Learning on Graphs Conference – Tromsø Meetup · February 18, 2026

Graph Deep Learning for Time Series Processing

Forecasting, Reconstruction and Analysis

Andrea **Cini**, Ivan **Marisca**, Daniele **Zambon**

The Swiss AI Lab IDSIA
Università della Svizzera italiana

Introduction

Traffic monitoring

Smart cities

Energy analitics

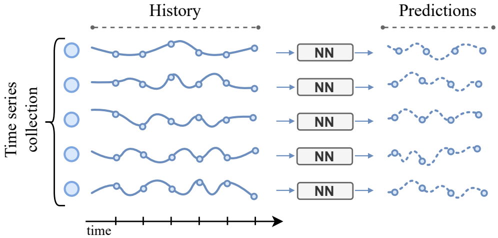
Physics

Stock markets

Deep learning for time series forecasting

Modern deep learning forecasting methods rely on a single neural network trained on a collection of related time series.

- ☺ Each time series is processed **independently**.
- ☺ Parameters are **shared**.
- ☺ Effective and **sample efficient**.
- ☹ **Dependencies are neglected**.



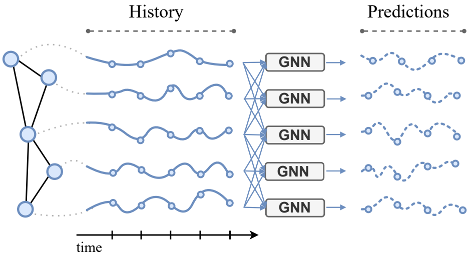
[1] Salinas *et al.*, “DeepAR: Probabilistic forecasting with autoregressive recurrent networks”, IJF 2020.

[2] Benidis *et al.*, “Deep Learning for Time Series Forecasting: Tutorial and Literature Survey”, ACM CS 2022.

Graph deep learning for time series forecasting

We will show graph deep learning (GDL) provides appropriate operators to go beyond these limitations.

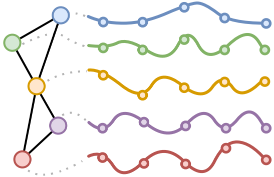
- ☺ **Dependencies** are **embedded into the processing** as inductive biases.
- ☺ Operate on **sets** of **correlated time series**.
- ☺ Parameters are **shared**.



☹ There are inherent **challenges** in applying this processing to data from the real world.

What this tutorial is about

This tutorial presents advances coming from the combination of



1. deep learning for time series and
2. deep learning on graphs.

The objective of the tutorial is to provide:

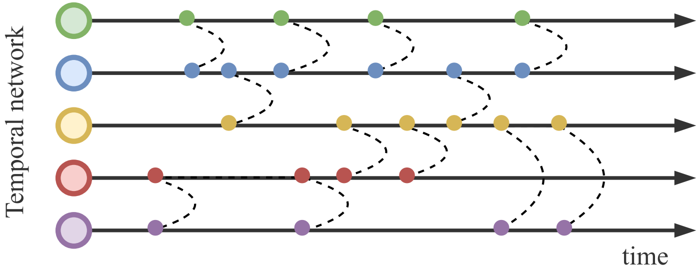
1. a comprehensive framework for graph-based time series processing models;
2. methods to address challenges and potential pitfalls;
3. tools and guidelines for real-world applications and developing new methods.

This presentation is complemented by a demo and a tutorial paper [3].

[3] Cini *et al.*, “Graph Deep Learning for Time Series Forecasting”, ACM Comput. Surv. 2025.

What this tutorial is not about

⚠ This tutorial is **not** about processing sequences of interactions in **temporal networks**.



→ Graphs will be a representation of the (dynamic) relationships among (irregular) time series.

Tutorial outline

Part 1

1.1) Correlated time series

1.2) Graph-based representation

1.3) STGNN architectures

1.4) Global and local models

Part 2

2.1) Dealing with missing data

2.2) Latent graph learning

2.3) Computational scalability

2.4) Model quality assessment

🏆 Additional Directions & Conclusions

</> Software demo

Part 1

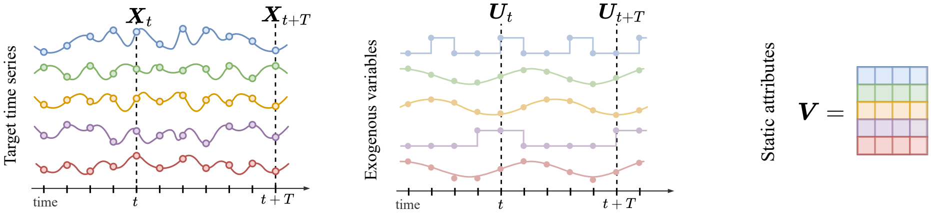
Graph-based Processing of Correlated Time series

Correlated time series

Collections of time series

We consider a set $\mathcal{D}$ of N correlated time series. Each i-th time series can be associated with:

- **observations** $x_t^i \in \mathbb{R}^{d_x}$ at each time step t;
- **exogenous variables** $u_t^i \in \mathbb{R}^{d_u}$ at each time step t;
- a vector of **static (time-independent) attributes** $v^i \in \mathbb{R}^{d_v}$.



Capital letters denote the stacked N time series, i.e., $\boldsymbol{X}_t \in \mathbb{R}^{N \times d_x}$, $\boldsymbol{U}_t \in \mathbb{R}^{N \times d_u}$.

→ We call spatial the dimension spanning the collection.

[3] Cini *et al.*, “Graph Deep Learning for Time Series Forecasting”, ACM Comput. Surv. 2025.

Correlated time series

We consider a time-invariant stochastic process generating each time series as

$$\boldsymbol{x}_t^i \sim p^i\left(\boldsymbol{x}_t^i \middle| \boldsymbol{X}_{<t}, \boldsymbol{U}_{\leq t}, \boldsymbol{V}\right) \quad \text{for all } i = 1 \dots N, t = 0, \dots, T-1$$

and assume the existence of a causality à la Granger among time series.

Furthermore time series

- are assumed

  a) homogenous, b) synchronous, c) regularly sampled.
- can be generated by different processes.

Notation:

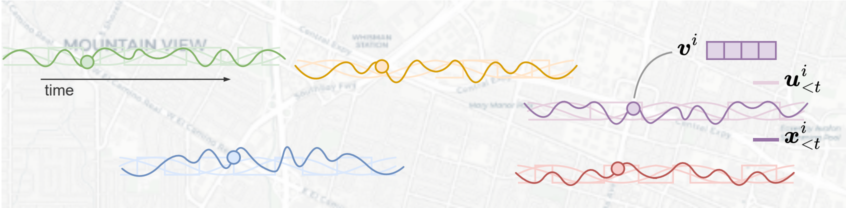
$$\mathcal{X}_t = \langle \boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{V} \rangle$$

$$\mathcal{X}_{<t} = [\mathcal{X}_0, \cdots, \mathcal{X}_{t-2}, \mathcal{X}_{t-1}]$$

❗ Assumptions a),b),c) can be relaxed as we will discuss in the 2nd part.

Example: Traffic monitoring system

Consider a sensor network monitoring the speed of vehicles at crossroads.



- $\boldsymbol{X}_{<t}$ collects past traffic speed measurements.
- $\boldsymbol{U}_t$ stores identifiers for time-of-the-day and day-of-the-week.
- $\boldsymbol{V}$ collects static sensor's features, e.g., type or number of lanes of the monitored road.

$\rightarrow$ Strong dependencies among time series that reflect the road network.

Forecasting

Time series forecasting

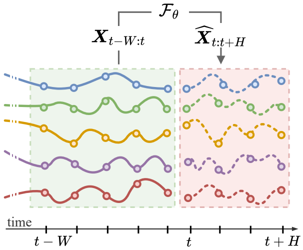
Multi-step time-series forecasting

Given a window of $W \geq 1$ past values

$$\mathcal{X}_{t-W:t} = [\mathcal{X}_{t-W}, \dots, \mathcal{X}_{t-1}],$$

predict $H \geq 1$ future observations

$$\boldsymbol{X}_{t+h} \qquad h = 1, \dots, H.$$



In particular, we are interested in learning a parametric model $\mathcal{F}\left(\,\cdot\,;\boldsymbol{\theta}\right)$ s.t.

$$\mathcal{F}\left(\mathcal{X}_{t-W:t}, \boldsymbol{U}_{t:t+H}; \boldsymbol{\theta}\right) = \widehat{\boldsymbol{X}}_{t:t+H} \approx E_p\left[\boldsymbol{X}_{t:t+H}\right].$$

Probabilistic predictors can be considered as well, but we focus on point forecasts.

Training objective

For point predictors, parameters $\boldsymbol{\theta}$ can be learned by minimizing a cost function $\ell(\,\cdot\,,\,\cdot\,)$ (e.g., MSE) on a training set

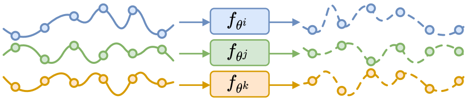
$$\begin{aligned}\widehat{\boldsymbol{\theta}} &= \arg\min_{\boldsymbol{\theta}} \frac{1}{NT}\sum_{t=1}^{T} \ell\left(\widehat{\boldsymbol{X}}_{t:t+H}, \boldsymbol{X}_{t:t+H}\right)\\ &= \arg\min_{\boldsymbol{\theta}} \frac{1}{NT}\sum_{t=1}^{T} \left\|\boldsymbol{X}_{t:t+H} - \widehat{\boldsymbol{X}}_{t:t+H}\right\|_2^2.\end{aligned}$$

! Choosing a different cost function allows for predicting other values.

→ **Example:** minimizing the MAE results in forecasts of the median.

Global and local predictors

Local models



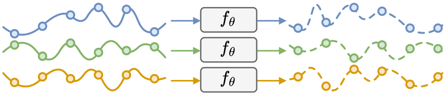
$$\hat{\boldsymbol{x}}^i_{t+h} = f\left(\boldsymbol{x}^i_{t-W:t}, \dots; \boldsymbol{\theta}^i\right)$$

Example: Box-Jenkins method

☺ Tailored to each time series.

☹ Inefficient.

Global models



$$\hat{\boldsymbol{x}}^i_{t+h} = f\left(\boldsymbol{x}^i_{t-W:t}, \dots; \boldsymbol{\theta}\right)$$

Example: DeepAR [1]

☺ Sample efficient.

☺ Allows for more complex models.

☹ Both approaches neglect dependencies among time series.

[1] Salinas *et al.*, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks", IJF 2020.

[4] Montero-Manso *et al.*, "Principles and algorithms for forecasting groups of time series: Locality and globality", IJF 2021.

Accounting for spatial dependencies

- One option is to consider the input as single multivariate time series

  → Resulting predictors are **local**: $\widehat{\boldsymbol{X}}_{t+h} = f\left(\boldsymbol{X}_{t-W:t}, \ldots; \boldsymbol{\theta}\right).$

  ☹ High sample complexity and poor scalability.

- Models operating on sets of time series would allow to keep parameters shared.

  → Resulting predictors are **global**: $\widehat{\boldsymbol{X}}^{\mathcal{S}}_{t+h} = \mathcal{F}\left(\boldsymbol{X}^{\mathcal{S}}_{t-W:t}, \ldots; \boldsymbol{\theta}\right), \qquad \forall \mathcal{S} \subseteq \mathcal{D}$

  ☺ Can be implemented by attention-based models (e.g, Transformers).

  ☹ Does not exploit structural priors, high computational and sample complexity.

- Other methods (e.g., [5]) rely on dimensionality reduction to extract shared latent factors.

  ☺ Might work well if data are low-rank.

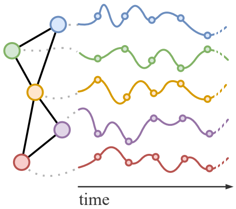
  ☹ Local and relational information are lost and can still suffer from, scalability issues.

[2] Benidis *et al.*, “Deep Learning for Time Series Forecasting: Tutorial and Literature Survey”, ACM CS 2022.

[5] Sen *et al.*, “Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting”, NeurIPS 2019.

Graph-based representation

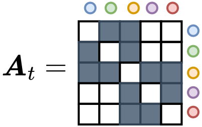
Relational information



Exploit functional dependencies as an inductive bias to improve the forecasts.

We can model pairwise relationships existing at time step t with adjacency matrix $\boldsymbol{A}_t \in \{0,1\}^{N \times N}$.

- $\boldsymbol{A}_t$ can be **asymmetric** and **dynamic** (can vary with t).

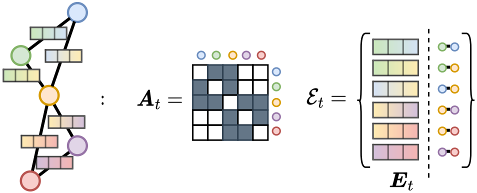


Relational information with attributes

Optional edge attributes $\boldsymbol{e}_t^{ij} \in \mathbb{R}^{d_e}$ can be associated to each non-zero entry of $\boldsymbol{A}_t$.

The set of attributed edges is denoted by

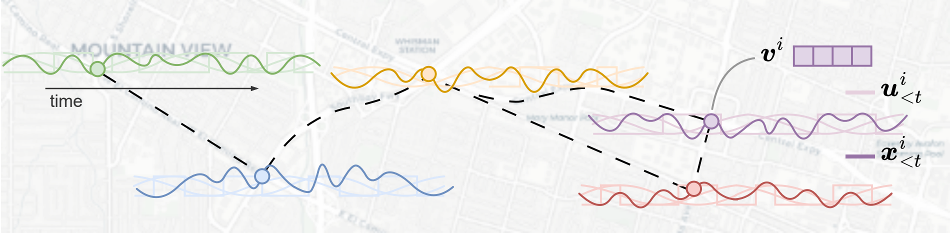
$$\mathcal{E}_t \doteq \{\langle (i,j), \boldsymbol{e}_t^{ij} \rangle \mid \forall i,j : \boldsymbol{A}_t[i,j] \neq 0\}.$$



→ Edge attributes can be both **categorical** or **numerical**.

