Graph Deep Learning for Spatiotemporal Time Series

Forecasting, Reconstruction and Analysis

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

ECML/PKDD, Turin · September 22, 2023

Graph Machine Learning Group (gmlg.ch)
The Swiss AI Lab IDSIA
Università della Svizzera italiana

Traffic monitoring

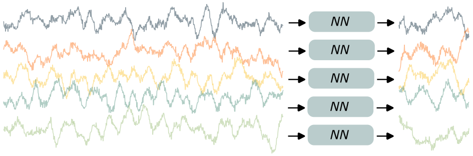
Smart cities

Energy analitics

Physics

Stock markets

Deep learning for time series forecasting



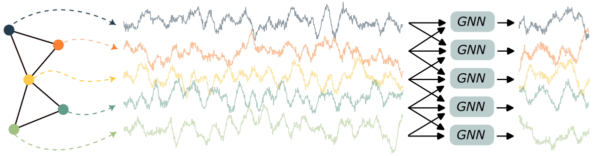
The standard deep learning approach to time series forecasting consists in training a single neural network on a collection of time series.

- Each time series is treated independently from the others.
- A single set of shared learnable parameters is used to predict each time series.
- Resulting models are effective and efficient.

Dependencies across time series are often discarded.

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

Relational inductive biases



One way out is to embed such relational structure as an architectural bias into the processing.

Graph neural networks provide appropriate neural operators.

- Message-passing blocks allow for localizing the predictions
  $\rightarrow$ conditioning on observations at related time series (neighboring nodes).
- Parameters are shared and the model can operate on arbitrary sets of time series.

[2] D. Bacciu *et al.*, “A gentle introduction to deep learning for graphs”, NN 2020.
[3] M. M. Bronstein *et al.*, “Geometric deep learning: Grids, groups, graphs, geodesics, and gauges” 2021.

What this tutorial is about

This tutorial aims at merging two active and prominent research fields:

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

- We provide a unified exposition of the recent advancements in graph-based time series processing, highlighting challenges and pitfalls.
- We offer researchers and practitioners a complete toolset of methodological guidelines, best practices, and software to exploit such framework in real-world problems.

A reference paper and Python notebook complement this presentation.

[4] A. Cini *et al.*, “Graph Deep Learning for Time Series Forecasting: A Comprehensive Methodological Framework” 2023.

Tutorial outline

Part 1

1.1) Spatiotemporal time series

1.2) Spatiotemporal GNNs

1.3) Global and local models

1.4) Model quality assessment

Software demo

Conclusions

Part 2

2.1) Latent graph learning

2.2) Learning in non-stationary environments

2.3) Scalability

2.4) Dealing with missing data

Part 1

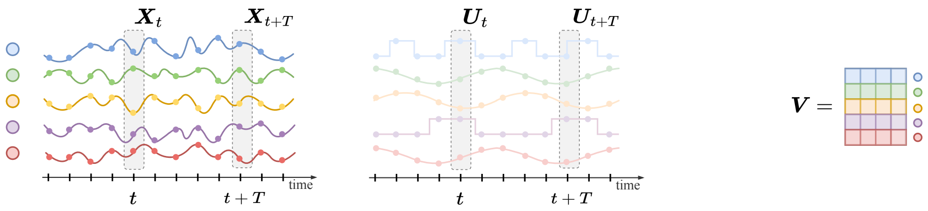
Graph-based Processing of Spatiotemporal Time series

Spatiotemporal time series

Collections of time series

We consider a set of N correlated time series, where each i-th time series is associated with:

- an observation vector $\boldsymbol{x}_t^i \in \mathbb{R}^{d_x}$ at each time step t;
- a vector of exogenous variable $\boldsymbol{u}_t^i \in \mathbb{R}^{d_u}$ at each time step t;
- a vector of static (time-independent) attributes $\boldsymbol{v}^i \in \mathbb{R}^{d_v}$.



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

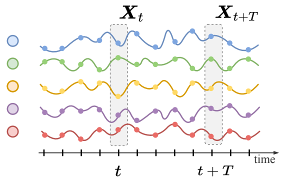
[4] A. Cini *et al.*, “Graph Deep Learning for Time Series Forecasting: A Comprehensive Methodological Framework” 2023.

Correlated time series

We assume a time-invariant stochastic process

$$\boldsymbol{x}_t^i \sim p^i\left(\boldsymbol{x}_t^i \middle| \boldsymbol{X}_{<t}, \boldsymbol{U}_{\leq t}, \boldsymbol{V}\right)$$

generating the data $\boldsymbol{x}_t^i$ for all $i = 1 \dots N$ and $t \in \mathbb{N}$.



Note that the time series:

- can be generated by different processes,
- can depend on each other,
- are assumed

  homogenous, synchronous, regularly sampled.

Notation:

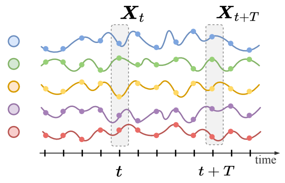
$$\boldsymbol{X}_{t:t+T} = [\boldsymbol{X}_t, \cdots, \boldsymbol{X}_{t+T-1}]$$
$$\boldsymbol{X}_{<t} = [\boldsymbol{X}_0, \cdots, \boldsymbol{X}_{t-2}, \boldsymbol{X}_{t-1}]$$

$\rightarrow$ These assumptions can be relaxed, as we will discuss in the 2nd part.

Relational information

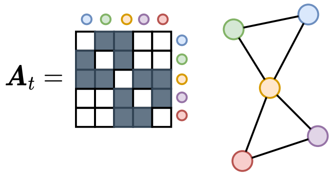
We assume the existence of functional dependencies between the time series.

$\rightarrow$ e.g., forecasts for one time series can be improved by accounting for the past values of other time series.



We 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).



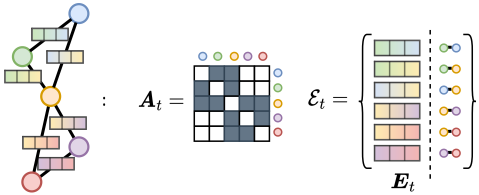
$\rightarrow$ We call spatial the dimension spanning the time series collection.

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 $\boldsymbol{A}_t$.

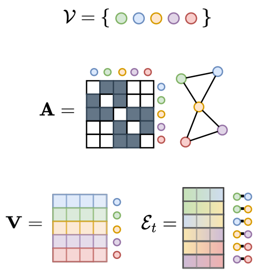
The set of attributed edges encoding all the available relational information is denoted by

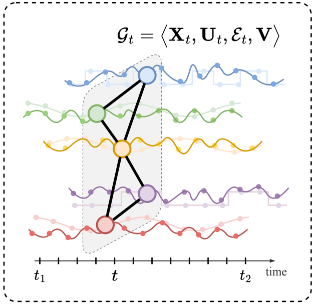
$$\mathcal{E}_t \doteq \{\langle (i,j), \boldsymbol{e}_t^{ij} \rangle \mid \forall i,j : \boldsymbol{A}_t[i,j] \neq 0\}.$$

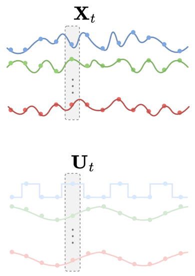


$\rightarrow$ For many applications, $\boldsymbol{A}_t$ changes slowly over time and can be considered as constant within a short window of observations.

Spatiotemporal time series




$\mathcal{G}_t = \langle \mathbf{X}_t, \mathbf{U}_t, \mathcal{E}_t, \mathbf{V} \rangle$
t_1
t
t_2
time



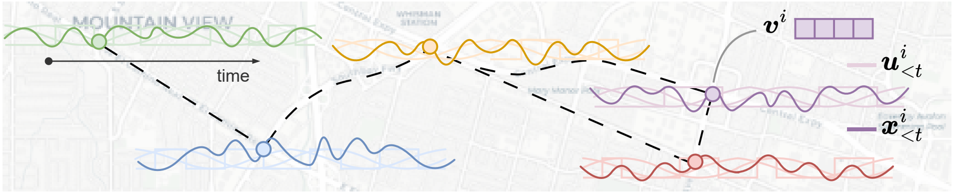
We use the terms node and sensor to indicate the N entities generating the time series.

