
Graph Deep Learning for Irregular Spatiotemporal Data

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Ivan Marisca

under the supervision of
Prof. Cesare Alippi

July 2025

Dissertation Committee

Prof. Luca Maria Gambardella	Università della Svizzera italiana, Switzerland
Prof. Marc Langheinrich	Università della Svizzera italiana, Switzerland
Prof. Xiaowen Dong	University of Oxford, UK
Prof. Paolo Frasconi	University of Firenze, Italy
Prof. Marco Gori	University of Siena, Italy

Dissertation accepted on 29 July 2025



Research Advisor

Prof. Cesare Alippi

PhD Program Director

Prof. Walter Binder / Prof. Stefan Wolf

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Ivan Marisca
Lugano, 29 July 2025

Abstract

Irregular spatiotemporal data – with observations acquired unevenly across space and time – are common in real-world applications such as climate modeling, social media analytics, and transportation systems. In these settings, capturing underlying structural dependencies is critical for effective pattern recognition and learning. While the existing literature has predominantly focused on regularly sampled data, irregular sampling introduces significant challenges that require novel methodological approaches. My doctoral research aims to advance predictive modeling for irregular spatiotemporal data through a graph-based representation, focusing specifically on imputation, filtering, and prediction across both temporal and spatial dimensions. Graphs naturally lend themselves to modeling irregularities, with edges encoding relationships between data-generating processes, making graph deep learning a suitable and powerful processing framework. Imputation and filtering methods are proposed to reconstruct missing observations and improve data quality by enforcing spatiotemporal consistency. Prediction approaches are explored to forecast future observations by leveraging underlying dependencies. A theoretical analysis of information propagation in spatiotemporal architectures completes the thesis, providing insights and guidelines for the design of future models.

Contents

Contents	iii
List of Figures	ix
List of Tables	xi
List of Acronyms	xiii
1 Introduction	1
1.1 Research lines	3
1.2 Challenges	5
1.3 Contributions	6
1.4 Publications and dissemination	9
1.5 Thesis outline	10
2 Problem formalization and preliminaries	13
2.1 Problem setting	13
2.1.1 Relational dependencies	15
2.1.2 Changes in topology and sensor heterogeneity	16
2.1.3 Missing data	17
2.2 Imputation, filtering, and prediction	18
2.2.1 Prediction	18
2.2.2 Imputation and filtering	20
2.3 Graph neural networks	21
2.3.1 Over-smoothing, over-squashing, and under-reaching . . .	23
2.3.2 Isotropic vs anisotropic message-passing	25
2.4 Deep learning for temporal processing	26
2.4.1 Recurrent neural networks	27
2.4.2 Temporal convolutional networks	28

2.4.3	Attention-based models	29
2.5	Spatiotemporal graph neural networks	30
2.5.1	Spatiotemporal message-passing	30
2.5.2	Spatiotemporal message-passing operators	31
2.6	Datasets	33

I Imputation 41

3	Imputation of irregular spatiotemporal data	43
3.1	Graph deep learning for time series imputation	45
3.1.1	Graph-based filtering and virtual sensing	46
3.1.2	Operational settings	47
3.2	Graph-based autoregressive imputation	49
3.2.1	The autoregressive approach	50
3.2.2	Graph Recurrent Imputation Network	51
3.2.3	Model optimization: a filtering objective	56
3.3	Related work	57
3.4	Empirical evaluation	60
3.4.1	Imputation baselines	60
3.4.2	Air quality	61
3.4.3	Traffic and energy analytics	62
3.4.4	Ablation study	63
3.4.5	Virtual sensing	64
3.5	Discussion and future directions	65
4	Imputation with sparse observations	67
4.1	Learning from sparse spatiotemporal data	69
4.1.1	Sparse spatiotemporal attention	70
4.1.2	Two-phase propagation	72
4.1.3	Model optimization	73
4.2	Addressing the computational scalability	73
4.2.1	Graph subsampling	73
4.2.2	Hierarchical attention	74
4.3	Spatiotemporal positional encoding	75
4.4	Related work	76
4.5	Empirical evaluation	76
4.5.1	Imputation benchmarks	77
4.5.2	Sensitivity analysis	78

4.5.3	Ablation study	80
4.5.4	Virtual sensing	80
4.6	Discussion and future directions	81
II	Prediction	83
5	Prediction under irregular spatial sampling	85
5.1	Graph deep learning for time series forecasting	87
5.1.1	Globality and locality in STGNNs	88
5.1.2	Forecasting processing paradigms	90
5.2	Reference forecasting STGNNs	92
5.3	Global-local graph-based predictors	94
5.3.1	Learnable node embeddings	95
5.3.2	Structuring the embedding space	96
5.3.3	Empirical validation	97
5.4	Scalable graph-based predictors	100
5.4.1	Preliminaries: echo state networks	101
5.4.2	Randomized spatiotemporal encoding	102
5.4.3	Trainable multi-scale decoder	103
5.4.4	Computational complexity	105
5.5	Related work	105
5.6	Empirical evaluation	107
5.6.1	Forecasting baselines	108
5.6.2	Global, local, and global-local architectures	109
5.6.3	Large-scale forecasting	110
5.6.4	Forecasting benchmarks	112
5.7	Discussion and future directions	113
6	Prediction under irregular spatiotemporal sampling	115
6.1	Spatiotemporal downsampling	117
6.2	Scalable hierarchical downsampling	119
6.2.1	Learning hierarchical representations	120
6.2.2	Adaptive decoder	122
6.2.3	Computational scalability	124
6.3	Related work	124
6.4	Empirical evaluation	126
6.4.1	Controlled environment	127
6.4.2	Forecasting benchmarks	129

6.5	Interpretability of decoder weights	131
6.6	Discussion and future directions	132
7	Spatiotemporal over-squashing	135
7.1	Spatial over-squashing and sensitivity analysis	137
7.2	Spatiotemporal over-squashing	138
7.3	Over-squashing in TCNs	140
7.3.1	Information propagation in TCNs	141
7.3.2	Temporal over-squashing	142
7.3.3	Temporal graph rewiring	146
7.3.4	Empirical validation	148
7.4	Over-squashing in MPTCNs	150
7.4.1	Empirical validation	153
7.5	Related work	156
7.6	Discussion and future directions	157
8	Conclusion	159
8.1	Summary of contributions	159
8.2	Future directions	161
8.3	Final remarks	162
	Glossary	163
	Bibliography	167

Figures

1.1	Regular versus irregular spatiotemporal sampling.	2
1.2	Imputation, filtering, and prediction.	4
2.1	Representation of irregular spatiotemporal data.	14
2.2	The message-passing framework.	22
2.3	Cartesian product of spatial graph $\mathbf{A}$ and temporal graph $\mathbf{T}$	32
3.1	The autoregressive approach to imputation.	50
3.2	The unidirectional GRIN module.	53
3.3	Virtual sensing using GRIN.	65
4.1	The architecture of SPIN.	70
4.2	The spatiotemporal attention mechanism in SPIN.	71
4.3	Virtual sensing using SPIN-H.	81
5.1	Globality and locality in forecasting architectures.	88
5.2	Receptive fields in T&S and TTS MPRNNs.	93
5.3	Node clusters in CER-E via embedding space regularization. . . .	99
5.4	Overview of the SGP encoder.	102
5.5	Standard vs. grouped feed-forward layer.	104
6.1	Schematic representation of the SRC pooling framework.	118
6.2	Overview of HD-TTS.	119
6.3	Architectural scheme of HD-TTS.	120
6.4	Details of the spatial processing procedure.	122
6.5	Generation of the GraphMSO dataset.	128
6.6	Interpretability of decoder weights in HD-TTS.	131
7.1	Evolution of receptive fields in TCNs.	146
7.2	Empirical validation of temporal over-squashing in TCNs.	149
7.3	RING and LOLLIPOP graphs used in the synthetic experiments. . . .	153