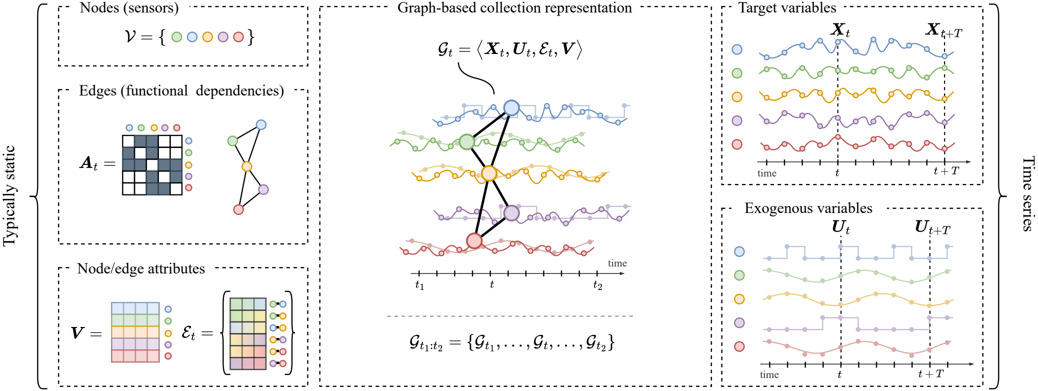
Example: Traffic monitoring system

Consider again the sensor network of the previous example.



- Edges in $\mathcal{E}$ can be obtained by considering the road network.
  - → Road closures and traffic diversions can be accounted for with a dynamic topology $\mathcal{E}_t$.

Graph-based representations for correlated time series


Typically static
Nodes (sensors)
$\mathcal{V} = \{ ○ ○ ○ ○ ○ \}$
Edges (functional dependencies)
$\boldsymbol{A}_t =$
Node/edge attributes
$\boldsymbol{V} =$
$\mathcal{E}_t =$
Graph-based collection representation
$\mathcal{G}_t = \langle \boldsymbol{X}_t, \boldsymbol{U}_t, \mathcal{E}_t, \boldsymbol{V} \rangle$
time
t_1
t
t_2
$\mathcal{G}_{t_1:t_2} = \{\mathcal{G}_{t_1}, \dots, \mathcal{G}_t, \dots, \mathcal{G}_{t_2}\}$
Target variables
$\boldsymbol{X}_t$
$\boldsymbol{X}_{t+T}$
time
t
$t+T$
Exogenous variables
$\boldsymbol{U}_t$
$\boldsymbol{U}_{t+T}$
time
t
$t+T$
Time series

$\mathcal{G}_t \doteq \langle \boldsymbol{X}_t, \boldsymbol{U}_t, \mathcal{E}_t, \boldsymbol{V} \rangle$ contains the available information w.r.t. time step t.

[3] Cini *et al.*, “Graph Deep Learning for Time Series Forecasting”, ACM Comput. Surv. 2025.

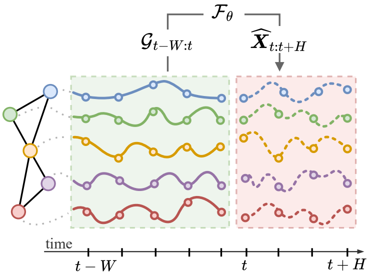
Relational inductive biases for time series forecasting

Forecasts can be conditioned on the available relational information $\mathcal{E}_{t-W:t}$

$$\widehat{\boldsymbol{X}}^{\mathcal{S}}_{t:T+H} = \mathcal{F}\left(\mathcal{G}^{\mathcal{S}}_{t-W:t}, \boldsymbol{U}^{\mathcal{S}}_{t:t+H}; \boldsymbol{\theta}\right) \qquad \forall \mathcal{S} \in \mathcal{D}$$

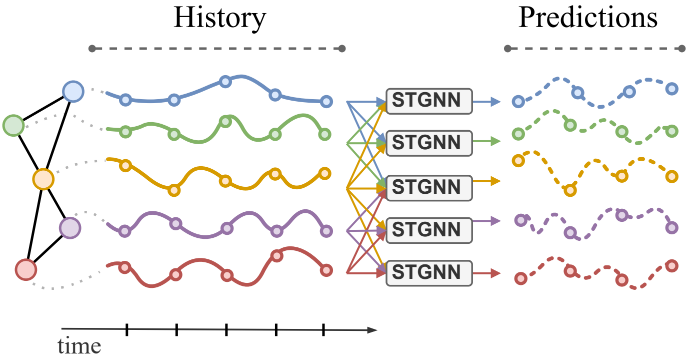
The conditioning can act as a regularization to localize predictions w.r.t. each node.

- ☺ Relational priors prune spurious correlations.
- ☺ More scalable than standard multivariate models.
- ☺ Can forecast any subset of correlated time series.



Spatiotemporal graph neural networks

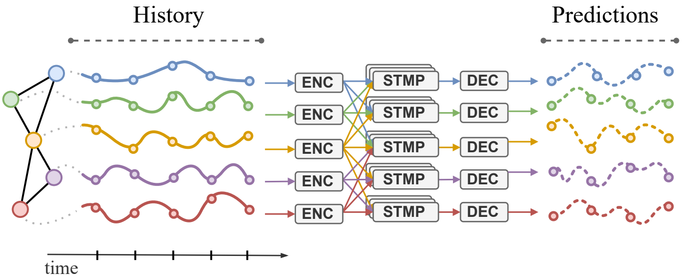
We call spatiotemporal graph neural networks (STGNNs) a neural network exploiting both temporal and spatial relations of the input spatiotemporal time series.



We focus on models based on message passing (MP).

A general recipe for building STGNNs

We consider STGNNs consisting of three main components



- $\textsc{Enc}(\,\cdot\,)$ is the **encoding** layer, e.g., implemented by an MLP.
- $\textsc{STMP}(\,\cdot\,)$ is a stack of **spatiotemporal message-passing (STMP)** layers.
- $\textsc{Dec}(\,\cdot\,)$ is the **readout** layer, e.g., implemented by an MLP.

[3] Cini *et al.*, “Graph Deep Learning for Time Series Forecasting”, ACM Comput. Surv. 2025.

A closer look

Representations are updated as follows.

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

$$\boldsymbol{H}^{l+1}_{t-1} = \text{STMP}^l\left(\boldsymbol{H}^l_{\leq t-1}, \mathcal{E}_{\leq t-1}\right), \quad l = 0, \dots, L-1 \tag{2}$$

$$\hat{\boldsymbol{x}}^i_{t:t+H} = \textsc{Decoder}\left(\boldsymbol{h}^{i,L}_{t-1}, \boldsymbol{u}^i_{t:t+H}\right). \tag{3}$$

- $\textsc{Enc}(\,\cdot\,)$ process each observation independently.
- $\text{STMP}(\,\cdot\,)$ is where propagation through time and space happens.
- $\textsc{Dec}(\,\cdot\,)$ maps each representation to predictions.

[3] Cini *et al.*, “Graph Deep Learning for Time Series Forecasting”, ACM Comput. Surv. 2025.

Spatiotemporal message-passing (STMP)

STMP blocks can be defined as:

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

Each block processes **sequences** while accounting for **relational dependencies**.

As in standard MP operators:

- $\textsc{Msg}^l(\,\cdot\,)$ is a **message function**, e.g., implemented by *temporal convolutional layers*.
- $\textsc{Aggr}\{\,\cdot\,\}$ is a permutation invariant **aggregation function**.
- $\textsc{Up}^l(\,\cdot\,)$ is an **update function**, e.g., implemented by an RNN.

❗ Blocks can be implemented by composing MP and sequence modeling operators.

→ Many possible designs exist.

[3] Cini *et al.*, “Graph Deep Learning for Time Series Forecasting”, ACM Comput. Surv. 2025.

[6] Gilmer *et al.*, “Neural message passing for quantum chemistry”, ICML 2017.

Design paradigms for STGNNs

Depending on the implementation of the STMP blocks, we categorize STGNNs into:

- **Time-and-Space (T&S)**
  Temporal and spatial processing cannot be factorized in two separate steps.
- **Time-then-Space (TTS)**
  Each time series is embedded in a vector and then representations are propagated on the graph.
- **Space-then-Time (STT)**
  Spatial propagation is performed before processing the resulting time series.

Time-and-Space

In T&S models, representations at every node and time step are obtained by jointly propagating representation through time and space.

$$\boldsymbol{H}_{t-1}^{l+1} = \mathsf{STMP}^{l}\left(\boldsymbol{H}_{\leq t-1}^{l}, \mathcal{E}_{\leq t-1}\right)$$

Several options exist.

- Integrate MP into neural operators for sequential data.
  - Graph recurrent architectures, spatiotemporal convolutions, spatiotemporal attention, ...
- Use sequence molding operators to compute messages.
  - Temporal graph convolutions, spatiotemporal cross-attention, ...
- Product graph representations.

Example 1: From Recurrent Neural Networks...

Consider a standard GRU cell [7].

$$\boldsymbol{r}_t^i = \sigma\left(\boldsymbol{\Theta}_r\left[\boldsymbol{x}_t^i || \boldsymbol{h}_{t-1}^i\right] + \boldsymbol{b}_r\right) \tag{4}$$

$$\boldsymbol{u}_t^i = \sigma\left(\boldsymbol{\Theta}_u\left[\boldsymbol{x}_t^i || \boldsymbol{h}_{t-1}^i\right] + \boldsymbol{b}_u\right) \tag{5}$$

$$\boldsymbol{c}_t^i = \tanh\left(\boldsymbol{\Theta}_c\left[\boldsymbol{x}_t^i || \boldsymbol{r}_t^i \odot \boldsymbol{h}_{t-1}^i\right] + \boldsymbol{b}_c\right) \tag{6}$$

$$\boldsymbol{h}_t^i = \left(1 - \boldsymbol{u}_t^i\right) \odot \boldsymbol{c}_t^i + \boldsymbol{u}_t^i \odot \boldsymbol{h}_{t-1}^i \tag{7}$$

Time series are processed independently for each node or as a single multivariate time series.

[7] Chung *et al.*, “Empirical evaluation of gated recurrent neural networks on sequence modeling” 2014.

...to Graph Convolutional Recurrent Neural Networks

We can obtain a T&S model by implementing the gates of the GRU with MP blocks:

$$\boldsymbol{Z}_t^l = \boldsymbol{H}_t^{l-1} \tag{8}$$

$$\boldsymbol{R}_t^l = \sigma\left(\text{MP}_r^l\left(\left[\boldsymbol{Z}_t^l || \boldsymbol{H}_{t-1}^l\right], \mathcal{E}_t\right)\right), \tag{9}$$

$$\boldsymbol{O}_t^l = \sigma\left(\text{MP}_o^l\left(\left[\boldsymbol{Z}_t^l || \boldsymbol{H}_{t-1}^l\right], \mathcal{E}_t\right)\right), \tag{10}$$

$$\boldsymbol{C}_t^l = \tanh\left(\text{MP}_c^l\left(\left[\boldsymbol{Z}_t^l || \boldsymbol{R}_t^l \odot \boldsymbol{H}_{t-1}^l\right], \mathcal{E}_t\right)\right), \tag{11}$$

$$\boldsymbol{H}_t^l = \boldsymbol{O}_t^l \odot \boldsymbol{H}_{t-1}^l + (1 - \boldsymbol{O}_t^l) \odot \boldsymbol{C}_t^l, \tag{12}$$

These T&S models are known as graph convolutional recurrent neural networks (GCRNNs) [8].

[8] Seo *et al.*, “Structured sequence modeling with graph convolutional recurrent networks”, ICONIP 2018.

Popular GCRNNs

The first GCRNN has been introduced in [8], with message passing (MP) blocks implemented as polynomial graph convolutional filters.

GCRNNs have become popular in traffic forecasting with the Diffusion Convolutional Recurrent Neural Network (DCRNN) architecture [9].

DCRNN relies on a bidirectional diffusion convolution:

$$\boldsymbol{H}_t' = \sum_{k=0}^{K} \left(\boldsymbol{D}_{t,\text{out}}^{-1}\boldsymbol{A}_t\right)^k \boldsymbol{H}_t \boldsymbol{\Theta}_1^{(k)} + \left(\boldsymbol{D}_{t,\text{in}}^{-1}\boldsymbol{A}_t^{\top}\right)^k \boldsymbol{H}_t \boldsymbol{\Theta}_2^{(k)} \tag{13}$$

[8] Seo *et al.*, “Structured sequence modeling with graph convolutional recurrent networks”, ICONIP 2018.

[9] Li *et al.*, “Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting”, ICLR 2018.

Example 2: Spatiotemporal convolutional networks (i)

Spatiotemporal convolutional networks (STCNs) instead **alternate spatial and temporal convolutions**:

1. Compute intermediate representations by using a temporal convolutional layer:

$$\boldsymbol{z}_{t-W:t}^{i,l} = \mathsf{TCN}^{l}\left(\boldsymbol{h}_{t-W:t}^{i,l-1}\right) \qquad \forall\, i$$

   where TCN^{l} indicates a temporal convolutional layer.

2. Compute the updated representation at each time step by using a graph convolution:

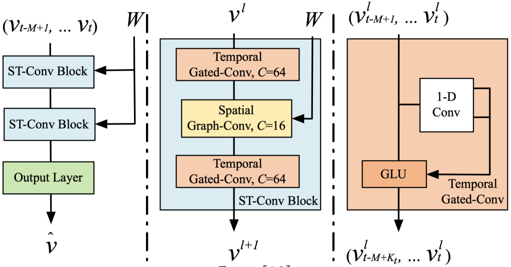
$$\boldsymbol{H}_{t}^{l} = \mathsf{MP}^{l}\left(\boldsymbol{Z}_{t}^{l}, \mathcal{E}_{t}\right) \qquad \forall\, t$$

Spatiotemporal convolutional networks (ii)

The first example of such architecture is the STGCN by Yu et al. [10].