$\rightarrow$ We refer to the node set together with the relational information as sensor network.

The tuple $\mathcal{G}_t \doteq \langle \boldsymbol{X}_t, \boldsymbol{U}_t, \mathcal{E}_t, \boldsymbol{V} \rangle$ contain all the available information associated with time step t.

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

Time series forecasting

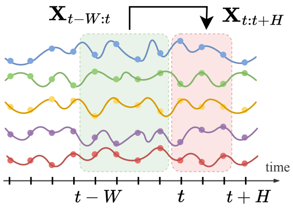
Multi-step time-series forecasting

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

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

predict $H \geq 1$ future observations

$$\boldsymbol{x}^i_{t+h}, \qquad i = 1 \cdots N,\ h = 1 \cdots H.$$



In particular, we are interested in learning a parametric model $p_{\boldsymbol{\theta}}$ approximating the unknown data distribution p

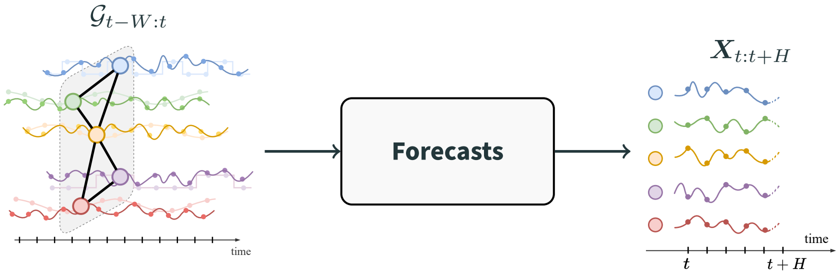
$$p_{\boldsymbol{\theta}}\left(\boldsymbol{x}^i_{t+h} \middle| \boldsymbol{X}_{t-W:t}, \boldsymbol{U}_{t-W:t+h}, \boldsymbol{V}\right) \approx p^i\left(\boldsymbol{x}^i_{t+h} \middle| \boldsymbol{X}_{<t}, \boldsymbol{U}_{\leq t+h}, \boldsymbol{V}\right).$$

- $\boldsymbol{\theta}$ is the model parameter vector.

Time series forecasting + relational inductive biases

Condition the model on the relational information $\mathcal{E}_{t-W:t}$

$$p_{\boldsymbol{\theta}}\left(\boldsymbol{x}^i_{t+h} \middle| \mathcal{G}_{t-W:t}, \boldsymbol{U}_{t-W:t+h}, \boldsymbol{V}\right)$$



⚡ The conditioning on the sequence of attributed graphs acts as a regularization to localize predictions w.r.t. the neighborhood of each node.

Point forecasts

For simplicity, we focus here on point forecasts, rather than the modeling of full data distributions p, and consider predictive model

$$\widehat{\boldsymbol{x}}^i_{t+h} = \mathcal{F}\left(\mathcal{G}_{t-W:t}, \boldsymbol{U}_{t:t+h}; \boldsymbol{\theta}\right)$$

where $\widehat{\boldsymbol{x}}^i_{t+h}$ approximates, e.g., $\mathbb{E}_p\left[\boldsymbol{x}^i_{t+h}\right]$.

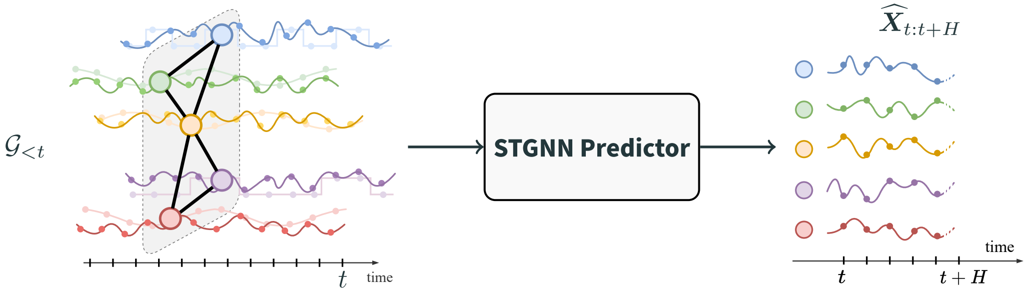
Parameters $\boldsymbol{\theta}$ can be learned by minimizing a cost function $\ell(\,\cdot\,,\,\cdot\,)$ (e.g., MSE) on a training set

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

Spatiotemporal Graph Neural Networks

Spatiotemporal Graph Neural Networks

We call Spatiotemporal Graph Neural Network (STGNN) a neural network exploiting both temporal and spatial relations of the input spatiotemporal time series.



We focus on models based on message passing.

Message-passing neural networks

To process the spatial dimension, we rely on the **message-passing (MP)** framework

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

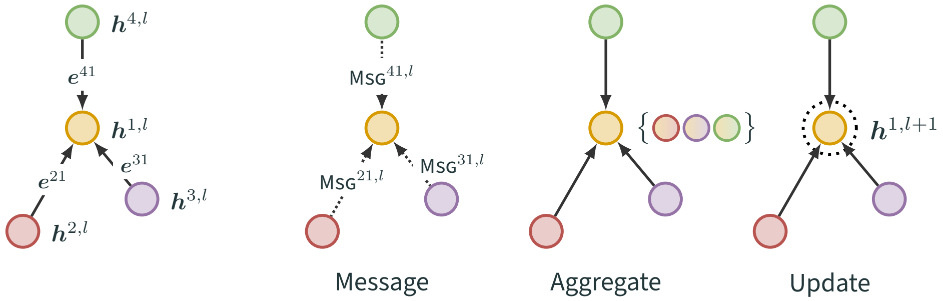
Where:

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

Aggregation is performed over $\mathcal{N}(i)$, i.e., the set of neighbors of node i.

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

Message passing in action



Spatiotemporal message passing

Starting from the MP framework, we can define a general scheme for spatiotemporal message-passing (STMP) networks:

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

Rather than vectors, STMP blocks process **sequences**.

$\rightarrow$ STMP blocks must be implemented with operators that work on sequences!

We will look at different implementations of STMP blocks in the following.

[4] A. Cini *et al.*, "Graph Deep Learning for Time Series Forecasting: A Comprehensive Methodological Framework" 2023.

A general recipe

We consider STGNNs can be expressed as a sequence of three operations:

$$\boldsymbol{h}^{i,0}_{t-1} = \textsc{Encoder}\left(\boldsymbol{x}^i_{t-1}, \boldsymbol{u}^i_{t-1}, \boldsymbol{v}^i\right), \tag{2}$$

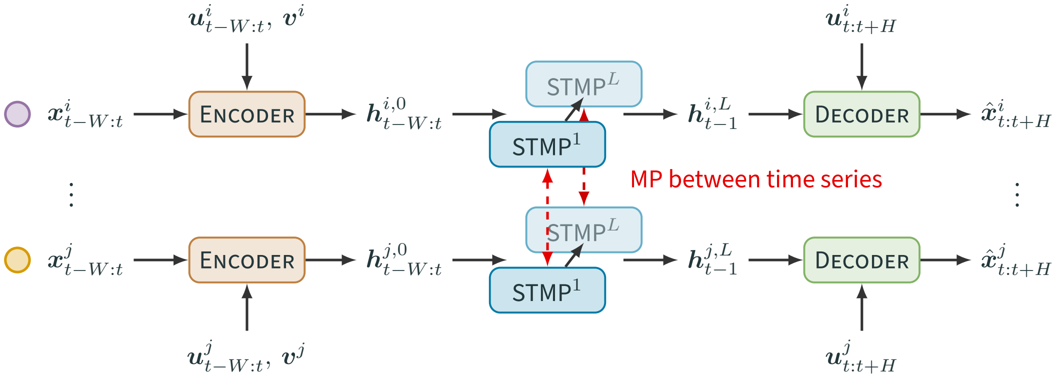
$$\boldsymbol{H}^{l+1}_{t-1} = \textsf{STMP}^{l}\left(\boldsymbol{H}^{l}_{\leq t-1}, \mathcal{E}_{\leq t-1}\right), \quad l = 0, \dots, L-1 \tag{3}$$

$$\hat{\boldsymbol{x}}^i_{t:t+H} = \textsc{Decoder}\left(\boldsymbol{h}^{i,L}_{t-1}, \boldsymbol{u}^i_{t:t+H}\right). \tag{4}$$

Where:

- $\textsc{Encoder}(\,\cdot\,)$ is the encoding layer, e.g., implemented by an MLP.
- STMP is a stack of STMP layers.
- $\textsc{Decoder}(\,\cdot\,)$ is the readout layer, e.g., implemented by an MLP.

