Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations

Anonymous Author(s)
Affiliation
Address
email

Abstract

Modeling multivariate time series as temporal signals over a (possibly dynamic) graph is an effective representational framework that allows for developing models for time series analysis. In fact, discrete sequences of graphs can be processed by autoregressive graph neural networks to recursively learn representations at each discrete point in time and space. Spatiotemporal graphs are often highly sparse, with time series characterized by multiple, concurrent, and even long sequences of missing data, e.g., due to the unreliable underlying sensor network. In this context, autoregressive models can be brittle and exhibit unstable learning dynamics. The objective of this paper is, then, to tackle the problem of learning effective models to reconstruct, i.e., impute, missing data points by conditioning the reconstruction only on the available observations. In particular, we propose a novel class of attention-based architectures that, given a set of highly sparse discrete observations, learn a representation for points in time and space by exploiting a spatiotemporal diffusion architecture aligned with the imputation task. Representations are trained end-to-end to reconstruct observations w.r.t. the corresponding sensor and its neighboring nodes. Compared to the state of the art, our model handles sparse data without propagating prediction errors or requiring a bidirectional model to encode forward and backward time dependencies. Empirical results on representative benchmarks show the effectiveness of the proposed method.

1 Introduction

Exploiting structure – both temporal and spatial – is arguably the key ingredient for the success of modern deep learning architectures and models. Structures and invariances allow imposing inductive biases to learning systems that act as strong regularizations, thus limiting the space of possible models to the most plausible ones and, consequently, greatly reducing sample complexity. This is the case with spatiotemporal graph neural networks (STGNNs) [1–3] which learn to process multivariate time series while taking into account underlying space and time dependencies by encoding structural spatiotemporal inductive biases in their architectures. In particular, graphs are used to model the presence of spatial relationships which can be thought of as soft functional dependencies existing among the sensors generating the time series [4]. However, even when spatiotemporal relationships are present, available data are almost always incomplete and irregularly sampled both spatially and temporally. This is definitely true for data coming from real sensor networks (SNs), where the presence of missing data is a phenomenon inherent to any data acquisition or communication process. In this case, a common approach is to reconstruct missing time series observations with simple interpolation strategies before proceeding with the downstream task. However, arguably, this induces

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

a bias in the inference procedure, possibly made even worse by the strong regularization imposed on the model. More advanced methods deal with missing data by autoregressively replacing missing observations with predicted ones, eventually using bidirectional architectures [5, 6] to exploit both forward and backward temporal dependencies. To account also for spatial dependencies, Cini et al. [4] introduced a method, named GRIN, combining a bidirectional autoregressive architecture with message passing graph neural networks [7–10] for spatiotemporal imputation. Despite being the state of the art in imputation, GRIN suffers from the error propagation typical of autoregressive models that bootstrap future inferences from their own predictions. In fact, we argue that the propagation of imputed (biased) values through space and time combined with noisy observations might exacerbate error accumulation in highly sparse data and drive the hidden state of GRIN-like models to possibly drift away. In this paper, we aim at tackling this problem by designing an architecture based on a novel attention mechanism that takes spatiotemporal sparsity into account while learning representations and imputing missing values.

Attention-based models allow distributed representations to emerge by letting each discrete element – representing an entity or point within a structured environment – interact with each other. In fact, these mechanisms for propagating information through structures have been recently linked to collective intelligence [11]. In our context, each token, represented as a node in a sequence of graphs, represents a point in space and time. A trivial way to account for sparsity in this setting is to constrain the attention coefficients of time steps corresponding to missing data to be zero or simply to add an auxiliary token to indicate missing observations; basically, the resulting reconstruction model would behave similarly to a denoising autoencoder, in the same spirit of BERT-like encoders popular in natural language processing [12]. Conversely, we propose a novel architecture exploiting an inter-node sparse spatiotemporal attention mechanism within the neural message-passing framework. In particular, we seek to design an architecture where each processing stage is aligned with the task of reconstructing missing spatiotemporal observations. Our model can be seen as a learned, self-organizing, spatiotemporal propagation process that, as we will motivate throughout the paper, is more apt to the purpose of missing data reconstruction than standard encoder-decoder architectures. In fact, compared with the alternatives discussed so far, our method exploits the aforementioned propagation process to learn a predictive representation for each missing observation by relying only on observed values propagated through the spatiotemporal structure. This approach achieves the twofold objective of avoiding propagating biased representation – typical in the autoregressive framework – and reconstructing observations at arbitrary nodes in the sensor network. In summary, our main contributions are:

1. introduction of a sparse spatiotemporal attention mechanism to learn, from sparse data, representations localized in time and space;
2. design of a novel graph neural network architecture based on the aforementioned spatiotemporal attention mechanism and equipped with inductive biases that make the model tailored for the multivariate time series imputation task;
3. an empirical assessment of the proposed method showing how it overcomes the limits of existing approaches, particularly in settings with highly sparse data.

The paper is structured as follows. In Section 2 we discuss related works and contrast them with ours. We formulate the problem of multivariate time series imputation in the context of spatiotemporal graphs in Section 3, and present our approach in Section 4, providing an in-depth discussion of motivations, design choices, as well as methodological and technical issues. Finally, in Section 5, we report the empirical results on several relevant benchmarks. Conclusions and future works are discussed in Section 6.

2 Related works

Multivariate time series imputation is a core task in time series analysis. Besides standard statistical approaches based on linear autoregressive models [13–15] or interpolation [16], methods based on

matrix factorization are widely popular [17] which can also incorporate temporal [18] and graph-side information [19]. Deep learning methods are also commonly used in this regard. In particular, deep autoregressive models based on recurrent neural networkss (RNNs) are currently among the most widely adopted methods [20, 5, 6, 21]. In this direction, BRITS [6] is archetypal of several related works which exploit bidirectional RNNs to perform imputation. Notably, differently from a denoising autoencoder, BRITS uses one-step-ahead forecasting as a surrogate task to learn an imputation model while using a simple linear regression layer to incorporate spatial information. Several approaches in the literature, then, rely on generative adversarial neural networks [22] – often paired with RNNs – to generate imputed subsequences by matching the underlying data distribution [23, 24, 21]. Recently, several attention-based imputation techniques have also been proposed [25–27], however, none of these explicitly account for spatial dependencies within the graph processing framework and overlook the spatial dimension of the problem. Other works, instead, address this problem in the context of continuous-time models [28]. Huang et al. [29], in particular, exploit graph representations to model spatial dynamics. The limits of deep autoregressive approaches in data reconstruction have also been tackled by using hierarchical imputation methods [30, 31]. More related to our work, GRIN [4] uses a bidirectional graph RNN, paired with a message passing spatial decoder, to impute time series based on spatiotemporal dependencies. Other graph-based architectures have been used in application-specific settings, such as traffic data [32, 33] and load profiles from smart grids [34]. While GRIN achieves remarkable performance, we argue that the spatial regularization might not be enough to prevent errors to accumulate in the states of the recurrent graph network.

The attention mechanism has been exploited in several contexts within the graph deep learning literature, in particular in anisotropic graph convolutional filters [35–38]. Among STGNNs, attention-based architectures have been exploited in time series forecasting [37, 39–41]. In particular, TraverseNet [41] is specially related to our work, since it relies on spatiotemporal autoregressive attention to compute messages exchanged between nodes. One striking example of attention being used successfully to process incomplete data is in pretraining routines for representation learning in natural language processing [12, 42]. Finally, graph neural networks are also popular for reconstructing missing features in static graphs [43–45].