The model is obtained by stacking STMP blocks consisting of

- a (gated) temporal convolution;
- a polynomial graph convolution;
- a second (gated) temporal convolution.


$(v_{t-M+1}, \dots v_t)$
W
ST-Conv Block
ST-Conv Block
Output Layer
$\hat{v}$
v^l
W
Temporal Gated-Conv, C=64
Spatial Graph-Conv, C=16
Temporal Gated-Conv, C=64
ST-Conv Block
v^{l+1}
$(v^l_{t-M+1}, \dots v^l_t)$
1-D Conv
GLU
Temporal Gated-Conv
$(v^l_{t-M+K_t}, \dots v^l_t)$

From [10].

More advanced implementations exist, e.g., see Graph Wavenet [11].

[10] Yu *et al.*, “Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting”, IJCAI 2018.
[11] Wu *et al.*, “Graph wavenet for deep spatial-temporal graph modeling”, IJCAI 2019.

Example 3: Temporal Graph Convolution

A more integrated approach instead consists of implementing a temporal propagation mechanism in the message function.

For example, we can design STMP layers s.t.

$$\boldsymbol{h}^{i,l}_{t-W:t} = \mathsf{TCN}^{l}_{1}\left(\boldsymbol{h}^{i,l-1}_{t-W:t}, \underset{j\in\mathcal{N}_t(i)}{\textsc{Aggr}}\left\{\mathsf{TCN}^{l}_{2}\left(\boldsymbol{h}^{i,l-1}_{t-W:t}, \boldsymbol{h}^{j,l-1}_{t-W:t}, \boldsymbol{e}^{ji}_{t-W:t}\right)\right\}\right).$$

💡 Analogous models can be built with any sequence modeling architecture.

→ **Example:** many rely on attention-based operators [12][13].

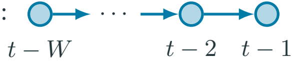
[12] Marisca *et al.*, “Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations”, NeurIPS 2022.
[13] Wu *et al.*, “TraverseNet: Unifying Space and Time in Message Passing for Traffic Forecasting”, TNNLS 2022.

Example 4: Product graph representations

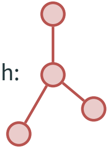
An alternative option is to consider the sequence $\mathcal{G}_{t-W:t}$ as a **single graph** with temporal and spatial edges.

In particular, **product graph representations** can be obtained by combining the two edge sets.

Temporal graph:



Spatial graph:

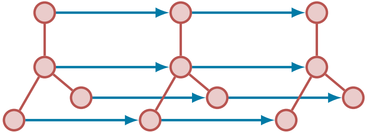


The resulting graph can be processed by any MP neural network.

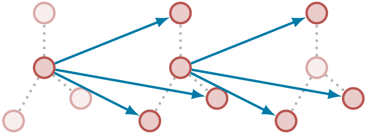
[14] Sabbaqi *et al.*, “Graph-time convolutional neural networks: Architecture and theoretical analysis”, TPAMI 2023.

Building product graph representations

- **Cartesian product**
  Spatial graphs are kept and each node is connected to itself in the previous time instant.



- **Kronecker product**
  Each node is connected **only** to its neighbors in the previous time instant.



- ...

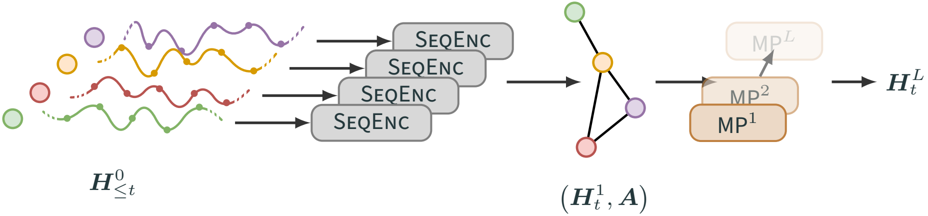
Time-then-Space models

The general recipe for a TTS model consists in:

1. Embedding each node-level time series in a vector.
2. Propagating obtained encodings throughout the graph with a stack of MP layers.

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

2. $\boldsymbol{H}_t^{l+1} = \text{MP}^l\left(\boldsymbol{H}_t^l, \mathcal{E}_t\right)$


$\boldsymbol{H}_{\leq t}^0$
SeqEnc
SeqEnc
SeqEnc
SeqEnc
$(\boldsymbol{H}_t^1, \boldsymbol{A})$
MP^L
MP^2
MP^1
$\boldsymbol{H}_t^L$

Pros & Cons of TTS models

Pros:

- ☺ Easy to implement and computationally efficient.
- ☺ We can reuse operators we already know.

Cons:

- ☹ The 2-step encoding might introduce information bottlenecks.
- ☹ Accounting for changes in topology and dynamic edge attributes can be more problematic.

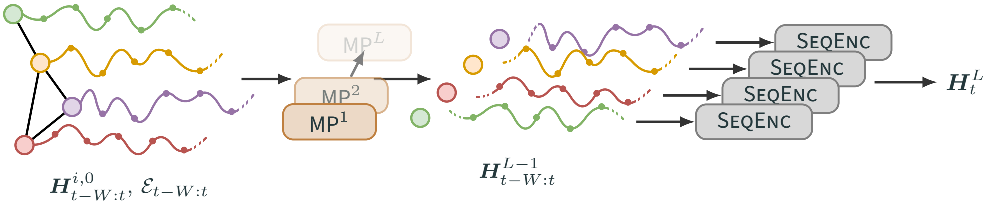
Space-then-Time

In STT approaches the two processing steps of TTS models are inverted:

1. Observations are propagated among nodes w.r.t. each time step using a stack of MP layers.
2. Each sequence of representations is processed by a sequence encoder.

1. $\boldsymbol{H}_t^{i,l} = \mathsf{MP}^l\left(\boldsymbol{H}_t^{i,l-1}, \mathcal{E}_t\right)$

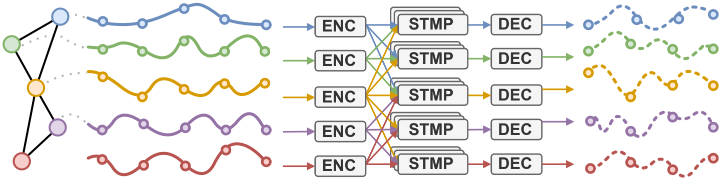
2. $\boldsymbol{h}_t^{i,L} = \textsc{SeqEnc}\left(\boldsymbol{h}_{t-W:t}^{i,L-1}\right)$


MP^L
MP^2
MP^1
SeqEnc
SeqEnc
SeqEnc
SeqEnc
$\boldsymbol{H}_t^L$
$\boldsymbol{H}_{t-W:t}^{i,0}, \mathcal{E}_{t-W:t}$
$\boldsymbol{H}_{t-W:t}^{L-1}$

☹ They do not have the same computational advantages of TTS models.

Global and local models

Globality and locality in STGNNs

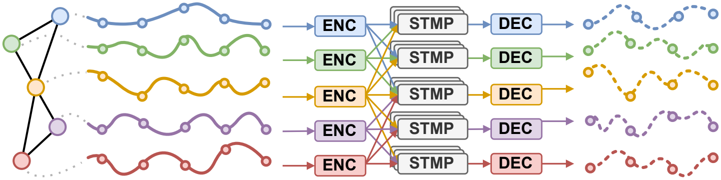


Standard STGNNs are **global** models.

- ☺ Can handle arbitrary node sets.
- ☺ Neighbors provide further conditioning on the predictions.
- ☹ Might struggle with local effects.
- ☹ Might need long windows and high model capacity.

💡 Use hybrid global-local STGNNs.

Global-local STGNNs

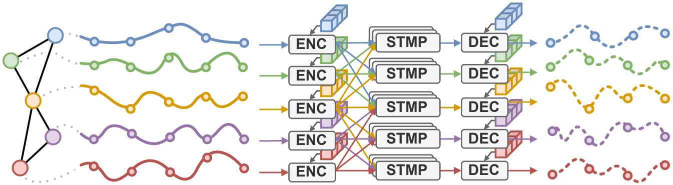


💡 We can turn some global components of the architecture into local.

☺ Resulting models can capture local effects.

☹ Might require a large number of local parameters.

Global-local STGNNs with node embeddings



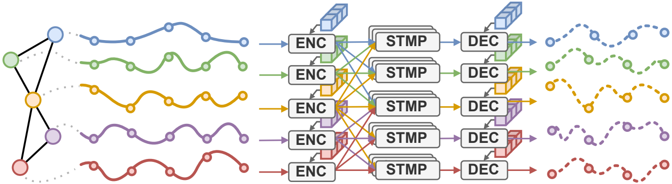
Node embeddings can amortize the learning of local components.

Node embeddings are a table of **learnable parameters** $Q \in \mathbb{R}^{N \times d_q}$ associated with **each node**.

- ☺ Fed into encoder/decoder, amortize the learning of local components.
- ☺ Most of the model’s parameters remain shared.
- 😐 Number of parameters scales linearly with the number of time series . . .
  - → One might consider intermediate solutions, e.g., learning embeddings for clusters of time series.

[15] Cini *et al.*, “Taming Local Effects in Graph-based Spatiotemporal Forecasting”, NeurIPS 2023.

Transferability



❗ Hybrid global-local STGNNs are not inductive models.

However, the cost of transfer learning can be reduced.

☺ Keep shared parameters fixed and finetune local parameters only.

☺ Node embeddings can be regularized to facilitate transfer further.

[15] Cini *et al.*, "Taming Local Effects in Graph-based Spatiotemporal Forecasting", NeurIPS 2023.

[16] Butera *et al.*, "On the Regularization of Learnable Embeddings for Time Series Processing", TMLR 2025.

Some empirical results

Models	MetrLA	PemsBAY	CER-E	AQI	MetrLA	PemsBAY	CER-E	AQI
Reference arch.	Global models				+ Local node embeddings			
RNN	3.54±.00	1.77±.00	4.57±.01	14.02±.04	**3.15±.03**	**1.59±.00**	**4.22±.02**	**13.73±.04**
GCRNN-IMP	3.35±.01	1.70±.01	4.44±.01	12.87±.02	**3.10±.01**	**1.59±.00**	**4.18±.01**	**12.48±.03**
RNN+IMP	3.34±.01	1.72±.00	4.39±.01	12.74±.02	**3.08±.01**	**1.58±.00**	**4.12±.03**	**12.33±.02**
GCRNN-AMP	3.22±.02	1.65±.00	4.57±.04	12.29±.02	**3.07±.02**	**1.59±.00**	**4.17±.02**	**12.17±.05**
RNN+AMP	3.24±.01	1.66±.00	4.31±.01	12.30±.02	**3.06±.01**	**1.58±.01**	**4.13±.01**	**12.15±.02**
Baseline arch.	Original				+ Local node embeddings			
DCRNN	3.22±.01	1.64±.00	4.28±.01	12.96±.03	**3.07±.02**	**1.60±.00**	**4.13±.02**	**12.53±.02**
GraphWaveNet	3.05±.03	**1.56±.01**	**3.97±.01**	12.08±.11	**2.99±.02**	1.58±.00	4.01±.01	**11.81±.04**
AGCRN	3.16±.01	**1.61±.00**	4.45±.01	13.33±.02	**3.14±.00**	1.62±.00	**4.37±.02**	**13.28±.03**

Table 1: MAE on benchmark datasets.

Transfer learning results

We consider datasets coming from four different traffic networks.

→ **One of the networks is left out** at training time and used for evaluating **transferability**.

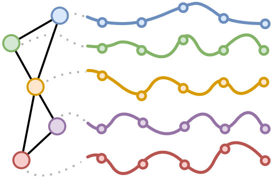
	RNN+IMP	PEMS03	PEMS04	PEMS07	PEMS08
Fine-tuning	Global	15.30 ± 0.03	21.59 ± 0.11	23.82 ± 0.03	15.90 ± 0.07
	Embeddings	14.64 ± 0.05	20.27 ± 0.11	**22.23 ± 0.08**	**15.45 ± 0.06**
	– **Variational**	**14.56 ± 0.03**	20.19 ± 0.05	22.43 ± 0.02	**15.41 ± 0.06**
	– **Clustering**	**14.60 ± 0.02**	**19.91 ± 0.11**	**22.16 ± 0.07**	**15.41 ± 0.06**
	Zero-shot	18.20 ± 0.09	23.88 ± 0.08	32.76 ± 0.69	20.41 ± 0.07

Table 2: Transfer learning results (MAE) after fine-tuning on a week of data.

[15] Cini *et al.*, “Taming Local Effects in Graph-based Spatiotemporal Forecasting”, NeurIPS 2023.

End of Part 1: what we have so far

1. We formalized the problem of processing correlated time series.
2. Graph representations allows for modeling dependencies among them.
3. We discussed forecasting problem and global/local deep learning models for time series.
4. We saw approaches to building spatiotemporal graph neural networks and the associated trade-offs.



In the next part, we will discuss challenges typical of many practical applications.

Part 2

Challenges

Challenges

- **Dealing with missing data**
  How to deal with missing observations within the time series?
- **Latent graph learning**
  What to do when the underlying graph is not known?
- **Scalability**
  How to deal with large collections of time series?
- **Model quality assessment**
  How to evaluate our graph-based model?

Dealing with missing data

The problem of missing data

So far, we assumed to deal with **complete sequences**.

- i.e., to have valid observations associated with each node (sensor) and time step.

However, time series collected by real-world sensor networks often have missing data, due to:

- faults, of either transient or permanent nature;
- asynchronicity among the time series;
- communication errors...