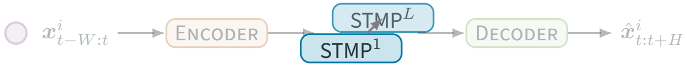
Framework overview



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)**
  Embed each time series in a vector, which is then propagated over the graph.
- **Space-then-Time (STT)**
  Propagate nodes features at first and then process the resulting time series.



Time-and-Space

In T&S models, representations at every node and time step are the results of a joint temporal and spatial encoding

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

Several options exist.

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

Example 1: From Recurrent Neural Networks...

Consider a standard GRU [6] cell.

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

$$\boldsymbol{u}_t^i = \sigma\left(\boldsymbol{\Theta}_u\left[\boldsymbol{x}_t^i || \boldsymbol{h}_{t-1}^i\right] + \boldsymbol{b}_u\right) \tag{6}$$

$$\boldsymbol{c}_t^i = \tanh\left(\boldsymbol{\Theta}_c\left[\boldsymbol{x}_t^i || \boldsymbol{r}_t^i \odot \boldsymbol{h}_{t-1}^i\right] + \boldsymbol{b}_c\right) \tag{7}$$

$$\boldsymbol{h}_t^i = \left(1 - \boldsymbol{u}_t^i\right) \odot \boldsymbol{c}_t^i + \boldsymbol{u}_t^i \odot \boldsymbol{h}_{t-1}^i \tag{8}$$

Time series can be processed **independently** for each node or as a **single multivariate** time series.

[6] J. 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:

$$\boldsymbol{Z}_t^l = \boldsymbol{H}_t^{l-1} \tag{9}$$

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

$$\boldsymbol{O}_t^l = \sigma\left(\mathsf{MP}_o^l\left(\left[\boldsymbol{Z}_t^l || \boldsymbol{H}_{t-1}^l\right], \mathcal{E}_t\right)\right), \tag{11}$$

$$\boldsymbol{C}_t^l = \tanh\left(\mathsf{MP}_c^l\left(\left[\boldsymbol{Z}_t^l || \boldsymbol{R}_t^l \odot \boldsymbol{H}_{t-1}^l\right], \mathcal{E}_t\right)\right), \tag{12}$$

$$\boldsymbol{H}_t^l = \boldsymbol{O}_t^l \odot \boldsymbol{H}_{t-1}^l + (1 - \boldsymbol{O}_t^l) \odot \boldsymbol{C}_t^l, \tag{13}$$

These T&S models are known as graph convolutional recurrent neural networks (GCRNNs) [7].

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

Popular GCRNNs

The first GCRNN has been introduced in [7], with MP blocks implemented as polynomial graph convolutional filters.

GCRNNs have become popular in the traffic forecasting context with the Diffusion Convolutional Recurrent Neural Network (DCRNN) architecture [8].

In DCRNN, MP is performed through bidirectional diffusion convolution:

$$\boldsymbol{H}_t' = \sum_{k=0}^{K} \left(\boldsymbol{D}_{t,\text{out}}^{-1}\boldsymbol{A}_t\right)^k \boldsymbol{H}_t\boldsymbol{\Theta}_1^{(k)} + \left(\boldsymbol{D}_{t,\text{in}}^{-1}\boldsymbol{A}_t^\top\right)^k \boldsymbol{H}_t\boldsymbol{\Theta}_2^{(k)} \tag{14}$$

[7] Y. Seo *et al.*, “Structured sequence modeling with graph convolutional recurrent networks”, ICONIP 2018.
[8] Y. Li *et al.*, “Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting”, ICLR 2018.

Example 2: Spatiotemporal convolutional networks (i)

A completely different approach is that of spatiotemporal convolutional networks (STCNs), that **alternate spatial and temporal convolutional filters**:

- Compute intermediate representations by using a node-wise temporal convolutional layer:

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

  where TCN^l indicates a temporal convolutional network layer.

- Then, compute the updated representation by using a time-wise graph convolution:

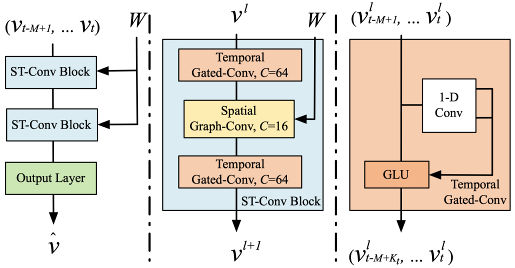
$$\boldsymbol{H}^l_t = \mathsf{MP}^l\left(\boldsymbol{Z}^l_t, \mathcal{E}_t\right) \qquad \forall\, t$$

Spatiotemporal convolutional networks (ii)

The first example of such architecture is the STGCN by Yu et al. [9].

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
$(v^l_{t-M+K_t}, \dots v^l_t)$

Courtesy of [9].

[9] B. Yu *et al.*, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting", IJCAI 2018.

Example 3: Temporal Graph Convolution

A more integrated approach instead consists of using temporal operators to compute messages.

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

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

Analogous models can be built by exploiting attention-based operators [10], [11].

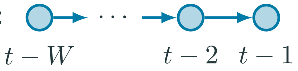
[10] I. Marisca *et al.*, “Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations”, NeurIPS 2022.
[11] Z. Wu *et al.*, “TraverseNet: Unifying Space and Time in Message Passing for Traffic Forecasting”, TNNLS 2022.

Example 4: Product graph representations

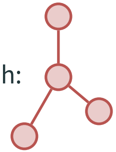
Finally, an orthogonal option to those seen so far is to consider $\mathcal{G}_{t-W:t}$ as a single spatiotemporal graph $\mathcal{S}_t$.

Such **product graph** can be obtained by combining temporal and spatial graphs.

Temporal graph:



Spatial graph:

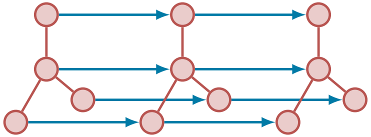


The resulting graph can be processed by any MP neural network.

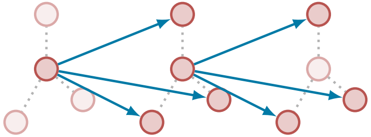
[12] M. Sabbaqi *et al.*, "Graph-time convolutional neural networks: Architecture and theoretical analysis" 2022.

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.



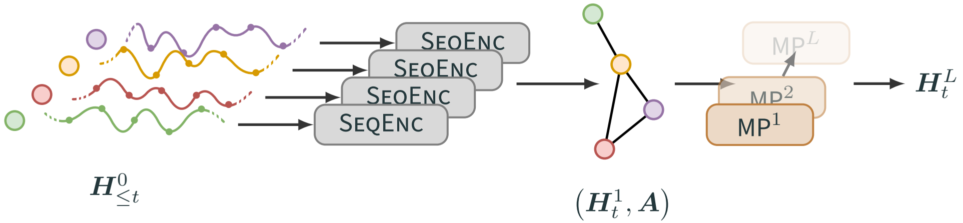
- ...