7.4	Validation of spatiotemporal over-squashing in MPTCNs.	154
-----	--	-----

Tables

2.1	Metadata of datasets used in the experiments.	34
3.1	Imputation error of GRIN on AQI datasets.	61
3.2	Imputation error of GRIN on METR-LA, PEMS-BAY, and CER-E.	63
3.3	Ablation study of GRIN.	64
4.1	Imputation error of SPIN on imputation benchmarks.	77
4.2	Performance of SPIN with increasing data sparsity.	78
4.3	Performance of SPIN with increasing number of failures.	79
4.4	Ablation study of SPIN.	80
5.1	Forecasting error in the transfer learning setting.	98
5.2	Forecasting error on PEMS04 in the transfer learning setting by varying fine-tuning set size.	98
5.3	Forecasting error for global, local, and global-local models.	109
5.4	Forecasting error and computational performance on large-scale datasets.	111
5.5	Forecasting architectures performance across five benchmarks.	112
6.1	Forecasting error on GraphMSO.	128
6.2	Parameters used to generate the missing data masks.	129
6.3	Forecasting error of HD-TTS across missing data distributions.	130
7.1	Forecasting error with fixed budget in real-world benchmarks.	155

Acronyms

BRITS Bidirectional Recurrent Imputation for Time Series	51
CNN convolutional neural network	1
DAG directed acyclic graph	27
ESN echo state network	101
GNN graph neural network	3
GRIN Graph Recurrent Imputation Network	49
GRU Gated Recurrent Unit	27
GSO graph shift operator	25
HD-TTS Hierarchical Downsampling Time-Then-Space	119
LSTM Long Short-Term Memory	27
MAE mean absolute error	20
MAR missing at random	17
MCAR missing completely at random	17
MLP multilayer perceptron	22
MNAR missing not at random	17
MP message passing	22
MPGRU message-passing GRU	33

MPNN message-passing neural network	21
MPRNN message-passing recurrent neural network	32
MPTCN message-passing temporal convolutional network	32
MSE mean squared error	20
RNN recurrent neural network	1
SGP Scalable Graph Predictor	100
SPIN Spatiotemporal Point Inference Network	69
STGNN spatiotemporal graph neural network	30
STMP spatiotemporal message-passing	30
STT space-then-time	90
T&S time-and-space	90
TCN temporal convolutional network	28
TMP temporal message-passing	26
TTS time-then-space	90

Chapter 1

Introduction

Spatiotemporal data refers to information collected across both space and time, capturing observations that are paired with spatial and temporal indices. The spatial dimension typically corresponds to a set of data-generating entities (i.e., *sensors*) constrained by relationships, which may represent physical systems (e.g., traffic sensors or weather stations) or abstract entities (e.g., stocks in financial markets or users in a network). Common examples include satellite image sequences, traffic monitoring data capturing speed and congestion levels, or stock market time series, to mention a few application domains.

The growing presence of Internet of Things technologies [Ashton et al., 2009] and pervasive computing has led to the proliferation of spatiotemporal data across a wide range of domains, creating a need for machine learning models that can effectively process such data at scale. Spatiotemporal data often exhibits strong dependencies along both spatial and temporal axes [Atluri et al., 2018], with nearby observations in space or time typically showing higher correlation. Many modern learning systems leverage this property through inductive biases that prioritize local information, embedding these constraints directly into their architectures.

Historically, most successful approaches to spatiotemporal learning have focused on discrete-time sequences and Euclidean domains, where observations are regularly sampled over time and space, as in video data. Deep learning models [LeCun et al., 2015; Schmidhuber, 2015] have excelled in this regime, offering powerful tools to capture complex structural dependencies [Frasconi et al., 1998]. Amongst others, convolutional neural networks (CNNs) [LeCun and Bengio, 1998] have proven effective for spatial feature extraction, while recurrent neural

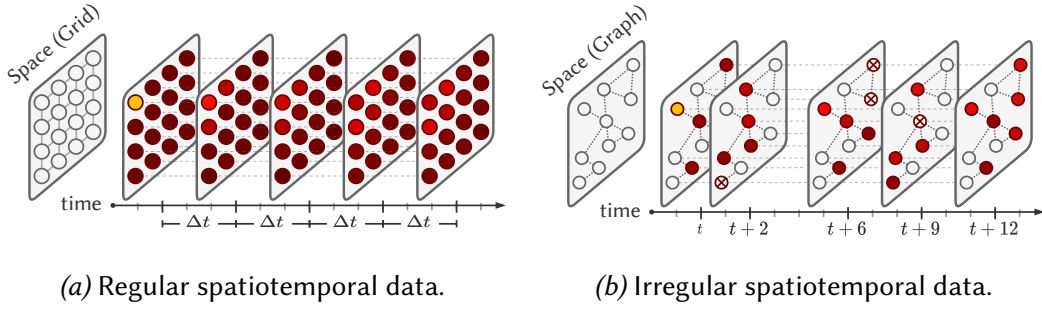


Figure 1.1. A spatiotemporal diffusion process observed under regular (a) and irregular (b) sampling in space and time. Crossed circles ($\otimes$) denote acquisition failures resulting in missing values.

networks (RNNs) [Elman, 1990; Hochreiter and Schmidhuber, 1997] have been widely used to model temporal dynamics. Hybrid architectures have also been proposed for jointly capturing spatiotemporal patterns [Shi et al., 2015]. More recently, attention-based architectures such as Transformers [Vaswani et al., 2017] have been adapted to images [Dosovitskiy et al., 2021; Arnab et al., 2021] and videos [Bertasius et al., 2021], with applications ranging from healthcare [Tran et al., 2023] to live-cell imaging [Pulfer et al., 2024].

However, real-world spatiotemporal data are rarely regular, with irregularities commonly arising in both spatial and temporal dimensions. In space, sensors are often scattered, with varying distances depending on physical constraints, expert knowledge, or deployment costs, as is typical in environmental monitoring. In time, irregularities result from asynchronous sampling across sensors or missing data due to malfunctions, communication errors, or other transient issues. Non-physical domains, such as social media or financial markets, are also not exempt from irregularities. Here, temporal gaps may be caused, for instance, by privacy concerns [Langheinrich, 2001] or access limitations, while spatial relationships – between users or assets – reflect behavioral or functional affinities rather than physical proximity. Such irregularities challenge conventional methods reliant on inputs with regular underlying structure and call for frameworks that move beyond traditional assumptions of uniform sampling and Euclidean spaces. **Figure 1.1** shows the foundational difference between the two settings with an example of a diffusion process under regular (left) and irregular (right) sampling.

In the spatial dimension, the key to addressing this challenge lies in effectively modeling structural information, which is typically relational. Graphs offer a

natural and powerful representation to model these dependencies, with nodes corresponding to data-generating entities – the sensors – and edges encoding the relationships between them, whether these are physical distances, statistical correlations, or domain-specific constraints. This graph-based perspective has a twofold advantage. First, it explicitly encodes non-Euclidean structural information without requiring regular sampling or grid-like arrangements. Second, it provides flexibility to represent diverse relationship types, allowing the same methodological framework to address both physical and non-physical domains.

Building on this representation, graph signal processing extends traditional discrete signal processing to data defined on graphs, offering a principled framework for data analysis [Shuman et al., 2013; Ortega et al., 2018; Dong et al., 2020; Leus et al., 2023]. More recently, the emerging field of geometric deep learning [Bronstein et al., 2021] has focused specifically on developing neural architectures for non-Euclidean domains like graphs. Within this context, graph neural networks (GNNs) [Sperduti and Starita, 1997; Gori et al., 2005; Scarselli et al., 2008; Battaglia et al., 2018; Bacciu et al., 2020] have proven effective at capturing spatial dependencies through iterative message-passing mechanisms [Gilmer et al., 2017] that aggregate information from neighboring nodes.

