Learning on Graphs Conference Italy meetup · December 6, 2024

Graph Deep Learning for Time Series Processing

Forecasting, Reconstruction and Analysis

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

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

Introduction

Traffic monitoring

Smart cities

Energy analitics

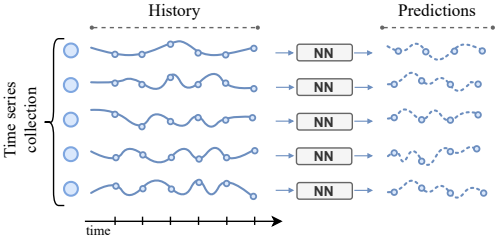
Physics

Stock markets

Deep learning for time series forecasting

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

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

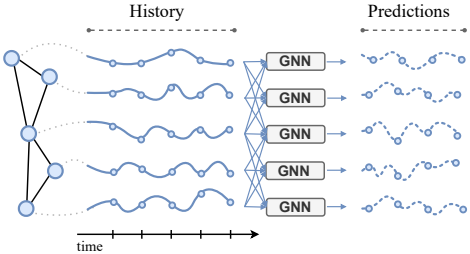


[1] Salinas *et al.*, “DeepAR: Probabilistic forecasting with autoregressive recurrent networks”, IJF 2020.
[2] Benidis *et al.*, “Deep Learning for Time Series Forecasting: Tutorial and Literature Survey”, ACM CS 2022.

Graph deep learning for time series forecasting

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

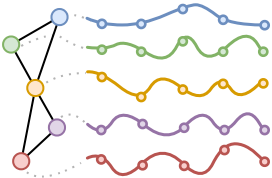
- ☺ **Dependencies** are **embedded into the processing** as inductive biases.
- ☺ Operate on **sets** of **correlated time series**.
- ☺ Parameters are **shared**.



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

What this tutorial is about

This tutorial presents advances coming from the combination of



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

The objective of this short tutorial is to provide:

1. a framework for graph-based time series processing models;
2. a discussion of selected challenges and future directions.

There is a longer version of this tutorial[1], complemented by a software demo and a paper [3].

[3] Cini, Marisca, Zambon, and Alippi, "Graph Deep Learning for Time Series Forecasting", Preprint 2023.

[1] Available at `gmlg.ch`

Part 1

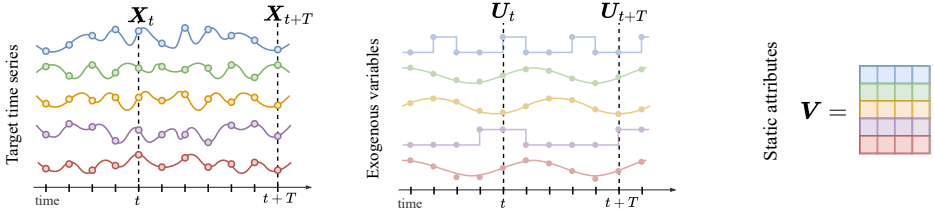
Graph-based Processing of Correlated Time series

Correlated time series

Collections of time series

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

- **observations** $\boldsymbol{x}_t^i \in \mathbb{R}^{d_x}$ at each time step t;
- **exogenous variables** $\boldsymbol{u}_t^i \in \mathbb{R}^{d_u}$ 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 N time series, i.e., $\boldsymbol{X}_t \in \mathbb{R}^{N \times d_x}$, $\boldsymbol{U}_t \in \mathbb{R}^{N \times d_u}$.

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

[3] Cini *et al.*, "Graph Deep Learning for Time Series Forecasting", Preprint 2023.

Correlated time series

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

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

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

Furthermore time series

- are assumed

  a) homogenous, b) synchronous, c) regularly sampled.

- can be generated by different processes.

Notation:

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

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

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

Example: Traffic monitoring system

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

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

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

Forecasting

Time series forecasting

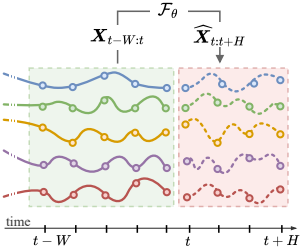
Multi-step time-series forecasting

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

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

predict $H \geq 1$ future observations

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



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

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

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

Global and local predictors

Local models

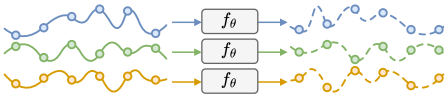
$$\hat{\boldsymbol{x}}^i_{t+h} = f\left(\boldsymbol{x}^i_{t-W:t}, \ldots; \boldsymbol{\theta}^i\right)$$

Example: Box-Jenkins method

☺ Tailored to each time series.

☹ Inefficient.

Global models



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

Example: DeepAR [1]

☺ Sample efficient.

☺ Allows for more complex models.

☹ Both approaches neglect dependencies among time series.

[1] Salinas *et al.*, “DeepAR: Probabilistic forecasting with autoregressive recurrent networks”, IJF 2020.
[4] Montero-Manso *et al.*, “Principles and algorithms for forecasting groups of time series: Locality and globality”, IJF 2021.

Accounting for spatial dependencies

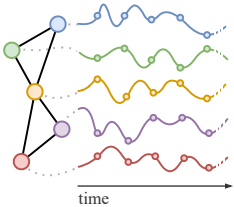
- One option is to consider the input as single multivariate time series
  - → Resulting predictors are **local**: $\widehat{\boldsymbol{X}}_{t+h} = f\left(\boldsymbol{X}_{t-W:t}, \ldots; \boldsymbol{\theta}\right).$
    - ☹ High sample complexity and poor scalability.
- Models operating on sets of time series would allow to keep parameters shared.
  - → Resulting predictors are **global**: $\widehat{\boldsymbol{X}}^{\mathcal{S}}_{t+h} = \mathcal{F}\left(\boldsymbol{X}^{\mathcal{S}}_{t-W:t}, \ldots; \boldsymbol{\theta}\right), \qquad \forall \mathcal{S} \subseteq \mathcal{D}$
    - ☺ Can be implemented by attention-based models (e.g, Transformers).
    - ☹ Does not exploit structural priors, high computational and sample complexity.
- Other methods (e.g., [5]) rely on dimensionality reduction to extract shared latent factors.
    - ☺ Might work well if data are low-rank.
    - ☹ Local and relational information are lost and can still suffer from, scalability issues.

[2] Benidis *et al.*, "Deep Learning for Time Series Forecasting: Tutorial and Literature Survey", ACM CS 2022.
[5] Sen *et al.*, "Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting", NeurIPS 2019.