Time-then-Space models

The general recipe for a TTS model consists in:

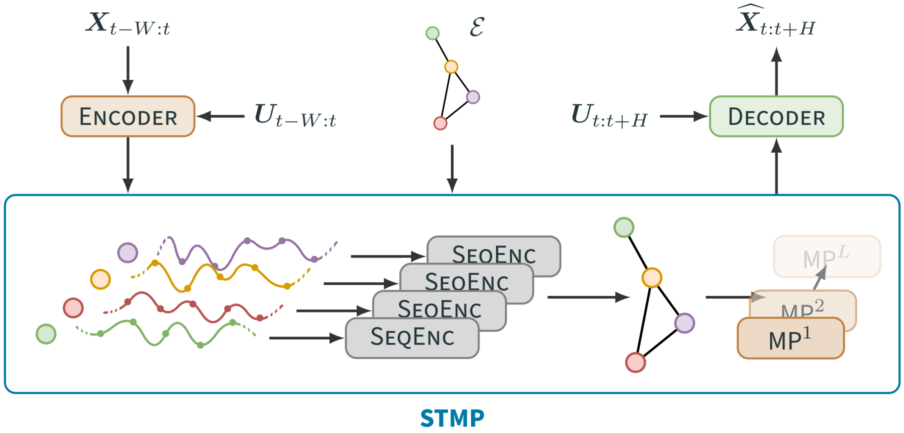
1. Embedding each node-level time series in a vector.
2. Propagating obtained encodings throughout the graph with a stack of MP layers.

1. $\boldsymbol{h}_t^{i,1} = \textsc{SeqEnc}\left(\boldsymbol{h}_{\leq t}^{i,0}\right)$
2. $\boldsymbol{H}_t^{l+1} = \text{MP}^l\left(\boldsymbol{H}_t^l, \mathcal{E}_t\right)$



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

Full TTS model



Pros & Cons of TTS models

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

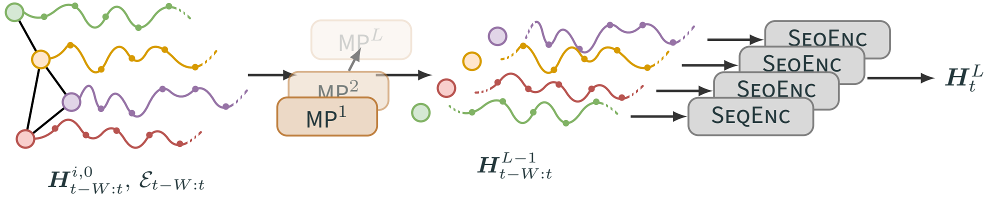
Cons:
- ☹ The 2-step encoding might introduce information bottlenecks.
- ☹ Accounting for changes in topology and dynamic edge attributes can be more problematic.

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.

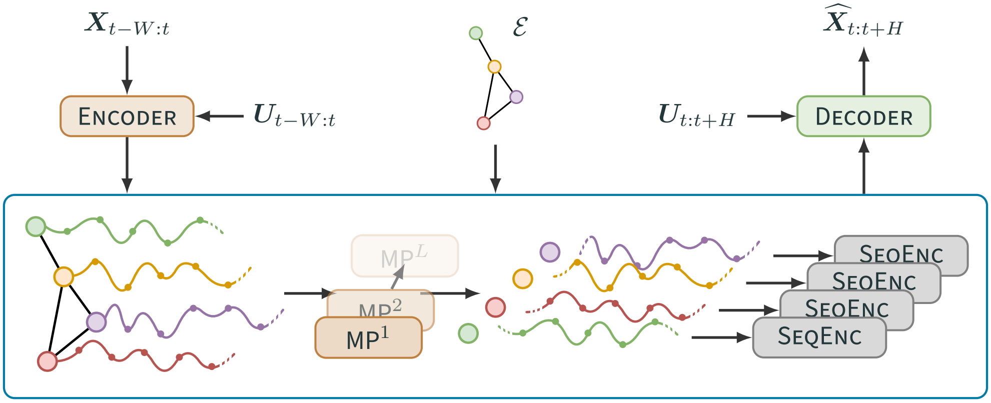
1. $\boldsymbol{H}_t^{i,l} = \text{MP}^l\left(\boldsymbol{H}_t^{i,l-1}, \mathcal{E}_t\right)$ 2. $\boldsymbol{h}_t^{i,L} = \text{SeqEnc}\left(\boldsymbol{h}_{t-W:t}^{i,L-1}\right)$



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.

Full STT model



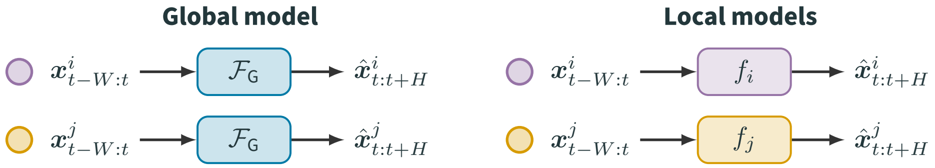
STMP

Global and local models

Global vs local

A forecasting model is called **global** if its parameters are fitted to a group of time series
$\rightarrow$ either univariate or multivariate.

Conversely, **local** models are specific to a single (possibly multivariate) time series.


Global model
$\boldsymbol{x}^i_{t-W:t}$
$\mathcal{F}_\mathsf{G}$
$\hat{\boldsymbol{x}}^i_{t:t+H}$
$\boldsymbol{x}^j_{t-W:t}$
$\mathcal{F}_\mathsf{G}$
$\hat{\boldsymbol{x}}^j_{t:t+H}$
Local models
$\boldsymbol{x}^i_{t-W:t}$
f_i
$\hat{\boldsymbol{x}}^i_{t:t+H}$
$\boldsymbol{x}^j_{t-W:t}$
f_j
$\hat{\boldsymbol{x}}^j_{t:t+H}$

A global model does not have any time-series-specific (local) parameters.

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

Trade-offs

Global models

- Just a single model needs to be trained and maintained.
- Larger amount of data available for training.
- Can be used in inductive learning scenarios (on unseen target time series).
- Theoretically, it can be as expressive as fitting a set of local models to each time series.

Local models

- Can more easily model time-series-specific dynamics.
- Often require shorter input windows.
- No problem in dealing with heterogeneous/asynchronous time series.

Globality and locality in STGNNs

STGNNs are typically global models: they do not rely upon node-specific parameters.

→ But they condition representations on each node's **neighborhood**, thus accounting for spatial dependencies.

Nonetheless, entirely global models might struggle to model local effects[1] and might require:

- ☹ impractically long observation windows;
- ☹ large model capacity.

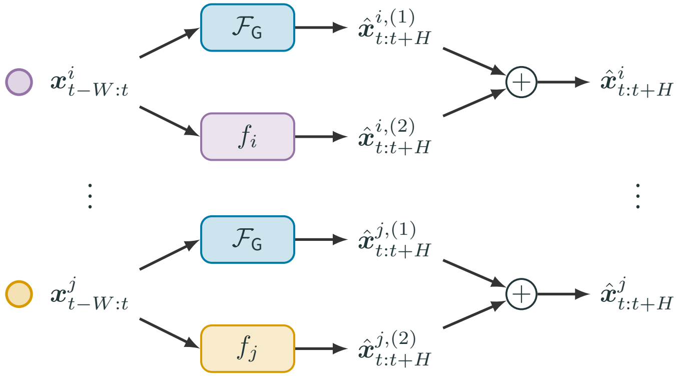
💡 We can use hybrid global-local STGNNs with specialized local components.

[1] Dynamics proper of each time series in the collection.