3 Preliminaries

We model multivariate spatiotemporal time series as observations coming from a SNs. In a SN every i-th node, i.e., *sensor*, acquires a d-dimensional $\boldsymbol{x}_t^i \in \mathbb{R}^d$ observation at each t-th time step. We denote by $\boldsymbol{X}_t \in \mathbb{R}^{N_t \times d}$ the matrix collecting the measurements of N_t sensors at time step t, with $\boldsymbol{X}_{t:t+T}$ being the sequence of T measurements collected in the time interval $[t, t+T)$. We model functional relationships among the sensors as graph edges. Relationships can be often inferred from available side information, e.g., one can extract a graph from the position of each sensor and their reciprocal physical proximity or the structure of the physical system where sensors are placed. In other cases instead, functional dependencies can be inferred directly from data by exploiting some affinity score (e.g., Pearson correlation, Granger causality [46], correntropy [47], etc.), or more advanced techniques (e.g., graph learning methods [48]). We model the extracted relational information with a weighted, possibly asymmetric adjacency matrix $\boldsymbol{A}_t \in \mathbb{R}^{N_t \times N_t}$, in which each nonzero entry $a_t^{i,j}$ indicates the weight of the edge going from the i-th node to the j-th. While our framework can account for dynamic relationships, we mostly focus on settings where the topology is static, i.e., $\boldsymbol{A}_t = \boldsymbol{A}$ and $N_t = N$. Finally, we assume to have available sensor-level covariates $\boldsymbol{Q}_t \in \mathbb{R}^{N \times d_q}$ that act as spatiotemporal coordinates to localize a point in time and space (e.g., date/time features and geographic location). Note that coordinates $\boldsymbol{q}_t^i$ are assumed available for each node at each time step; in section 4 we discuss how $\boldsymbol{Q}_t$ can be learned as a spatiotemporal positional encoding. We then model observations as a discrete sequence of spatiotemporal graphs, where each graph is a triplet $\mathcal{G}_t = \langle \boldsymbol{X}_t, \boldsymbol{Q}_t, \boldsymbol{A} \rangle$. Note that, in general, we indicate with capital letters network-level attributes, while we use the lowercase for the sensor level.

As already mentioned, observations might be incomplete due to faults, partial observability, and costs of the data acquisition process. We model data availability with a binary mask $\boldsymbol{m}_t^i \in \{0, 1\}$ which is

1 if the measurements associated with the i-th sensor are valid at time step t. Conversely, if $\boldsymbol{m}_t^i = 0$, we consider the measurements $\boldsymbol{x}_t^i$ to be completely missing, with the exogenous variables $\boldsymbol{q}_t^i$ being instead available. Notice that usually each $\boldsymbol{m}_t^i$ are realizations of random variables with non-trivial distributions exhibiting complex spatiotemporal dynamics and correlations. For instance, a sensor fault may last for more than one time step, or spatially correlated errors may result in missing data in an entire sub-region of the network.

Multivariate Time Series Imputation Given $\mathcal{G}_{t:t+T}$ with mask $\boldsymbol{M}_{t:t+T}$, we denote by $\widetilde{\boldsymbol{X}}_{t:t+T}$ the unknown corresponding complete sequence, i.e., the sequence where no observation is missing. Formally, the goal of multivariate time series imputation (MTSI) is to find an estimate $\widehat{\boldsymbol{X}}_{t:t+T}$ minimizing the reconstruction error

$$\mathcal{L}\left(\widehat{\boldsymbol{X}}_{t:t+T}, \widetilde{\boldsymbol{X}}_{t:t+T}, \boldsymbol{M}_{t:t+T}\right) = \sum_{\tau=t}^{t+T} \frac{\sum_{i=1}^{N} \overline{\boldsymbol{m}}_\tau^i \cdot \ell\left(\hat{\boldsymbol{x}}_\tau^i, \tilde{\boldsymbol{x}}_\tau^i\right)}{\sum_{i=1}^{N} \overline{\boldsymbol{m}}_\tau^i}, \tag{1}$$

where $\ell(\,\cdot\,,\,\cdot\,)$ is an element-wise error function and $\overline{\boldsymbol{m}}_\tau^i$ is the logical binary complement of $\boldsymbol{m}_\tau^i$. Notice that, since $\widetilde{\boldsymbol{X}}_{t:t+T}$ is not available, one should find a surrogate objective or simulate the presence of missing data, for which the reconstruction error can be computed.

4 Learning representations from sparse spatiotemporal data

The autoregressive approach to reconstruction consists in directly modeling distributions $p\big(\boldsymbol{x}_t^i \mid \boldsymbol{X}_{<t}\big)$ and using one-step-ahead forecasting as a surrogate objective to learn how to recover missing observations. To exploit available data subsequent to the target time step, it is common to use a bidirectional architecture, i.e., to mirror time w.r.t. the time step of interest and have a second model for estimating $p\big(\boldsymbol{x}_t^i \mid \boldsymbol{X}_{>t}\big)$ [49, 6]. Moreover, a third component $p\big(\boldsymbol{x}_t^i \mid \{\boldsymbol{x}_t^{j\neq i}\}\big)$ must be introduced to account for spatial information at each step. Architectures like GRIN, follow exactly this scheme with different components dedicated to model each of these three aspects. While being effective in practice, these approaches can have multiple drawbacks. Besides the computational overhead of having three separate components and the compounding error in the autoregressive models, they can fall short in capturing global context, as the processing of the structural information is decomposed. Furthermore, merging the information coming from the different modules is also problematic, yielding to further compounding of errors. Finally, in the case of highly sparse observations, the spatial processing should be dealt with special care as propagating information through partially observed spatiotemporal graphs adds another layer of complexity. For instance, simply masking out faulty sensors would compromise the flow of information through message passing layers [9].

Model overview To address the limitation of existing methods, we act on the problem more directly by aligning the structure of our proposed architecture closely with the reconstruction task. We denote as *observed set* $\mathcal{X}_{t:t+T} = \big\{\big\langle \boldsymbol{x}_\tau^i, \boldsymbol{q}_\tau^i \big\rangle \mid \boldsymbol{m}_\tau^i = 1 \wedge \tau \in [t, t+T)\big\}$ the set of all observations, paired with their spatiotempotal coordinates. Conversely, we name *target set* $\mathcal{Y}_{t:t+T} = \big\{\boldsymbol{q}_\tau^i \mid \boldsymbol{m}_\tau^i = 0 \wedge \tau \in [t, t+T)\big\}$ the complement set collecting the coordinates of the discrete spatiotemporal points for which we want to reconstruct an observation. We refer to the set of observed and target points of the i-th node as $\mathcal{X}_{t:t+T}^i$ and $\mathcal{Y}_{t:t+T}^i$, respectively. Then, for all $\boldsymbol{q}_\tau^i \in \mathcal{Y}_{t:t+T}$, we aim at learning a structured model for

$$p\big(\boldsymbol{x}_\tau^i \mid \boldsymbol{q}_\tau^i, \mathcal{X}_{t:t+T}, \boldsymbol{A}\big). \tag{2}$$