However, despite major advances in the literature, relatively little attention has been given to the spatiotemporal setting, where information associated with graph nodes evolves over time [Seo et al., 2018]. The challenge becomes even more pronounced in the case of irregular sampling, which remains largely underexplored. Thus, the literature on graph-based modeling of irregular spatiotemporal data remains in its early stages: the development of rigorous methodologies represents a significant opportunity for advancing the field, with potential for major impacts on real-world problems.

1.1 Research lines

This thesis advances the modeling and analysis of irregular spatiotemporal data through the lens of graph deep learning, structured around three interconnected research lines:

- R1. Imputation and filtering** — These tasks address data refinement within a temporal window. Imputation focuses on estimating missing values by exploiting spatiotemporal dependencies, while filtering aims to suppress noise or inconsistent patterns that may hinder downstream modeling.

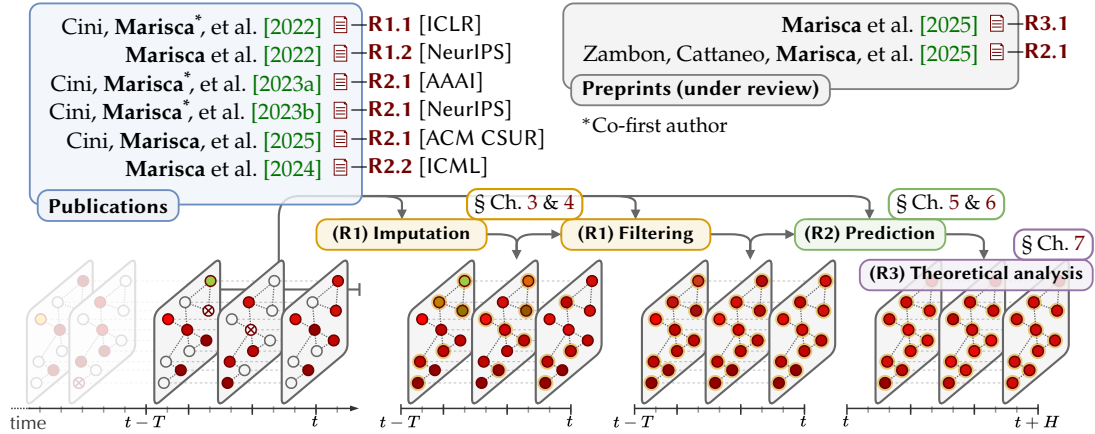


Figure 1.2. The interconnected tasks of imputation, filtering, and prediction of irregular spatiotemporal data. While imputation and filtering can be end goals, they can also serve as pre-processing for downstream tasks such as prediction. Colored boxes display authored publications and preprints associated with the corresponding research line and contribution.

Both are essential in real-world settings where data incompleteness and measurement errors are actual challenges. Moreover, they often serve as critical preprocessing steps for downstream tasks.

R2. Prediction — Forecasting future observations involves inferring values beyond the available time interval, based on partially observed data. This task is central to numerous domains, from climate science to healthcare, where anticipating the future is crucial for planning and intervention. In irregular settings, the challenge becomes more pronounced due to incomplete, asynchronous, or non-uniformly sampled inputs, which introduce additional uncertainty.

R3. Theoretical analysis — Understanding how information propagates across space and time in graph-based models is fundamental to both explaining their behavior and guiding architectural improvements. Research in this direction aims to provide theoretical insights into how design choices, such as spatial and temporal processing schemes, affect a model’s ability to integrate and preserve information.

These research directions are tightly coupled: advances in imputation and filtering often lead to improved predictive performance, while theoretical insights guide the refinement of models across tasks. Importantly, from a predictive

standpoint, perfect imputation is not always necessary: models can achieve accurate forecasts even when reconstructed inputs deviate from the true values [Le Morvan et al., 2021]. Nevertheless, some form of imputation remains often necessary to provide a consistent input representation, stabilize training, and enable the use of architectures that require fully observed sequences. At the same time, imputation is valuable as a standalone goal, e.g., to support interpretability of the data, anomaly detection, and virtual sensing. Hence, imputation and prediction should be regarded as complementary objectives, while theoretical analysis provides the unifying principles underlying model design. Together, these three directions contribute to a comprehensive framework for graph-based spatiotemporal learning under irregular sampling conditions. **Figure 1.2** illustrates this processing pipeline and highlights the contributions made in each research line throughout my doctoral studies.

1.2 Challenges

Across these research directions, three fundamental challenges emerge that guide the methodological developments in this thesis:

Irregularity. The fundamental challenge lies in effectively modeling data that are unevenly sampled in time and structured over complex, non-Euclidean spaces. Unlike traditional approaches that assume regular grids or uniform sampling, methods for irregular spatiotemporal data must incorporate the availability of observations and the underlying structural dependencies as intrinsic elements of the learning process. This requires architectures that can flexibly adapt to varying observation patterns while preserving the ability to extract meaningful spatiotemporal patterns.

Sparsity. Beyond mere irregularity, many real-world applications suffer from extreme sparsity, where only a small subset of potential observations is available. This sparsity challenges standard modeling assumptions, particularly those relying on continuity, as large gaps in space or time limit the effectiveness of local inductive biases, which depend on the availability of nearby observations. Consequently, models must be able to operate robustly under sparse conditions, effectively propagating information across distant points in the spatiotemporal domain to preserve predictive accuracy and reliability.

Scalability. The computational burden of spatiotemporal modeling compounds across both dimensions, particularly for large-scale applications involving many sensors and long time horizons. As the number of time series and the volume of collected data grow, memory and computational constraints become increasingly limiting. This necessitates architectures that balance expressiveness with efficiency, avoiding the high complexity that naïve approaches would incur when processing both spatial and temporal information simultaneously.

Addressing these challenges jointly enables the development of both theoretical and practical contributions for modeling irregular spatiotemporal data in real-world scenarios, in which they often coexist. Indeed, a dataset can be simultaneously sparse at the local level – with individual sensors or regions providing few or intermittent observations – yet involving a large-scale system – with potentially thousands of sensors and long temporal horizons. For instance, in nation-wide traffic forecasting, the sensor network may involve thousands of detectors (posing a scalability challenge), while individual stations frequently experience outages or entire districts remain uncovered (posing a sparsity challenge). Models must therefore reconcile these two aspects, propagating information efficiently across large graphs while remaining robust to the limited availability of local observations.

1.3 Contributions

This doctoral research advances the state of the art in graph deep learning for irregular spatiotemporal data through a comprehensive set of methodological and theoretical contributions. The research spans the presented research lines and addresses the core challenges of irregularity, sparsity, and scalability. The main contributions are summarized below:

R1.1. Imputation of irregular spatiotemporal data

This contribution introduces a novel graph-based imputation framework [Cini, Marisca, et al., 2022] that leverages graph neural networks embedded within a bidirectional recurrent architecture. This enables spatially-coherent, structure-aware reconstructions of partially observed data, significantly outperforming previous methods across multiple real-world benchmarks.

R1.2. Imputation with sparse observations

To address error propagation in autoregressive models under extreme sparsity, this contribution proposes an attention-based imputation ar-

chitecture [Marisca et al., 2022] specifically tailored for scattered inputs. The model formulates imputation as a point-wise inference task using sparse spatiotemporal attention, allowing robust predictions even when observations are few and unevenly distributed across time and space. Together with R1.1, these contributions provide a comprehensive toolbox for graph-based imputation across varying regimes.

R2.1. Prediction under irregular spatial sampling

This contribution spans several works and establishes a comprehensive methodological framework for graph-based time series forecasting [Cini, Marisca, et al., 2023a; Cini, Marisca, et al., 2023b; Cini, Marisca, et al., 2025], hence addressing domains with regular temporal sampling but spatial irregularities. The framework systematically analyzes core design principles, such as globality and locality, and proposes architectural elements to balance generalization with task-specific accuracy [Cini, Marisca, et al., 2023b]. Moreover, it tackles the scalability bottleneck that hinders practical deployment in real-world systems by proposing lightweight predictors capable of maintaining predictive accuracy under constrained resources and large graph sizes [Cini, Marisca, et al., 2023a].