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

Global-local STGNNs (Example 1)

A simple approach consists of combining a global model and a (simpler) local one:

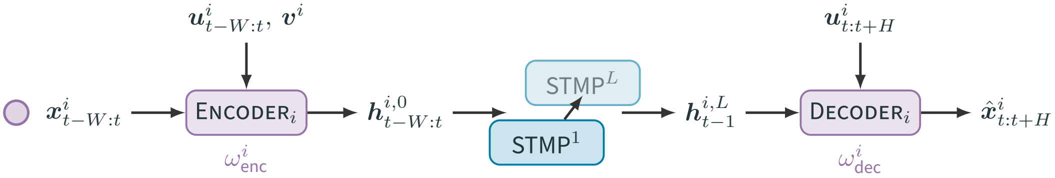


$\boldsymbol{x}^i_{t-W:t}$
$\mathcal{F}_\mathsf{G}$
f_i
$\hat{\boldsymbol{x}}^{i,(1)}_{t:t+H}$
$\hat{\boldsymbol{x}}^{i,(2)}_{t:t+H}$
$\hat{\boldsymbol{x}}^{i}_{t:t+H}$
$\boldsymbol{x}^j_{t-W:t}$
$\mathcal{F}_\mathsf{G}$
f_j
$\hat{\boldsymbol{x}}^{j,(1)}_{t:t+H}$
$\hat{\boldsymbol{x}}^{j,(2)}_{t:t+H}$
$\hat{\boldsymbol{x}}^{j}_{t:t+H}$

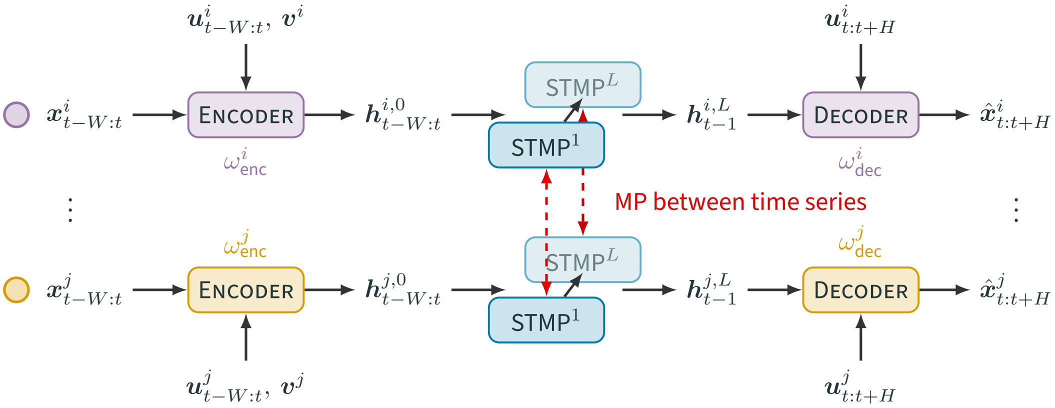
Global-local STGNNs (Example 2)

Another possibility is to use different weights for each time series at the encoding (ω^i_{enc}) and decoding (ω^i_{dec}) steps:

$$\boldsymbol{h}_t^{i,0} = \textsc{Encoder}_i\left(\boldsymbol{x}^i_{t-1}, \boldsymbol{u}^i_{t-1}, \boldsymbol{v}^i; \omega^i_{\text{enc}}\right) \qquad \hat{\boldsymbol{x}}^i_{t:t+H} = \textsc{Decoder}_i\left(\boldsymbol{h}_t^{i,L}, \boldsymbol{u}_{t:t+H}; \omega^i_{\text{dec}}\right)$$


$\boldsymbol{u}^i_{t-W:t}, \boldsymbol{v}^i$
$\boldsymbol{x}^i_{t-W:t}$
$\textsc{Encoder}_i$
ω^i_{enc}
$\boldsymbol{h}^{i,0}_{t-W:t}$
STMP^1
STMP^L
$\boldsymbol{h}^{i,L}_{t-1}$
$\boldsymbol{u}^i_{t:t+H}$
$\textsc{Decoder}_i$
ω^i_{dec}
$\hat{\boldsymbol{x}}^i_{t:t+H}$

Global-local STGNNs (Example 2)



$\boldsymbol{u}^i_{t-W:t}, \boldsymbol{v}^i$
$\boldsymbol{x}^i_{t-W:t}$
ENCODER
ω^i_{enc}
$\boldsymbol{h}^{i,0}_{t-W:t}$
STMP^L
STMP^1
$\boldsymbol{h}^{i,L}_{t-1}$
$\boldsymbol{u}^i_{t:t+H}$
DECODER
ω^i_{dec}
$\hat{\boldsymbol{x}}^i_{t:t+H}$
MP between time series
ω^j_{enc}
$\boldsymbol{x}^j_{t-W:t}$
ENCODER
$\boldsymbol{h}^{j,0}_{t-W:t}$
STMP^L
STMP^1
$\boldsymbol{h}^{j,L}_{t-1}$
ω^j_{dec}
DECODER
$\hat{\boldsymbol{x}}^j_{t:t+H}$
$\boldsymbol{u}^j_{t-W:t}, \boldsymbol{v}^j$
$\boldsymbol{u}^j_{t:t+H}$

Pros & Cons of global-local STGNNs

How to balance between the global and local modeling paradigms is problem-dependent.

Introducing local components specific to each time series in a global STGNN has several effects.

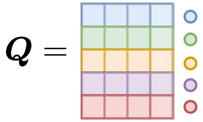
- ☺ Node-level effects are captured more efficiently than by fully global models.
- ☺ Forecasting accuracy on the task is usually higher empirically.
- ☹ The model's inductive capabilities are compromised (hard to handle unseen time series).
- ☹ The number of learnable parameters can be much larger compared to fully global models.

⚡ We can mitigate these two drawbacks by associating each node with a learnable **embedding**.

Learnable node embeddings

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

They can be fed into modules of a global STGNN and learned end-to-end.



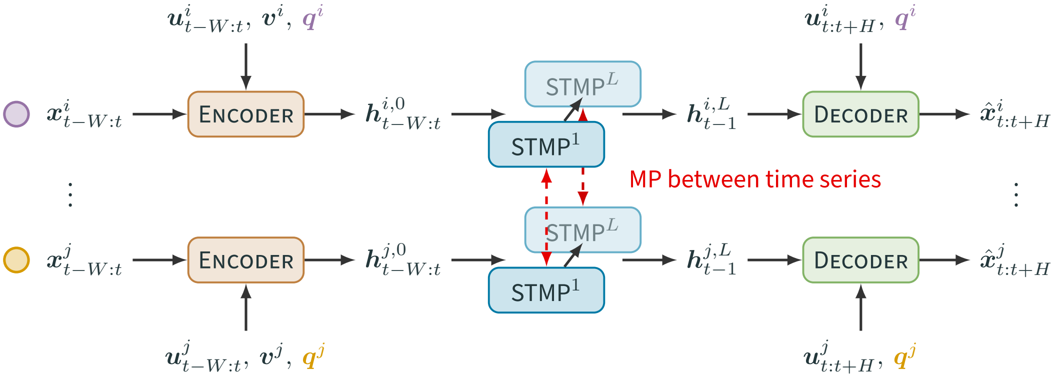
Example: node embeddings can be used to condition the encoding and decoding steps:

$$\boldsymbol{h}_t^{i,0} = \textsc{Encoder}\left(\boldsymbol{x}_{t-1}^i, \boldsymbol{u}_{t-1}^i, \boldsymbol{v}^i, \boldsymbol{q}^i\right) \qquad \hat{\boldsymbol{x}}_{t:t+H}^i = \textsc{Decoder}\left(\boldsymbol{h}_t^{i,L}, \boldsymbol{u}_{t:t+H}^i, \boldsymbol{q}^i\right)$$

Note: all the weights of the Encoder and Decoder modules can be shared among all the nodes.

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

Node embeddings in action


$\boldsymbol{u}^i_{t-W:t}, \boldsymbol{v}^i, \boldsymbol{q}^i$
$\boldsymbol{x}^i_{t-W:t}$
Encoder
$\boldsymbol{h}^{i,0}_{t-W:t}$
STMP^L
STMP^1
$\boldsymbol{h}^{i,L}_{t-1}$
$\boldsymbol{u}^i_{t:t+H}, \boldsymbol{q}^i$
Decoder
$\hat{\boldsymbol{x}}^i_{t:t+H}$
MP between time series
$\boldsymbol{x}^j_{t-W:t}$
Encoder
$\boldsymbol{h}^{j,0}_{t-W:t}$
STMP^L
STMP^1
$\boldsymbol{h}^{j,L}_{t-1}$
Decoder
$\hat{\boldsymbol{x}}^j_{t:t+H}$
$\boldsymbol{u}^j_{t-W:t}, \boldsymbol{v}^j, \boldsymbol{q}^j$
$\boldsymbol{u}^j_{t:t+H}, \boldsymbol{q}^j$

Advantages of node embeddings

Using node **embeddings** to make an STGNN global-local allows us to:

1. **Amortize** the cost of specializing the model to each time series;
   - A single d_q-dimensional vector for each node is added to the model's parameters;
   - The same vector can be used in multiple components of the architecture.
2. **Transfer** the learned model to a different set of time series $\mathcal{V}'$ more easily.
   - Only $|\mathcal{V}'|d_q$ parameters need to be tuned, while the shared components are fixed;
   - The embedding space can be regularized to better fit embeddings of new nodes [14].