Graph-based representation

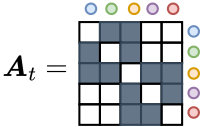
Relational information

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



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

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

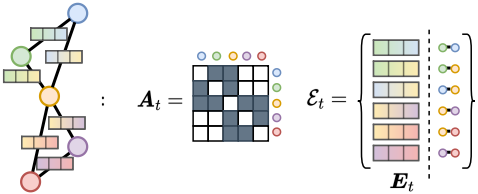


Relational information with attributes

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

The set of attributed edges is denoted by

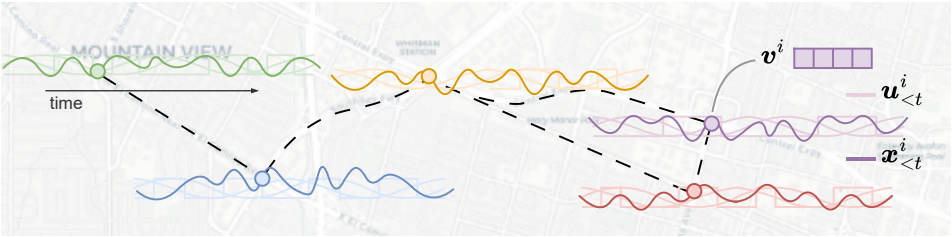
$$\mathcal{E}_t \doteq \{\langle (i,j), \boldsymbol{e}_t^{ij} \rangle \mid \forall i,j : \boldsymbol{A}_t[i,j] \neq 0\}.$$



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

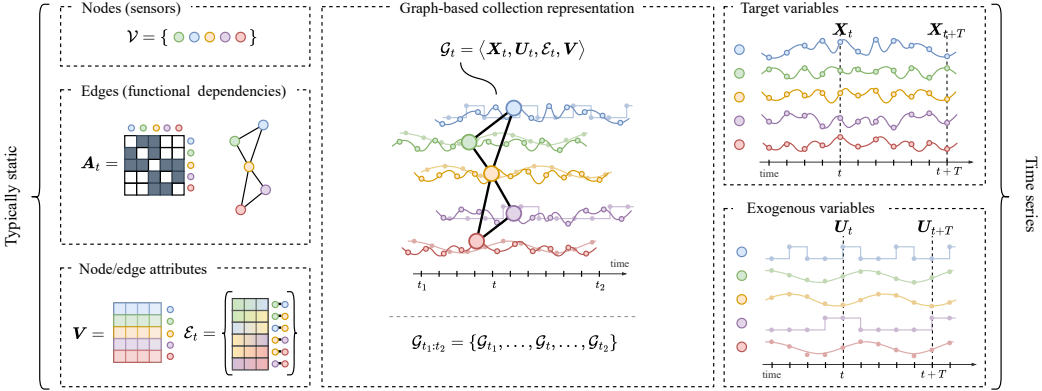
Example: Traffic monitoring system

Consider again the sensor network of the previous example.



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

Graph-based representations for correlated time series



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

[3] Cini *et al.*, “Graph Deep Learning for Time Series Forecasting”, Preprint 2023.

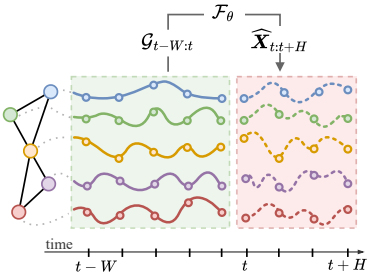
Relational inductive biases for time series forecasting

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

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

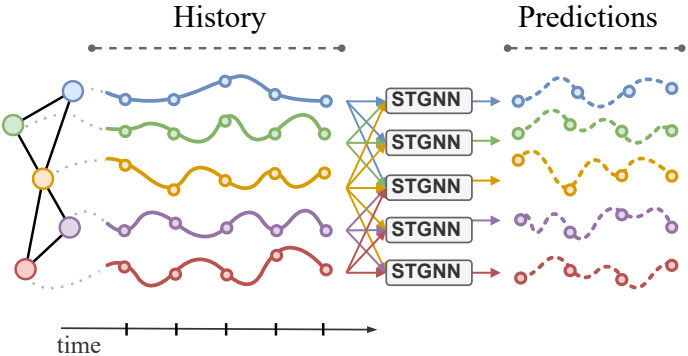
The conditioning can act as a regularization to localize predictions w.r.t. each node.

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



Spatiotemporal graph neural networks

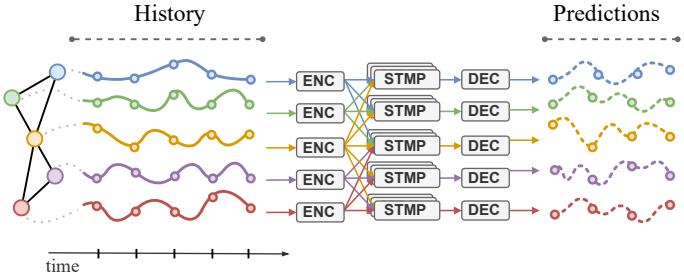
We call spatiotemporal graph neural networks (STGNNs) a neural network exploiting both temporal and spatial relations of the input spatiotemporal time series.



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

A general recipe for building STGNNs

We consider STGNNs consisting of three main components



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

[3] Cini *et al.*, "Graph Deep Learning for Time Series Forecasting", Preprint 2023.

Spatiotemporal message-passing (STMP)

STMP blocks can be defined as:

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

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

As in standard MP operators:

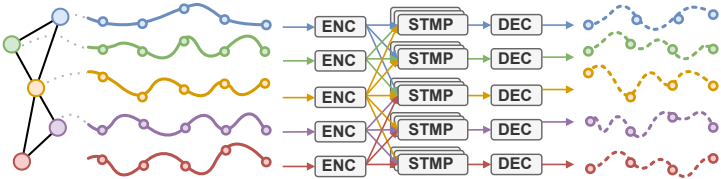
- $\textsc{Msg}^l(\,\cdot\,)$ is a **message function**, e.g., implemented by *temporal convolutional layers*.
- $\textsc{Aggr}\{\,\cdot\,\}$ is a permutation invariant **aggregation function**.
- $\textsc{Up}^l(\,\cdot\,)$ is an **update function**, e.g., implemented by an RNN.

❗ Blocks can be implemented by composing MP and sequence modeling operators.
→ Many possible designs exist.

[3] Cini *et al.*, "Graph Deep Learning for Time Series Forecasting", Preprint 2023.
[6] Gilmer *et al.*, "Neural message passing for quantum chemistry", ICML 2017.

