# Spatiotemporal Graph Deep Learning

Traffic monitoring

Smart cities

Energy analitics

Physics

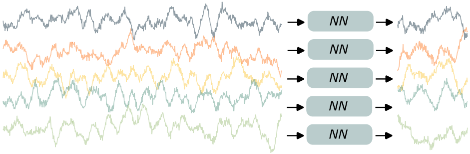
Stock markets

# Outline

1) Spatiotemporal time series
2) Spatiotemporal GNNs
3) Dealing with missing data

</> demo

# Deep learning for time series forecasting



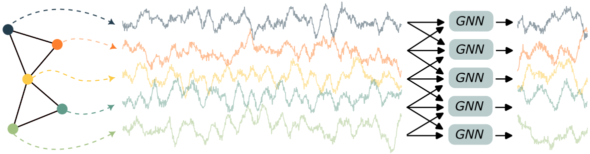

The standard deep learning approach to time series forecasting consists in training a single neural network on a collection of time series.

- 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.

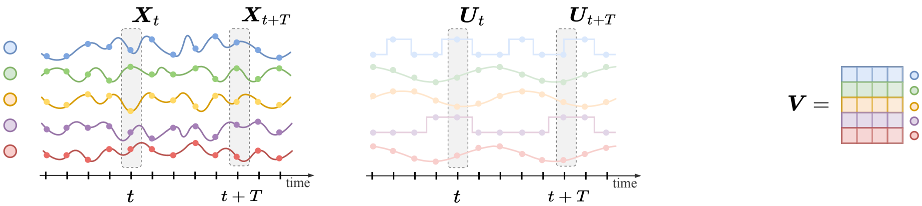
[2] D. Bacciu *et al.*, “A gentle introduction to deep learning for graphs”, NN 2020.
[3] M. M. Bronstein *et al.*, “Geometric deep learning: Grids, groups, graphs, geodesics, and gauges” 2021.

# 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., $\boldsymbol{X}_t \in \mathbb{R}^{N \times d_x}$, $\boldsymbol{U}_t \in \mathbb{R}^{N \times d_u}$.

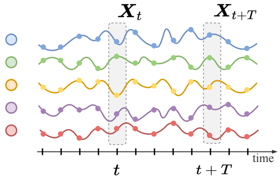
[4] A. Cini *et al.*, “Graph Deep Learning for Time Series Forecasting: A Comprehensive Methodological Framework” 2023.

# 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 $t \in \mathbb{N}$.




Notation:

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

Note that the time series:

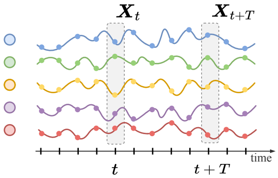
- can be generated by different processes,
- can depend on others,
- are assumed homogenous, synchronous, regularly sampled.

$\rightarrow$ These assumptions can be relaxed

# Relational information

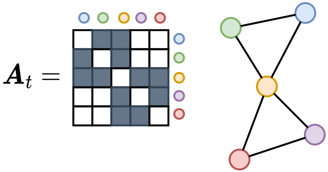
We assume the existence of functional dependencies between the time series.

$\rightarrow$ 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 $t$).



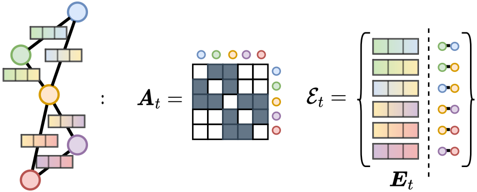

$\rightarrow$ We call spatial the dimension spanning the time series collection.