Our approach, named *Spatiotemporal Point Inference Network* (SPIN), is a graph attention network for MTSI, designed to learn representations of discrete points associated with nodes of a sequence of spatiotemporal graphs. Given disjoint observed and target sets $\mathcal{X}_{t:t+T}$ and $\mathcal{Y}_{t:t+T}$, SPIN is trained to learn a model

$$f_\theta(\boldsymbol{q}_\tau^i \mid \mathcal{X}_{t:t+T}, \boldsymbol{A}) \approx \mathbb{E}\left[p\big(\boldsymbol{x}_\tau^i \mid \boldsymbol{q}_\tau^i, \mathcal{X}_{t:t+T}, \boldsymbol{A}\big)\right] \tag{3}$$

for all discrete points $\boldsymbol{x}_\tau^i \in \mathcal{Y}_{t:t+T}$. To this end, SPIN learns a parameterized propagation process where each representation, corresponding to a specific node and time step, is updated by aggregating

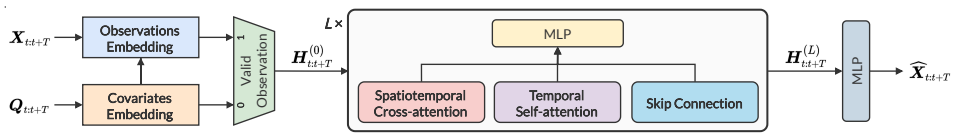

Figure 1: The architecture of SPIN. At first, we encode observations $\boldsymbol{X}_{t:t+T}$ and spatiotemporal coordinates $\boldsymbol{Q}_{t:t+T}$, obtaining initial representations $\boldsymbol{H}^{(0)}_{t:t+T}$. The representations are updated by a stack of L sparse spatiotemporal attention blocks. Final imputations are obtained from $\boldsymbol{H}^{(L)}_{t:t+T}$ with a nonlinear readout.

information from all the available observations acquired at neighboring nodes weighted by input-dependent attention coefficients. Figure 1 shows an overview of the architecture. In the next paragraphs we will go through each component explaining in details and providing motivations for each design choice. We start by describing how we set up the propagation process.

Sparse spatiotemporal attention The core component of SPIN is a novel *sparse spatiotemporal attention* layer used to propagate information at the level of single observations. Indeed, leveraging on the attention mechanism, we learn representations for each i-th node at each τ-th time step by simultaneously aggregating information from (1) the observed set of i-th node $\mathcal{X}^i_{t:t+T}$ (2) the observed set $\mathcal{X}^j_{t:t+T}$ of its neighbors $j \in \mathcal{N}(i)$. Figure 2 shows a schematic representation of this procedure.

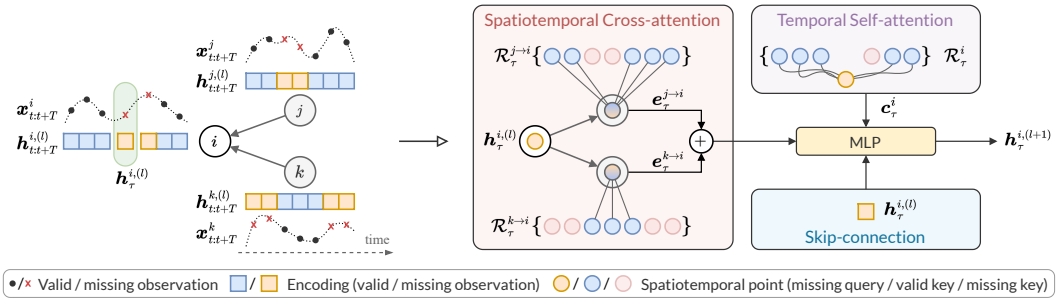

Figure 2: Example of the sparse spatiotemporal attention layer. On the left, the input spatiotemporal graph, with time series associated with every node. On the right, how the layer acts to update target representation $\boldsymbol{h}^{i,(l)}_\tau$ (highlighted by the green box), by simultaneously performing inter-node spatiotemporal cross-attention (red block) and intra-node temporal self-attention (violet block).

Let $\boldsymbol{h}^{i,(l)}_\tau \in \mathbb{R}^{d_h}$ be the learned representation for the i-th node and time step τ at the l-th layer. The encoding is initialized as

$$\boldsymbol{h}^{i,(0)}_\tau = \begin{cases} \mathrm{MLP}\big(\boldsymbol{q}^i_\tau\big) & \boldsymbol{q}^i_\tau \in \mathcal{Y}_{t:t+T} \\ \mathrm{MLP}\big(\boldsymbol{x}^i_\tau, \boldsymbol{q}^i_\tau\big) & \big\langle \boldsymbol{x}^i_\tau, \boldsymbol{q}^i_\tau \big\rangle \in \mathcal{X}_{t:t+T} \end{cases} \tag{4}$$

where $\boldsymbol{q}^i_\tau$ indicates a positional encoding, as previously anticipated, and MLP is a generic multi-layer perceptron. The representation is then updated by joint temporal and spatiotemporal attention operations. To describe inner working of the proposed attention mechanism we adopt the terminology of [50] and indicate as *query* the token for which we want to compute an updated representation, *key* a representation of the source tokens, and *value* the content representation of each token. The next steps involve computations of spatiotemporal messages, i.e., representations computed to propagate information from one discrete space-time point to another. We indicate the propagation along the temporal dimension from time step s to time step τ as subscripts $s \to \tau$. Conversely, superscripts $j \to i$ indicate messages sent from the j-th node to the i-th. To avoid overloading the notation, we omit the layer superscript in the following. The message $\boldsymbol{r}^{j\to i}_{s\to\tau} \in \mathbb{R}^{d_h}$ from the j-th node at time step s to the i-th node at time step τ is computed as

$$\boldsymbol{r}^{j\to i}_{s\to\tau} = \mathrm{MLP}\big(\boldsymbol{h}^j_s, \boldsymbol{h}^i_\tau\big), \tag{5}$$

where, following the adopted terminology, target point representation $\boldsymbol{h}_\tau^i$ acts as query, source point representation $\boldsymbol{h}_s^j$ as key and the computed message $\boldsymbol{r}_{t\to\tau}^{j\to i}$ is the value. Notice that, differently from the dot-product attention mechanism by Vaswani et al. [50], we take into account both source and target representations to compute the value, similarly to the Bahdanau attention [51]. To account for spatial information, this mechanism is used to perform an *inter-node* temporal cross-attention. More precisely, for every neighbor $j \in \mathcal{N}(i)$, we use $\boldsymbol{h}_\tau^i$ to query – separately – every encoding in $\boldsymbol{h}_{t:t+T}^j$ associated with a valid observation, and collect the messages in set

$$\mathcal{R}_\tau^{j\to i} = \{\boldsymbol{r}_{s\to\tau}^{j\to i} \mid \langle \boldsymbol{x}_s^j, \boldsymbol{q}_s^j \rangle \in \mathcal{X}_{t:t+T}\}. \tag{6}$$

Then, we linearly transform messages in $\mathcal{R}_\tau^{j\to i}$ (using trainable weights $\boldsymbol{w}_\alpha \in \mathbb{R}^{d_h \times 1}$) and obtain *message scores* $\alpha_{s\to\tau}^{j\to i}$ with a softmax layer, i.e.,

$$\alpha_{s\to\tau}^{j\to i} = \frac{\exp(\boldsymbol{r}_{s\to\tau}^{j\to i}\boldsymbol{w}_\alpha)}{\sum_{\boldsymbol{r}\in\mathcal{R}_\tau^{j\to i}} \exp(\boldsymbol{r}\boldsymbol{w}_\alpha)}\,. \tag{7}$$