R2.2. Prediction under irregular spatiotemporal sampling

Extending forecasting to scenarios with both spatial and temporal irregularity, this contribution introduces a hierarchical downsampling framework for forecasting with missing data [Marisca et al., 2024]. The proposed architecture extracts multi-scale representations and combines them adaptively conditioned on the missing data pattern. The resulting model achieves state-of-the-art accuracy in real-world applications across different data sparsity regimes.

R3.1. Spatiotemporal over-squashing

To address representational bottlenecks that limit long-range dependency modeling in GNNs, this contribution defines and studies *spatiotemporal over-squashing* [Marisca et al., 2025], extending the notion of topological bottlenecks to dynamic settings. A theoretical analysis reveals how architectural choices and graph topology influence information propagation in spatiotemporal architectures. The work proposes mitigation strategies based on temporal rewiring, backed by both synthetic and real-world evidence.

These contributions collectively advance the field of spatiotemporal graph learning by addressing the identified fundamental challenges. In addition to

the methodological advancements, the research led to impactful outcomes for industry and practitioners.

Industrial application This research contributed to the Innosuisse project “GraPV – Graph-based Predictors for Photovoltaic Power Forecasting,” conducted in collaboration with DXT Commodities SA. The project focused on developing a graph-based forecasting approach for day-ahead photovoltaic power production across a distributed portfolio of solar plants. Building on the methodological advances introduced in this thesis, the project delivered a model achieving remarkable predictive performance. This success led to the founding of GraphSight SA, a startup focused on AI-driven forecasting solutions for renewable energy. This outcome highlights how fundamental research in graph-based spatiotemporal modeling can translate into impactful, industry-ready applications.

Software Beyond methodological contributions, my research has actively supported open science by releasing datasets and software tailored to irregular spatiotemporal data. These resources address critical challenges in benchmarking machine learning models on dynamic, non-Euclidean domains. In particular, I contributed to the development and publication of two real-world weather datasets, EngRAD [Marisca et al., 2024] and PeakWeather [Zambon, Cattaneo, Marisca, et al., 2025], both described in detail in Section 6.4.2. These datasets provide standardized benchmarks for evaluating models under spatiotemporal irregularity. I am also the co-creator of Torch Spatiotemporal [Cini and Marisca, 2022], an open-source Python library for deep learning on spatiotemporal data. Torch Spatiotemporal offers a collection of ready-to-use models, along with flexible modules for building new ones, and has been widely adopted by researchers and practitioners in the community. More recently, I co-released Torch Geometric Pool [Bianchi and Marisca, 2025], a library that provides a suite of graph pooling operators compatible with Pytorch Geometric [Fey and Lenssen, 2019]. Torch Geometric Pool extends the work presented in [Marisca et al., 2024] and has broader applications also in static domains, further advancing the tools available for graph-based learning.

The contributions of this research extend beyond academic publications, demonstrating tangible industrial and societal impact. The developed methods have supported the deployment of AI-driven solutions in real-world applications, contributed to improved benchmarking practices for irregular spatiotemporal data, and fostered innovation through the release of open-source tools.

1.4 Publications and dissemination

During my doctoral studies, I co-authored 8 papers. Most of them have been published in top-tier venues of the field, including the ACM journal *Computing Surveys* [Cini, Marisca, et al., 2025] and conferences such as NeurIPS [Marisca et al., 2022; Cini, Marisca, et al., 2023b], ICML [Marisca et al., 2024], ICLR [Cini, Marisca, et al., 2022], and AAI [Cini, Marisca, et al., 2023a]. As part of my dissemination efforts, I co-organized and delivered tutorials on graph deep learning for time series forecasting at major venues, including ECML PKDD 2023 and the Learning on Graphs (LoG) conference in 2024. My work has been featured in invited talks at both academic and industrial venues, such as the Learning on Graphs and Geometry (LoG2) seminar series at the University of Oxford. Notably, the work on scalable spatiotemporal graph neural networks [Cini, Marisca, et al., 2023a] received the Best Paper Award at the NeurIPS 2022 Temporal Graph Learning Workshop, the leading event in this area. Below is a list of all publications and preprints currently under review.

Journal papers

Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi (June 2025). **Graph Deep Learning for Time Series Forecasting**. In: *ACM Computing Surveys* 57.12. ISSN: 0360-0300.

Conference papers

In the following list of publications, I was a co-first author on all papers where I am listed second.

Andrea Cini, Ivan Marisca, and Cesare Alippi (Jan. 2022). **Filling the Gaps: Multivariate Time Series Imputation by Graph Neural Networks**. In: *International Conference on Learning Representations*.

Ivan Marisca, Andrea Cini, and Cesare Alippi (2022). **Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations**. In: *Advances in Neural Information Processing Systems*. Vol. 35, pp. 32069–32082.

Andrea Cini, Ivan Marisca, Filippo Maria Bianchi, and Cesare Alippi (June 2023a). **Scalable Spatiotemporal Graph Neural Networks**. In: *Proceedings of the 37th AAAI Conference on Artificial Intelligence* 37.6, pp. 7218–7226.

Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi (2023b). **Taming Local Effects in Graph-based Spatiotemporal Forecasting**. In: *Advances in Neural Information Processing Systems*. Vol. 36, pp. 55375–55393.

Ivan Marisca, Cesare Alippi, and Filippo Maria Bianchi (2024). **Graph-based Forecasting with Missing Data through Spatiotemporal Downsampling**. In: *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. Proceedings of Machine Learning Research, pp. 34846–34865.

Preprints

Ivan Marisca, Jacob Bamberger, Cesare Alippi, and Michael M. Bronstein (2025). **Over-squashing in Spatiotemporal Graph Neural Networks**. In: *arXiv preprint arXiv:2506.15507*.

Daniele Zambon, Michele Cattaneo, Ivan Marisca, Jonas Bhend, Daniele Nerini, and Cesare Alippi (2025). **PeakWeather: MeteoSwiss Weather Station Measurements for Spatiotemporal Deep Learning**. In: *arXiv preprint arXiv:2506.13652*.

1.5 Thesis outline

The remainder of this thesis is organized as follows.

Chapter 2 formalizes the problem of learning from irregular spatiotemporal data. It introduces the tasks of imputation, filtering, and prediction, and reviews relevant background in graph and sequence modeling. It concludes with a discussion of commonly used datasets, including those contributed as part of this thesis.

Chapter 3 addresses missing data imputation in irregular spatiotemporal data (**R1**). It presents a bidirectional spatiotemporal architecture for imputation that integrates message passing within a recurrent network [Cini, Marisca, et al., 2022] (**R1.1**). We show that introducing relational inductive biases leads to state-of-the-art imputation accuracy and provides smoothed reconstructions as a natural byproduct. **Chapter 4** tackles the more challenging regime of extremely sparse inputs (**R1.2**), where autoregressive imputation becomes unreliable. It presents a non-autoregressive attention-based model that directly estimates missing values from available context [Marisca et al., 2022].

Chapter 5 shifts the focus to prediction (**R2**), under the assumption of regular

temporal sampling but spatial irregularity (**R2.1**). The chapter first formalizes the spatiotemporal forecasting problem for graph-structured time series and compares different processing paradigms. It then presents two main contributions: (i) a global-local prediction framework using trainable node embeddings to balance generalization and specialization [Cini, Marisca, et al., 2023b]; and (ii) a scalable graph-based predictor [Cini, Marisca, et al., 2023a] for efficient large-scale forecasting. **Chapter 6** addresses the most general setting of prediction under both spatial and temporal irregularities (**R2.2**), introducing a hierarchical spatiotemporal downsampling strategy to deal with varying patterns of data availability [Marisca et al., 2024]. **Chapter 7** investigates the theoretical limits of information propagation in spatiotemporal models (**R3**). It introduces the concept of spatiotemporal over-squashing [Marisca et al., 2025], generalizing known bottlenecks in GNNs to dynamic settings and proposing architectural remedies (**R3.1**).