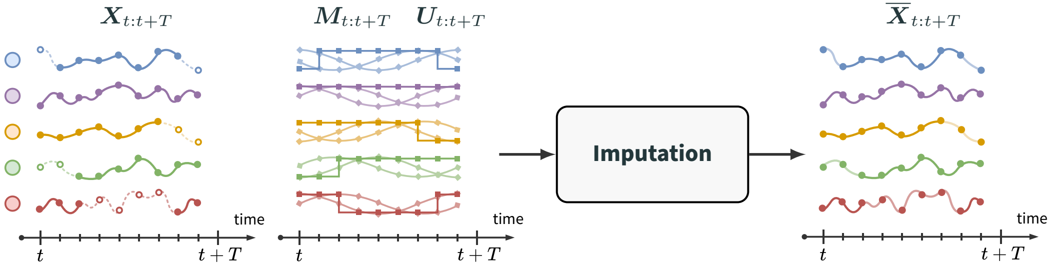
Most time series analysis methods operate on complete sequences.

→ We need a way to impute, i.e., *reconstruct*, missing data.

Time series imputation

Time series imputation (TSI)

Given a window of observations $\boldsymbol{X}_{t:t+T}$, mask $\boldsymbol{M}_{t:t+T}$, and covariates $\boldsymbol{U}_{t:t+T}$, the goal is to estimate the missing observations in the sequence $\overline{\boldsymbol{X}}_{t:t+T}$.



→ We use a **mask** $m_t^i \in \{0, 1\}$ to distinguish between missing (0) and valid (1) observations.

Missing data distribution

We can categorize missing data patterns according to the conditional distribution $p\left(\boldsymbol{m}_t^i \mid \boldsymbol{M}_{\leq t}\right)$.

- **Point missing**
  $p\left(\boldsymbol{m}_t^i = \mathbf{0}\right)$ is the same across nodes and time steps, i.e., RVs associated to each $\boldsymbol{m}_t^i$ are iid.

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

- **Block missing**
  $p\left(\boldsymbol{m}_t^i = \mathbf{0}\right)$ is not independent from missing data at other nodes and/or time steps.

Temporal block missing $\quad p\left(\boldsymbol{m}_t^i \mid \boldsymbol{m}_{t-1}^i\right) \neq p\left(\boldsymbol{m}_t^i\right)$

Spatial block missing $\quad p\left(\boldsymbol{m}_t^i \mid \left\{\boldsymbol{m}_t^j\right\}^{j \neq i}\right) \neq p\left(\boldsymbol{m}_t^i\right)$

Spatiotemporal block missing $\quad p\left(\boldsymbol{m}_t^i \mid \boldsymbol{m}_{t-1}^i, \left\{\boldsymbol{m}_t^j\right\}^{j \neq i}\right) \neq p\left(\boldsymbol{m}_t^i\right)$

Learning to reconstruct missing observation

Parameters $\boldsymbol{\theta}$ can be learned by minimizing a loss function $\ell(\,\cdot\,,\,\cdot\,)$ on valid observations in a training set:

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} \sum_{i=1}^{N} \frac{\left\| \boldsymbol{m}_t^i \odot \ell\left(\hat{\boldsymbol{x}}_t^i, \boldsymbol{x}_t^i\right) \right\|_1}{\left\| \boldsymbol{m}_t^i \right\|_1}. \qquad \leftarrow \qquad \text{e.g., } \ell = \left(\hat{\boldsymbol{x}}_t^i - \boldsymbol{x}_t^i\right)^2$$

For imputation, we mark some valid observations as missing with mask $\overline{\boldsymbol{m}}_t^i$ to obtain ground-truth labels:

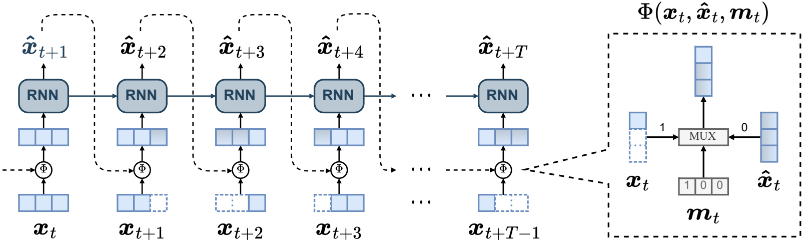
$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} \sum_{i=1}^{N} \frac{\left\| \overline{\boldsymbol{m}}_t^i \odot \ell\left(\overline{\boldsymbol{x}}_t^i, \boldsymbol{x}_t^i\right) \right\|_1}{\left\| \overline{\boldsymbol{m}}_t^i \right\|_1}.$$

⚠ Data where $\overline{\boldsymbol{m}}_t^i = \mathbf{1}$ must not be used in the model to obtain the imputations.

Deep learning for TSI

Besides standard statistical methods, deep learning approaches have become a popular alternative.

- In particular, autoregressive models (e.g., RNNs).


$\hat{\boldsymbol{x}}_{t+1}$
$\hat{\boldsymbol{x}}_{t+2}$
$\hat{\boldsymbol{x}}_{t+3}$
$\hat{\boldsymbol{x}}_{t+4}$
$\hat{\boldsymbol{x}}_{t+T}$
RNN
RNN
RNN
RNN
RNN
Φ
$\boldsymbol{x}_t$
$\boldsymbol{x}_{t+1}$
$\boldsymbol{x}_{t+2}$
$\boldsymbol{x}_{t+3}$
$\boldsymbol{x}_{t+T-1}$
$\Phi(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t, \boldsymbol{m}_t)$
1
MUX
0
$\boldsymbol{x}_t$
1 0 0
$\hat{\boldsymbol{x}}_t$
$\boldsymbol{m}_t$

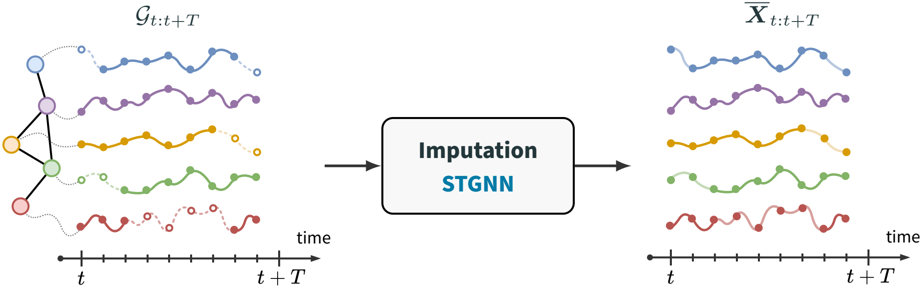
☺ Effective in exploiting past (and future, with bidirectional models) node observations.

☹ Struggle in capturing nonlinear space-time dependencies.

Time series imputation + relational inductive biases

Again, we can use the available relational information to condition the model, i.e.,

$$\boldsymbol{x}^i_{t+k} \sim p\left(\boldsymbol{x}^i_{t+k} \mid \boldsymbol{X}_{t:t+T} \odot \boldsymbol{M}_{t:t+T}, \boldsymbol{A}\right) \qquad k \in [0, T)$$



Graph Recurrent Imputation Network (GRIN) – Idea

We can integrate graph processing into the autoregressive approach for imputation [17].

In these approaches, the distribution $p\left(\boldsymbol{x}_t^i \mid \boldsymbol{X}_{t_0:t_T} \odot \boldsymbol{M}_{t_0:t_T}\right)$ is modeled into three independent steps:

Information from previous observations.

$$p\left(\boldsymbol{x}_t^i \mid \boldsymbol{X}_{<t} \odot \boldsymbol{M}_{<t}\right)$$

Information from subsequent observations.

$$p\left(\boldsymbol{x}_t^i \mid \boldsymbol{X}_{>t} \odot \boldsymbol{M}_{>t}\right)$$

Information from related concurrent observations.

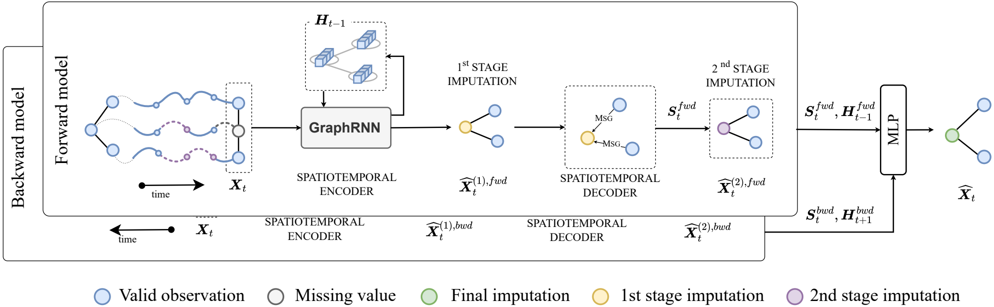
$$p\left(\boldsymbol{x}_t^i \mid \left\{\boldsymbol{x}_t^j \odot \boldsymbol{m}_t^j\right\}^{j\neq i}\right)$$

Typically modeled by bidirectional autoregressive models.

Enabled by message passing.

[17] Cini *et al.*, “Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks”, ICLR 2022.

Graph Recurrent Imputation Network (GRIN) – Overview



GRIN [17] is a bidirectional autoregressive STGNN.

- A **graph RNN** encodes the sequence of **valid observations**.
- An **final message-passing decoder** accounts for **concurrent observations at neighbors**

[17] Cini *et al.*, “Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks”, ICLR 2022.

Graph Recurrent Imputation Network (GRIN)

- The GCRNN cell encodes the sequence of valid observations as:

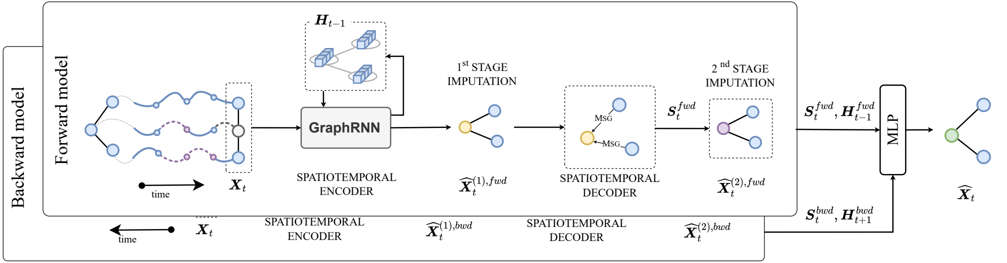
$$\boldsymbol{Z}_t = \mathsf{STMP}\left(\boldsymbol{X}_{<t} \odot \boldsymbol{M}_{<t} + \widehat{\boldsymbol{X}}_{<t} \odot (1 - \boldsymbol{M}_{<t}), \mathcal{E}_{<t}\right).$$

- The final message-passing decoder accounts for concurrent observations at neighbors:

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

[17] Cini *et al.*, “Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks”, ICLR 2022.

Graph Recurrent Imputation Network (GRIN)


Backward model
Forward model
$\boldsymbol{H}_{t-1}$
GraphRNN
time
$\boldsymbol{X}_t$
SPATIOTEMPORAL ENCODER
1st STAGE IMPUTATION
$\widehat{\boldsymbol{X}}_t^{(1),fwd}$
MSG
MSG
SPATIOTEMPORAL DECODER
$\boldsymbol{S}_t^{fwd}$
2 nd STAGE IMPUTATION
$\widehat{\boldsymbol{X}}_t^{(2),fwd}$
$\boldsymbol{S}_t^{fwd}, \boldsymbol{H}_{t-1}^{fwd}$
MLP
$\widehat{\boldsymbol{X}}_t$
time
$\boldsymbol{X}_t$
SPATIOTEMPORAL ENCODER
$\widehat{\boldsymbol{X}}_t^{(1),bwd}$
SPATIOTEMPORAL DECODER
$\widehat{\boldsymbol{X}}_t^{(2),bwd}$
$\boldsymbol{S}_t^{bwd}, \boldsymbol{H}_{t+1}^{bwd}$

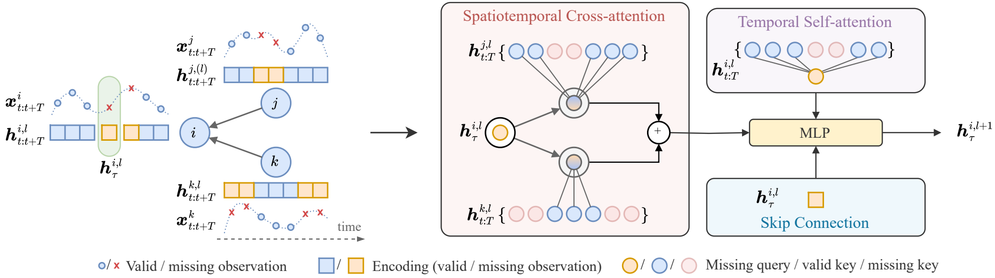
○ Valid observation ○ Missing value ○ Final imputation ○ 1st stage imputation ○ 2nd stage imputation

☺ Extremely effective in many settings.

☹ However, autoregressive models can accumulate reconstruction error.

[17] Cini *et al.*, “Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks”, ICLR 2022.

Spatiotemporal Point Inference Network (SPIN)


$\boldsymbol{x}^i_{t:t+T}$
$\boldsymbol{h}^{i,l}_{t:t+T}$
$\boldsymbol{h}^{i,l}_{\tau}$
$\boldsymbol{x}^j_{t:t+T}$
$\boldsymbol{h}^{j,(l)}_{t:t+T}$
j
i
k
$\boldsymbol{h}^{k,l}_{t:t+T}$
$\boldsymbol{x}^k_{t:t+T}$
time
Spatiotemporal Cross-attention
$\boldsymbol{h}^{j,l}_{t:T}$
$\boldsymbol{h}^{i,l}_{\tau}$
$\boldsymbol{h}^{k,l}_{t:T}$
+
Temporal Self-attention
$\boldsymbol{h}^{i,l}_{t:T}$
MLP
$\boldsymbol{h}^{i,l+1}_{\tau}$
$\boldsymbol{h}^{i,l}_{\tau}$
Skip Connection
Valid / missing observation
Encoding (valid / missing observation)
Missing query / valid key / missing key

💡 Use attention-based operators.