Then, we aggregate temporal messages coming from each j-th node separately, and obtain an edge-level context vector $\boldsymbol{e}_\tau^{j\to i}$

$$\boldsymbol{e}_\tau^{j\to i} = \sum_{\boldsymbol{r}_{s\to\tau}^{j\to i}\in\mathcal{R}_\tau^{j\to i}} \alpha_{s\to\tau}^{j\to i} \cdot \boldsymbol{r}_{s\to\tau}^{j\to i}, \tag{8}$$

which encodes the observed sequence at each j-th node w.r.t. the i-th node and time step τ. Analogously, to account for the observed sequence of the i-th node itself, we exploit an *intra-node* temporal self-attention mechanism to query the encodings $\boldsymbol{h}_{t:t+T}^i$ corresponding to valid observations. Then, resulting representations are weighted and aggregated to obtain a temporal context vector $\boldsymbol{c}_\tau^i$ as

$$\boldsymbol{r}_{s\to\tau}^i = \mathrm{MLP}\big(\boldsymbol{h}_s^i, \boldsymbol{h}_\tau^i\big) \tag{9}$$

$$\mathcal{R}_\tau^i = \{\boldsymbol{r}_{s\to\tau}^i \mid \langle \boldsymbol{x}_s^i, \boldsymbol{q}_s^i \rangle \in \mathcal{X}_{t:t+T}\} \tag{10}$$

$$\boldsymbol{c}_\tau^i = \sum_{\boldsymbol{r}_{s\to\tau}^i\in\mathcal{R}_\tau^i} \alpha_{s\to\tau}^i \cdot \boldsymbol{r}_{s\to\tau}^i \tag{11}$$

where message score $\alpha_{s\to\tau}^i$ is obtained similarly as Eq. (7). Note that parameters are not shared between the cross-attention and self-attention blocks. After having obtained context vectors $\boldsymbol{c}_{\tau,(l)}^i$ and $\boldsymbol{e}_\tau^{j\to i,(l)}$, the encoding $\boldsymbol{h}_\tau^{i,(l)}$ is updated with a final aggregation step as

$$\boldsymbol{h}_\tau^{i,(l+1)} = \mathrm{MLP}\Big(\boldsymbol{h}_\tau^{i,(l)},\ \boldsymbol{c}_\tau^{i,(l)},\ \sum\nolimits_{j\in\mathcal{N}(i)} \boldsymbol{e}_\tau^{j\to i,(l)}\Big). \tag{12}$$

After L layers, we obtain imputations for all spatiotemporal points in $\mathcal{Y}_{t:t+T}$ with a nonlinear readout

$$\widehat{\mathcal{Y}}_{t:t+T} = \{\hat{\boldsymbol{x}}_\tau^i = \mathrm{MLP}\big(\boldsymbol{h}_\tau^{i,(L)}\big) \mid \boldsymbol{q}_\tau^i \in \mathcal{Y}_{t:t+T}\}. \tag{13}$$

Two-phase propagation Masking out tokens in the target set allows SPIN to propagate only valid information. As a downside, this results in blocking the flow of information on paths through points in the target set. This can be problematic when the input observations are extremely sparse. Nonetheless, it is reasonable to assume that, after only a few propagation steps, the available information has already been partially diffused to locations with missing observations. At this point, interrupted paths can be unlocked, allowing for reaching higher-order neighborhoods. In practice, we introduce a hyperparameter η to control the number of layers with masked connections and effectively split the propagation process into two phases. It is important to notice that what is being propagated in the second phase are learned representations, not observations (unavailable for masked tokens).

Graph subsampling and hierarchical attention Roughly speaking, the proposed spatiotemporal attention mechanism can be viewed as performing attention over the spatiotemporal graph $\mathcal{S}$, obtained by considering the product graph between space and time dimensions – with some connections pruned w.r.t. unavailable data. Let $N_{\max}, E_{\max}$ be the largest number of nodes and edges, respectively, among graphs in $\mathcal{G}_{t:t+T}$. Performing graph attention on the surrogate graph $\mathcal{S}$ has time and memory

complexities that scale with $O((N_{\max} + E_{\max})T^2)$ (sparse implementations can alleviate this complexity by replacing T^2 with the number of valid time step pairs). To reduce this computational burden – which undermines the application of the proposed method to large graphs and long time horizons – we propose two different approaches. The straightforward approach consists in training the model by exploiting graph subsampling, using one of the many possible subsampling strategies from the literature (e.g., [52]). In practice, at training time, we sample a k-hop subgraph centered on n target nodes and then compute the loss only w.r.t. these n nodes. In this way, we can reduce the amount of computation required by acting on n and k. This type of subsampling can also be seen as a form of regularization [53].

A more interesting and orthogonal approach to reduce complexity is to rewire the attention mechanism to be hierarchical [54]. We do this by adding K dummy nodes that act as hubs for propagating information. Let $\boldsymbol{Z}^i \in \mathbb{R}^{K \times d_z}$ be the the hub nodes' representations for central node i, and then, for hub k:

1. update $\boldsymbol{z}_k^i$ by querying $\{\boldsymbol{h}_\tau^i \mid \langle \boldsymbol{x}_\tau^i, \boldsymbol{q}_\tau^i \rangle \in \mathcal{X}_{t:t+T}\}$, i.e., node encodings associated with valid observations, obtaining $\tilde{\boldsymbol{z}}_k^i$;
2. update node encoding $\boldsymbol{h}_\tau^i$ by querying updated $\widetilde{\boldsymbol{Z}}^i$ and $\widetilde{\boldsymbol{Z}}^j$ of every j-th neighbor in $\mathcal{N}(i)$.

The spatiotemporal attention is effectively split in two phases. At first, we update each hub node representation similarly as Eq. (9-11):

$$\boldsymbol{r}_{\tau,k}^i = \mathrm{MLP}\big(\boldsymbol{h}_\tau^i, \boldsymbol{z}_k^i\big) \tag{14}$$

$$\mathcal{R}_k^i = \{\boldsymbol{r}_{\tau,k}^i \mid \langle \boldsymbol{x}_\tau^i, \boldsymbol{q}_\tau^i \rangle \in \mathcal{X}_{t:t+T}\} \tag{15}$$

$$\boldsymbol{c}_k^i = \sum\nolimits_{\boldsymbol{r}_{\tau,k}^i \in \mathcal{R}_k^i} \alpha_{\tau,k}^i \cdot \boldsymbol{r}_{\tau,k}^i \tag{16}$$

$$\tilde{\boldsymbol{z}}_k^i = \mathrm{MLP}\big(\boldsymbol{z}_k^i, \boldsymbol{c}_k^i\big) \tag{17}$$

Then, we obtain context vectors from the updated hub representations as:

$$\boldsymbol{r}_{k,\tau}^i = \mathrm{MLP}\big(\tilde{\boldsymbol{z}}_k^i, \boldsymbol{h}_\tau^i\big) \tag{18}$$

$$\boldsymbol{c}_\tau^i = \sum\nolimits_k \alpha_{k,\tau}^i \cdot \boldsymbol{r}_{k,\tau}^i \tag{19}$$

$$\boldsymbol{r}_{k,\tau}^{j \to i} = \mathrm{MLP}\big(\tilde{\boldsymbol{z}}_k^j, \boldsymbol{h}_\tau^i\big) \tag{20}$$

$$\boldsymbol{e}_\tau^{j \to i} = \sum\nolimits_k \alpha_{k,\tau}^{j \to i} \cdot \boldsymbol{r}_{k,\tau}^{j \to i} \tag{21}$$

and update node representation $\boldsymbol{h}_\tau^i$ as in Eq. (12). In this way, we can reduce the spatiotemporal attention complexity to $O((N_{\max} + E_{\max})KT)$ with $K \ll T$, at the cost of introducing an information bottleneck. We initialize the hub representations at layer $l = 0$ with random trainable parameters.