Globality and locality in STGNNs

Globality and locality in STGNNs

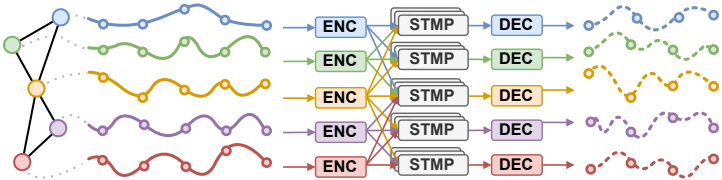


Standard STGNNs are **global** models.

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

💡 Use hybrid global-local STGNNs.

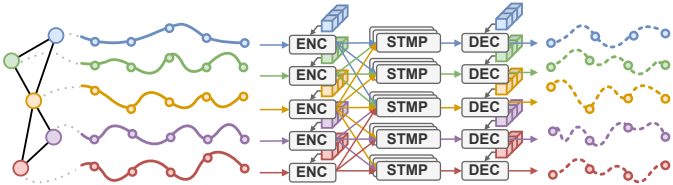
Global-local STGNNs



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

- ☺ Resulting models can capture local effects.
- ☹ Might require a large number of local parameters.

Global-local STGNNs with node embeddings



Node embeddings can amortize the learning of local components.

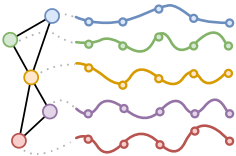
Node embeddings are a table of **learnable parameters** $Q \in \mathbb{R}^{N \times d_q}$ associated with **each node**.

- ☺ Most of the model's parameters remain shared.
- ☺ Can facilitate transfer learning.
- 😐 Number of parameters scales linearly with the number of time series . . .
  - → One might consider intermediate solutions, e.g., learning embeddings for clusters of time series.

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

What we have seen so far

1. Introduced the problem of processing correlated time series.
2. Graph representations allows for modeling dependencies.
3. Discussed the forecasting problem and associated predictors.
4. Saw recipes for building (global/local) STGNNs.



In the following, we will look into

- dealing with partial observations;
- latent graph learning;
- a selection of future directions.

Checkout the full tutorial for more on: computational scalability, model quality assessment, software libraries, . . .

Part 2

Challenges

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...

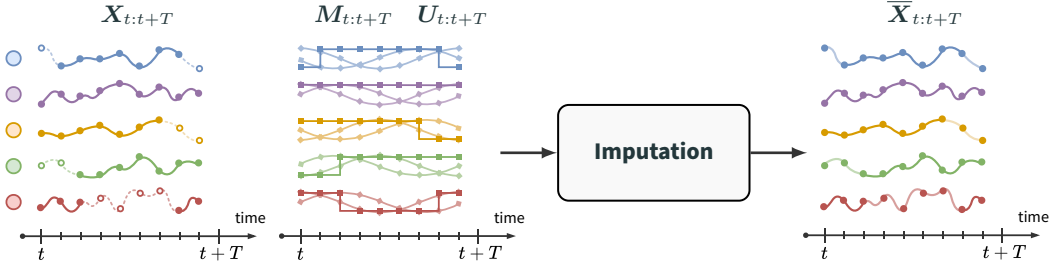
Most forecasting methods operate on complete sequences.

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

Time series imputation

Time series imputation (TSI)

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



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

Missing data types

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

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

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

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

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

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

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

Optimization

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

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

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

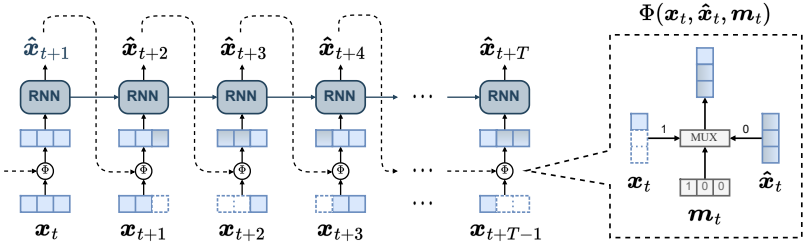
$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} \sum_{i=1}^{N} \frac{\left\| \overline{\boldsymbol{m}}_t^i \odot \ell\left(\overline{\boldsymbol{x}}_t^i, \boldsymbol{x}_t^i\right) \right\|_1}{\left\|\overline{\boldsymbol{m}}_t^i\right\|_1}.$$

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

Deep learning for TSI

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

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



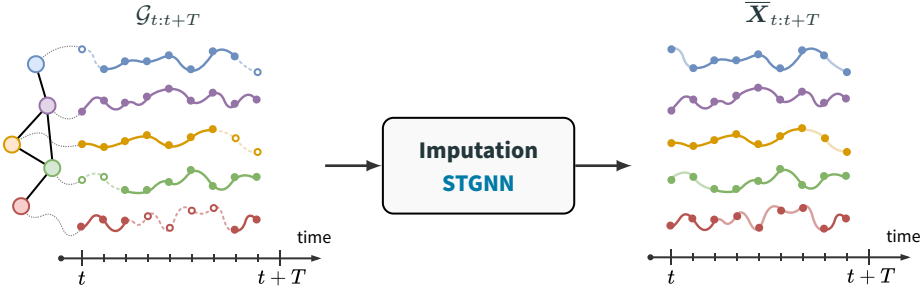
☺ Effective in exploiting past (and future, with bidirectional models) node observations.

☹ Struggle in capturing nonlinear space-time dependencies.

Time series imputation + relational inductive biases

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

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



Graph Recurrent Imputation Network (GRIN)

Similarly to GCRNN for forecasting, we can integrate graph processing into the autoregressive approach for imputation [8].

In these approaches, the distribution $p\left(\boldsymbol{x}_t^i \mid \boldsymbol{X}_{0:\infty} \odot \boldsymbol{M}_{0:\infty}\right)$ is modeled into three independent steps:

Information from previous observations.

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

Information from subsequent observations.

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

Information from related concurrent observations.

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

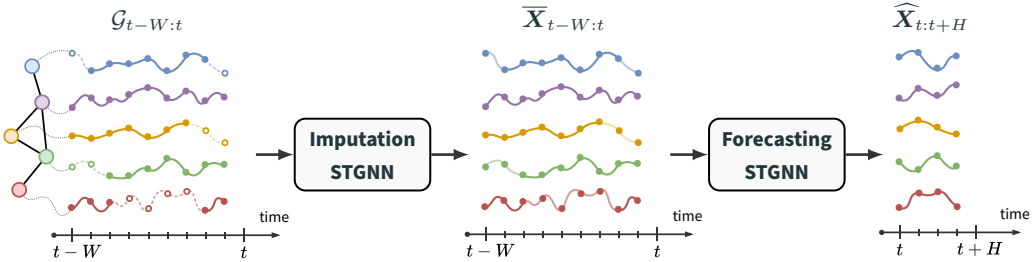
Typically modeled by bidirectional autoregressive models.