- ☺ We can exploit cross-attention to propagate information among related time series.
- ☺ Prevents autoregressive error accumulation.
- ☹ Can be computationally expensive.
  - → hierarchical computation can reduce the cost from $\mathcal{O}\left(T^2|\mathcal{E}|\right)$ to $\mathcal{O}\left(TK|\mathcal{E}|\right)$ ☺

[12] Marisca *et al.*, "Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations", NeurIPS 2022.

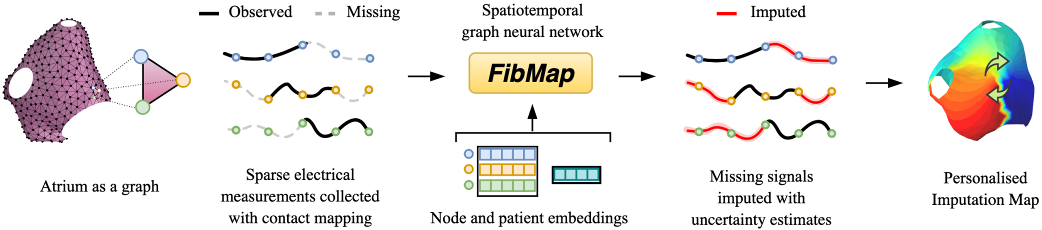
Some empirical results

We compared the introduced **graph-based** methods against several state-of-the-art approaches including standard **attention-based** architectures and popular **RNN** models.

Model	Block missing		Point missing		Simulated failures	
	PEMS-BAY	METR-LA	PEMS-BAY	METR-LA	AQI-36	AQI
Mean	5.46±.00	7.48±.00	5.42±.00	7.56±.00	53.48±.00	39.60±.00
VAR	2.09±.10	3.11±.08	1.30±.00	2.69±.00	15.64±.08	22.95±.30
rGAIN	2.18±.01	2.90±.01	1.88±.02	2.83±.01	15.37±.26	21.78±.50
BRITS	1.70±.01	2.34±.01	1.47±.00	2.34±.00	14.50±.35	20.21±.22
SAITS	1.56±.01	2.30±.01	1.40±.03	2.26±.00	18.16±.42	21.33±.15
STTr	1.70±.02	3.54±.00	0.74±.00	2.16±.00	11.98±.53	18.11±.25
GRIN	1.14±.01	2.03±.00	**0.67±.00**	**1.91±.00**	12.08±.47	14.73±.15
SPIN	**1.06±.02**	**1.98±.01**	0.70±.01	**1.90±.01**	11.77±.54	**13.92±.15**
SPIN-H	**1.05±.01**	2.05±.02	0.73±.01	1.96±.03	**10.89±.27**	14.41±.13

Table 3: Reconstruction MAE.

Application: Mapping the Dynamics of Atrial Fibrillation


Observed
Missing
Spatiotemporal graph neural network
FibMap
Imputed
Atrium as a graph
Sparse electrical measurements collected with contact mapping
Node and patient embeddings
Missing signals imputed with uncertainty estimates
Personalised Imputation Map

💡 Reconstruct atrial fibrillation (AF) dynamics modeling the atrium surface as a graph.

❗ Possibly enabling personalized AF care and improving patient outcomes.

[18] Jenkins *et al.*, “Learning to Predict Global Atrial Fibrillation Dynamics from Sparse Measurements” 2025.

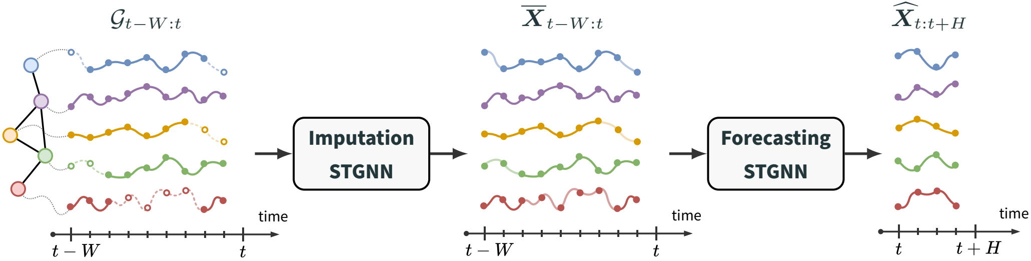
Graph-based Imputation Architectures

- ⚡ Graph-based imputation is a very active field!
  - → Very relevant in practical applications.
- Many different architectures/methods.
  - → Diffusion-based approaches have been particularly successful [19]–[21].
- ❓ A necessary preprocessing step for forecasting?

[19] Tashiro *et al.*, “CSDI: Conditional score-based diffusion models for probabilistic time series imputation”, NeurIPS 2021.
[20] Liu *et al.*, “Pristi: A conditional diffusion framework for spatiotemporal imputation”, ICDE 2023.
[21] Wen *et al.*, “DiffSTG: Probabilistic spatio-temporal graph forecasting with denoising diffusion models”, SIGSPATIAL 2023.

Imputation as a preprocessing step

TSI is often used as a preprocessing step for a downstream task, e.g., forecasting.



- 😊 Allows for using standard forecasting methods for regularly sampled time series.
- ☹ Often **necessary**, but might introduce biases due to errors in estimated values.
- ❗ More accurate reconstruction does not imply more accurate forecasts.

Forecasting from partial observations

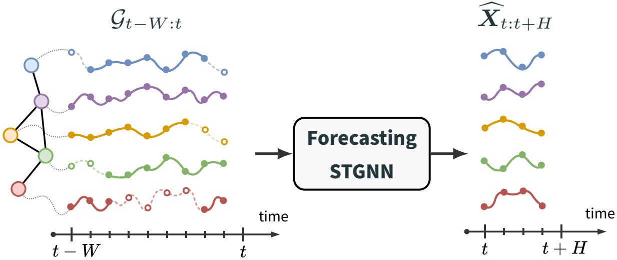
A more direct approach: avoid the reconstruction step!

→ Design forecasting architecture to directly deal with irregular observations.

Benefits

☺ Leverage only valid observations specifically for the task at hand.

☺ Avoid the computational cost of imputing missing values.



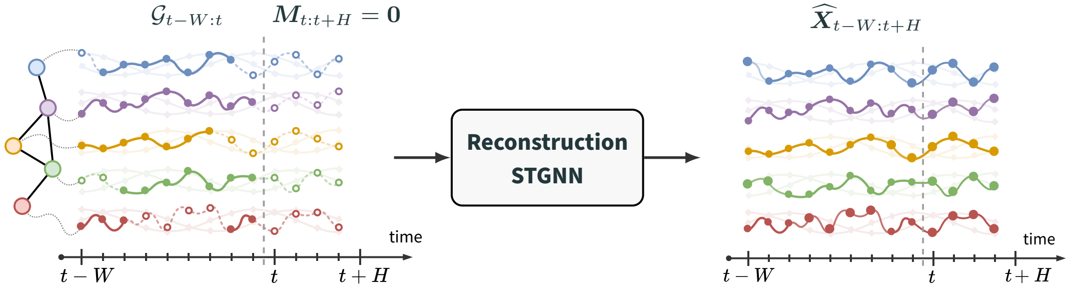
[22] Zhang *et al.*, “Graph-guided network for irregularly sampled multivariate time series”, ICLR 2022.

[23] Zhong *et al.*, “Heterogeneous spatio-temporal graph convolution network for traffic forecasting with missing values”, IEEE ICDCS 2021.

[24] Marisca *et al.*, “Graph-based Forecasting with Missing Data through Spatiotemporal Downsampling”, ICML 2024.

Forecasting as a reconstruction task

Reconstruction methods can also be adapted to perform both imputation and forecasting simultaneously.



Reconstruction STGNN

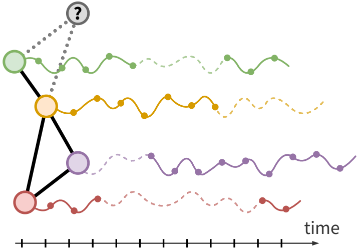
$\mathcal{G}_{t-W:t}$
$\boldsymbol{M}_{t:t+H} = \mathbf{0}$
time
$t-W$
t
$t+H$
$\widehat{\boldsymbol{X}}_{t-W:t+H}$
time
$t-W$
t
$t+H$

- ☺ Allows for building general purples architectures.
- ☹ Might require sub-optimal designs.

Virtual sensing

Virtual sensing

Infer observations at **completely unobserved location** by exploiting spatial dependencies to obtain a **virtual sensor**.



- ☺ Relational learning conditions estimates on neighboring sensors.
- ☺ The inductive property of MP allows us to handle new nodes and edges.

[25] Wu *et al.*, “Inductive Graph Neural Networks for Spatiotemporal Kriging”, AAAI 2021.

[26] De Felice *et al.*, “Graph-Based Virtual Sensing from Sparse and Partial Multivariate Observations”, ICLR 2024.

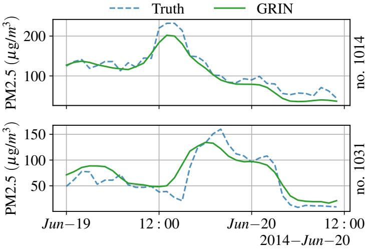
Graph imputation for virtual sensing

Add a virtual node with **no data** and let the model infer the corresponding time series.

Clearly, several assumptions are needed

- high degree of homogeneity of sensors,
- capability to reconstruct from observations at neighboring sensors,
- and many more...

Two virtual sensors for air quality. (from [17])

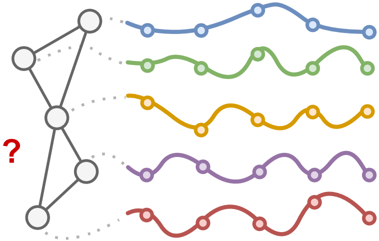


[17] Cini *et al.*, “Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks”, ICLR 2022.

Latent graph learning

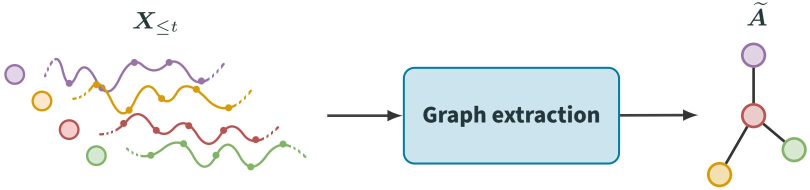
Learning an adjacency matrix

- ☹ Relational information is not always available.
- ☺ Relational architectural biases can nonetheless be exploited . . .
- ☺ . . . by learning a graph from data.



Graph structure learning

Infer a graph from the time series or node attributes.



$\boldsymbol{X}_{\leq t}$

Graph extraction

$\widetilde{\boldsymbol{A}}$

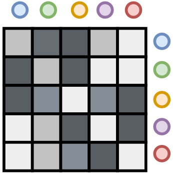
! When possible, the graph should be sparse.
$\longrightarrow$ It can be interpreted as regularizing a spatial attention operator.

- This task is found under different names:
  graph structure learning, latent graph learning, graph/relational inference, ...

Time-series similarities

Probably, the simplest approach to extract a graph from the time series is by computing time series similarity scores.

- Pearson correlation
- Correntropy
- Granger causality
- Kernels for time series
- . . .



→ Thresholding might be necessary to obtain binary and sparse graphs.

Inferring latent structures from time series

More advanced approaches model the graph as a latent variable connected to the observed time series.

$\rightarrow$ Often rely on assumptions, e.g., signal smoothness and/or the existence of an underlying diffusion process with some properties.

The graph is obtained by minimizing loss functions encoding such assumptions, e.g.,

$$\text{trace}(\boldsymbol{X}_t^\top \boldsymbol{L} \boldsymbol{X}_t) = \frac{1}{2} \sum_{ij} \boldsymbol{A}_{i,j} \| \boldsymbol{x}_t^i - \boldsymbol{x}_t^j \|_2^2$$

constraining $\boldsymbol{L}$ (or $\boldsymbol{A}$) to be a Laplacian (adjacency matrix) and promoting sparsity.

$\rightarrow$ These approaches are common in graph signal processing.

[27] Dong *et al.*, “Learning Laplacian matrix in smooth graph signal representations”, IEEE TSP 2016.

[28] Mateos *et al.*, “Connecting the dots: Identifying network structure via graph signal processing”, IEEE SP Mag 2019.

End-to-end latent graph learning

We focus on an integrated approach: learn the **relations** end-to-end with the downstream task

→ e.g., by minimizing the forecasting error (MAE, MSE...).

Two main approaches:

1. learning an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ directly;
2. learning a probability distribution over graphs p_Φ generating $\boldsymbol{A}$ (often $\in \{0, 1\}^{N \times N}$).

⚠ One key challenge is keeping both $\boldsymbol{A}$ and the subsequent computations sparse.

→ Non-trivial with gradient-based optimization.

Direct approach

Learn $\widetilde{\boldsymbol{A}}$ as function $\xi(\,\cdot\,)$ of edge scores $\Phi \in \mathbb{R}^{N \times N}$ as

$$\widetilde{\boldsymbol{A}} = \xi\left(\Phi\right)$$

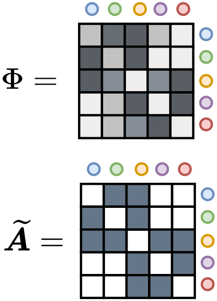
Edge scores Φ

- can be a table of learnable model parameters,
- obtained as a function of the inputs and/or other parameters:

$$\Phi_t = \Phi\left(\boldsymbol{X}_{<t}, \dots, \widetilde{\Phi}\right).$$

Function $\xi(\,\cdot\,)$ can introduce structure in $\widetilde{\boldsymbol{A}}$, e.g.,

→ make $\widetilde{\boldsymbol{A}}$ binary, a k-NN graph, a tree...