Spatiotemporal positional encoding We now come back to the problem of learning the spatiotemporal positional encodings $\boldsymbol{Q}_{t:t+T}$. The encoding has to capture both spatial and temporal structure and be decoupled from the observations $\boldsymbol{x}_t^i$ to allow inference for points in the target set. These encodings can be directly extracted from available exogenous information (e.g., sensor location and date/time features) or learned end-to-end jointly with the other model parameters. We propose a spatiotemporal positional encoding $\boldsymbol{q}_t^i = \rho(\boldsymbol{u}_t, \boldsymbol{v}^i)$ obtained by combining a *temporal encoding* $\boldsymbol{U} \in \mathbb{R}^{T \times d_u}$ and a *spatial encoding* $\boldsymbol{V} \in \mathbb{R}^{N \times d_v}$ with a non linear transformation ρ, e.g.,

$$\boldsymbol{q}_t^i = \mathrm{MLP}\big(\boldsymbol{u}_t, \boldsymbol{v}^i\big). \tag{22}$$

For the temporal encoding $\boldsymbol{u}_t$, we use sine and cosine transforms of the time step t w.r.t. a period of interest (e.g., day and/or week), to account for seasonalities. For the spatial encoding $\boldsymbol{v}^i$, we resort to a vector of learnable parameters different for each i-th node. More advanced methods could be considered [55, 56].

5 Empirical evaluation

In this section, we evaluate our method on three real-world datasets and compare the performance against state-of-the-art methods and standard approaches for MTSI. As the objective of our approach is to address the imputation problem in highly sparse settings, in a second experiment we assess how performance changes as the percentage of missing values increases.

Table 1: Performance (in terms of MAE) averaged over multiple independent runs.

	Block missing		Point missing		Simulated failures	
	PEMS-BAY	METR-LA	PEMS-BAY	METR-LA	AQI-36	AQI
`Mean`	5.46 ± 0.00	7.48 ± 0.00	5.42 ± 0.00	7.56 ± 0.00	53.48 ± 0.00	39.60 ± 0.00
`KNN`	4.30 ± 0.00	7.79 ± 0.00	4.30 ± 0.00	7.88 ± 0.00	30.21 ± 0.00	34.10 ± 0.00
`MF`	3.28 ± 0.01	5.46 ± 0.02	3.29 ± 0.01	5.56 ± 0.03	30.54 ± 0.26	26.74 ± 0.24
`MICE`	2.94 ± 0.02	4.22 ± 0.05	3.09 ± 0.02	4.42 ± 0.07	30.37 ± 0.09	26.98 ± 0.10
`VAR`	2.09 ± 0.10	3.11 ± 0.08	1.30 ± 0.00	2.69 ± 0.00	15.64 ± 0.08	22.95 ± 0.30
`rGAIN`	2.18 ± 0.01	2.90 ± 0.01	1.88 ± 0.02	2.83 ± 0.01	15.37 ± 0.26	21.78 ± 0.50
`BRITS`	1.70 ± 0.01	2.34 ± 0.01	1.47 ± 0.00	2.34 ± 0.00	14.50 ± 0.35	20.21 ± 0.22
`SAITS`	1.56 ± 0.01	2.30 ± 0.01	1.40 ± 0.03	2.26 ± 0.00	18.16 ± 0.42	21.33 ± 0.15
Transformer	1.70 ± 0.02	3.54 ± 0.00	0.74 ± 0.00	2.16 ± 0.00	11.98 ± 0.53	18.11 ± 0.25
`GRIN`	1.14 ± 0.01	2.03 ± 0.00	**0.67 ± 0.00**	**1.91 ± 0.00**	12.08 ± 0.47	14.73 ± 0.15
SPIN	**1.06 ± 0.01**	**1.97 ± 0.01**	0.71 ± 0.01	**1.90 ± 0.01**	**11.77 ± 0.74**	**14.00 ± 0.13**
SPIN-H	**1.06 ± 0.01**	2.05 ± 0.03	0.74 ± 0.02	1.96 ± 0.04	**11.08 ± 0.06**	14.39 ± 0.03

5.1 Experimental setting

In following experiments, we consider both **SPIN** and the hierarchical version **SPIN-H** (Sec. 4). The figure of merit used is the *mean absolute error* (MAE), averaged across imputation windows. We consider only the *out-of-sample* scenario [4], in which every parametric model is trained and tested on disjoint sets. All the baselines have been implemented in PyTorch [57] using, whenever available, public code provided by the authors. Please refer to the appendix for more details about the experimental setup.

Datasets We consider three openly available datasets coming from real-world SNs. The first two, namely **PEMS-BAY** and **METR-LA** [2], are two widely used benchmarks in spatiotemporal forecasting literature. Each of them records traffic measurements every 5 minutes from 325 speed sensors in San Francisco Bay Area and 207 in Los Angeles County Highway, respectively. Since the original datasets have a low number of missing values, we use the same setup of [4] to inject missing data with two different policies: 1) *Block missing*, in which we randomly mask out 5% of the available data and, in addition, we simulate a failure lasting for $S \sim \mathcal{U}(12, 48)$ steps with 0.15% probability; 2) *Point missing*, in which we randomly drop 25% of the available data. As a third dataset, we consider **AQI** [58], which collects one year of hourly measurements of air pollutants from 437 air quality monitoring stations over 43 cities in China. We consider also a smaller version of this dataset (**AQI-36**) with only the 36 sensors scattered over the city of Beijing. This dataset is a popular benchmark for imputation for the high number of missing values (25.67% in the complete dataset) and provides a mask for evaluation that simulates the true missing data distribution [15]. For a given month, such a mask replicates the missing values patterns of the previous month, making this scenario more similar to the block missing setting, for a total of $\approx 36\%$ missing data. In all settings, all the valid observations masked out are used as targets for evaluation. We obtain an adjacency matrix from the pairwise distance of sensors following previous works [2–4].

Baselines As the target of our approach are spatiotemporal graphs with missing observations, we compare our methods against `GRIN` [4], a graph-based bidirectional RNN for MTSI with state-of-the-art performance. We then consider a spatiotemporal Transformer, where we alternate temporal and spatial Transformer encoder layers from [50] and replace missing values with a `[MASK]` token (as in [12]). We consider also other deep imputation methods: 1) `SAITS` [25], a recent attention-based architecture based on diagonally-masked self-attention; 2) `BRITS` [6], which leverages on a bidirectional RNN; 3) `rGAIN`, an adversarial approach which shares similarities with GAIN [23] and SSGAN [21]. Finally, we report results of simpler methods that imputes missing values using 4) node-level sequence mean (`MEAN`) or 5) neighbors mean (`KNN`); 6) Matrix Factorization (`MF`); 7)

Table 2: Performance (MAE) with increasing data sparsity (average over 5 evaluation masks).