Enabled by message passing.

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

Imputation *before* forecasting

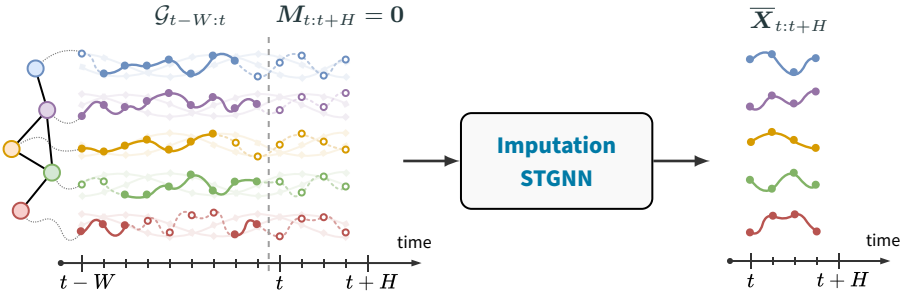
TSI is often used as a preprocessing step for a downstream task, e.g., forecasting.



- ☹ Often necessary to use standard forecasting methods with irregular time series.
- ☹ Might introduce biases due to errors in estimated values.

Imputation *in place of* forecasting

Imputation methods can also be adapted to perform forecasting.



- ☹ It is a workaround (this is not their purpose).
- ☹ Might perform poorly due to the absence of values in the forecasting horizon.

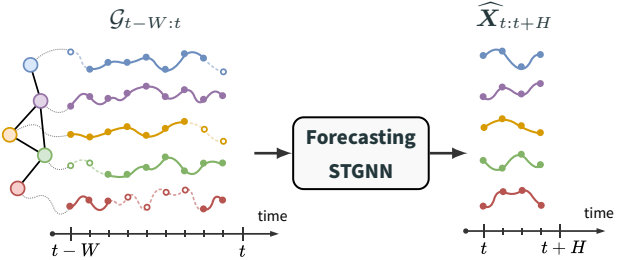
Forecasting from partial observations

A more direct approach: avoid the reconstruction step!

→ Design forecasting architecture to directly deal with irregular observations.

Benefits

- ☺ Learn how to leverage only valid observations specifically for the task at hand.
- ☺ Avoid the computational burden of imputing missing values.



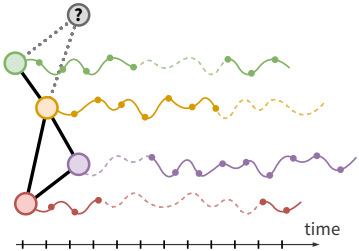
[9] Zhang *et al.*, "Graph-guided network for irregularly sampled multivariate time series", ICLR 2022.
[10] Zhong *et al.*, "Heterogeneous spatio-temporal graph convolution network for traffic forecasting with missing values", IEEE ICDCS 2021.
[11] Marisca *et al.*, "Graph-based Forecasting with Missing Data through Spatiotemporal Downsampling", ICML 2024.

Virtual sensing

The practice of estimating unmeasured states using models and existing observations.

The power of graphs:

- ☺ The relational processing allows us to condition estimates on data close in *space*.
- ☺ The inductive property of MP allows us to handle new nodes and edges.
- ☺ Useful in applications where sensing has a cost.



[12] Wu *et al.*, "Inductive Graph Neural Networks for Spatiotemporal Kriging", AAAI 2021.
[13] De Felice *et al.*, "Graph-Based Virtual Sensing from Sparse and Partial Multivariate Observations", ICLR 2024.

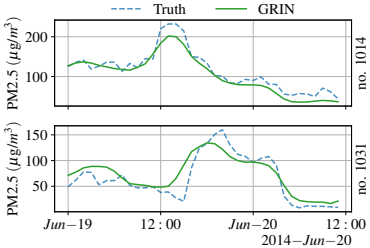
Graph imputation for virtual sensing

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

Clearly, several assumptions are needed

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

Two virtual sensors for air quality. (from [8])



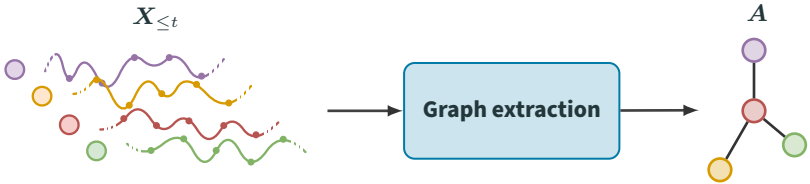
[8] Cini *et al.*, “Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks”, ICLR 2022.
[14] Marisca *et al.*, “Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations”, NeurIPS 2022.

Latent graph learning

Learning an adjacency matrix

☹ Relational information is not always (or only partially) available,
☹ or might be ineffective in capturing spatial dynamics.

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

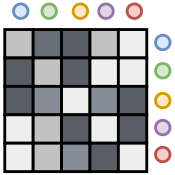


- When possible, the learned graph should be sparse.
- It can be interpreted as regularizing a spatial attention operator.
- This task is found under different names:
  graph structure learning, latent graph learning, graph inference...

Time-series similarities

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

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



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

Inferring latent structures from time series

Model the graph as a latent variable determining the realizations of the time series.

- They rely on assumptions, such as of signal smoothness and of a diffusion process.

Dedicated loss functions are formulated and minimized, e.g.,

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

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

$\rightarrow$ These approaches are commonly derived from a graph signal processing point of view.

[15] Dong *et al.*, "Learning Laplacian matrix in smooth graph signal representations", IEEE TSP 2016.
[16] Mateos *et al.*, "Connecting the dots: Identifying network structure via graph signal processing", IEEE SP Mag 2019.

Task-oriented latent graph learning

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

$\rightarrow$ e.g., by minimizing the forecasting error (MAE, MSE...).

Two different formulations:

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

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

$\rightarrow$ non-trivial 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

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

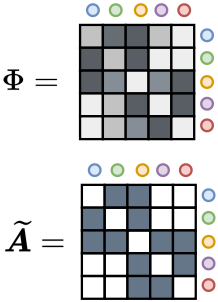
Edge scores Φ

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

$$\Phi = \Phi(\boldsymbol{X}, \Phi).$$

Function $\xi(\,\cdot\,)$ can enforce structures on $\boldsymbol{A}$, like,

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

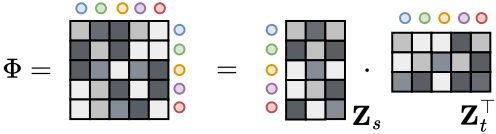


Edge score factorization

The number of possible edge scores is quadratic in the number of nodes ($\Phi \in \mathbb{R}^{N \times N}$)

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

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



with

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

$\boldsymbol{Z}_s$ and $\boldsymbol{Z}_t$ can be learned as tables of (local) parameters or as a function of the input window.