# 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 $\boldsymbol{A}_t$.

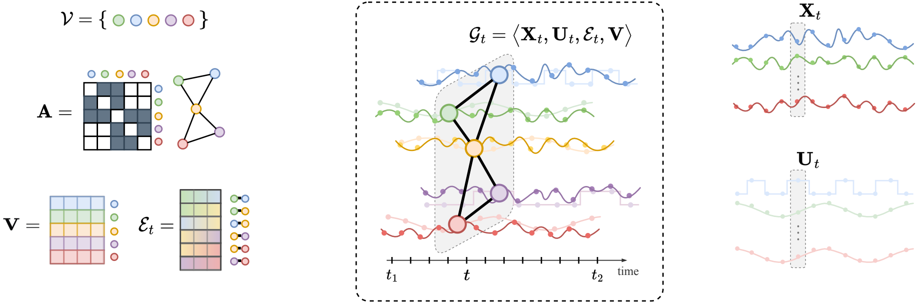
The set of attributed edges encoding all the available relational information is denoted by

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




$\rightarrow$ 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




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

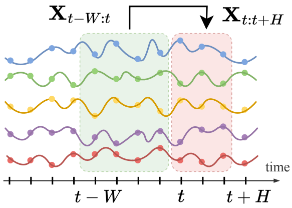
**Multi-step time-series forecasting**

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 $p$

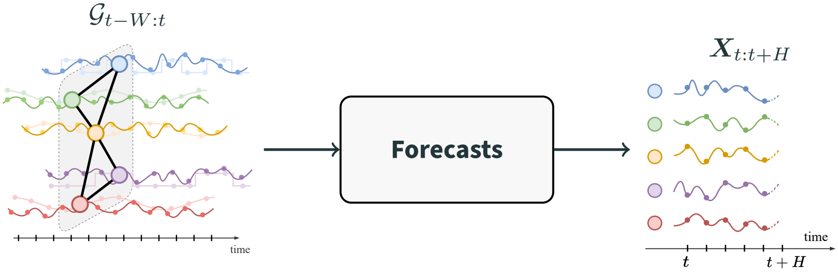
$$p_{\boldsymbol{\theta}}\left(\boldsymbol{x}^i_{t+h} \middle| \boldsymbol{X}_{t-W:t}, \boldsymbol{U}_{t-W:t+h}, \boldsymbol{V}\right) \approx p^i\left(\boldsymbol{x}^i_{t+h} \middle| \boldsymbol{X}_{<t}, \boldsymbol{U}_{\leq t+h}, \boldsymbol{V}\right).$$

- $\boldsymbol{\theta}$ 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 $\widehat{\boldsymbol{x}}^i_{t+h}$ estimates $\mathbb{E}_p\left[\boldsymbol{x}^i_{t+h}\right]$

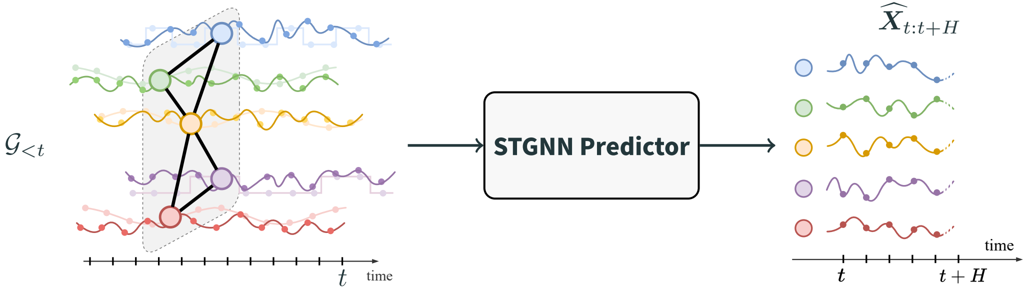
Parameters $\boldsymbol{\theta}$ can be learned by minimizing a cost function $\ell(\,\cdot\,,\,\cdot\,)$ (e.g., MSE) on a training set

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

# 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

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

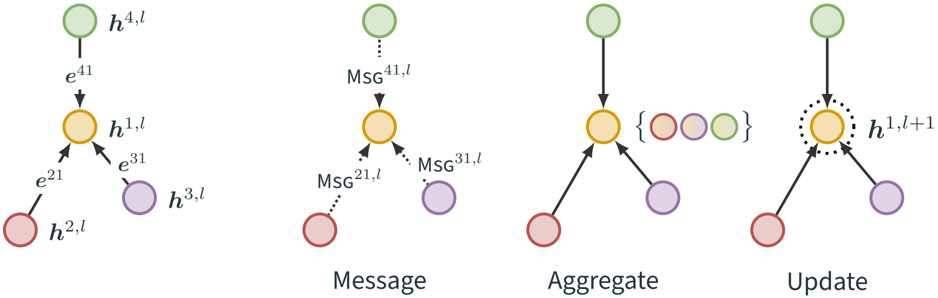
Where:

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

STGNNs can be expressed as a sequence of three operations:

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

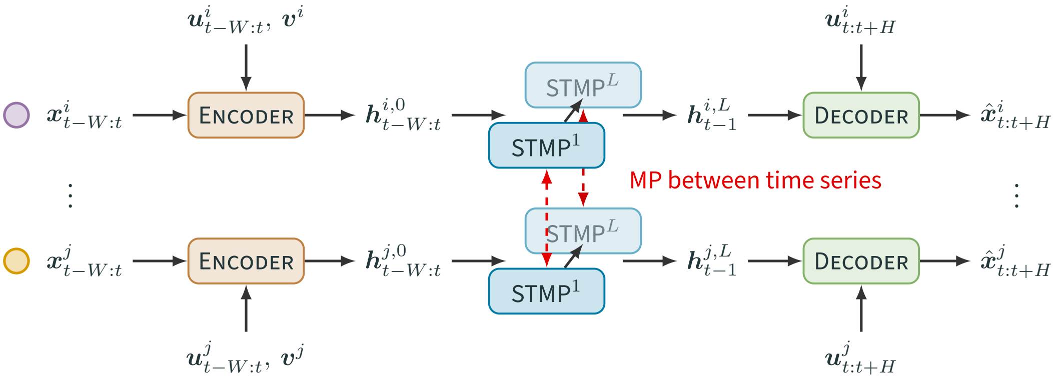
$$\boldsymbol{H}^{l+1}_{t-1} = \textsc{STMP}^{l}\left(\boldsymbol{H}^{l}_{\leq t-1}, \mathcal{E}_{\leq t-1}\right), \quad l = 0, \dots, L-1 \tag{3}$$

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

Where:

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

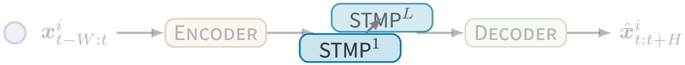
# Framework overview



# 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 $\mathsf{TCN}^{l}$ indicates a temporal convolutional network layer.

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

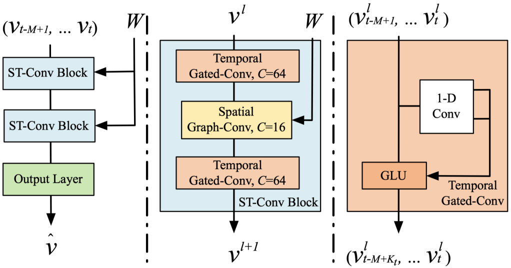
$$\boldsymbol{H}^{l}_{t} = \mathsf{MP}^{l}\left(\boldsymbol{Z}^{l}_{t}, \mathcal{E}_t\right) \qquad \forall\, t$$

# 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.




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.

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

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

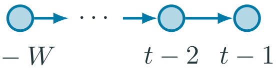
[10] I. Marisca *et al.*, “Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations”, NeurIPS 2022.
[11] Z. Wu *et al.*, “TraverseNet: Unifying Space and Time in Message Passing for Traffic Forecasting”, TNNLS 2022.

# Example 4: Product graph representations

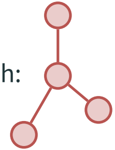
Finally, an orthogonal option to those seen so far is to consider $\mathcal{G}_{t-W:t}$ as a single spatiotemporal graph $\mathcal{S}_t$.

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

Temporal graph:




Spatial graph:

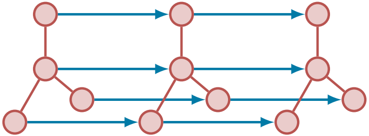


The resulting graph can be processed by any MP neural network.