	METR-LA			PEMS-BAY			AQI		
	50 %	75 %	95 %	50 %	75 %	95 %	50 %	75 %	95 %
BRITS	2.52 ± 0.00	3.02 ± 0.00	5.19 ± 0.02	1.55 ± 0.00	2.17 ± 0.00	3.91 ± 0.02	14.90 ± 0.03	18.29 ± 0.03	29.83 ± 0.07
SAITS	2.48 ± 0.00	3.74 ± 0.01	6.72 ± 0.01	1.50 ± 0.00	2.96 ± 0.01	7.40 ± 0.01	15.36 ± 0.02	20.64 ± 0.05	34.57 ± 0.05
Transformer	2.31 ± 0.00	2.71 ± 0.00	5.13 ± 0.01	0.85 ± 0.00	1.13 ± 0.00	2.70 ± 0.01	9.11 ± 0.02	12.56 ± 0.05	25.65 ± 0.11
GRIN	2.05 ± 0.00	2.39 ± 0.00	4.08 ± 0.02	**0.79 ± 0.00**	1.09 ± 0.00	2.70 ± 0.01	8.43 ± 0.01	10.97 ± 0.02	20.38 ± 0.10
SPIN	2.02 ± 0.00	2.24 ± 0.00	2.89 ± 0.01	**0.79 ± 0.00**	1.00 ± 0.00	1.71 ± 0.00	**8.15 ± 0.01**	**9.96 ± 0.02**	**15.51 ± 0.08**
SPIN-H	**2.01 ± 0.00**	**2.20 ± 0.00**	**2.82 ± 0.00**	**0.79 ± 0.00**	**0.97 ± 0.00**	**1.68 ± 0.00**	8.67 ± 0.02	10.27 ± 0.02	15.75 ± 0.07

MICE [59]; 8) VAR, a vector autoregressive one-step-ahead predictor. Whenever possible, we use results from [4].

5.2 Results

Table 1 reports the experiments results. Both SPIN methods outperform the baselines in almost all scenarios. Not surprisingly, improvements are more evident when entire blocks of data are missing, as in AQI datasets and block missing settings. Differently from autoregressive methods, the sparse spatiotemporal attention mechanism of SPIN, in fact, is not autoregressive and allows an unbounded memory capacity. Conversely, in the point missing setting, SPIN methods perform on par with the state of the art. With respect to the spatiotemporal Transformer, SPIN performs better in all settings except for AQI-36, which can be attributed to the ineffectiveness of spatial attention alone in determining the dependencies among the different spatial locations. Notice also that in almost all cases SPIN-H performs on par with SPIN (even better in the smaller dataset AQI-36), making it a valid lightweight alternative to SPIN.

To further show the advantages of SPIN in sparsely observed structured sequences, we compare our methods against the most relevant baselines on the same datasets by progressively increasing the sparseness in the input data distribution. In practice, we test the models on the same test horizon of the previous experiment but, instead of using the previous evaluation masks, we reduce the number of available observations by associating a probability p of being removed to each observation. The higher is p, the more likely it is to have blocks of consecutive missing values. For the traffic datasets, we use the weights of the models trained on the point missing setting. Table 2 shows results (in terms of MAE) averaged over 5 different evaluation masks for each missing probability p. In all the considered scenarios, SPIN-based models rank as best performing methods for any sparsity level. Moreover, they are more robust to changes in missing data distribution. In fact, compared to the baselines, the performance of both SPIN and SPIN-H deteriorates at a slower rate as the sparsity data increases. Notably, GRIN would require much more data to match the performance that SPIN is able to achieve with only 5% of valid observations. Improvements over attention-based methods SAITS and Transformer are also striking.

6 Conclusion and Future Works

We introduced a graph-based architecture, named SPIN, to reconstruct missing observations in sparse spatiotemporal time series. To overcome the major limitations of autoregressive methods, we designed a novel sparse spatiotemporal attention mechanism to propagate valid observations through discrete points in time and space, jointly. Furthermore, we showed how the time and space complexities of the approach can be drastically reduced considering a novel hierarchical attention mechanism. Empirical analysis shows that the proposed method outperforms by a wide margin state-of-the-art methods for imputation in highly sparse settings. We noticed that in some extremely sparse *training* settings, SPIN might suffer from a lack of supervision that may slow down learning; future works might try to address this problem by introducing an additional auxiliary learning task. Other possible extensions could investigate spatiotemporal positional encoding methods and their application in SNs.

References

[1] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, pages 362–373. Springer, 2018.

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

[3] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121*, 2019.

[4] Andrea Cini, Ivan Marisca, and Cesare Alippi. Filling the g_ap_s: Multivariate time series imputation by graph neural networks. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=kOu3-S3wJ7`.

[5] Jinsung Yoon, William R Zame, and Mihaela van der Schaar. Multi-directional recurrent neural networks: A novel method for estimating missing data. In *Time Series Workshop at the 34th International Conference on Machine*, pages 1–5, 2017.

[6] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Yitan Li, and Lei Li. Brits: bidirectional recurrent imputation for time series. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6776–6786, 2018.

[7] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[8] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34 (4):18–42, 2017.

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

[10] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[11] David Ha and Yujin Tang. Collective intelligence for deep learning: A survey of recent developments. *arXiv preprint arXiv:2111.14377*, 2021.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*. Oxford university press, 2012.

[14] J Kihoro, K Athiany, et al. Imputation of incomplete nonstationary seasonal time series data. *Mathematical Theory and Modeling*, 3(12):142–154, 2013.

[15] Xiuwen Yi, Yu Zheng, Junbo Zhang, and Tianrui Li. 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, page 2704–2710. AAAI Press, 2016. ISBN 9781577357704.

[16] Lorenzo Beretta and Alessandro Santaniello. Nearest neighbor imputation algorithms: a critical evaluation. *BMC medical informatics and decision making*, 16(3):197–208, 2016.

[17] Andrzej Cichocki and Anh-Huy Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 92(3):708–721, 2009.

[18] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. Temporal regularized matrix factorization for high-dimensional time series prediction. *Advances in neural information processing systems*, 29:847–855, 2016.

[19] Nikhil Rao, Hsiang-Fu Yu, Pradeep Ravikumar, and Inderjit S Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In *NIPS*, volume 2, page 7. Citeseer, 2015.

[20] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12, 2018.

[21] Xiaoye Miao, Yangyang Wu, Jun Wang, Yunjun Gao, Xudong Mao, and Jianwei Yin. Generative semi-supervised learning for multivariate time series imputation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8983–8991, 2021.

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

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

[24] Yonghong Luo, Ying Zhang, Xiangrui Cai, and Xiaojie Yuan. E²gan: End-to-end generative adversarial network for multivariate time series imputation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3094–3100. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[25] Wenjie Du, David Côté, and Yan Liu. Saits: Self-attention-based imputation for time series. *arXiv preprint arXiv:2202.08516*, 2022.

[26] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34, 2021.

[27] Satya Narayan Shukla and Benjamin Marlin. Multi-time attention networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2020.

[28] Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

[29] Zijie Huang, Yizhou Sun, and Wei Wang. Learning continuous system dynamics from irregularly-sampled partial observations. *Advances in Neural Information Processing Systems*, 33:16177–16187, 2020.

[30] Yukai Liu, Rose Yu, Stephan Zheng, Eric Zhan, and Yisong Yue. Naomi: Non-autoregressive multiresolution sequence imputation. *Advances in Neural Information Processing Systems*, 32: 11238–11248, 2019.