[17] Wu *et al.*, "Graph wavenet for deep spatial-temporal graph modeling", IJCAI 2019.

Pro & Cons of the direct approach

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

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

Probabilistic methods

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

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

Graphs of independent edges

For every edge (i, j)

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

Fixed-degree graphs

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

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

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

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

[18] Kazi *et al.*, “Differentiable graph module (dgm) for graph convolutional networks”, IEEE TPAMI 2022.
[19] Cini *et al.*, “Sparse graph learning from spatiotemporal time series”, JMLR 2023.

Learning graph distributions

Training typically involves optimizing terms similar to

$$\mathcal{L}(\theta, \Phi) = \mathbb{E}_{\boldsymbol{A} \sim p_\Phi}[L_\theta(\boldsymbol{A})]$$

which average a cost L_θ over all graphs according to p_Φ.

For example,

$$\begin{aligned}
&\mathcal{L}(\theta, \Phi) = \\
\text{(MSE)} \quad &= \mathbb{E}_{\boldsymbol{A} \sim p_\Phi} \left[\| \mathcal{F}_\theta(\boldsymbol{X}_{t-W:t}, \boldsymbol{A}) - \boldsymbol{X}_{t:t+H} \|^2 \right] . \\
\text{(CRPS)} \quad &= \mathbb{E}_{\boldsymbol{A} \sim p_\Phi} \left[\| \mathcal{F}_\theta(\boldsymbol{X}_{t-W:t}, \boldsymbol{A}) - \boldsymbol{X}_{t:t+H} \| \right] - \frac{1}{2} \mathbb{E}_{\boldsymbol{A}, \boldsymbol{A}' \sim p_\Phi} \left[\left\| \mathcal{F}_\theta(\boldsymbol{X}_{t-W:t}, \boldsymbol{A}) - \mathcal{F}_\theta(\boldsymbol{X}_{t-W:t}, \boldsymbol{A}') \right\| \right] .
\end{aligned}$$

(The expected value $\mathbb{E}_{\boldsymbol{X}, \boldsymbol{Y}}$ over the input and output data distribution is omitted for brevity.)

Gradient-based optimization and Monte Carlo sampling

Gradient-based optimization requires the computation of ∇_θ and ∇_Φ of $\mathcal{L}(\theta, \Phi) = \mathbb{E}_{\boldsymbol{A} \sim p_\Phi}[L_\theta(\boldsymbol{A})]$.

☺ Gradient $\nabla_\theta \mathcal{L}(\theta, \Phi)$ is can be estimated via Monte Carlo (MC) with standard tools

$$\nabla_\theta \mathcal{L}(\theta, \Phi) \overset{MC}{\approx} \nabla_\theta \frac{1}{M} \sum_m L_\theta(\boldsymbol{A}^m) = \frac{1}{M} \sum_m \nabla_\theta L_\theta(\boldsymbol{A}^m)$$

with $\{\boldsymbol{A}^m\}_{m=1}^M$ being a set of i.i.d. M samples from p_Φ.

☹ Estimating gradient $\nabla_\Phi \mathcal{L}(\theta, \Phi)$ via MC is less straightforward:

$$\nabla_\Phi \mathcal{L}(\theta, \Phi) = \nabla_\Phi \mathbb{E}_{\boldsymbol{A} \sim p_\Phi}[L_\theta(\boldsymbol{A})]$$

☹ Expanding the gradient leads to

$$\nabla_\Phi \mathcal{L}(\theta, \Phi) = \int L_\theta(\boldsymbol{A}) \nabla_\Phi p_\Phi(\boldsymbol{A}) \mathrm{d}\boldsymbol{A}.$$

- not in the form of an expected value,
- analytical computation is often unfeasible.

Reparametrization trick

One approach is to reparametrize $\boldsymbol{A} \sim p_\Phi(\boldsymbol{A})$ as: $\boldsymbol{A} = g(\Phi, \varepsilon)\,, \qquad \varepsilon \sim p(\varepsilon)$

$\rightarrow$ for instance, $a \sim \mathcal{N}(\mu, \sigma)$ can be written as $a = \mu + \varepsilon\sigma$, with $\varepsilon \sim \mathcal{N}(0, 1)$.

Above rewriting decouples parameters Φ from the random component ε:

$$\nabla_\Phi \mathbb{E}_{\boldsymbol{A} \sim p_\Phi}[L_\theta(\boldsymbol{A})] = \mathbb{E}_\varepsilon\left[\nabla_\Phi L(g(\Phi, \varepsilon)\right].$$

If $\boldsymbol{A} \in \{0, 1\}$, gradient $\nabla_\Phi g_\Phi(A) = 0$ almost everywhere and undefined otherwise.

$\rightarrow$ Continuous relaxation is used, e.g., Concrete distribution.

☺ Relatively easy to implement,

☹ relies on continuous relaxations: subsequent computations scale with $\mathcal{O}(N^2)$.

[20] Kipf *et al.*, "Neural relational inference for interacting systems", ICML 2018.

[21] Elinas *et al.*, "Variational inference for graph convolutional networks in the absence of graph data and adversarial settings", NeurIPS 2020.

Score-function gradient estimator

Score-function gradient estimators rely on the relation

$$\nabla_\Phi \mathbb{E}_{p_\Phi}\left[L_\theta(\boldsymbol{A})\right] = \mathbb{E}_{p_\Phi}\left[L_\theta(\boldsymbol{A}) \nabla_\Phi \log p_\Phi(\boldsymbol{A})\right]$$

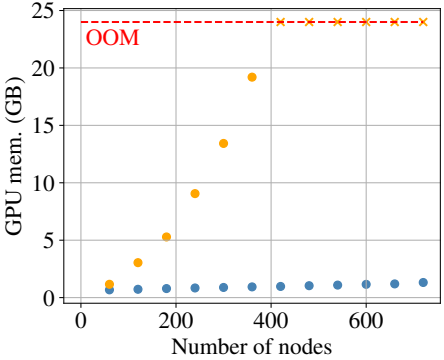
In our forecasting settings, it reads

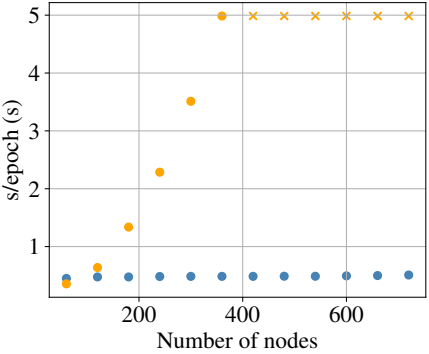
$$\nabla_\Phi \mathcal{L}(\theta, \Phi) \overset{MC}{\approx} \frac{1}{M} \sum_{m=1}^{M} \ell\left(\mathcal{F}_\theta(\boldsymbol{X}_{t-W:t}, \boldsymbol{A}), \boldsymbol{X}_{t:t+H}\right) \nabla_\Phi \log p_\Phi(\boldsymbol{A}_m).$$

