Graph Deep Learning for Spatiotemporal Time Series

Forecasting, Reconstruction and Analysis

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

ECML/PKDD, Turin · September 22, 2023

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


Traffic monitoring

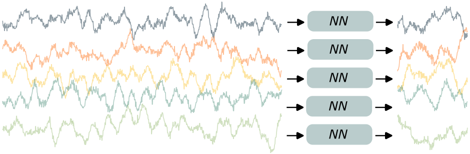

Smart cities


Energy analitics


Physics


Stock markets

Deep learning for time series forecasting



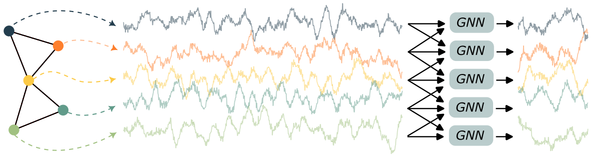
The standard deep learning approach to time series forecasting consists in training a single neural network on a collection of time series.

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

⚠ Dependencies across time series are often discarded.

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

Relational inductive biases



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

Graph neural networks provide appropriate neural operators.

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

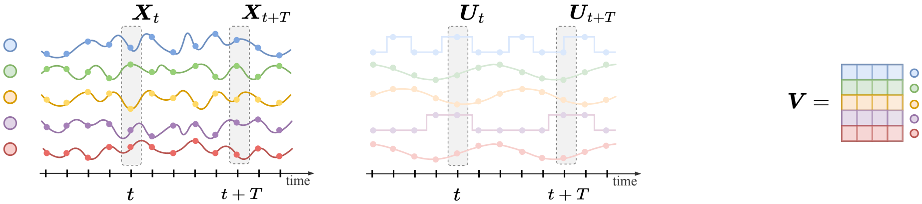
[2] D. Bacciu *et al.*, "A gentle introduction to deep learning for graphs", NN 2020.
[3] M. M. Bronstein *et al.*, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges" 2021.

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

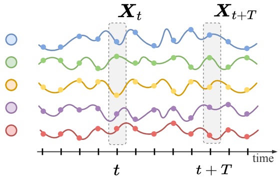
[4] A. Cini *et al.*, “Graph Deep Learning for Time Series Forecasting: A Comprehensive Methodological Framework” 2023.

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 $i = 1 \ldots N$ and $t \in \mathbb{N}$.



Note that the time series:

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

  homogenous, synchronous, regularly sampled.

Notation:

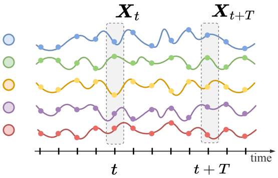
$$\boldsymbol{X}_{t:t+T} = [\boldsymbol{X}_t, \cdots, \boldsymbol{X}_{t+T-1}]$$
$$\boldsymbol{X}_{<t} = [\boldsymbol{X}_0, \cdots, \boldsymbol{X}_{t-2}, \boldsymbol{X}_{t-1}]$$

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

Relational information

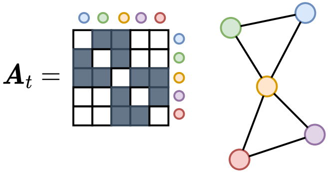
We assume the existence of functional dependencies between the time series.

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



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

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



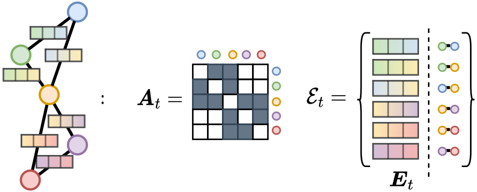
$\rightarrow$ We call spatial the dimension spanning the time series collection.