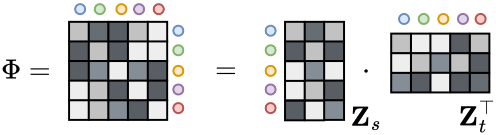


Edge score factorization

The number of scores to compute is quadratic in the number of nodes as $\Phi \in \mathbb{R}^{N \times N}$.

$\rightarrow$ a common approach is to factorize Φ:

$$\widetilde{\boldsymbol{A}} = \xi\left(\Phi\right) \qquad \Phi = \boldsymbol{Z}_s \boldsymbol{Z}_t^\top$$



with

- $\boldsymbol{Z}_s \in \mathbb{R}^{N \times d}$ source node embeddings
- $\boldsymbol{Z}_t \in \mathbb{R}^{N \times d}$ target node embeddings

$\boldsymbol{Z}_s$ and $\boldsymbol{Z}_t$ can be tables of learnable parameters or obtained as a function of the input window.

[11] Wu *et al.*, “Graph wavenet for deep spatial-temporal graph modeling”, IJCAI 2019.

Pro & Cons of the direct approach

- ☺ Easy to implement.
- ☺ Many possible parametrizations.
- ☺ Edge scores are usually easy to learn end-to-end.

- ☹ It often results in dense computations with $\mathcal{O}(N^2)$ complexity.
- ☹ Sparsifying $\boldsymbol{A}$ results in sparse gradients.
- ☹ Encoding prior structural information requires smart parametrizations.

Probabilistic methods

In this context, probabilistic methods aim at learning a parametric distribution p_Φ for $\boldsymbol{A}$.

- Different parametrizations of p_Φ allow for embedding graph structural priors on the sampled graphs, e.g., edge density, bounded node degree.

Graphs of independent edges

For every edge (i, j)

$$\boldsymbol{A}_{i,j} \sim \text{Bernoulli}(\sigma(\Phi_{i,j})).$$

Fixed-degree graphs

For each node i, sample w/o replacement k nodes from

$$\text{Categorical}\left(\text{SoftMax}(\Phi_{i,1}, \dots, \Phi_{i,N})\right).$$

- As seen before, Φ can be factorized and p_Φ made input dependent, e.g.,

$$\Phi = \xi\left(\boldsymbol{Z}_s \boldsymbol{Z}_t^\top\right), \qquad \boldsymbol{A}_t \sim p_\Phi\left(\boldsymbol{A} | \boldsymbol{X}_{<t}, \boldsymbol{U}_{<t}, \boldsymbol{V}\right).$$

[29] Kazi *et al.*, “Differentiable graph module (dgm) for graph convolutional networks”, IEEE TPAMI 2022.

[30] Cini *et al.*, “Sparse graph learning from spatiotemporal time series”, JMLR 2023.

Learning graph distributions

The problem consists of learning p_Φ by minimizing a loss based on, e.g., point predictions

$$\mathcal{L}(\Phi) = \mathbb{E}_{\boldsymbol{A}\sim p_\Phi}\left[\ell\left(\widehat{\boldsymbol{X}}_{t:t+H}, \boldsymbol{X}_{t:t+H}\right)\right], \qquad \mathcal{L}(\Phi) = \ell\left(\mathbb{E}_{\boldsymbol{A}\sim p_\Phi}\left[\widehat{\boldsymbol{X}}_{t:t+H}\right], \boldsymbol{X}_{t:t+H}\right),$$

where $\boldsymbol{X}_{t:t+H} = \mathcal{F}(\boldsymbol{X}_{t-W:t}, \boldsymbol{A}; \theta)$.

More generally, we can consider losses comparing distributions by relying on a divergence measure Δ"

$$\mathcal{L}(\Phi) = \Delta\left(p_\Phi(\widehat{\boldsymbol{X}}_{t:t+H}), p(\boldsymbol{X}_{t:t+H})\right).$$

⚠ Gradient-based optimization requires computing $\nabla_\Phi \mathcal{L}(\Phi)$,
→ i.e., differentiating w.r.t. the parameters of the target distribution.

- ☹ Analytical gradients are often untractable;
- ☹ Monte Carlo approximations are not straightforward.

Monte Carlo gradient estimators

💡 One approach is to reparametrize $\widetilde{\boldsymbol{A}} \sim p_\Phi(\boldsymbol{A})$ as: $\widetilde{\boldsymbol{A}} = g\left(\Phi, \boldsymbol{\varepsilon}\right), \quad \boldsymbol{\varepsilon} \sim p(\boldsymbol{\varepsilon})$
decoupling parameters Φ from the random component $\boldsymbol{\varepsilon}$: $\nabla_\Phi \mathcal{L}(\Phi) = \mathbb{E}_{\boldsymbol{\varepsilon}}\left[\nabla_\Phi \ell(\widehat{\boldsymbol{X}}, \boldsymbol{X})\right]$.

- ☺ Practical and easy to implement; resulting estimators have low variance.
- ☹ Rely on continuous relaxations and make subsequent computations scale with $\mathcal{O}(N^2)$.

💡 Conversely, score-function (SF) gradient estimators rely on the relation

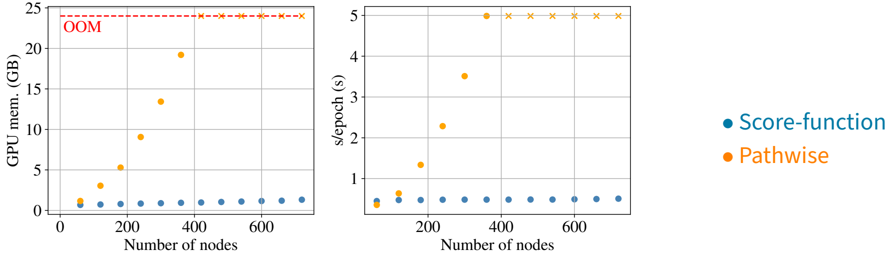
$$\nabla_\Phi \mathbb{E}_{p_\Phi}\left[\ell(\widehat{\boldsymbol{X}}, \boldsymbol{X})\right] = \mathbb{E}_{p_\Phi}\left[\ell(\widehat{\boldsymbol{X}}, \boldsymbol{X}) \nabla_\Phi \log p_\Phi\right]$$

- ☹ Suffer from high variance. → Use variance reduction techniques.
- ☺ Allow to keep computations sparse through the model.

Both approaches allow for **Monte Carlo** estimation.

[31] Kipf *et al.*, “Neural relational inference for interacting systems”, ICML 2018.
[30] Cini *et al.*, “Sparse graph learning from spatiotemporal time series”, JMLR 2023.

Computational efficiency



With score-based gradient estimators $\nabla_\Phi \mathcal{L}(\Phi) = \mathbb{E}_{\boldsymbol{A}\sim p_\Phi}\left[\ell(\widehat{\boldsymbol{X}}, \boldsymbol{X})\nabla_\Phi \log p_\Phi(\boldsymbol{A})\right]$.

☺ They are computationally efficient as ∇_Φ is computed with respect to $\log p_\Phi(\boldsymbol{A})$.

- Do not rely on continuous relaxation of discrete random variables;
- Allow for sparse message passing to compute $\widehat{\boldsymbol{X}}$ (and, in turn, of $\ell(\widehat{\boldsymbol{X}}, \boldsymbol{X})$) by rely on sparse matrices $\boldsymbol{A}$.

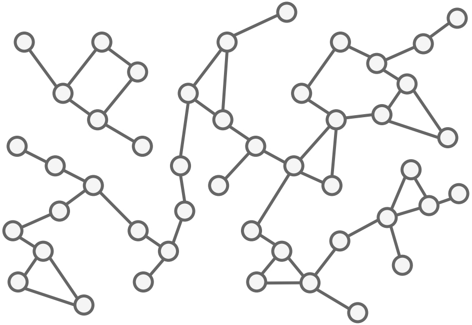
☹ They can be sample inefficient due to the high variance of the gradient estimates.

Scalability

☺ STGNNs are scalable architectures

Graph-based processing allows us to

- ☺ learn a single inductive (global) model...
- ☺ ...while conditioning on related time series in a **sparse** fashion.
- ☺ Cost reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(|\mathcal{E}_t|)$ w.r.t. attention-based architectures.



☹ Often it might not be enough

Spatiotemporal data span – as the name suggests – **two dimensions**:

- the spatial dimension – the number of time series.
- the time dimension – the number of time steps per time series.

In the real world, dealing with high-frequency, large-scale time series data is quite common.

– E.g., smart cities, environmental monitoring, finance

☹ A large amount of data needs to be **processed at once**.

→ We need this to model **long-range** spatiotemporal dependencies.

Computational complexity of STGNNs

W: length of time series · N: number of nodes · $|\mathcal{E}_t|$: number of edges · L: number of MP layers

The computational complexity of T&S models is given by:

- node-wise temporal processing – $\mathcal{O}(WN)$;
- L MP layers **for each time step** – $\mathcal{O}(WL|\mathcal{E}_t|)$.

$$\rightarrow \mathcal{O}(W(N + L|\mathcal{E}_t|))$$

A first step toward improving scalability is represented by TTS models, which perform:

- node-wise temporal processing – $\mathcal{O}(WN)$;
- L MP layers **at the last time step** – $\mathcal{O}(L|\mathcal{E}_t|)$.

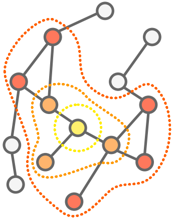
$$\rightarrow \mathcal{O}(WN + L|\mathcal{E}_t|)$$

STT models, instead, do not have computational advantages over T&S models.

Graph subsampling

Computations can be reduced by training on subgraphs of the full network.

- sampling the K-th **order neighborhood** of a subset of nodes;
- **rewiring** the graph to reduce the total number of edges.



Mostly adapted from methods developed in **static graph processing** (e.g., [33], [34]).

☹ Subsampling might break long-range spatiotemporal dependencies.

☹ The learning signal may be noisy.

[32] Gandhi *et al.*, “Spatio-Temporal Multi-graph Networks for Demand Forecasting in Online Marketplaces”, ECML-PKDD 2021.

[33] Hamilton *et al.*, “Inductive representation learning on large graphs”, NeurIPS 2017.

[34] Rong *et al.*, “DropEdge: Towards Deep Graph Convolutional Networks on Node Classification”, ICLR 2020.

Pre-computation

Pre-processing methods (e.g., [35]) enable scalability to large graphs by:

- precomputing a representation for each node's neighborhood **ahead of training**;
- processing the obtained node representations as if they were **i.i.d. samples**.

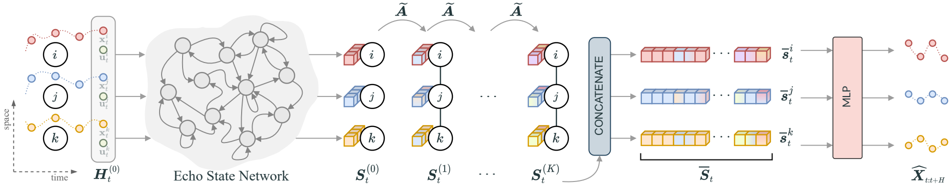
An extension to spatiotemporal data is given by SGP [36], which acts in 2 steps:

1. obtain a temporal encoding at each time step with a deep echo state network[1];
2. propagate such encodings through the graph using powers of a graph shift operator.

[35] Frasca *et al.*, "SIGN: Scalable inception graph neural networks" 2020.
[36] Cini *et al.*, "Scalable Spatiotemporal Graph Neural Networks", AAAI 2023.
[37] Liu *et al.*, "Do we really need graph neural networks for traffic forecasting?" Preprint 2023.
[1] A randomized recurrent neural networks

SGP: Scalable Graph Predictor [36]

Extracted representations can be sampled uniformly across time and space during training.



- ☺ The cost of a training step is independent of W, N and $|\mathcal{E}_t|$.
- ☺ Performance matches state of the art.
- ☹ More storage space is required – the number of extracted features is $\gg d_x$.
- ☹ More reliant on hyperparameter selection than end-to-end approaches.

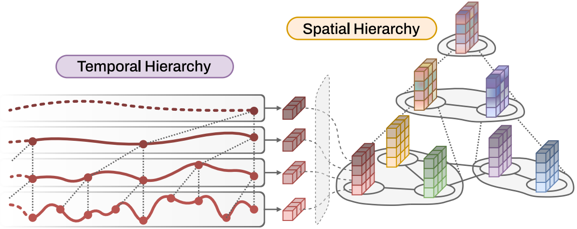
[36] Cini *et al.*, “Scalable Spatiotemporal Graph Neural Networks”, AAAI 2023.

Hierarchical processing

We can reduce computational complexity by using coarser-grained representations of the input.

In space, this can be achieved through graph pooling [38].

- ☺ Reduced number of operations to reach the same receptive field.
- ☹ Might introduce bottlenecks.



[38] Grattarola *et al.*, “Understanding Pooling in Graph Neural Networks” 2024.

Model quality assessment

Questions to answer

Consider a predictor $\mathcal{F}$ trained to solve a time-series forecasting problem.

1. Is the predictor optimal for the problem at hand?
2. Where does the predictor appear sub-optimal?
3. How can we improve the predictor?

Remark: Multiple optimality criteria can be considered.

⚡ Relational inductive biases can help us here too.

Performance at task

Given two predictors $\mathcal{F}_a, \mathcal{F}_b$ and performance metric M (e.g., MAE, MSE).

- we consider $\mathcal{F}_a$ better than $\mathcal{F}_b$ if $M(\mathcal{F}_a)$ is *statistically* better than $M(\mathcal{F}_b)$.
- we consider $\mathcal{F}_a$ optimal if there is no other model $\mathcal{F}_b$ better than $\mathcal{F}_a$.

? Can we tell if the current model $\mathcal{F}_a$ **can be improved** only by **looking at performance**?

- Either we find a new model $\mathcal{F}_*$ better than $\mathcal{F}_a$.
- Or, we have some prior knowledge on the data distribution.