- ☹ suffer from high variance (use variance reduction techniques),
- ☺ allow to keep computations sparse through the model.
  - do not rely on continuous relaxation of discrete random variables;
  - allow for sparse message passing in $\mathcal{F}(\boldsymbol{X}_{t-W:t}, \boldsymbol{A})$ by relying on sparse matrices $\boldsymbol{A}$.

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

Computational efficiency





● Score-function ● Reparametrization trick

Uncertainty quantification

While probabilistic models have been used to enable the **learning of discrete variables** (graph edges), the associated edge probabilities can carry information about the relevance of the associated connections.

→ It enables some degree of **explainability** and better informed **decision-making**.

⚠ Assessing the calibration of latent variables is hard on real data.

→ This is due to their latent nature, for which observations are not available.

[22] Gray *et al.*, "Bayesian inference of network structure from information cascades", IEEE TSIPN 2020.

Learning guarantees for latent graph calibration

☺ Under appropriate assumptions, we can achieve: $p_\Phi(\hat{\boldsymbol{X}}_{t:t+H} | \boldsymbol{X}_{t-W:t}) = p(\boldsymbol{X}_{t:t+H} | \boldsymbol{X}_{<t})$.

→ This means that the model output is calibrated.

😐 Calibration of $\boldsymbol{A}$ is not implied from that of the model output.

→ Conditions on the function $\boldsymbol{A} \mapsto \widehat{\boldsymbol{X}}_{t:t+H} = \mathcal{F}_\Phi(\boldsymbol{X}_{t-W:t}, \boldsymbol{A})$ are requested.

☺ For graphs and graph neural networks, these conditions appear easier to meet!

[23] Gneiting *et al.*, "Probabilistic forecasting", Annu. Rev. Stat. Appl. 2014.
[24] Manenti *et al.*, *Learning Latent Graph Structures and Their Uncertainty*, Preprint 2024.

Part 3

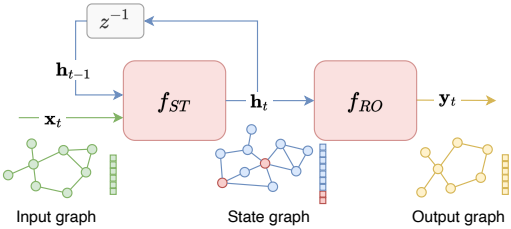
Future Directions

Graph State-Space Models

State-space models

$$\begin{cases} \mathbf{h}_t = f_{\mathsf{ST}}(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}, \boldsymbol{\eta}_{t-1}) \\ \mathbf{y}_t = f_{\mathsf{RO}}(\mathbf{h}_t, \boldsymbol{\nu}_t) \end{cases}$$

- Inputs $\mathbf{x}_t$, states $\mathbf{h}_t$, and outputs $\mathbf{y}_t$ are different attributed graphs.
- $\boldsymbol{\eta}_t, \boldsymbol{\nu}_t$ are noise terms at the node/edge level.

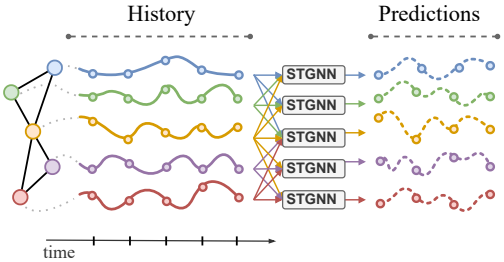


[25] Rangapuram *et al.*, "Deep State Space Models for Time Series Forecasting", NeurIPS 2018.
[26] Zambon *et al.*, *Graph State-Space Models*, Preprint 2023.
[27] Alippi *et al.*, *Graph Kalman Filters*, Preprint 2023.
[28] Buchnik *et al.*, "GSP-kalmannet: Tracking graph signals via neural-aided Kalman filtering", IEEE TSP 2024.
[29] Chouzenoux *et al.*, "Sparse graphical linear dynamical systems", JMLR 2024.

Hierarchical processing

What we achieved so far

- ☺ Pairwise dependencies are embedded into the processing.
- ☺ Predictions are localized w.r.t. a node and its neighbors.
- ☹ Operate at a fixed spatiotemporal scale.
- ☹ Higher-order dependencies are not explicitly modeled.

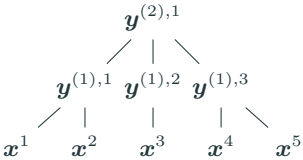


Hierarchical forecasting

- Hierarchical forecasting is about making predictions at multiple resolutions.
- Coherency constraints provide a regularization mechanism.
- Predictions are coherent iff:

$$\boldsymbol{Q}\widehat{\boldsymbol{Y}}_t = \left[\boldsymbol{I} \mid -\boldsymbol{C}\right] \widehat{\boldsymbol{Y}}_t = \boldsymbol{0},$$

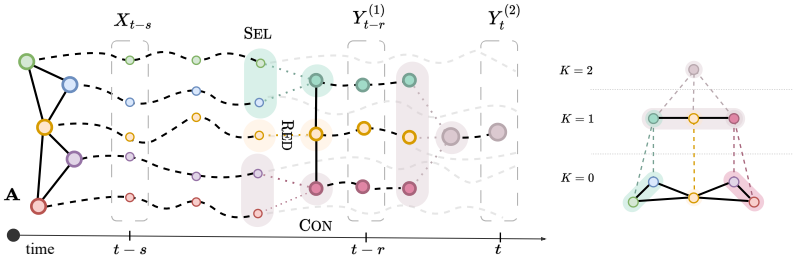
where $\boldsymbol{Y}_t$ contains stacked predictions for each level.



$$\boldsymbol{C} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

[30] Hyndman *et al.*, “Optimal combination forecasts for hierarchical time series”, Elsevier CSDA 2011.