Chapter 8 concludes the thesis by summarizing the main contributions, discussing the key insights and limitations, and outlining several promising research directions.

Chapter 2

Problem formalization and preliminaries

2.1 Problem setting

While the field of spatiotemporal data analysis is vast and encompasses several data representations following diverse sampling modalities, we consider the setting where data are sampled irregularly over time from entities scattered over space. This setting is also referred to as *raster data with irregular sampling* in the literature [Atluri et al., 2018]. Throughout this thesis, we call data acquired under this regime *irregular spatiotemporal data* and *nodes* (or *sensors*) the entities generating the data.

We denote by $\mathcal{V}$ the set of N data-generating entities, with matrix $\mathbf{V} \in \mathbb{R}^{N \times d_v}$ representing their static attributes. Each row $\mathbf{v}^i = \mathbf{V}[i]$ encodes information about the i -th node in $\mathcal{V}$, such as its location or type in the context of physical sensor networks. We use capital letters to refer to set-level attributes (e.g., $\mathbf{V}$) and lowercase letters for node-specific quantities (e.g., $\mathbf{v}^i$). We denote by $\mathbf{x}_t^i \in \mathbb{R}^{d_x}$ the d_x -dimensional observation at time step t for the i -th node. We assume multivariate time series at each node (i.e., $d_x > 1$), with all nodes measuring the same set of variables to ensure channel homogeneity across nodes.

To unify different sampling rates, we define a common temporal resolution Δt interleaving between any two $\mathbf{x}_t^i$ and $\mathbf{x}_{t+1}^j$ as the greatest common divisor that factorizes all sampling intervals across collected data. Temporal irregularities are modeled by allowing $\mathbf{x}_t^i$ to be missing for some i and t , due to reasons

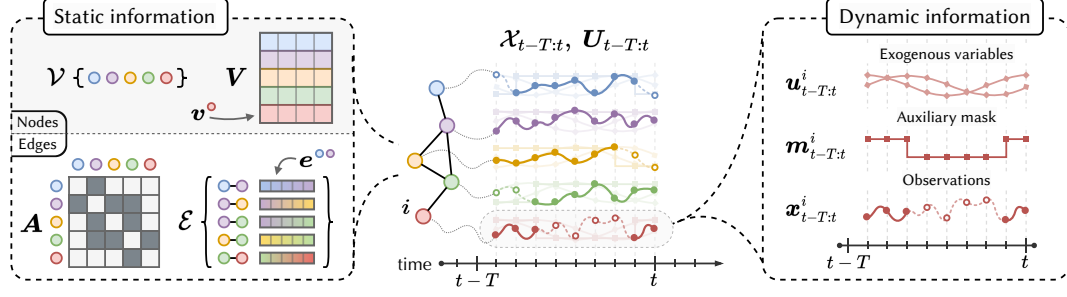


Figure 2.1. Representation of irregular spatiotemporal data with a distinction between static (left) and dynamic (right) information.

such as sensor asynchrony or acquisition errors. This modeling choice recasts the problem as regularly sampled data aligned on a fixed temporal grid with resolution Δt , where some values may be missing. At each time step t , the observation set $\mathcal{X}_t = \{\mathbf{x}_t^i\}_i$ contains all available node-level observations.

To facilitate alignment with representations used in regularly sampled settings, we adopt a matrix-based notation for node observations. At each time step t , we define $\mathbf{X}_t \in \mathbb{R}^{N \times d_x}$ as the matrix obtained by stacking the observation vectors $\mathbf{x}_t^i$ across all nodes, such that $\mathbf{X}_t[i] = \mathbf{x}_t^i$ if the observation is available ($\mathbf{x}_t^i \in \mathcal{X}_t$) and $\mathbf{X}_t[i] = \mathbf{0}$ otherwise. In addition, we introduce an auxiliary mask $\mathbf{M}_t \in \{0, 1\}^{N \times d_x}$ where zero elements denote a missing observation at the corresponding position in $\mathbf{X}_t$. Finally, we denote by $\mathbf{u}_t^i \in \mathbb{R}^{d_u}$ the exogenous information associated with node i at time t . We assume this information is always available – even when $\mathbf{x}_t^i$ is missing – and may include features such as acquisition timestamps or environmental conditions. Throughout the text, we use the subscript $\cdot_{t-T:t}$ to denote sequences within the time interval $(t - T, t]$ (e.g., $\mathbf{X}_{t-T:t}$), and $\cdot_{<t}$ ($\cdot_{\leq t}$) for sequences encompassing all time steps up to (and including) t .

Temporal dependencies We express temporal dependencies existing among observations acquired in the interval $(t - T, t]$ with a *temporal graph* $\mathbf{T} \in \{0, 1\}^{T \times T}$. A nonzero entry $\mathbf{T}[i, j] = 1$ represents a directed edge from time step $t - i$ to $t - j$, and indicates that information from $t - i$ contributes to the representation at $t - j$. In most cases, $\mathbf{T}$ is defined as a directed path graph over the sequence, such that $\mathbf{T}[i, j] = 1$ if and only if $i - j = 1$. Under this convention, $\mathbf{T}$ acts as a *backward shift operator* [Box et al., 2015], shifting temporal signals forward by one step. When this is the case, we omit $\mathbf{T}$ and instead assume temporal structure is induced by the natural ordering of the sequence.

2.1.1 Relational dependencies

We assume the existence of dyadic – possibly asymmetric – relational dependencies between sensors, modeled as a graph with attributed edges. These relationships are interpreted in the sense of predictive (Granger) causality [Granger, 1969]: past information from neighboring nodes can help improve predictive performance for a given node. The graph structure also serves as a sparsification prior, suppressing spurious correlations and emphasizing only the most informative dependencies across nodes. We refer to this relational structure as *spatial dependencies*, where the term *spatial* does not necessarily imply physical distance but rather any domain-specific notion of proximity or affinity encoded in the graph. In this sense, the node dimension plays the role of the spatial dimension.

Formally, we define the edge set as $\mathcal{E} = \{(i, j, \mathbf{e}^{ij}) \mid (i, j) \subseteq \mathcal{V} \times \mathcal{V}\}$, where each triplet represents a directed edge from node i to node j , and $\mathbf{e}^{ij} \in \mathbb{R}^{d_e}$ denotes the attributes associated with that edge. This graph structure can also be captured by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, where the entry $a^{ij} = \mathbf{A}[i, j]$ is either a binary indicator or a weighted value (potentially derived from $\mathbf{e}^{ij}$), and is nonzero if and only if $(i, j, \mathbf{e}^{ij}) \in \mathcal{E}$.

Figure 2.1 illustrates the key elements introduced above, and Example 1 complements this with a practical example.

Learning the adjacency matrix When explicit relational information is unavailable, several strategies can be employed to infer the adjacency matrix $\mathbf{A}$. A simple option is to compute pairwise similarity scores – such as Pearson correlation – between node observations and construct edges based on these affinities. This is often done when no side information is available or when computational resources are limited and prevent more advanced learning approaches. Alternatively, the adjacency can be derived from static node features $\mathbf{V}$, e.g., using distances or other similarity functions. More appealing techniques in the field of graph structure learning aim to obtain an adjacency matrix directly from data during model training [Dong et al., 2016; Dong et al., 2019; Cini et al., 2023c; Manenti et al., 2025]. A common and straightforward approach is to leverage spatial attention scores computed between each pair of nodes as

$$a^{ij} = \mathbf{q}^{i\top} \mathbf{k}^j, \quad (2.1)$$

where $\mathbf{q}^i \in \mathbb{R}^d$ and $\mathbf{k}^j \in \mathbb{R}^d$ are the source and target learnable node embeddings associated with the i -th and j -th node, respectively [Wu et al., 2019].