$\ldots$ is there a better way?

Model	M
$\mathcal{F}_a$	$0.145_{\pm 0.002}$
$\mathcal{F}_b$	$0.176_{\pm 0.005}$
$\vdots$	
$\mathcal{F}_n$	$0.158_{\pm 0.004}$
$\mathcal{F}_*$	$0.139_{\pm 0.001}$

Residual correlation analysis

We can design tests for correlation between prediction residuals $\boldsymbol{r}_t^i \doteq \boldsymbol{x}_{t:t+H}^i - \hat{\boldsymbol{x}}_{t:t+H}^i$ to assess model optimality.

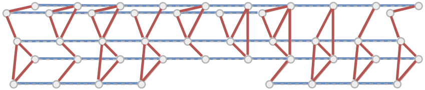
> If residuals are dependent
> $\implies$ there is information that the model hasn't captured
> $\implies$ model predictions can be improved.

Remarks: Residual correlation analysis

- ☺ Independent of specific performance measures.
- ☹ Does not quantify how much a model can improve w.r.t. a specific performance metric.
- ☺ Does not rely on comparisons with other models.

Research focused mainly on either serial correlation [39]–[41] or spatial correlation [42], [43].

AZ-Whiteness test: a spatio-temporal test



The test is defined by statistic

$$C(\{\boldsymbol{r}\}) = \underbrace{\sum_t \sum_{(i,j)\in\mathcal{E}_t} w_{ijt}\,\mathsf{sgn}(\langle \boldsymbol{r}_t^i, \boldsymbol{r}_t^j\rangle)}_{\text{spatial edge}} + \underbrace{\sum_t \sum_i w_{it}\,\mathsf{sgn}(\langle \boldsymbol{r}_t^i, \boldsymbol{r}_{t+1}^i\rangle)}_{\text{temporal edge}} \qquad \to \mathcal{N}(0,1)$$

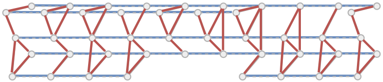
- ☺ Distribution-free and residuals can be non-identically distributed.
- ☺ Computation is linear in the number of edges and time steps.

[44] Zambon *et al.*, “AZ-whiteness Test: A Test for Signal Uncorrelation on Spatio-Temporal Graphs”, NeurIPS 2022.

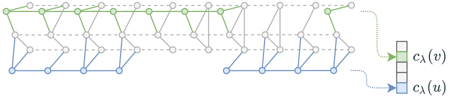
Where can we improve?

Analyzing the AZ-whiteness test statistic computed on subgraphs of the spatio-temporal graph allows for discovering insightful correlation patterns.

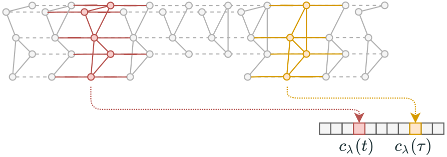
Spatial (or temporal) edges



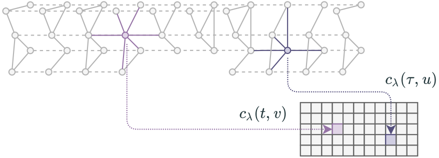
Edges related to a node



Edges related to a time step



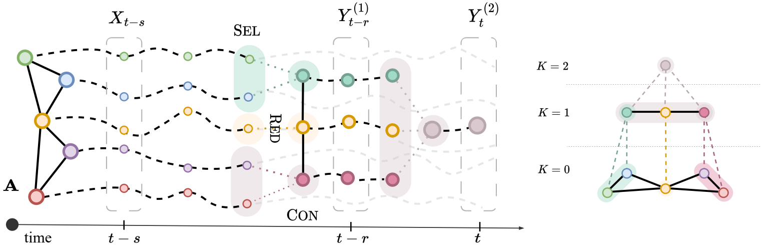
Edges related to a node



[45] Zambon *et al.*, “Where and How to Improve Graph-based Spatio-temporal Predictors” 2023.

Additional Directions

Higher-order dependencies & hierarchical processing



- ☹ Standard STGNNs operate at a fixed spatiotemporal scale.
- 💡 Combine hierarchical and graph-based representations.
- ☺ Exploit higher-order dependencies by operating on hierarchical representations of the input.
- ☺ Can also be used for hierarchical forecasting and to obtain reconciled predictions.

[46] Yu *et al.*, “ST-Unet: A spatio-temporal U-network for graph-structured time series modeling” 2019.

[47] Cini *et al.*, “Graph-based Time Series Clustering for End-to-End Hierarchical Forecasting”, ICML 2024.

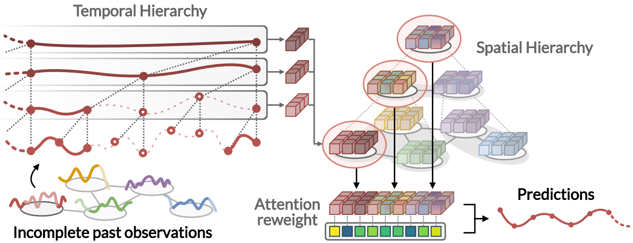
[24] Marisca *et al.*, “Graph-based Forecasting with Missing Data through Spatiotemporal Downsampling”, ICML 2024.

Multi-scale spatiotemporal representations

Similar hierarchical processing can be jointly performed also over the temporal dimension.

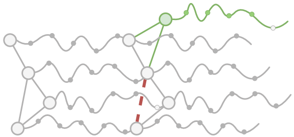
This gives a hierarchy of multi-scale representations, each accounting for a specific space-time resolution.

- ☺ Different scales might capture different dynamics.
- ☺ Helps with noisy and missing data.



[24] Marisca *et al.*, “Graph-based Forecasting with Missing Data through Spatiotemporal Downsampling”, ICML 2024.
[46] Yu *et al.*, “ST-Unet: A spatio-temporal U-network for graph-structured time series modeling” 2019.

Inductive settings & foundation models



In real-world applications, one often needs to

- operate under changes in the network connectivity;
- make predictions for newly added nodes;
- transfer the model to different networks.

Useful in several tasks, like, forecasting, missing data imputation, and virtual sensing.

⚠ Performance can easily degrade if the target distribution deviates from that at training nodes.
→ A lot of active research on foundation models for spatiotemporal data (not trivial)[48], [49].

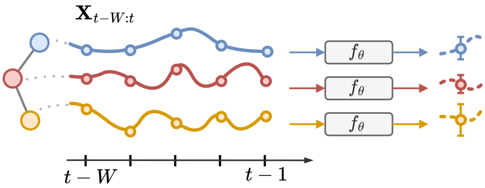
[15] Cini *et al.*, "Taming Local Effects in Graph-based Spatiotemporal Forecasting", NeurIPS 2023.
[48] Yuan *et al.*, "Diffusion Transformers as Open-World Spatiotemporal Foundation Models", NeurIPS 2025.
[49] Yuan *et al.*, "Spatio-Temporal Few-Shot Learning via Diffusive Neural Network Generation", ICLR 2024.

Uncertainty quantification

💡 Condition uncertainty estimates on observations at correlated time series.

→ This can provide reliable confidence intervals to support decision-making.

- Relational priors can be used, e.g., to regularize covariance estimates.



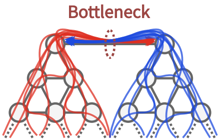
[50] Cini *et al.*, “Relational Conformal Prediction for Correlated Time Series”, ICML 2025.
[51] Zhang *et al.*, “Topology-Aware Conformal Prediction for Stream Networks”, NeurIPS 2025.
[52] Zhuang *et al.*, “SAUC: Sparsity-aware uncertainty calibration for spatiotemporal prediction with graph neural networks”, SIGSPATIAL 2024.

Additional directions

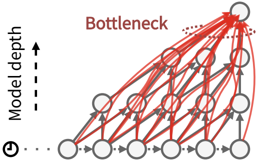
Architectural biases of STGNNs

GNNs suffer from over-squashing, where information is compressed and lost through bottlenecks.

STGNNs introduce a temporal dimension, compounding the issue [53], [54].



Over-squashing in GNNs



Over-squashing in TCNs

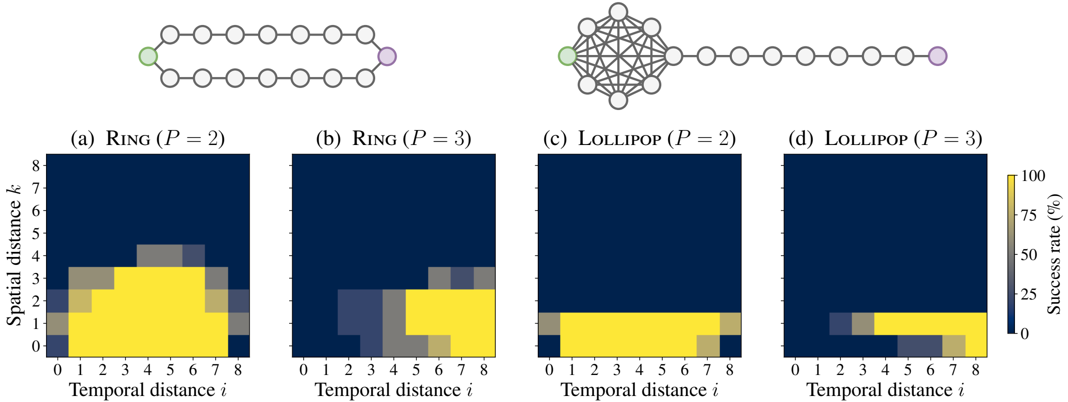
How does spatiotemporal over-squashing affect representation learning in STGNNs?

One approach to study it is through the spectral norm of the STGNN's Jacobian:

$$\left\| \nabla_{\substack{u \\ i}} \boldsymbol{h}_t^{v(L)} \right\| = \left\| \frac{\partial \boldsymbol{h}_t^{v(L)}}{\partial \boldsymbol{h}_{t-i}^{u(0)}} \right\| .$$

[55] Marisca *et al.*, "Over-squashing in Spatiotemporal Graph Neural Networks", NeurIPS 2025.

Some results for TCN-based STGNNs



[55] Marisca *et al.*, “Over-squashing in Spatiotemporal Graph Neural Networks”, NeurIPS 2025.

Additional directions

Benchmarks

Open datasets

In line with OGB [56], TGB [57], TGB 2.0 [58].

- Energy analytics [36]
- Weather forecasting [59]
- Traffic flow (LargeST) [60]
- ...

Software

Standard model evaluation platforms

- **Torch Spatiotemporal** [61]
- BasicTS [62]
- ...

[59] Zambon *et al.*, “PeakWeather: MeteoSwiss Weather Station Measurements for Spatiotemporal Deep Learning” 2025.
[36] Cini *et al.*, “Scalable Spatiotemporal Graph Neural Networks”, AAAI 2023.
[60] Liu *et al.*, “Largest: A benchmark dataset for large-scale traffic forecasting”, NeurIPS (D&B) 2024.
[61] Cini *et al.*, *Torch Spatiotemporal*, https://github.com/TorchSpatiotemporal/tsl 2022.
[62] Shao *et al.*, “Exploring Progress in Multivariate Time Series Forecasting: Comprehensive Benchmarking and Heterogeneity Analysis”, IEEE TKDE 2024.

Conclusions

Some Takeaways

Deep Learning for **time series** + Deep Learning on **graphs**

- ⚡ Relational inductive biases allow for exploiting dependencies among the time series,
- ☺ ...while sharing most of the model parameters,
- ☺ ...and overcoming limits due to irregularities in time and space.
- 💡 Whenever possible, global-local models are a safe starting point.

Challenges. Scalability • Missing data • Latent graph learning • Model quality assessment

Resources. 📰 Tutorial paper [3] • Open-source library [61]

[3] Cini, Marisca, Zambon, and Alippi, “Graph Deep Learning for Time Series Forecasting”, ACM Comput. Surv. 2025.

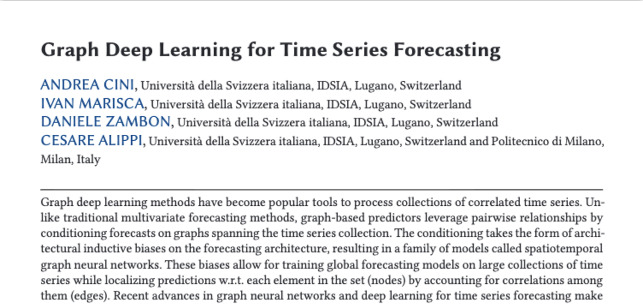
[61] Cini and Marisca, *Torch Spatiotemporal*, https://github.com/TorchSpatiotemporal/tsl 2022.