Hierarchical Graph Predictor (HiGP)



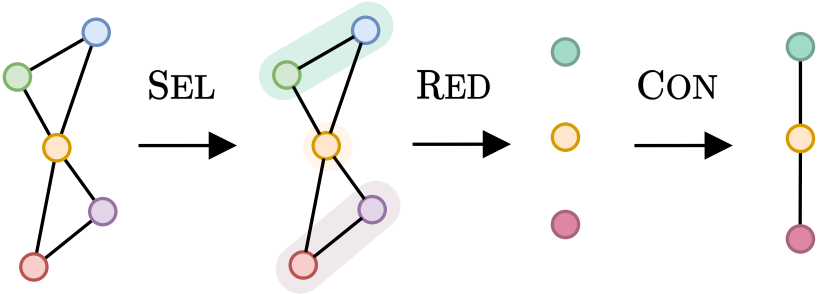
Combine hierarchical and graph-based forecasting.

We introduced a framework unifying the two.

- Operates at different spatial resolutions exploring a pyramidal graph structure.
- Exploits higher-order dependencies by operating on aggregated time series.
- Hierarchical time series clusters are learned end-to-end.

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

Select, Reduce, Connect (SRC)



- **Select** maps nodes into supernodes, i.e., node clusters.
- **Reduce** specifies how observations should be aggregated.
- **Connect** specifies how to rewire the graph after pooling.

[32] Grattarola *et al.*, "Understanding Pooling in Graph Neural Networks", IEEE TNNLS 2022.

A possible implementation

By exploiting the SRC framework to define the proper operators, hierarchical architectural biases can be embedded into a time-then-space STGNN architecture as

$$\boldsymbol{h}_t^{(k),i,0} = \textsc{SeqEnc}^{(k)}\left(\boldsymbol{y}_{t-W:t}^{(k),i}, \boldsymbol{u}_{t-W:t}^{(k),i}, \boldsymbol{v}^{(k),i}\right), \qquad \textbf{Temporal enc.}$$

$$\boldsymbol{Z}_t^{(k),l} = \mathsf{MP}_l^{(k)}\left(\boldsymbol{H}_t^{(k),l}, \boldsymbol{A}^{(k)}\right), \qquad \textbf{Intra-level prop.}$$

$$\boldsymbol{H}_t^{(k),l+1} = \mathsf{MLP}_l^{(k)}\left(\boldsymbol{Z}_t^{(k),l}, \underbrace{\boldsymbol{S}^{(k)^T}\boldsymbol{Z}_t^{(k-1),l}}_{\textsc{Red}^{(k)}}, \underbrace{\boldsymbol{S}^{(k)}\boldsymbol{Z}_t^{(k+1),l}}_{\textsc{Lift}^{(k)}}\right). \qquad \textbf{Inter-level prop.}$$

Representations can then be mapped to prediction using a **readout**.

Making coherent hierarchical forecasts

- **Learning time series clusters end-to-end**

  We learn **probabilistic cluster assignments** and use a **MinCut** [33] regularize.

$$\boldsymbol{S}^{(k)} \sim P(\boldsymbol{S}_{ij}^{(k)} = 1) = \frac{e^{\phi_{ij}^{(k)}/\tau}}{\sum_j e^{\phi_{ij}^{(k)}/\tau}}, \quad \boldsymbol{\Phi}^{(k)} = \mathcal{F}_{\boldsymbol{\psi}}\left(\boldsymbol{Y}_{t-W:t}^{(k-1)}, \boldsymbol{A}^{(k-1)}, \boldsymbol{V}^{(k-1)}\right).$$

  Forecasting the resulting aggregates provides a **self-supervised** learning mechanism.

- **Forecast reconciliation**

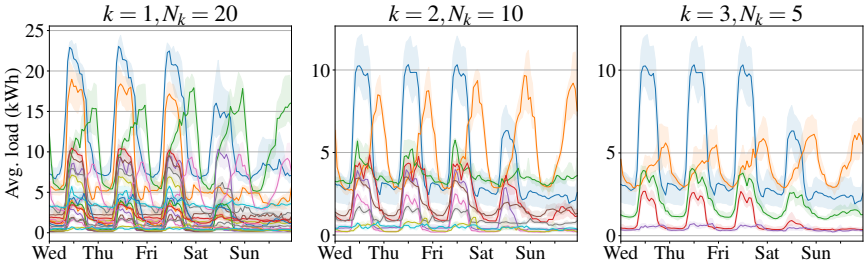
  A differentiable reconciliation step ensures **coherent forecasts** by recombining predictions as

$$\boldsymbol{P} \doteq \boldsymbol{I} - \boldsymbol{Q}^T \left(\boldsymbol{Q}\boldsymbol{Q}^T\right)^{-1} \boldsymbol{Q}, \qquad \overline{\boldsymbol{Y}}_t = \boldsymbol{P}\widehat{\boldsymbol{Y}}_t.$$

  ☹ Computing the inverse has a cubic cost, a soft regularization can be used alternatively.

[33] Bianchi *et al.*, “Spectral clustering with graph neural networks for graph pooling”, ICML 2020.
[34] Rangapuram *et al.*, “End-to-end learning of coherent probabilistic forecasts for hierarchical time series”, ICML 2021.

Example of learned clusters



Learned hierarchical clusters from CER (energy consumption profiles).

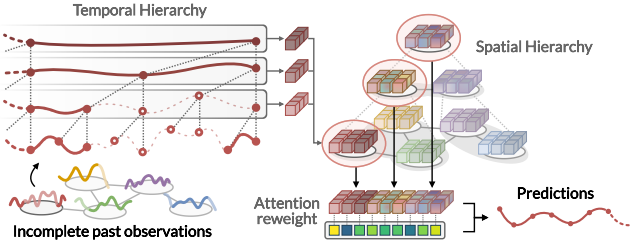
[31] Cini *et al.*, "Graph-based Time Series Clustering for End-to-End Hierarchical Forecasting", ICML 2024.

Multi-scale spatiotemporal representations

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

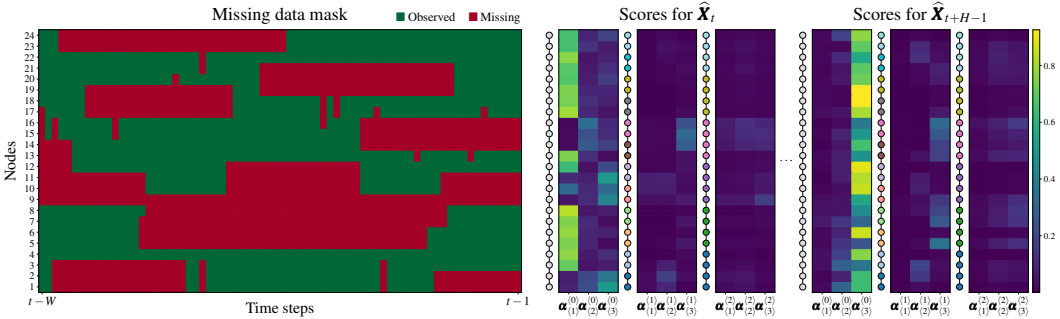
This gives a hierarchy of multi-scale representations, each accounting for a specific space-time resolution.