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

Example: global-local TTS

As an example, one can build a global-local TTS model by simply exploiting node embeddings and global RNN and MP layers as

$$\begin{aligned}
\boldsymbol{h}_t^{i,0} &= \textsc{Encoder}\left(\boldsymbol{x}_{t-1}^i, \boldsymbol{u}_{t-1}^i, \boldsymbol{v}^i, \boldsymbol{q}^i\right), \\
\boldsymbol{h}_t^{i,1} &= \text{RNN}\left(\boldsymbol{h}_{\leq t}^{i,0}\right), \\
\boldsymbol{H}_t^{l+1} &= \text{MP}^l\left(\boldsymbol{H}_{<t-1}^l, \mathcal{E}_{\leq t-1}\right), \quad l = 1, \dots, L-1 \\
\hat{\boldsymbol{x}}_{t:t+H}^i &= \textsc{Decoder}\left(\boldsymbol{h}_{t-1}^{i,L}, \boldsymbol{u}_{t:t+H}^i, \boldsymbol{v}^i, \boldsymbol{q}^i\right).
\end{aligned}$$

Some empirical results

Models		**GPVAR-Global** (MAE)	**GPVAR-Local** (MAE)
Local	FC-RNN	$.4393_{\pm .0024}$	$.5978_{\pm .0149}$
	Local RNNs	$.4047_{\pm .0001}$	$.4610_{\pm .0003}$
Global	RNN	$.3999_{\pm .0000}$	$.5440_{\pm .0003}$
	RNN+MP	$\mathbf{.3193}_{\pm \mathbf{.0000}}$	$.3587_{\pm .0049}$
Global-local (w/ Emb.)	RNN	$.3991_{\pm .0001}$	$.4612_{\pm .0003}$
	RNN+MP	$\mathbf{.3194}_{\pm \mathbf{.0001}}$	$\mathbf{.3199}_{\pm \mathbf{.0001}}$
Optimal model		.3192	.3192

- **Global** models can fall short in certain scenarios.
- **Local** multivariate models can easily overfit.
- **Global-local** models can strike a good compromise.

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

Consider predictors $\mathcal{F}_a, \mathcal{F}_b$ from a set $\mathbb{F}$ of models 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 $\mathcal{F}_b \in \mathbb{F}$ better than $\mathcal{F}_a$.

Can we further improve over the best model so far $\mathcal{F}_a$?

$\rightarrow$ Either we find a new model $\mathcal{F}_*$ better than $\mathcal{F}_a$

$\rightarrow$ or we need prior knowledge about the modeled system.

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

Studying the correlation between prediction residuals $\boldsymbol{r}_t^i \doteq \boldsymbol{x}_{t:t+H}^i - \hat{\boldsymbol{x}}_{t:t+H}^i$ allows for testing model optimality.

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

Serial correlation

Correlation between residuals at different time steps.

Spatial correlation

Correlation between residuals at different graph nodes.

Most of the research focused on either serial correlation [15]–[17] or spatial correlation [18], [19].

Statistical tests for residual correlation

Whiteness test

H_0 : residuals are uncorrelated $\qquad$ H_1 : some residuals correlate

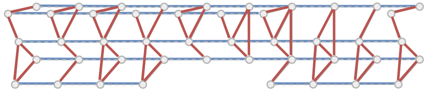
Define a test statistic $C(\{\boldsymbol{r}_t^i\}) = C(\mathcal{F}, \{\boldsymbol{x}_t^i\})$ and a threshold γ such that

If $|C(\{\boldsymbol{r}_t^i\})| > \gamma \implies$ reject H_0.

Remarks: Residual correlation analysis

- ☺ Is 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.

AZ-Whiteness test: a spatio-temporal test



The test is defined by statistic

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

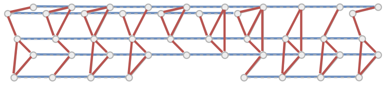
- distribution-free and residuals can be non-identically distributed.
- computation is linear in the number of edges and time steps.

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

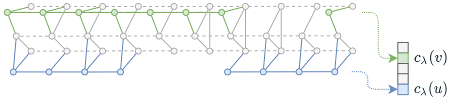
Where can we improve?

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

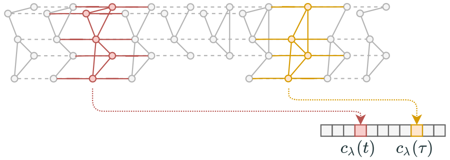
Spatial (or temporal) edges



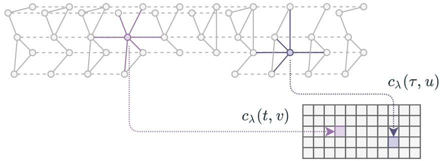
Edges related to a node



Edges related to a time step



Edges related to a node



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

Part 2

Challenges

Challenges

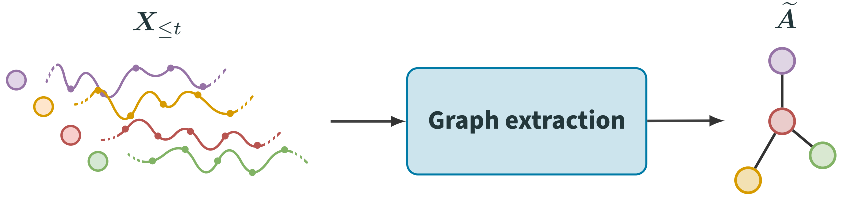
- **Latent graph learning**
  What to do when the underlying graph is not known?
- **Learning in non-stationary environments**
  What to do when the environment changes?
- **Scalability**
  How to deal with large collections of time series?
- **Dealing with missing data**
  How to deal with missing observations within the time series?

Latent graph learning

Learning and adjacency matrix

☹ Relational information is not always available
☹ or might be ineffective in capturing spatial dynamics.

☺ Relational architectural biases can nonetheless be exploited
$\rightarrow$ **extract a graph** from the time series or node attributes



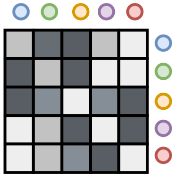
- It can be interpreted as regularizing a spatial attention operator.

[22] A. Cini *et al.*, "Sparse graph learning from spatiotemporal time series", JMLR 2023.

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
- . . .



$\rightarrow$ Thresholding might be necessary to obtain binary and sparse graphs.

Latent graph learning

An integrated approach: learn the **relations** end-to-end with the downstream task

- as a function of the input data,
- as trainable parameters of the model,
- or both.

This problem is known as latent graph learning (or latent graph inference).

Two different approaches:

1. learning directly an adjacency matrix $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{N \times N}$;
2. learning a probability distribution over graphs p_{Φ} generating $\widetilde{\boldsymbol{A}}$.

⚠ One key challenge is keeping both $\widetilde{\boldsymbol{A}}$ and the subsequent computations sparse.
$\rightarrow$ challenging with gradient-based optimization.

Direct approach

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

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

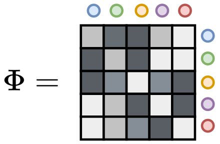
Edge scores Φ

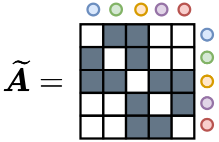
$\rightarrow$ can be a table of learnable model parameters,

$\rightarrow$ obtained as a function of the inputs and/or other parameters.

Function $\xi(\,\cdot\,)$ is a nonlinear activation

$\rightarrow$ it can be exploited to make $\widetilde{\boldsymbol{A}}$ sparse.





Direct approach: factorization methods

Many of the methods directly learning $\widetilde{\boldsymbol{A}}$, learn a factorization of the former to amortize the cost of the inference:

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

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 learned as tables of (local) parameters or as a function of the input window.