Collaborators

Andrea **Cini**

Ivan **Marisca**

Daniele **Zambon**

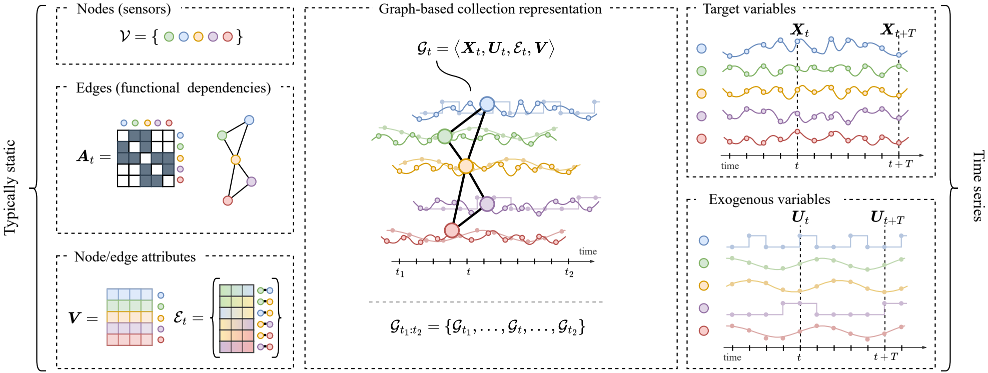


Graph Deep Learning for Time Series Forecasting

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

Graph deep learning methods have become popular tools to process collections of correlated time series. Unlike traditional multivariate forecasting methods, graph-based predictors leverage pairwise relationships by conditioning forecasts on graphs spanning the time series collection. The conditioning takes the form of architectural inductive biases on the forecasting architecture, resulting in a family of models called spatiotemporal graph neural networks. These biases allow for training global forecasting models on large collections of time series while localizing predictions w.r.t. each element in the set (nodes) by accounting for correlations among them (edges). Recent advances in graph neural networks and deep learning for time series forecasting make

Graph Deep Learning for Time Series Processing



Typically static

Nodes (sensors)

$\mathcal{V} = \{ ○ ○ ○ ○ ○ \}$

Edges (functional dependencies)

$\boldsymbol{A}_t =$

Node/edge attributes

$\boldsymbol{V} =$

$\mathcal{E}_t =$

Graph-based collection representation

$\mathcal{G}_t = \langle \boldsymbol{X}_t, \boldsymbol{U}_t, \mathcal{E}_t, \boldsymbol{V} \rangle$

$\mathcal{G}_{t_1:t_2} = \{\mathcal{G}_{t_1}, \dots, \mathcal{G}_t, \dots, \mathcal{G}_{t_2}\}$

Target variables

Exogenous variables

Time series

Questions?

DEMO

Coding Spatiotemporal GNNs

tsl: PyTorch Spatiotemporal Library

tsl (Torch Spatiotemporal) is a python library built upon PyTorch and PyG to accelerate research on neural spatiotemporal data processing methods, with a focus on **Graph Neural Networks**.

- torch-spatiotemporal.readthedocs.io
- github.com/TorchSpatiotemporal/tsl

Notebook

Spatiotemporal Graph Neural Networks with tsl

Open in Colab

[61] Cini and Marisca, *Torch Spatiotemporal*, https://github.com/TorchSpatiotemporal/tsl 2022.

References i

[1] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, **“DeepAR: Probabilistic forecasting with autoregressive recurrent networks,”** *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[2] K. Benidis, S. S. Rangapuram, V. Flunkert, *et al.*, **“Deep learning for time series forecasting: Tutorial and literature survey,”** *ACM Comput. Surv.*, vol. 55, no. 6, Dec. 2022, ISSN: 0360-0300. DOI: 10.1145/3533382. [Online]. Available: https://doi.org/10.1145/3533382.

[3] A. Cini, I. Marisca, D. Zambon, and C. Alippi, **“Graph deep learning for time series forecasting,”** *ACM Comput. Surv.*, 2025.

[4] P. Montero-Manso and R. J. Hyndman, **“Principles and algorithms for forecasting groups of time series: Locality and globality,”** *International Journal of Forecasting*, vol. 37, no. 4, pp. 1632–1653, 2021.

[5] R. Sen, H.-F. Yu, and I. S. Dhillon, **“Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting,”** *Advances in Neural Information Processing Systems*, vol. 32, 2019.

References ii

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

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, **"Empirical evaluation of gated recurrent neural networks on sequence modeling,"** *arXiv preprint arXiv:1412.3555*, 2014.

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

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

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

[11] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, **"Graph wavenet for deep spatial-temporal graph modeling,"** in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 1907–1913.

References iii

[12] I. Marisca, A. Cini, and C. Alippi, **"Learning to reconstruct missing data from spatiotemporal graphs with sparse observations,"** in *Advances in Neural Information Processing Systems*, 2022.

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

[14] M. Sabbaqi and E. Isufi, **"Graph-time convolutional neural networks: Architecture and theoretical analysis,"** *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 12, pp. 14 625–14 638, Dec. 2023, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2023.3311912.

[15] A. Cini, I. Marisca, D. Zambon, and C. Alippi, **"Taming local effects in graph-based spatiotemporal forecasting,"** *arXiv preprint arXiv:2302.04071*, 2023.

[16] L. Butera, G. De Felice, A. Cini, and C. Alippi, **"On the regularization of learnable embeddings for time series processing,"** *Transactions on Machine Learning Research*, 2025.

[17] A. Cini, I. Marisca, and C. Alippi, **"Filling the g_ap_s: Multivariate time series imputation by graph neural networks,"** in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=kOu3-S3wJ7.

References iv

[18] A. Jenkins, A. Cini, J. Barker, *et al.*, **“Learning to predict global atrial fibrillation dynamics from sparse measurements,”** *arXiv preprint arXiv:2502.09473*, 2025, preprint.

[19] Y. Tashiro, J. Song, Y. Song, and S. Ermon, **“CSDI: Conditional score-based diffusion models for probabilistic time series imputation,”** *Advances in neural information processing systems*, vol. 34, pp. 24 804–24 816, 2021.

[20] M. Liu, H. Huang, H. Feng, L. Sun, B. Du, and Y. Fu, **“Pristi: A conditional diffusion framework for spatiotemporal imputation,”** in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, IEEE, 2023, pp. 1927–1939.

[21] H. Wen, Y. Lin, Y. Xia, *et al.*, **“DiffSTG: Probabilistic spatio-temporal graph forecasting with denoising diffusion models,”** in *Proceedings of the 31st ACM international conference on advances in geographic information systems*, 2023, pp. 1–12.

[22] X. Zhang, M. Zeman, T. Tsiligkaridis, and M. Zitnik, **“Graph-guided network for irregularly sampled multivariate time series,”**, 2022.

References v

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

[24] I. Marisca, C. Alippi, and F. M. Bianchi, **“Graph-based forecasting with missing data through spatiotemporal downsampling,”** in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235, PMLR, 2024, pp. 34 846–34 865.

[25] Y. Wu, D. Zhuang, A. Labbe, and L. Sun, **“Inductive Graph Neural Networks for Spatiotemporal Kriging,”** in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 4478–4485.

[26] G. De Felice, A. Cini, D. Zambon, V. Gusev, and C. Alippi, **“Graph-based Virtual Sensing from Sparse and Partial Multivariate Observations,”** in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: `https://openreview.net/forum?id=CAqdG2dy5s`.

[27] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, **“Learning laplacian matrix in smooth graph signal representations,”** *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, 2016.

[28] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, **“Connecting the dots: Identifying network structure via graph signal processing,”** *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 16–43, 2019.

References vi

[29] A. Kazi, L. Cosmo, S.-A. Ahmadi, N. Navab, and M. M. Bronstein, **"Differentiable graph module (dgm) for graph convolutional networks,"** *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, pp. 1606–1617, 2022.

[30] A. Cini, D. Zambon, and C. Alippi, **"Sparse graph learning from spatiotemporal time series,"** *Journal of Machine Learning Research*, vol. 24, no. 242, pp. 1–36, 2023.

[31] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, **"Neural relational inference for interacting systems,"** in *International conference on machine learning*, PMLR, 2018, pp. 2688–2697.

[32] A. Gandhi, Aakanksha, S. Kaveri, and V. Chaoji, **"Spatio-temporal multi-graph networks for demand forecasting in online marketplaces,"** in *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part IV*, 2021, pp. 187–203, ISBN: 978-3-030-86513-9. DOI: `10.1007/978-3-030-86514-6_12`. [Online]. Available: `https://doi.org/10.1007/978-3-030-86514-6_12`.

[33] W. Hamilton, Z. Ying, and J. Leskovec, **"Inductive representation learning on large graphs,"** *Advances in neural information processing systems*, vol. 30, 2017.

References vii

[34] Y. Rong, W. Huang, T. Xu, and J. Huang, **"Dropedge: Towards deep graph convolutional networks on node classification,"** in *International Conference on Learning Representations*, 2020.

[35] F. Frasca, E. Rossi, D. Eynard, B. Chamberlain, M. Bronstein, and F. Monti, **"SIGN: Scalable inception graph neural networks,"** *arXiv preprint arXiv:2004.11198*, 2020.

[36] A. Cini, I. Marisca, F. M. Bianchi, and C. Alippi, **"Scalable spatiotemporal graph neural networks,"** *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 6, pp. 7218–7226, Jun. 2023. DOI: 10.1609/aaai.v37i6.25880.

[37] X. Liu, Y. Liang, C. Huang, *et al.*, **"Do we really need graph neural networks for traffic forecasting?"** *arXiv preprint arXiv:2301.12603*, 2023.

[38] D. Grattarola, D. Zambon, F. M. Bianchi, and C. Alippi, **"Understanding pooling in graph neural networks,"** *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 2, pp. 2708–2718, 2024.

[39] J. Hosking, **"Equivalent Forms of the Multivariate Portmanteau Statistic,"** *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 43, no. 2, pp. 261–262, 1981.

[40] Z. Li, C. Lam, J. Yao, and Q. Yao, **"On Testing for High-Dimensional White Noise,"** *The Annals of Statistics*, vol. 47, no. 6, pp. 3382–3412, 2019.

References viii

[41] A. Bose and W. Hachem, **"A Whiteness Test Based on the Spectral Measure of Large Non-Hermitian Random Matrices,"** in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 8768–8771.

[42] P. A. P. Moran, **"Notes on Continuous Stochastic Phenomena,"** *Biometrika*, vol. 37, no. 1/2, pp. 17–23, 1950, ISSN: 0006-3444. DOI: 10.2307/2332142.

[43] A. D. Cliff and K. Ord, **"Spatial Autocorrelation: A Review of Existing and New Measures with Applications,"** *Economic Geography*, vol. 46, pp. 269–292, 1970, ISSN: 0013-0095. DOI: 10.2307/143144.

[44] D. Zambon and C. Alippi, **"AZ-whiteness test: A test for signal uncorrelation on spatio-temporal graphs,"** in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022.

[45] D. Zambon and C. Alippi, **"Where and how to improve graph-based spatio-temporal predictors,"** *arXiv preprint arXiv:2302.01701*, 2023.

[46] B. Yu, H. Yin, and Z. Zhu, **"ST-Unet: A spatio-temporal U-network for graph-structured time series modeling,"** *arXiv preprint arXiv:1903.05631*, 2019.

References ix

[47] A. Cini, D. Mandic, and C. Alippi, **“Graph-based Time Series Clustering for End-to-End Hierarchical Forecasting,”** *International Conference on Machine Learning*, 2024.

[48] Y. Yuan, C. Han, J. Ding, G. Zhang, D. Jin, and Y. Li, **“Diffusion transformers as open-world spatiotemporal foundation models,”** in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[49] Y. Yuan, C. Shao, J. Ding, D. Jin, and Y. Li, **“Spatio-temporal few-shot learning via diffusive neural network generation,”** in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: `https://openreview.net/forum?id=QyFm3D3Tzi`.

[50] A. Cini, A. Jenkins, D. Mandic, C. Alippi, and F. M. Bianchi, **“Relational conformal prediction for correlated time series,”** in *International Conference on Machine Learning*, 2025.

[51] J. Zhang, F. Wang, Z. Song, P. S. Yu, K. Ding, and S. Zhu, **“Topology-aware conformal prediction for stream networks,”** in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. [Online]. Available: `https://openreview.net/forum?id=Ldni3xeyIa`.

References x

[52] D. Zhuang, Y. Bu, G. Wang, S. Wang, and J. Zhao, **"SAUC: Sparsity-aware uncertainty calibration for spatiotemporal prediction with graph neural networks,"** in *Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems*, 2024, pp. 160–172.

[53] Y. Bengio, P. Simard, and P. Frasconi, **"Learning long-term dependencies with gradient descent is difficult,"** *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[54] U. Alon and E. Yahav, **"On the bottleneck of graph neural networks and its practical implications,"** in *International Conference on Learning Representations*, 2021.

[55] I. Marisca, J. Bamberger, C. Alippi, and M. M. Bronstein, **"Over-squashing in spatiotemporal graph neural networks,"** *arXiv preprint arXiv:2506.15507*, 2025. [Online]. Available: `https://arxiv.org/abs/2506.15507`.

[56] W. Hu, M. Fey, M. Zitnik, *et al.*, **"Open graph benchmark: Datasets for machine learning on graphs,"** in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 22 118–22 133. [Online]. Available: `https://proceedings.neurips.cc/paper_files/paper/2020/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf`.

References xi

[57] S. Huang, F. Poursafaei, J. Danovitch, *et al.*, **"Temporal graph benchmark for machine learning on temporal graphs,"** in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36, Curran Associates, Inc., 2023, pp. 2056–2073. [Online]. Available: `https://proceedings.neurips.cc/paper_files/paper/2023/file/066b98e63313162f6562b35962671288-Paper-Datasets_and_Benchmarks.pdf`.

[58] J. Gastinger, S. Huang, M. Galkin, *et al.*, **"Tgb 2.0: A benchmark for learning on temporal knowledge graphs and heterogeneous graphs,"** in *Advances in Neural Information Processing Systems*, 2024.

[59] D. Zambon, M. Cattaneo, I. Marisca, J. Bhend, D. Nerini, and C. Alippi, **"Peakweather: Meteoswiss weather station measurements for spatiotemporal deep learning,"** *arXiv preprint arXiv:2506.13652*, 2025.

[60] X. Liu, Y. Xia, Y. Liang, *et al.*, **"Largest: A benchmark dataset for large-scale traffic forecasting,"** *Advances in Neural Information Processing Systems*, vol. 36, 2024.

References xii

[61] A. Cini and I. Marisca, ***Torch Spatiotemporal*****,** Mar. 2022. [Online]. Available: `https://github.com/TorchSpatiotemporal/tsl`.

[62] Z. Shao, F. Wang, Y. Xu, *et al.*, **“Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis,”** *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–14, 2024. DOI: `10.1109/TKDE.2024.3484454`.

[63] C. Gray, L. Mitchell, and M. Roughan, **“Bayesian inference of network structure from information cascades,”** *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, pp. 371–381, 2020.

[64] A. Manenti, D. Zambon, and C. Alippi, ***Learning Latent Graph Structures and their Uncertainty*****,** May 2024.