Example 1. Network of weather stations

Consider a collection of weather stations measuring wind across a geographic region. Each station is modeled as a node in $\mathcal{V}$, with its attributes v^i capturing static information such as geographic location or sensor aging. Spatial relationships between stations can be defined in multiple ways. As an example, an undirected edge can be added between nodes i and j in $\mathcal{E}$ if their physical distance falls below a specified threshold. This distance can be encoded as an edge attribute in e^{ij} . At each time step t , station i records the wind's strength and direction as x_t^i . If a reading is unavailable (e.g., due to a sensor malfunction or transmission failure), we set the corresponding mask $m_t^i = 0$. Additional weather information observed from the satellite, such as cloud movements or precipitation, can be treated as exogenous inputs and represented by u_t^i .

2.1.2 Changes in topology and sensor heterogeneity

The reference setting outlined above assumes a static topology and homogeneous sensor types. While these assumptions simplify modeling and reflect many practical applications, real-world data often exhibit more complex dynamics. Nonetheless, the framework can be extended to accommodate these additional challenges.

Dynamic graphs In many scenarios, the underlying set of sensors, as well as spatial dependencies, can evolve over time. For instance, in a transportation network, a new sensor can be added or a route might be deviated, leading to a time-varying graph topology. The dynamic nature might also arise when the dependencies are inferred from data, which are evolving by definition. We introduce $\mathbf{A}_t$ to denote the adjacency matrix at time t , with $\mathcal{E}_t$ the corresponding edge set. These structures can be modeled as sequences as well, i.e., $\mathbf{A}_{t-T:t}$ or $\mathcal{E}_{t-T:t}$, and given as input to the model. In our setting, however, we assume such evolution to be relatively slow, so that $\mathbf{A}_t$ can be treated as fixed within short intervals. Note that this assumption is realistic in most considered applications, where abrupt changes do not appear frequently.

Sensor heterogeneity The assumption that all sensors measure the same set of variables is not restrictive. When sensors differ in the variables they acquire (e.g., temperature, wind, or pollution levels), this heterogeneity can

be encoded explicitly by extending the feature space. Each measurement channel can be paired with a semantic type, and missing channels can be treated analogously to missing values using the auxiliary mask $\mathbf{m}_t^i$. Static node attributes $\mathbf{v}^i$ can also include sensor-specific metadata (e.g., type, measured channels, scaling factors) and be given to the model as part of the input. This enables architectures to handle heterogeneous collections of sensors without modifying their core processing mechanisms. When needed, more advanced and specialized approaches for heterogeneous data can be employed to better leverage inter-modality relations [Deng et al., 2018].

2.1.3 Missing data

Following prior work [Yi et al., 2016; Cini, Marisca, et al., 2022], we classify missing data patterns based on the conditional distribution $p(\mathbf{m}_t^i | \mathbf{M}_{\leq t})$. Note that this taxonomy differs from – and is orthogonal to – the classification in missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR) common in the literature [Rubin, 1976], which instead concerns the mechanisms that lead to missing values and is based on the conditional distribution $p(\mathbf{m}_t^i | \mathbf{x}_t^i)$. Throughout the thesis, we assume stationarity of the missing data distribution and MAR mechanism.

We call *point missing* (**Point**) the setting where the probability of a datum being missing is constant and independent across nodes and time steps. In this case, the mask follows an i.i.d. Bernoulli distribution:

$$p(\mathbf{m}_t^i) = \mathcal{B}(1 - \eta) \quad \forall t, \forall i \in \mathcal{V} \quad (2.2)$$

where $1 - \eta$ is the Bernoulli mean. This type of pattern is also referred to as *general missing* [Rubin, 1976], and realizations are spatially and temporally uncorrelated.

In many real-world scenarios, however, missing data exhibits structure and dependencies. For instance, hardware faults or power outages in a physical sensor network often affect multiple time steps or adjacent sensors. We refer to these structured patterns as *block missing*, which we further categorize by their dependency structure. In *temporal block missing* (**Block-T**), the probability of a value being missing depends on whether it was missing at the previous time step:

$$p(\mathbf{m}_t^i | \mathbf{m}_{t-1}^i) \neq p(\mathbf{m}_t^i). \quad (2.3)$$

In *spatial block missing* (**Block-S**), $p(\mathbf{m}_t^i)$ is instead influenced by simultaneous missing values at neighboring nodes:

$$p(\mathbf{m}_t^i \mid \{\mathbf{m}_t^j \mid j \in \mathcal{N}(i)\}) \neq p(\mathbf{m}_t^i), \quad (2.4)$$

where the neighborhood $\mathcal{N}(i) = \{j \mid (j, i, e^{ji}) \in \mathcal{E}\}$ contains all incoming neighbors of the i -th node. The most general case, *spatiotemporal block missing* (**Block-ST**), combines both forms:

$$p(\mathbf{m}_t^i \mid \mathbf{m}_{t-1}^i, \{\mathbf{m}_t^j \mid j \in \mathcal{N}(i)\}) \neq p(\mathbf{m}_t^i). \quad (2.5)$$

Note that in [Equations 2.2 to 2.5](#) we assume a univariate setting with $d_x = 1$. In the multivariate case ($d_x > 1$), we consider the missing data distribution of each channel to be independent from the others.

2.2 Imputation, filtering, and prediction

We consider a collection of T observation sets $\mathcal{X}_{t-T:t}$, paired with corresponding exogenous variables $\mathbf{U}_{t-T:t}$ and node-level attributes $\mathbf{V}$, as introduced in [Section 2.1](#). Given this context, we formalize the problems of imputation, filtering, and prediction within a unified probabilistic framework. Each task is framed as a distinct instance of conditional estimation over a partially observed spatiotemporal process. Although they share a common formulation, they differ in both objective and assumptions.

Imputation and filtering operate within a bounded temporal window, focusing on reconstructing missing or noisy values using surrounding observations. In contrast, prediction is a forward-looking task that aims to infer future values based on past observations. We begin by formalizing prediction as the natural estimation problem associated with the underlying data-generating process. We then analyze imputation and filtering, which are more tightly aligned with the challenges introduced by irregular sampling and often serve as preprocessing steps in spatiotemporal modeling pipelines.

2.2.1 Prediction

We assume that the i -th node observations are generated by a time-invariant spatiotemporal stochastic process p^i such that

$$\mathbf{x}_{t+1}^i \sim p^i(\mathbf{x}_{t+1}^i \mid \mathbf{X}_{\leq t}, \mathbf{U}_{\leq t+1}, \mathbf{V}) \quad i = 0, \dots, N-1. \quad (2.6)$$

We do not assume that all time series in the collection are generated by the same process; in general, $p^i \neq p^j$ for $i \neq j$, although time invariance holds for all nodes.

The goal of **prediction** is to forecast future values based on the available past information. Given a forecasting horizon $h \in \mathbb{N}^+$, the objective is to learn a model p_θ that approximates the h -step-ahead predictive distribution for each i -th node in $\mathcal{V}$, i.e.,

$$p_\theta(\mathbf{x}_{t+h}^i \mid \mathcal{X}_{t-T:t}, \mathbf{U}_{t-T:t+h}, \mathbf{V}) \approx p^i(\mathbf{x}_{t+h}^i \mid \mathbf{X}_{\leq t}, \mathbf{U}_{\leq t+h}, \mathbf{V}) \quad (2.7)$$

where θ denotes the set of learnable model parameters. Note that prediction can only condition on actually observed data $\mathcal{X}_{\leq t}$, not on the full latent state $\mathbf{X}_{\leq t}$, which may contain missing entries. Moreover, exogenous variables for future time steps $\mathbf{U}_{t:t+h}$ may be available only in some cases, e.g., for encoding of timestamps or simulated quantities; for all other cases, we consider data up to time t included only. The conditional distribution above acts as a *predictive prior* for the unknown future value $\mathbf{x}_{t+h}^i$. As such, it incorporates not only uncertainty due to irregular sampling but also the additional uncertainty introduced by extrapolating forward in time.