[23] Z. 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 $\widetilde{A}$ 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 $\widetilde{\boldsymbol{A}}$ by minimizing

$$\mathcal{L}(\Phi) = \mathbb{E}_{\widehat{\boldsymbol{A}} \sim p_\Phi}\left[\ell\left(\widehat{\boldsymbol{X}}_{t:t+H}, \boldsymbol{X}_{t:t+H}\right)\right]. \tag{15}$$

- Again, we can factorize Φ and make p_Φ input dependent, e.g.,

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

- Different parametrizations of p_Φ allow for embedding sparsity priors on the sampled graphs [22].

⚠ Gradient-based optimization requires $\nabla_\Phi \mathcal{L}(\Phi)$
$\rightarrow$ it can be challenging and computationally expensive.

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

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,
  - ☹ rely on continuous relaxations and make subsequent computations scale with $\mathcal{O}(N^2)$.

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

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

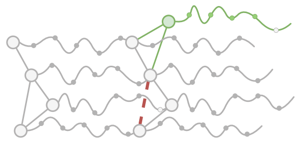
  - ☹ suffer from high variance (use variance reduction techniques),
  - ☺ allow to keep computations sparse.

$\rightarrow$ we can use Monte Carlo estimator.

[24] T. Kipf *et al.*, "Neural relational inference for interacting systems", ICML 2018.
[22] A. Cini *et al.*, "Sparse graph learning from spatiotemporal time series", JMLR 2023.

Learning in Non-Stationary Environments

Inductive learning



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 sensor networks (collections of time series)

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

⚠ Performance can easily degrade if the data distribution of target nodes

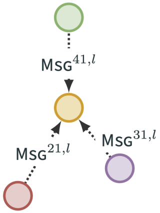
- deviates from that at training nodes
- changes over time.

[25] G. Ditzler *et al.*, "Learning in Nonstationary Environments: A Survey", IEEE CIM 2015.
[14] A. Cini *et al.*, "Taming Local Effects in Graph-based Spatiotemporal Forecasting", To appear in NeurIPS 2023.

Transferability of STGNNs

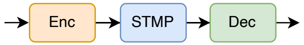
Global STGNNs are inductive and can directly be used in the above settings, provided that the training and target data are similar enough.

- MP operates on generic neighborhoods
- MP parameters are shared across nodes



Otherwise, STGNNs need to be adjusted

- fine-tuning (a subset of) the weights of the model on the new data
- exploiting transfer learning strategies



⚠ Global-local STGNNs reduce the cost of transfer learning

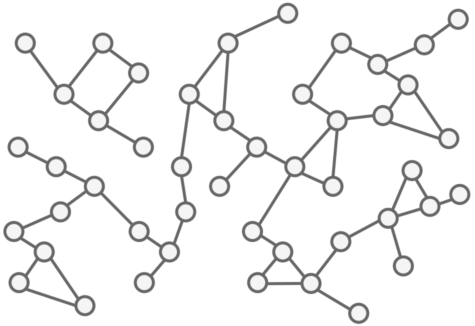
- sharing most of the parameters and finetuning node-specific parameters only
- node embeddings can be regularized to facilitate the learning further.

[26] G. Panagopoulos *et al.*, “Transfer graph neural networks for pandemic forecasting”, AAAI 2021.
[27] T. Mallick *et al.*, “Transfer learning with graph neural networks for short-term highway traffic forecasting”, ICPR 2021.
[14] A. Cini *et al.*, “Taming Local Effects in Graph-based Spatiotemporal Forecasting”, To appear in NeurIPS 2023.

Scalability

The scalability feature

- ☺ **Graph-based processing** allows us to learn a single model...
- ☺ ...able to deal with a **large collection** of time series...
- ☺ ...while accounting for the most relevant **relational information**.



The scalability issue

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

- the spatial dimension, corresponding to the number of time series (sensors).
- the time dimension, corresponding to the number of time steps (number of observations acquired per sensor).

In the real world, dealing with thousands of sensors acquiring data at high sampling rates is quite common (e.g., smart cities).

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

☹ In particular, to account for **long-range** spatiotemporal dependencies.

Computational complexity of STGNNs

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** – $\mathcal{O}(L|\mathcal{E}_t|)$.

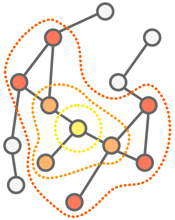
$$\rightarrow \mathcal{O}(WN + L|\mathcal{E}_t|)$$

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, e.g., by

- 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., [28], [29]).

☹ Subsampling might break long-range spatiotemporal dependencies.

☹ The learning signal may be noisy.

[28] W. Hamilton *et al.*, "Inductive representation learning on large graphs", NeurIPS 2017.
[29] Y. Rong *et al.*, "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification", ICLR 2020.

Pre-computation

Pre-processing methods (e.g., [30]) 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**.

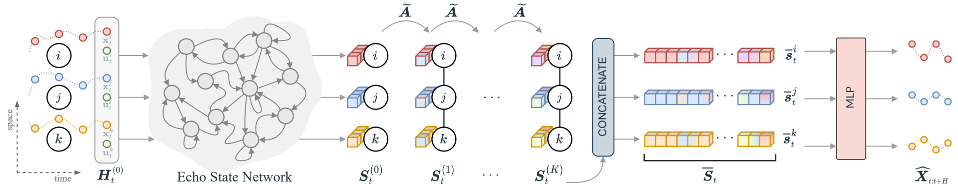
An extension to spatiotemporal data is given by SGP [31], which acts in 2 steps:

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

[30] F. Frasca *et al.*, "SIGN: Scalable inception graph neural networks" 2020.
[31] A. Cini *et al.*, "Scalable Spatiotemporal Graph Neural Networks", AAAI 2023.
[2] A randomized recurrent neural networks

SGP: Scalable Graph Predictor [31]

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, as the number of extracted features is much higher than d_x.
- ☹ More reliant on hyperparameter selection than end-to-end approaches.

[31] A. Cini *et al.*, "Scalable Spatiotemporal Graph Neural Networks", AAAI 2023.

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.

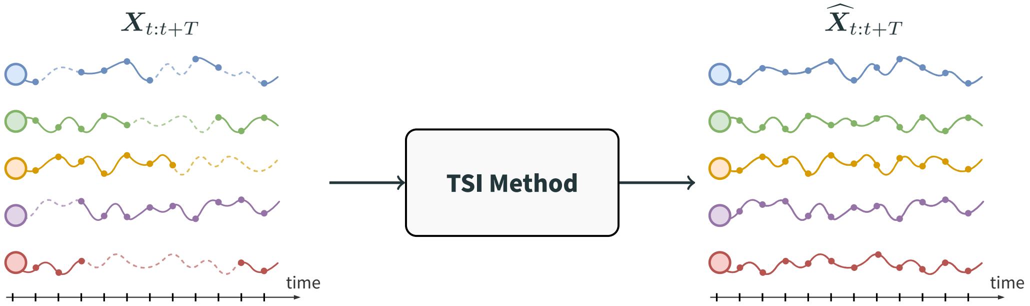
However, time series collected by real-world sensor networks often have missing data, due to:

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

Most forecasting methods operate on complete sequences.
$\rightarrow$ We need a way to impute, i.e., *reconstruct*, missing data.

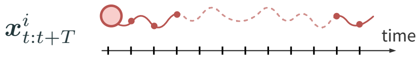
Time series imputation (i)

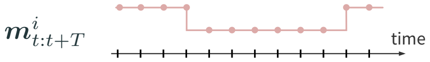
The problem of reconstructing missing values in a sequence of data is often referred to as time series imputation (TSI).