Relational information with attributes

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

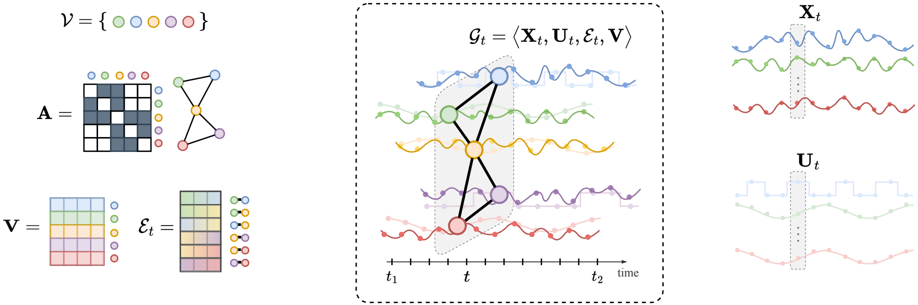
The set of attributed edges encoding all the available relational information is denoted by

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



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

Spatiotemporal time series



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

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

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

Example: Traffic monitoring system

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

- $\boldsymbol{X}_{<t}$ collects past traffic speed measurements.
- $\boldsymbol{U}_t$ stores identifiers for time-of-the-day and day-of-the-week.
- $\boldsymbol{V}$ collects static sensor's features, e.g., type or number of lanes of the monitored road.
- $\mathcal{E}$ can be obtained by considering the road network.
  - Road closures and traffic diversions can be accounted for with a dynamic topology $\mathcal{E}_t$.

Time series forecasting

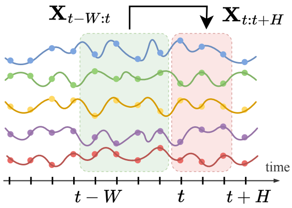
Multi-step time-series forecasting

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

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

predict $H \geq 1$ future observations

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



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

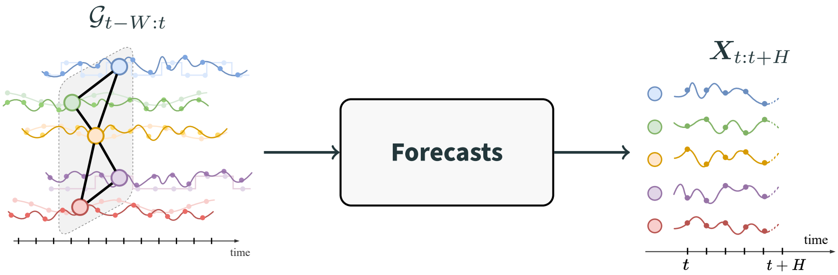
$$p_{\boldsymbol{\theta}}\left(\boldsymbol{x}^i_{t+h} \middle| \boldsymbol{X}_{t-W:t}, \boldsymbol{U}_{t-W:t+h}, \boldsymbol{V}\right) \approx p^i\left(\boldsymbol{x}^i_{t+h} \middle| \boldsymbol{X}_{<t}, \boldsymbol{U}_{\leq t+h}, \boldsymbol{V}\right).$$

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

Time series forecasting + relational inductive biases

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

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



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

Point forecasts

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

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

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

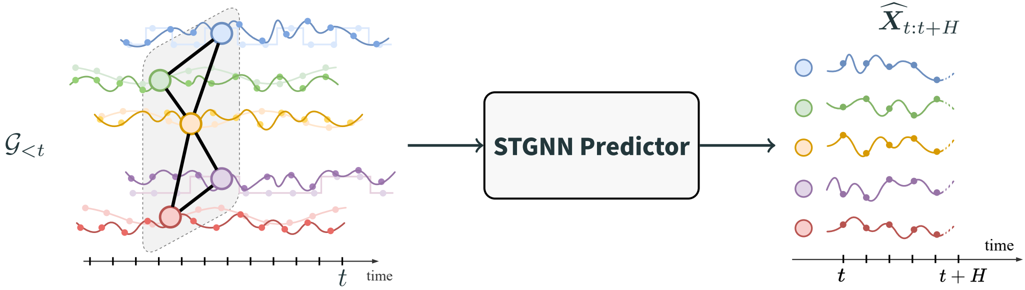
Parameters $\boldsymbol{\theta}$ can be learned by minimizing a cost function $\ell(\,\cdot\,,\,\cdot\,)$ (e.g., MSE) on a training set

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

Spatiotemporal Graph Neural Networks

Spatiotemporal Graph Neural Networks

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



We focus on models based on message passing.