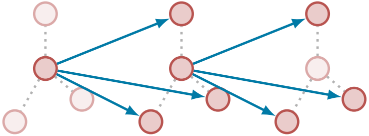
[12] M. Sabbaqi *et al.*, “Graph-time convolutional neural networks: Architecture and theoretical analysis” 2022.

# 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.



- ...

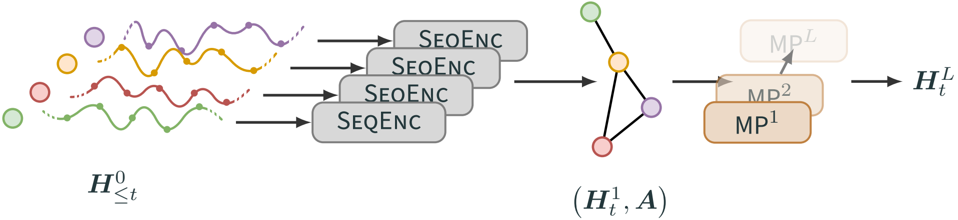
# Time-then-Space models

The general recipe for a TTS model consists in:

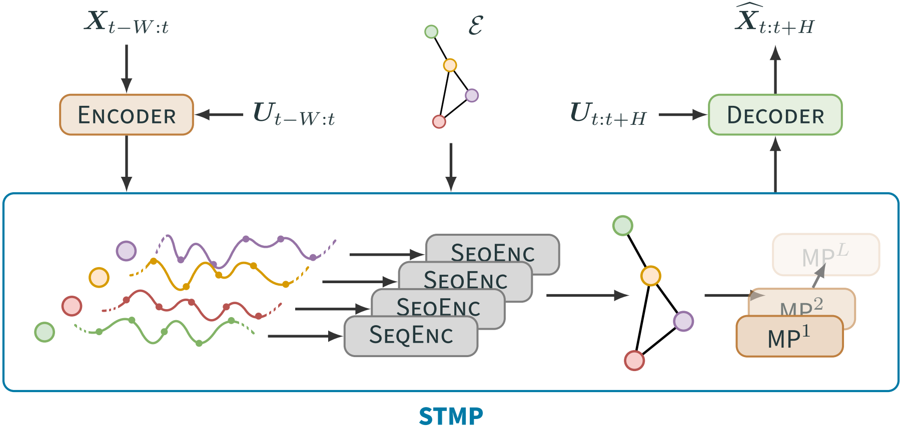
1. Embedding each node-level time series in a vector.
2. Propagating obtained encodings throughout the graph with a stack of MP layers.

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

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



# 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.

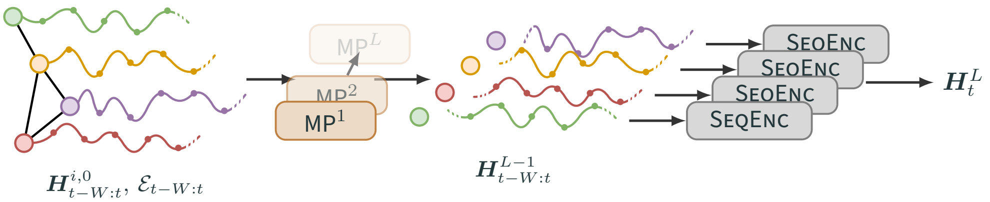
☹ Accounting for changes in topology and dynamic edge attributes can be more problematic.