[31] Siyuan Shan, Yang Li, and Junier B Oliva. Nrtsi: Non-recurrent time series imputation. *arXiv preprint arXiv:2102.03340*, 2021.

[32] Yuebing Liang, Zhan Zhao, and Lijun Sun. Dynamic spatiotemporal graph convolutional neural networks for traffic data imputation with complex missing patterns. *arXiv preprint arXiv:2109.08357*, 2021.

[33] Yongchao Ye, Shiyao Zhang, and James JQ Yu. Spatial-temporal traffic data imputation via graph attention convolutional network. In *International Conference on Artificial Neural Networks*, pages 241–252. Springer, 2021.

[34] Sanmukh R. Kuppannagari, Yao Fu, Chung Ming Chueng, and Viktor K. Prasanna. Spatio-temporal missing data imputation for smart power grids. In *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, e-Energy '21, page 458–465, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383332. doi: 10.1145/3447555.3466586. URL `https://doi.org/10.1145/3447555.3466586`.

[35] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*, 2018.

[36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

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

[38] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1548–1554, 2021.

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

[40] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 922–929, 2019.

[41] Zonghan Wu, Da Zheng, Shirui Pan, Quan Gan, Guodong Long, and George Karypis. Traversenet: Unifying space and time in message passing. *arXiv preprint arXiv:2109.02474*, 2021.

[42] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[43] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks*, 129:249–260, 2020.

[44] Jiaxuan You, Xiaobai Ma, Daisy Yi Ding, Mykel Kochenderfer, and Jure Leskovec. Handling missing data with graph representation learning. *Neural Information Processing Systems (NeurIPS*, 2020.

[45] Emanuele Rossi, Henry Kenlay, Maria I Gorinova, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael Bronstein. On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. *arXiv preprint arXiv:2111.12128*, 2021.

[46] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438, 1969.

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

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

[49] Jinsung Yoon, William R Zame, and Mihaela van der Schaar. Estimating missing data in temporal data streams using multi-directional recurrent neural networks. *IEEE Transactions on Biomedical Engineering*, 66(5):1477–1490, 2018.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[51] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[52] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BJe8pkHFwS`.

[53] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Hkx1qkrKPr`.

[54] Joshua Ainslie, Santiago Ontañón, Chris Alberti, Philip Pham, Anirudh Ravula, and Sumit Sanghai. Etc: Encoding long and structured data in transformers. *CoRR*, abs/2004.08483, 2020. URL `https://arxiv.org/abs/2004.08483`.

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

[56] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=wTTjnvGphYj`.

[57] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

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

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

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

[61] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[62] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[63] Andrea Cini and Ivan Marisca. Torch Spatiotemporal, 3 2022. URL https://github.com/TorchSpatiotemporal/tsl.

[64] neptune.ai. Neptune: Metadata store for mlops, built for research and production teams that run a lot of experiments, 2021. URL https://neptune.ai.

[65] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[66] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 1999.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 4 and 5.
   (b) Did you describe the limitations of your work? [Yes] Assumptions discussed in Section 3, limits in Sections 4 and 6.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We do not see any potential harm to society in the proposed method.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide the experimental setup, as well as all datasets used, as supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] As supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] As supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] We provide links to datasets and code that contain also information about the license.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Detailed experimental setup

In this appendix, we discuss in details the experimental settings. We use the same setup of Cini et al. [4][1][2]. Indeed, Table 1 reports results from [4] whenever possible. We refer to [4] for details on these baselines.

For SPIN, we use the same hyperparameters in all datasets: $L = 4$ layers, with first $\eta = 3$ layers with masked connections; hidden size $d_h = 32$; 2 layers with hidden size 32 for every MLP; ReLU activation functions. For SPIN-H, we use similar hyperparameters, but 5 layers, with $\eta = 3$; $K = 4$ hubs per node with $d_z = 128$ units each. These hyperparameters have been selected among a small subset of options on the validation set; we expect far better performance to be achievable with further hyperparameter tuning. Depending on the dataset, the number of parameters ranges from $\approx 55K$ to $\approx 95K$ for SPIN and $\approx 540K$ to $\approx 800K$ for SPIN-H. We use Adam optimizer [60], learning rate $lr = 0.0008$ and a cosine scheduler with a warm-up of 12 steps and (partial) restarts every 100 epochs. We train our models with 300 minibatches of 8 random samples per epoch, fixing the maximum number of epochs to 300 and using early stopping on the validation set with patience of 40 epochs. Due to constraints on memory capacity on some of the GPUs (see the description of the hardware resources below), for SPIN-H we set the batch size to 6 and 16 in AQI and AQI-36, respectively.

To train SPIN-based models, we minimize the following loss function:

$$\mathcal{L} = \sum_{l=1}^{L} \frac{\sum_{\boldsymbol{q}_\tau^i \in \mathcal{Y}_{t:t+T}} \ell\left(\hat{\boldsymbol{x}}_\tau^{i,(l)}, \boldsymbol{x}_\tau^i\right)}{|\mathcal{Y}_{t:t+T}|}, \tag{23}$$

where $\ell(\,\cdot\,,\,\cdot\,)$ is the absolute error and $\hat{\boldsymbol{x}}_\tau^{i,(l)}$ is l-th layer imputation for the i-th node at time step τ. Note that, to provide more supervision to the architecture, the loss is computed and backpropagated w.r.t. representations learned at each layer, not only at the last one. The error is computed only on data not seen by the model at each forward pass. For this reason, we randomly remove p ratio of the input data for each minibatch sample, with p sampled uniformly from $[0.2, 0.5, 0.8]$, and use them to compute the loss. We never use data masked for evaluation to train any model.

For the spatiotemporal Transformer baseline, we use the same training strategy and a similar hyperparameters configuration of SPIN-H: $L = 5$ layers; 4 attention heads; hidden size and feed-forward size of 64 and 128 units, respectively. For SAITS, we use the code provided by the authors[3]. Hyperparameters for SAITS have been selected on the validation set with a random search by using hyperparameter ranges from the original paper.

We recall that the time and memory complexities of SPIN and SPIN-H scale with $O((N_{\max} + E_{\max})T^2)$ and $O((N_{\max} + E_{\max})KT)$, respectively. For the sake of comparison, here we also report the asymptotic complexities of the spatiotemporal Transformer and GRIN. The Transformer alternates temporal attention (i.e., $O(N_{\max}T^2)$) and spatial attention (i.e., $O(TN_{\max}^2)$), with a resulting $O((N_{\max} + T)N_{\max}T)$ complexity. Let R be the spatial receptive field (i.e., number of graph convolution layers) of the inner MPGRU cell, the time complexity required to process a single direction in GRIN scales with $O(TRE_{\max})$. Note also that while most of the operations in the attention-based models can be executed in parallel, GRIN would need to process the entire sequence recurrently, with a consequent performance slowdown at execution time.

All the models were developed in Python [61] using PyTorch [57], PyG [62] and Torch Spatiotemporal [63]. We use Neptune[4] [64] for experiments tracking. The code to reproduce the experiments of

[1] `https://github.com/Graph-Machine-Learning-Group/grin`
[2] `https://github.com/TorchSpatiotemporal/tsl`
[3] `https://github.com/WenjieDu/SAITS`
[4] `https://neptune.ai/`

the paper is available as supplementary material. All the experiments have been run in a cluster using GPU-enabled nodes with different hardware setups. Running times of SPIN-H training on a node equipped with a 12GB NVIDIA Titan V GPU range from 4 to 14 hours (depending on the dataset). For SPIN we used a node with 40GB NVIDIA A100 GPU, with running times ranging from 4 to 26 hours.