In practice, we approximate the predictive distribution using a point estimate. To this end, we define a family of predictive models $f(\cdot)$ that estimate the conditional expectation of future observations:

$$\begin{aligned} \hat{\mathbf{x}}_{t+h}^i &= f(\mathcal{X}_{t-T:t}, \mathbf{U}_{t-T:t+h}, \mathbf{V}; \theta, i) \\ &\approx \mathbb{E}_p[p^i(\mathbf{x}_{t+h}^i \mid \mathbf{X}_{\leq t}, \mathbf{U}_{\leq t+h}, \mathbf{V})], \end{aligned} \quad (2.8)$$

where θ denotes the model parameters learned by minimizing a loss function and index i indicates the target node. We consider multi-step prediction and a pairwise loss function ℓ averaged over all available observations in the forecasting horizon $[t+1, t+H]$. The loss is weighted using the mask to ignore missing values:

$$\mathcal{L}(\widehat{\mathbf{X}}_{t:t+H}, \mathbf{X}_{t:t+H}, \mathbf{M}_{t:t+H}) = \sum_{h=1}^H \sum_{i=0}^{N-1} \frac{\|\mathbf{m}_{t+h}^i \odot \ell(\hat{\mathbf{x}}_{t+h}^i, \mathbf{x}_{t+h}^i)\|_1}{\|\mathbf{M}_{t:t+H}\|_1}, \quad (2.9)$$

where $\odot$ is the element-wise (Hadamard) product and $\|\cdot\|_1$ is the L^1 -norm. For matrices and tensors, we consider the entry-wise norm, such that $\|\mathbf{M}_{t:t+H}\|_1 = \|\text{vec}(\mathbf{M}_{t:t+H})\|_1 = \sum_{h=1}^H \sum_{i=0}^{N-1} \|\mathbf{m}_{t+h}^i\|_1$ is the total number of observed values in the prediction interval. Parameters θ are then obtained through loss

minimization:

$$\hat{\theta} = \arg \min_{\theta} \mathcal{L} \left(\widehat{\mathbf{X}}_{t:t+H}, \mathbf{X}_{t:t+H}, \mathbf{M}_{t:t+H} \right). \quad (2.10)$$

Typical choices for the loss function ℓ include the mean squared error (MSE):

$$\ell(\hat{\mathbf{x}}_t^i, \mathbf{x}_t^i) = \|\hat{\mathbf{x}}_t^i - \mathbf{x}_t^i\|_2^2, \quad (2.11)$$

and the mean absolute error (MAE):

$$\ell(\hat{\mathbf{x}}_t^i, \mathbf{x}_t^i) = \|\hat{\mathbf{x}}_t^i - \mathbf{x}_t^i\|_1. \quad (2.12)$$

2.2.2 Imputation and filtering

Unlike prediction, which aims to forecast future values, imputation and filtering focus on reconstructing missing or noisy observations from the past, i.e., estimating $\mathbf{x}_{t-k}^i$ using partial information available up to time t . In this setting, the goal is not to model the generative process, but to refine data quality by conditioning on observed evidence. Formally, for any $k \in \mathbb{N}_0$, this corresponds to:

$$\mathbf{x}_{t-k}^i \sim p^i(\mathbf{x}_{t-k}^i \mid \mathcal{X}_{\leq t}, \mathbf{U}_{\leq t}, \mathbf{V}) \quad i = 0, \dots, N-1. \quad (2.13)$$

This is a *posterior distribution* over a past value, updated using information that includes future observations relative to the time of interest $t-k$. Importantly, when using future information $\mathcal{X}_{>t-k}$ to constrain a past estimate $\mathbf{x}_{t-k}^i$, we still assume the data-generating process to be causal. Still, once future values are observed, they can be leveraged for refining past missing data.

As with forecasting, we approximate this probability distribution using point estimates. In our setting, **imputation** refers to the reconstruction of missing observations in the past, obtained by computing the conditional expectation under the posterior distribution:

$$\begin{aligned} \bar{\mathbf{x}}_{t-k}^i &= f(\mathcal{X}_{t-T:t}, \mathbf{U}_{t-T:t}, \mathbf{V}; \theta, i) \quad \text{for } m_{t-k}^i = 0 \\ &\approx \mathbb{E}_p[p^i(\mathbf{x}_{t-k}^i \mid \mathcal{X}_{\leq t}, \mathbf{U}_{\leq t}, \mathbf{V})]. \end{aligned} \quad (2.14)$$

Conversely, **filtering** targets observed values with the goal of refining potentially noisy or uncertain measurements. The idea is to incorporate both past and future information to produce a denoised estimate:

$$\begin{aligned} \tilde{\mathbf{x}}_{t-k}^i &= f(\mathcal{X}_{t-T:t}, \mathbf{U}_{t-T:t}, \mathbf{V}; \theta, i) \quad \text{for } m_{t-k}^i = 1 \\ &\approx \mathbb{E}_p[p^i(\mathbf{x}_{t-k}^i \mid \mathcal{X}_{\leq t}, \mathbf{U}_{\leq t}, \mathbf{V})]. \end{aligned} \quad (2.15)$$

The output $\tilde{x}_{t-k}^i$ is a filtered version of the original input, estimated under the same distribution used for imputation. Both tasks share the same formulation but differ in their targets: imputation estimates values that were missing, while filtering refines values that were observed. This distinction is also reflected in the loss functions used for training and evaluation. In imputation, we simulate missing values by artificially masking entries, then use them as ground-truth labels in the loss:

$$\mathcal{L}(\bar{\mathbf{X}}_{t-T:t}, \mathbf{X}_{t-T:t}, \mathbf{M}_{t-T:t}) = \sum_{k=0}^{T-1} \sum_{i=0}^{N-1} \frac{\|\bar{\mathbf{m}}_{t-k}^i \odot \ell(\bar{\mathbf{x}}_{t-k}^i, \mathbf{x}_{t-k}^i)\|_1}{\|\bar{\mathbf{M}}_{t-T:t}\|_1}, \quad (2.16)$$

where $\bar{\mathbf{m}}_t^i = 1$ flags observation $\mathbf{x}_t^i$, which is held out from the input but retained for the loss computation. Note that this surrogate objective is adopted to overcome the unavailability of the actual target value $\mathbf{x}_t^i$ for each $\mathbf{m}_t^i = 0$. For filtering, the model is trained to reconstruct all available values:

$$\mathcal{L}(\tilde{\mathbf{X}}_{t-T:t}, \mathbf{X}_{t-T:t}, \mathbf{M}_{t-T:t}) = \sum_{k=0}^{T-1} \sum_{i=0}^{N-1} \frac{\|\mathbf{m}_{t-k}^i \odot \ell(\tilde{\mathbf{x}}_{t-k}^i, \mathbf{x}_{t-k}^i)\|_1}{\|\mathbf{M}_{t-T:t}\|_1}. \quad (2.17)$$

In most practical pipelines, imputation and filtering are used as a preliminary step to improve data quality before performing downstream tasks such as prediction, as illustrated in [Figure 1.2 on Page 4](#). In this thesis, we mainly focus on imputation and prediction, as they address the core challenges introduced by irregular spatiotemporal sampling and represent the most relevant problems in real-world applications.

