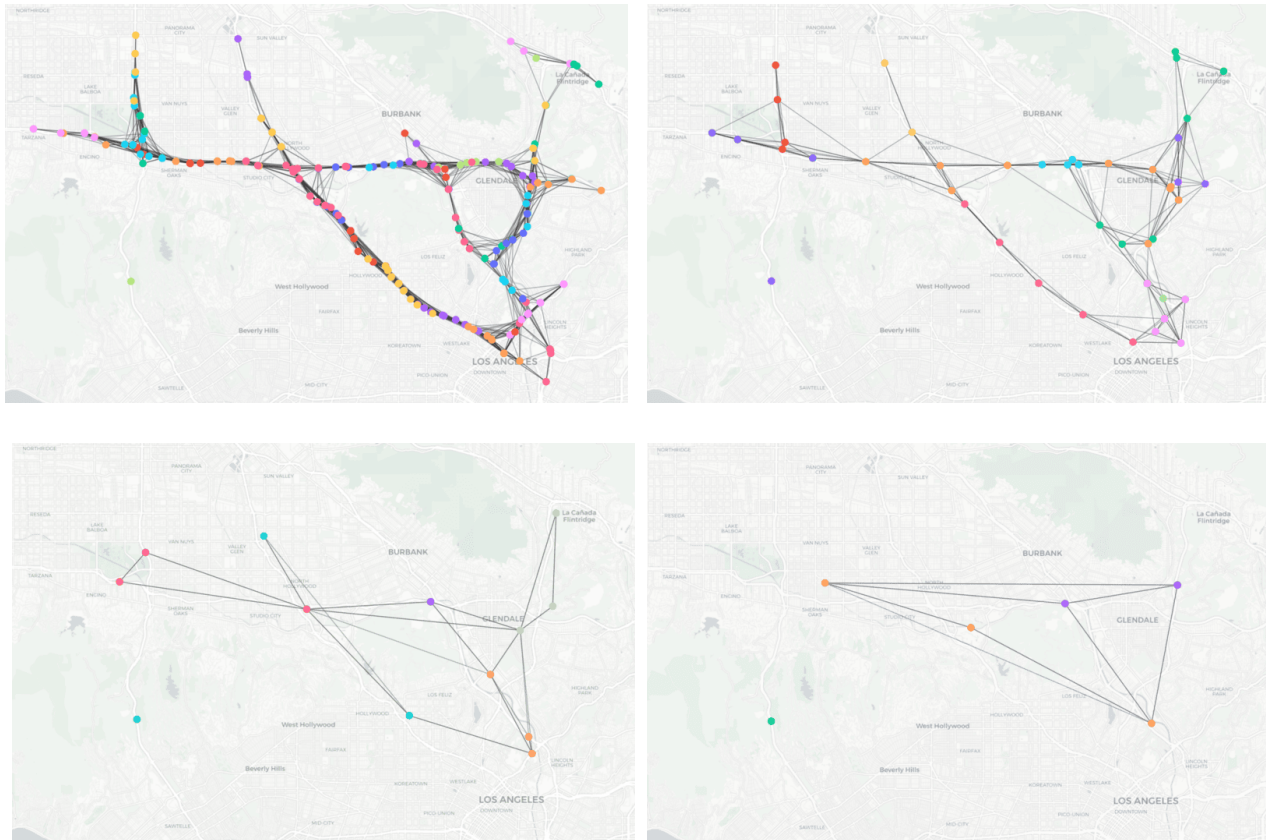
Figure 9. Coarsened graphs in METR-LA obtained by recursive application of k-MIS pooling.