Message-passing neural networks

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

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

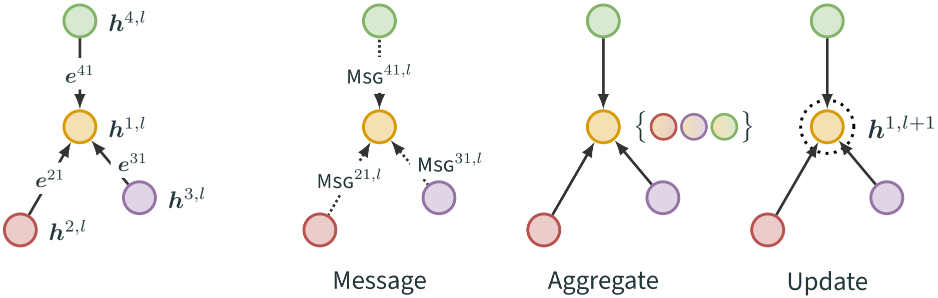
Where:

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

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

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

Message passing in action



Spatiotemporal message passing

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

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

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

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

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

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

A general recipe

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

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

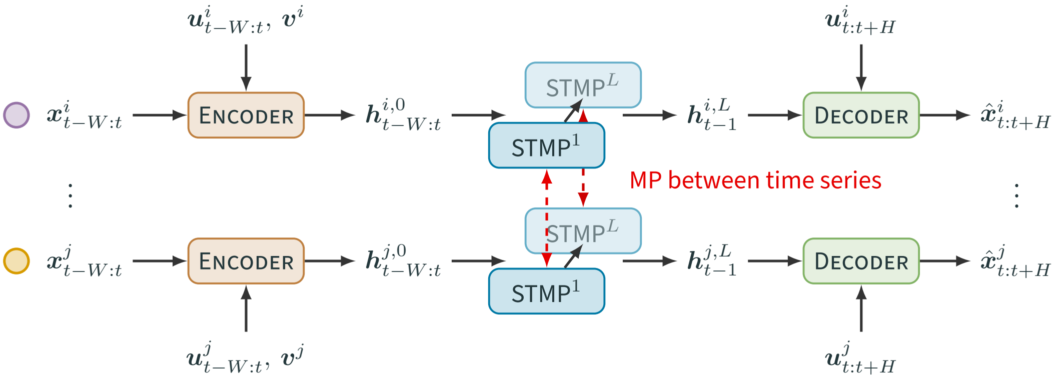
$$\boldsymbol{H}^{l+1}_{t-1} = \text{STMP}^l\left(\boldsymbol{H}^l_{\leq t-1}, \mathcal{E}_{\leq t-1}\right), \quad l = 0, \dots, L-1 \tag{3}$$

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

Where:

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

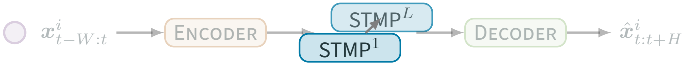
Framework overview



Design paradigms for STGNNs

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

- **Time-and-Space (T&S)**
  Temporal and spatial processing cannot be factorized in two separate steps.
- **Time-then-Space (TTS)**
  Embed each time series in a vector, which is then propagated over the graph.
- **Space-then-Time (STT)**
  Propagate nodes features at first and then process the resulting time series.



Time-and-Space

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

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

Several options exist.

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

Example 1: From Recurrent Neural Networks...

Consider a standard GRU [6] cell.

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

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

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

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

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

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

...to Graph Convolutional Recurrent Neural Networks

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

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

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

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

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

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

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

---

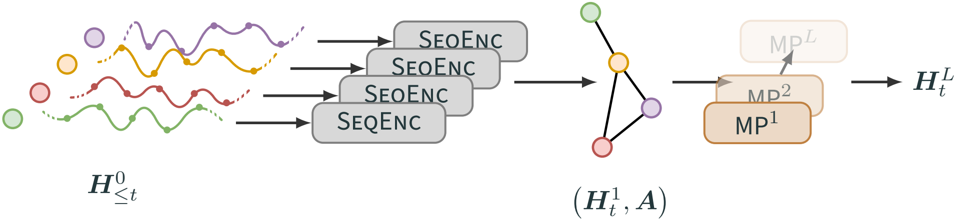
[7] Y. Seo *et al.*, "Structured sequence modeling with graph convolutional recurrent networks", ICONIP 2018.