# Space-then-Time

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

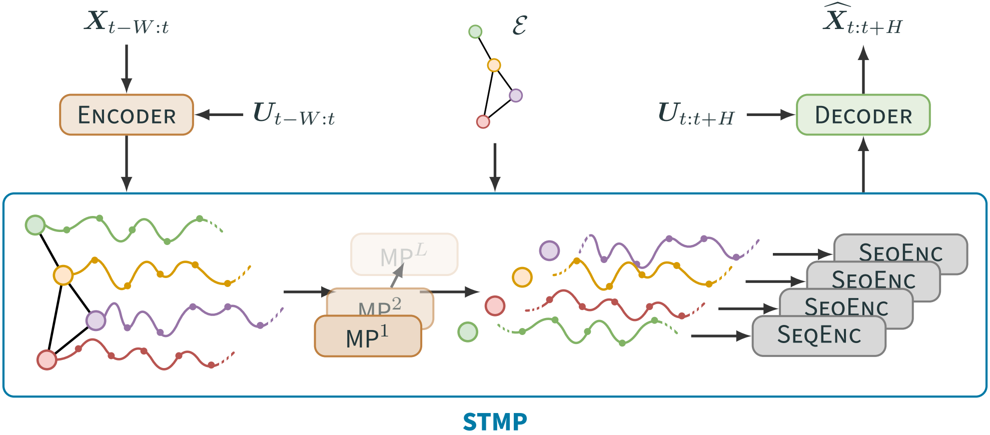
1. Observations are propagated among nodes w.r.t. each time step using a stack of MP layers.
2. Each sequence of representations is processed by a sequence encoder.

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




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

# Full STT model



# 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 to be sub-optimal?
3. How can we improve the predictor?

**Remark:** Multiple optimality criteria can be considered.

⚡ Relational inductive biases can help us

# 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 $\mathcal{F}_a$.

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

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.

# Residual correlation analysis

Studying the correlation between prediction residuals $r_t^i = x_{t:t+H}^i - \hat{x}_{t:t+H}^i$ allows us for testing model optimality.

If residuals are dependent

$\Longrightarrow$ there is information that the model hasn't captured

$\Longrightarrow$ model predictions can be improved.

**Temporal 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

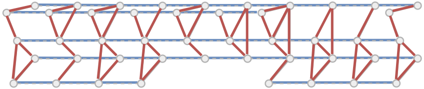
Define a test statistic $C(\{\boldsymbol{r}_t^i\}) = C(\mathcal{F}, \{\boldsymbol{x}_t^i\})$ and a threshold $\gamma$ such that

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

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

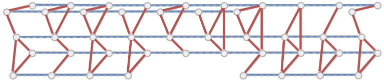
- distribution-free and residuals can be non-identically distributed.
- computation is linear in the number of edges and time steps.

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

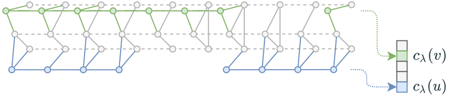
# Where can we improve?

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

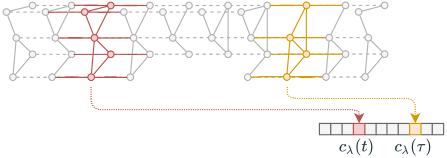
### Spatial (or temporal) edges



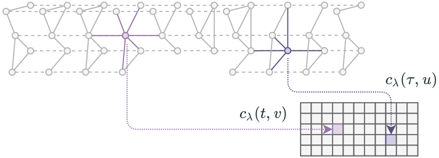
### Edges related to a node




### 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.

# 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.

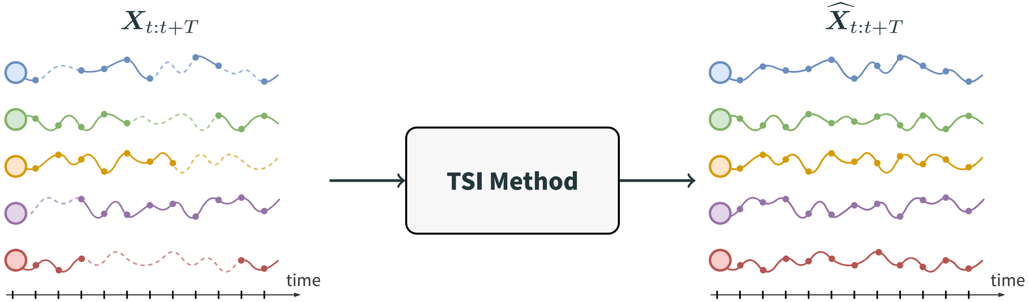
However, time series collected by real-world sensor networks often have missing data, due to:

- 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)

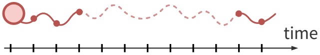
The problem of reconstructing missing values in a sequence of data is often referred to as time series imputation (TSI).