B Datasets

In this appendix, we provide details on datasets and preprocessing used for the experiments. We use temporal windows of $T = 24$ steps for all datasets except AQI-36, for which we set $T = 36$. For traffic datasets, we split the data sequentially as 70% for training, 10% for validation, and 20% for test. For air quality datasets, following Yi et al. [15], we consider as the test set the months of March, June, September, and December and we use valid observation $\boldsymbol{x}_\tau^i$ as ground-truth if the value is missing at the same hour and day in the following month. For data preprocessing we use the same approach of Cini et al. [4], by normalizing data across the feature dimension (graph-wise for graph-based models) to zero mean and unit variance.

In line with [3, 4], we obtain the adjacency matrix from the node pairwise geographical distances using a thresholded Gaussian kernel [65]

$$a^{i,j} = \begin{cases} \exp\left(-\frac{\mathrm{dist}(i,j)^2}{\gamma}\right) & \mathrm{dist}\,(i,j) \leq \delta \\ 0 & \text{otherwise} \end{cases}, \tag{24}$$

where $\mathrm{dist}\,(\,\cdot\,,\,\cdot\,)$ is the geographical distance operator, γ is a shape parameter and δ is the threshold.

C Virtual sensing

The spatiotemporal cross-attention mechanism allows SPIN variants to exploit valid observations at neighboring nodes to impute missing values at spatial locations where no sensor is available, thus performing *virtual sensing* (also referred to as *kriging* [66]). We assess the performance of SPIN-H in the virtual sensing task by completely masking out observations at two nodes during training and then using the trained model to infer the entire sequences at the masked nodes. For this experiment, we keep the same architecture hyperparameters and consider the dataset AQI-36, masking out the two nodes with the highest (station no. 1015) and lowest (no. 1032) number of neighbors, reproducing the experimental settings of [4]. Figure 3 shows the performance of SPIN-H in reconstructing a temporal window of 36 time steps for the two virtual sensors at test time. Results qualitatively show that our method is able to reconstruct observations for sensors with no available information other than their location in the network.

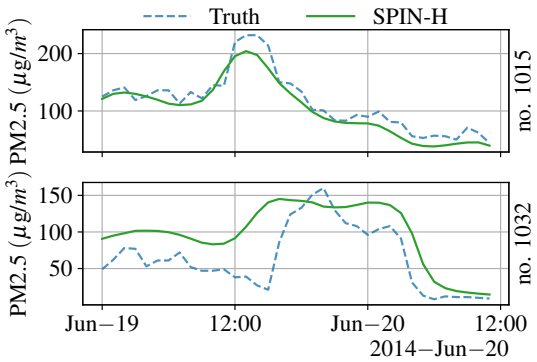

Figure 3: Visualization of completely missing sequences reconstructed by SPIN-H at nodes masked out during training. Imputations averaged over 5 independent runs.

D Ablation study

Table 3 shows the results of an ablation study on METR-LA (Point missing) and AQI-36. Here, we evaluate the performance in terms of mean absolute error (MAE) and mean relative error (MRE). We consider two different versions of SPIN-H in which we remove the spatiotemporal cross-attention

Table 3: Ablation study to assess the contribution of the single components in the spatiotemporal attention block. Performance averaged over multiple independent runs.

	METR-LA (P)		AQI-36	
	MAE	MRE (%)	MAE	MRE (%)
SPIN	**1.90** ± **0.01**	**3.29** ± **0.01**	**11.77** ± **0.74**	**16.56** ± **1.05**
SPIN-H	1.96 ± 0.04	3.39 ± 0.06	**11.08** ± **0.06**	**15.60** ± **0.09**
Without cross-attention	2.18 ± 0.01	3.78 ± 0.01	15.36 ± 0.09	21.62 ± 0.13
Without self-attention	2.24 ± 0.09	3.88 ± 0.16	13.63 ± 0.23	19.19 ± 0.32
Transformer	2.16 ± 0.00	3.74 ± 0.01	11.98 ± 0.53	16.87 ± 0.75

and the temporal self-attention components, respectively. We also report the performance of SPIN, SPIN-H and the Transformer for reference. Results clearly show that both components contribute positively to imputation accuracy. We also point out that in METR-LA (P) observations are masked out uniformly at random while the mask in AQI-36 reflects the empirical distribution of missing data in the dataset.

E Performance on data with simulated failures

In Table 2, we show the reconstruction error of the different methods as the number of valid observations in input sequences decreases. In detail, for every considered dataset, we remove observations uniformly at random and then compute the reconstruction error for the masked-out samples. To further assess the performance of our model, we show results for a different setting, where data are removed according to the *Block missing* policy (see section 5). In particular, we simulate the case in which, at each time step, every sensor has a constant probability $\bar{p}$ of going offline for a random number $S \sim \mathcal{U}(12, 36)$ of future (consecutive) time steps. Table 4 shows results for all datasets with $\bar{p} = 5\%$, $\bar{p} = 10\%$, and $\bar{p} = 15\%$. Note that, depending on the dataset, the portion of valid observations in each of these cases amounts to $\approx$ 25-30%, $\approx$ 8-10%, and $\approx$ 3-4%, respectively.

Table 4: Performance (MAE) with increasing number of simulated failures (average over 5 evaluation masks).

	METR-LA			PEMS-BAY			AQI		
	5 %	10 %	15 %	5 %	10 %	15 %	5 %	10 %	15 %
BRITS	5.87 ± 0.03	7.26 ± 0.06	8.29 ± 0.07	4.14 ± 0.05	5.41 ± 0.08	5.84 ± 0.04	24.09 ± 0.30	31.90 ± 0.26	37.62 ± 0.42
SAITS	4.73 ± 0.07	6.66 ± 0.05	7.27 ± 0.03	3.88 ± 0.09	7.62 ± 0.21	8.01 ± 0.11	20.78 ± 0.30	30.16 ± 0.39	36.34 ± 0.33
Transformer	6.03 ± 0.04	7.19 ± 0.05	8.06 ± 0.05	3.69 ± 0.06	5.09 ± 0.05	6.02 ± 0.04	29.21 ± 0.33	33.62 ± 0.16	37.31 ± 0.14
GRIN	3.05 ± 0.02	4.52 ± 0.05	5.82 ± 0.06	2.26 ± 0.03	3.45 ± 0.06	4.35 ± 0.04	15.62 ± 0.24	22.08 ± 0.39	29.03 ± 0.42
SPIN	2.71 ± 0.02	3.32 ± 0.02	3.87 ± 0.05	**1.78** ± **0.03**	**2.15** ± **0.03**	**2.41** ± **0.02**	**14.29** ± **0.24**	**18.71** ± **0.34**	**24.34** ± **0.46**
SPIN-H	**2.64** ± **0.02**	**3.17** ± **0.02**	**3.61** ± **0.04**	**1.75** ± **0.04**	**2.16** ± **0.03**	2.48 ± 0.02	**14.55** ± **0.26**	**19.37** ± **0.36**	25.38 ± 0.37

SPIN models, differently from the baselines, can handle all the considered scenarios. In particular, improvements in performance w.r.t. the best performing baseline (GRIN) are more evident as the number of available observations decreases.