Time-then-Space models

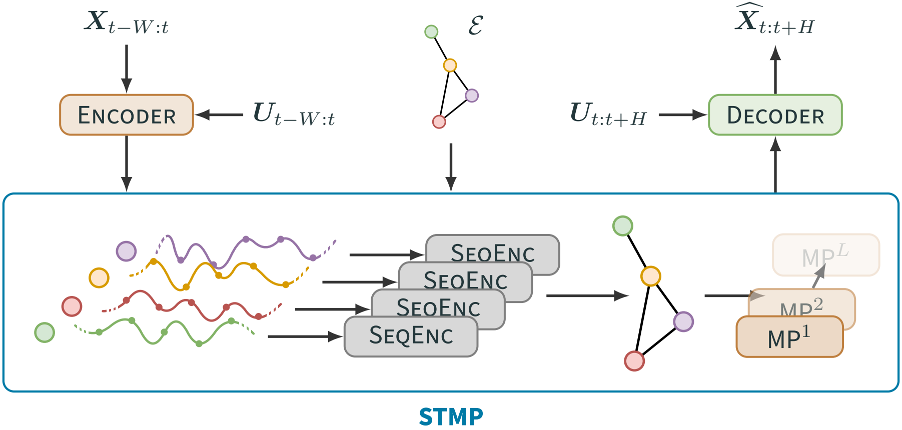
The general recipe for a TTS model consists in:

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.

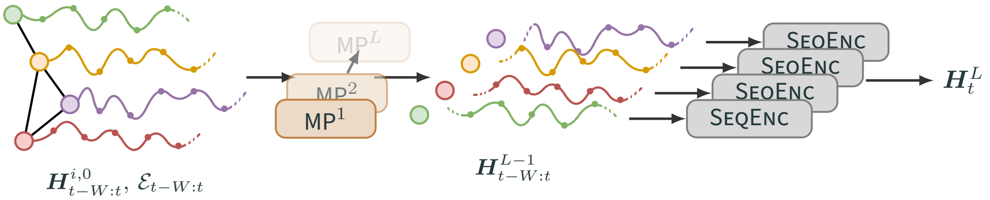
☹ Accounting for changes in topology and dynamic edge attributes can be more problematic.

Space-then-Time

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

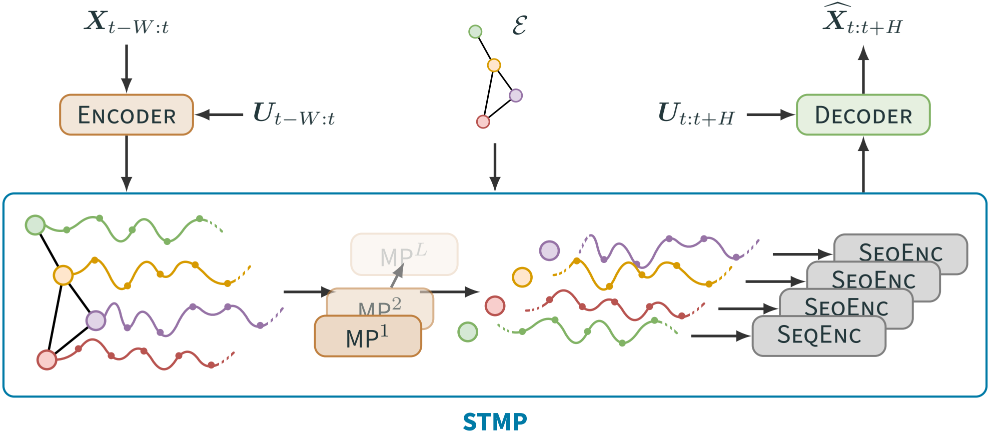
1. Observations are propagated among nodes w.r.t. each time step using a stack of MP layers.
2. Each sequence of representations is processed by a sequence encoder.

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



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