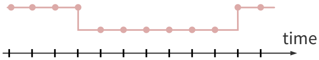
# Time series imputation (ii)

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

$\boldsymbol{x}^i_{t:t+T}$




$\boldsymbol{m}^i_{t:t+T}$




**Time series imputation**

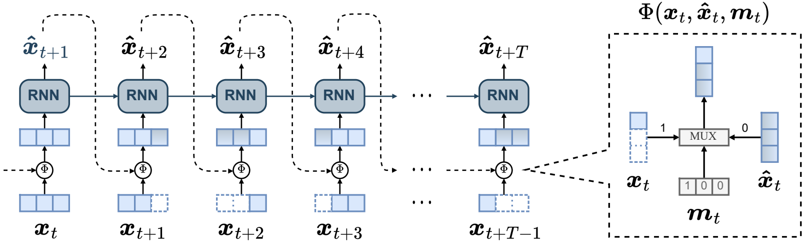
Given a window of $T \geq 1$ observations $\boldsymbol{X}_{<T}$ with missing values, the **time series imputation** problem consists in estimating the missing observations in the sequence

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

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

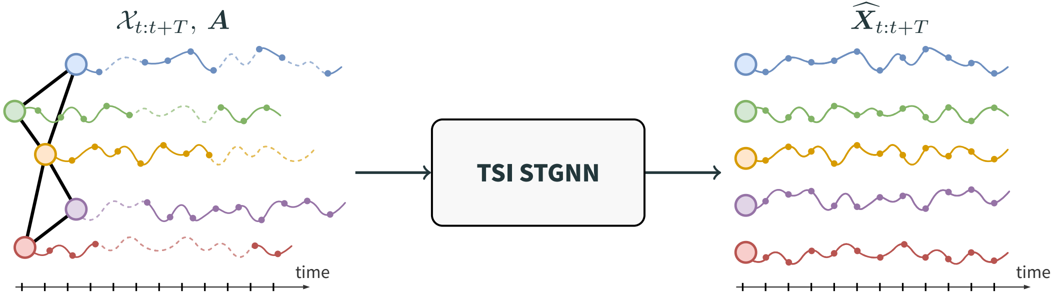
☹ ...but struggle in capturing **nonlinear space-time dependencies**.

# Time series imputation + relational inductive biases

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

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

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



# 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**:

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

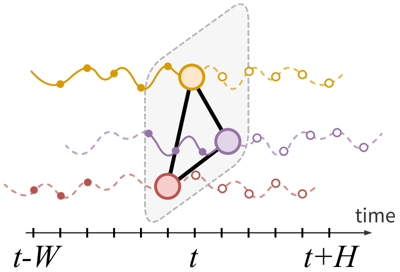
[32] A. Cini *et al.*, “Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks”, ICLR 2022.

# Forecasting from Partial Observations

A different approach to the problem avoids the reconstruction step and considers forecasting architectures that **directly deal with irregular observations**.

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

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

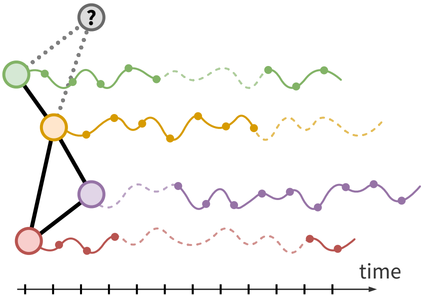


# Beyond imputation

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?



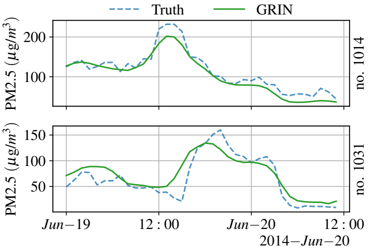
# Virtual sensing

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

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 [32])




[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

# 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.