Time series imputation (ii)

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





Time series imputation

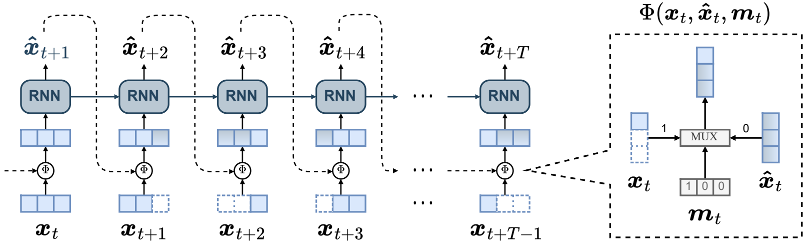
Given a window of $T \geq 1$ observations $\boldsymbol{X}_{<T}$ with missing values, the **time series imputation** problem consists in estimating the missing observations in the sequence

$$\boldsymbol{x}_t^i \sim p(\boldsymbol{x}_t^i \,|\, \mathcal{X}_{<T}) \qquad \forall\, i, t \text{ such that } \boldsymbol{m}_t^i = \boldsymbol{0}$$

with $\mathcal{X}_{<T} = \{\boldsymbol{x}_t^i \,|\, \boldsymbol{x}_t^i \in \boldsymbol{X}_{<T} \text{ and } \boldsymbol{m}_t^i = \boldsymbol{1}\}$ being the observed set.

Deep learning for TSI

Besides standard statistical methods, deep learning approaches have become a popular alternative, in particular, **autoregressive models** (e.g., RNNs).



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

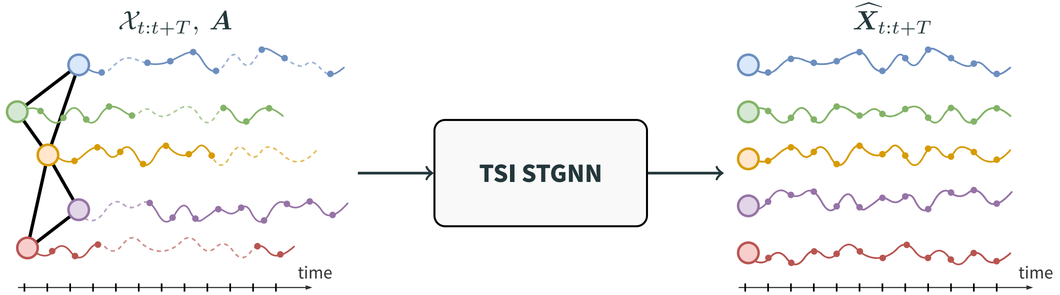
☹ ...but struggle in capturing **nonlinear space-time dependencies**.

Time series imputation + relational inductive biases

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

$$\boldsymbol{x}_t^i \sim p\left(\boldsymbol{x}_t^i \,|\, \mathcal{X}_{<T}, \boldsymbol{A}\right)$$

As done for the forecasting problem, we can use STGNNs to address the imputation task.



$\mathcal{X}_{t:t+T}, \ \boldsymbol{A}$
time
TSI STGNN
$\widehat{\boldsymbol{X}}_{t:t+T}$
time

Graph Recurrent Imputation Network

Cini et al. [32] propose a GCRNN that builds upon the autoregressive approach for imputation:

- A (graph-based) RNN (i.e., a GCRNN cell) is used to encode the sequence of only valid observations:

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

- An additional MP layer is used as spatial decoder, to account for **concurrent observations at neighbors**:

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

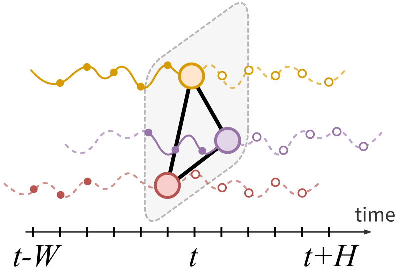
[32] A. Cini *et al.*, “Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks”, ICLR 2022.

Forecasting from Partial Observations

A more direct approach to the problem is to avoid the reconstruction step and consider forecasting architecture that can **directly deal with irregular observations**.

The mechanisms used in imputation models can be adapted to build forecasting architectures.

- ☺ Such models can be used to jointly impute missing observations and forecast future values.

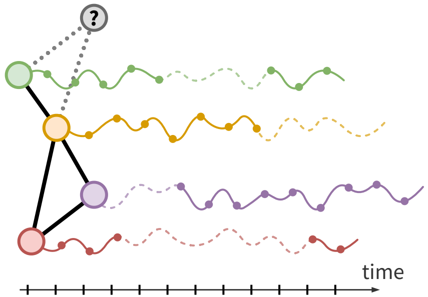


Beyond imputation

Graph-based imputation methods estimate missing values at an **existing node** by using available information at **neighboring nodes**.

Question:

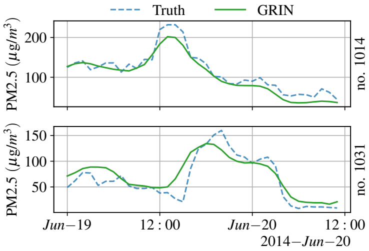
Can we use the same approach to **infer** observations of virtual sensors, i.e., fictitious nodes **not** associated with an existing sensor?



Virtual sensing

Simulate the presence of a sensor by adding a node with **no data**, then let the model infer the corresponding time series.

Two virtual sensors for air quality. (from [32])



Clearly, several assumptions are needed

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

[10] I. Marisca *et al.*, “Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations”, NeurIPS 2022.
[32] A. Cini *et al.*, “Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks”, ICLR 2022.
[33] Y. Wu *et al.*, “Inductive Graph Neural Networks for Spatiotemporal Kriging”, AAAI 2021.

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**.

Notebook

Spatiotemporal Graph Neural Networks with tsl

Open in Colab

Conclusions

Some Takeaways

DL for time series

+

DL on graphs

$\Longrightarrow$ Spatiotemporal Graph Neural Networks

- ☺ Relational inductive biases allow for exploiting dependencies among the time series
- ☺ while sharing most of the model parameters
- ☺ **Global-local STGNNs** are a safe choice in non-inductive settings

Challenges

- latent graph learning
- missing data imputation
- inductive learning
- scalability

Ivan **Marisca**

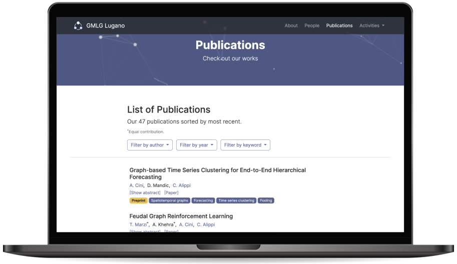
Andrea **Cini**

Daniele **Zambon**

Cesare **Alippi**

Graph Machine Learning Group

gmlg.ch



Publications

Check out our works

List of Publications

Our 47 publications sorted by most recent.

*Equal contribution.

Graph-based Time Series Clustering for End-to-End Hierarchical Forecasting

A. Cini, D. Mandic, C. Alippi

[Show abstract] [Paper]

Preprint Spatiotemporal graphs Forecasting Time series clustering Pooling

Feudal Graph Reinforcement Learning

T. Marzi*, A. Khehra*, A. Cini, C. Alippi

THE END

Questions?

References i

[1] 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, 2022, ISSN: 0360-0300. DOI: 10.1145/3533382. [Online]. Available: https://doi.org/10.1145/3533382.

[2] D. Bacciu, F. Errica, A. Micheli, and M. Podda, “A gentle introduction to deep learning for graphs,” *Neural Networks*, vol. 129, pp. 203–221, 2020.

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

[4] A. Cini, I. Marisca, D. Zambon, and C. Alippi, “Graph deep learning for time series forecasting: A comprehensive methodological framework,” *arxiv*, 2023.

[5] 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.

[6] 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.

References ii

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] M. Sabbaqi and E. Isufi, “Graph-time convolutional neural networks: Architecture and theoretical analysis,” *arXiv preprint arXiv:2206.15174*, 2022.

References iii

[13] 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.

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

[15] JRM 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.

[16] 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.

[17] 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.

[18] 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.

References iv

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

References v

[25] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, “Learning in nonstationary environments: A survey,” *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[26] G. Panagopoulos, G. Nikolentzos, and M. Vazirgiannis, “Transfer graph neural networks for pandemic forecasting,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 4838–4845.

[27] T. Mallick, P. Balaprakash, E. Rask, and J. Macfarlane, “Transfer learning with graph neural networks for short-term highway traffic forecasting,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, 2021, pp. 10 367–10 374.

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

[29] 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.

[30] 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.

References vi

[31] 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.

[32] 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.

[33] 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.