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



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

Downsampling with missing data



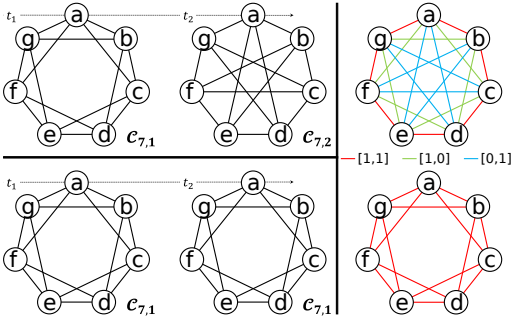
- The model focuses on the fine-grained temporal scale – if the most recent data are not missing.
- When data are missing at a given node, higher levels in the spatial hierarchy are given more weight.
- Slower dynamics become more relevant when long-range forecasting.

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

Theoretical properties

Theoretical properties of STGNNs

- High interest in studying the expressive power of GNNs in *static* graphs [36].
- Recent work extended the focus to dynamic settings, e.g., temporal graphs.
- **The important question**
  What's the impact of different spatiotemporal message-passing operators on the properties of the resulting STGNN?



From [37].

[37] Gao *et al.*, "On the Equivalence Between Temporal and Static Equivariant Graph Representations", ICML 2022.
[38] Gravina *et al.*, "Long Range Propagation on Continuous-Time Dynamic Graphs", ICML 2024.
[39] Beddar-Wiesing *et al.*, "Weisfeiler–Lehman goes dynamic: An analysis of the expressive power of graph neural networks for attributed and dynamic graphs", Neural Networks 2024.
[40] Wałęga *et al.*, "Expressive Power of Temporal Message Passing", Preprint 2024.

Conclusions

Some Takeaways

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

- Relational inductive biases allow for exploiting dependencies among the time series...
- ... and effectively processing spatiotemporal data,
- while sharing most of the model parameters.
- Global-local models are a good starting point.

Resources. Tutorial paper [3] • Open-source library [41]

[3] Cini, Marisca, Zambon, and Alippi, "Graph Deep Learning for Time Series Forecasting", Preprint 2023.
[41] Cini and Marisca, *Torch Spatiotemporal*, https://github.com/TorchSpatiotemporal/tsl 2022.

Andrea **Cini**

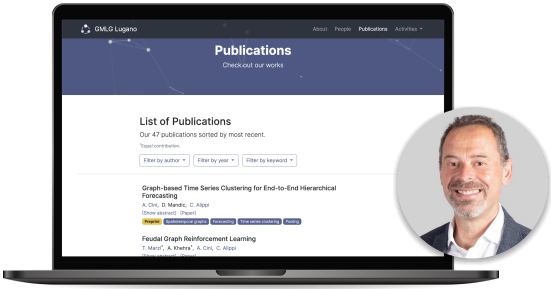
Ivan **Marisca**

Daniele **Zambon**

Graph Machine Learning Group

gmlg.ch

Group leader: Prof. Cesare **Alippi**



THE END

Questions?

References i

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

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

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

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

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

References ii

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

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

[8] 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`.

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

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

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

References iii

[12] 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.

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

[14] 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.

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

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

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

References iv

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

[19] 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.

[20] 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.

[21] P. Elinas, E. V. Bonilla, and L. Tiao, **"Variational inference for graph convolutional networks in the absence of graph data and adversarial settings,"** *Advances in neural information processing systems*, vol. 33, pp. 18 648–18 660, 2020.

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

[23] T. Gneiting and M. Katzfuss, **"Probabilistic forecasting,"** *Annual Review of Statistics and Its Application*, vol. 1, no. 1, pp. 125–151, 2014.

References v

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

[25] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, **"Deep State Space Models for Time Series Forecasting,"** in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018.

[26] D. Zambon, A. Cini, L. Livi, and C. Alippi, ***Graph state-space models***, Jan. 2023. DOI: `10.48550/arXiv.2301.01741`.

[27] C. Alippi and D. Zambon, ***Graph Kalman Filters***, Mar. 2023. DOI: `10.48550/arXiv.2303.12021`.

[28] I. Buchnik, G. Sagi, N. Leinwand, Y. Loya, N. Shlezinger, and T. Routtenberg, **"Gsp-kalmannet: Tracking graph signals via neural-aided kalman filtering,"** *IEEE Transactions on Signal Processing*, 2024.

[29] E. Chouzenoux and V. Elvira, **"Sparse graphical linear dynamical systems,"** *Journal of Machine Learning Research*, vol. 25, no. 223, pp. 1–53, 2024.

[30] R. J. Hyndman, R. A. Ahmed, G. Athanasopoulos, and H. L. Shang, **"Optimal combination forecasts for hierarchical time series,"** *Computational statistics & data analysis*, vol. 55, no. 9, pp. 2579–2589, 2011.

References vi

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

[32] D. Grattarola, D. Zambon, F. Bianchi, and C. Alippi, **"Understanding Pooling in Graph Neural Networks,"** *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–11, 2022. DOI: 10.1109/TNNLS.2022.3190922.

[33] F. M. Bianchi, D. Grattarola, and C. Alippi, **"Spectral clustering with graph neural networks for graph pooling,"** in *International conference on machine learning*, PMLR, 2020, pp. 874–883.

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

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

[36] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, **"How powerful are graph neural networks?"** In *International Conference on Learning Representations*, 2019.

References vii

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

[38] A. Gravina, G. Lovisotto, C. Gallicchio, D. Bacciu, and C. Grohnfeldt, **"Long range propagation on continuous-time dynamic graphs,"** in *Forty-first International Conference on Machine Learning*, 2024. [Online]. Available: `https://openreview.net/forum?id=gVg8V9isul`.

[39] S. Beddar-Wiesing, G. A. D'Inverno, C. Graziani, *et al.*, **"Weisfeiler–lehman goes dynamic: An analysis of the expressive power of graph neural networks for attributed and dynamic graphs,"** *Neural Networks*, vol. 173, p. 106 213, 2024.

[40] P. A. Wałęga and M. Rawson, **"Expressive power of temporal message passing,"** *arXiv preprint arXiv:2408.09918*, 2024.

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