2.3 Graph neural networks

Graph neural networks (GNNs) are a class of neural architectures tailored for processing graph-structured data [[Gori et al., 2005](#); [Scarselli et al., 2008](#); [Defferrard et al., 2016](#); [Bronstein et al., 2021](#)]. Their core mechanism involves iteratively updating each node’s representation by aggregating information from neighboring nodes. Over the past decade, GNNs have become the state-of-the-art approach for relational deep learning, achieving outstanding results in diverse domains, including drug discovery [[Stokes et al., 2020](#); [Liu et al., 2023a](#)], materials synthesis [[Merchant et al., 2023](#); [Szymanski et al., 2023](#)], and human behavior analysis [[Monti et al., 2019](#); [Corrias et al., 2023](#)]. Among the different GNNs variants, message-passing neural networks (MPNNs) – introduced by Gilmer et al. [[2017](#)] – represent the most widely adopted family.

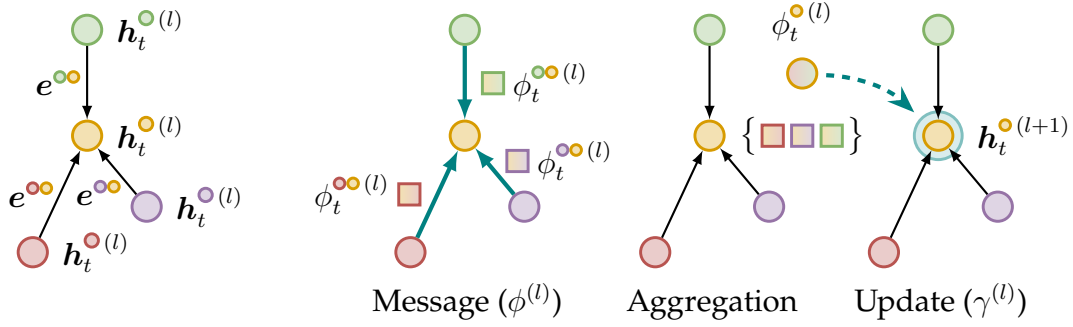


Figure 2.2. The message-passing framework. Node features $\mathbf{h}_t^{(l+1)}$ of node $\bullet$ at layer $l + 1$ are computed by (1) sending messages $\phi_t^{\bullet(l)}$ from each neighbor $\bullet$ in $\{\bullet, \bullet, \bullet\}$, (2) aggregating all messages in representation $\phi_t^{\bullet(l)}$, and (3) using it along with initial node features $\mathbf{h}_t^{(l)}$ within the update function $\gamma^{(l)}$.

Consider a graph with static node features $\mathbf{H}^{(0)} \in \mathbb{R}^{N \times d}$ and edge set $\mathcal{E}$, as introduced in Section 2.1. In an MPNN, the i -th node representation at layer l , denoted by $\mathbf{h}^{i(l)}$, is updated through the message passing (MP) function as

$$\mathbf{h}^{i(l+1)} = \gamma^{(l)} \left(\mathbf{h}^{i(l)}, \text{AGGR}_{j \in \mathcal{N}(i)} \{ \phi^{(l)}(\mathbf{h}^{i(l)}, \mathbf{h}^{j(l)}, \mathbf{e}^{ji}) \} \right), \quad (2.18)$$

where $\phi^{(l)}$ and $\gamma^{(l)}$ are learnable differentiable functions – typically multilayer perceptrons (MLPs) – called the *message* and *update* functions, respectively. The *aggregation* function $\text{AGGR} \{ \cdot \}$ is permutation-invariant (e.g., mean or sum), ensuring invariance to the ordering of neighbors. A visual illustration of this process is provided in Figure 2.2. We use MP to indicate the message-passing layer acting concurrently over all the nodes, i.e.,

$$\mathbf{H}^{(l+1)} = \text{MP}(\mathbf{H}^{(l)}, \mathcal{E}). \quad (2.19)$$

Notably, most existing GNNs can be written by following the message-passing paradigm, including GCN [Kipf and Welling, 2017], DCNN [Atwood and Towsley, 2016], GAT [Veličković et al., 2018], GatedGN [Bresson and Laurent, 2017]. A key characteristic of GNNs is their *inductive* nature [Hamilton et al., 2017; Ruiz et al., 2020], which enables them to generalize to unseen and variable-sized graphs. This is made possible by the parameter-sharing mechanism across nodes, with a node representation being localized by aggregating information from its local neighborhood. By increasing the depth of a GNN, i.e., stacking more layers, the model gains access to increasingly broader context within the graph. However, excessive message passing can lead to issues such as

representational rank collapse or the distortion of distant node information. These challenges are further discussed in the next subsection.

2.3.1 Over-smoothing, over-squashing, and under-reaching

As graph neural networks grow deeper, they may suffer from two phenomena that fundamentally limit their expressive power, named *over-smoothing* and *over-squashing*. Conversely, *under-reaching* affects representation learning in shallow architectures. In the following, we formally introduce these issues and discuss their underlying mechanisms and potential remedies.

The repeated aggregation of neighborhood information across GNN layers leads to the convergence of node representations, which become nearly indistinguishable – a phenomenon known as **over-smoothing** [Li et al., 2018a; Oono and Suzuki, 2020; Rusch et al., 2023]. From a spectral perspective, stacking isotropic graph filters repeatedly applies a low-pass operator that exponentially contracts the feature space, collapsing the rank of the node-feature matrix. This collapse can be formally characterized using similarity metrics, which tend to converge toward a constant vector shared across all nodes [Rusch et al., 2023]. A common way to quantify over-smoothing is through the Dirichlet energy E_D on attributed graphs:

$$E_D(\mathbf{H}^{(l)}) = \sum_{(i,j,e^{ij}) \in \mathcal{E}} \|\mathbf{h}^{i(l)} - \mathbf{h}^{j(l)}\|_2^2, \quad (2.20)$$

which decreases when the network’s depth increases in standard GNNs. Several strategies have been proposed to mitigate over-smoothing. One line of work focuses on anisotropic message passing, which alleviates the low-pass behavior of standard graph convolutions (see Section 2.3.2). Other approaches include architectural modifications such as residual or skip connections that reintroduce initial features [Li et al., 2023; Chen et al., 2020b], stochastic edge dropping to slow diffusion [Rong et al., 2020], and delay mechanisms in the messages [Gutteridge et al., 2023]. Each can be interpreted as either modifying the spectrum of the message-passing matrix, constraining contraction along leading eigenvectors, or re-injecting high-frequency components to maintain discernibility.

Unlike over-smoothing, **over-squashing** occurs when information from exponentially many distant nodes is compressed into fixed-size messages that must traverse the graph [Alon and Yahav, 2021]. If the average node degree is $D > 1$, then $\Theta(D^k)$ feature vectors must be summarized into a single embedding after k layers. Alon and Yahav [2021] formalized this via the *problem radius*

r , i.e., the minimum number of layers L required to solve the task, showing that when long-range interactions are needed ($r \gg 1$), standard GNNs suffer from a *bottleneck* that might yield information loss even at sufficient depths ($L \geq r$). Recent work has linked over-squashing to geometric properties of the graph, such as curvature and effective resistance: negatively curved edges produce large distortions, provably causing over-squashing [Topping et al., 2022; Di Giovanni et al., 2023a]. To address this challenge, two primary strategies have emerged. The first involves graph re-wiring, i.e., modifying the original topology by removing or adding edges, to break structural bottlenecks and facilitate smoother information flow [Gasteiger et al., 2019; Topping et al., 2022; Gutteridge et al., 2023; Barbero et al., 2024b]. Notably, SDRF [Topping et al., 2022] is a curvature-aware rewiring method that increases the balanced Forman/Ollivier curvature along problematic regions. A key nuance is the trade-off between over-squashing and over-smoothing: rewiring the graph can also enlarge the spectral gap, potentially accelerating smoothing. Successful rewiring balances these effects and uses multi-graph message passing to preserve locality given by the original graph while adding depth [Karhadkar et al., 2023].

The second mitigation strategy focuses on architectural design, incorporating constraints on the learned weights to better regulate information propagation and retention across layers [Chen et al., 2020a; Tortorella and Micheli, 2022; Gravina et al., 2023; Gravina et al., 2025]. Notably, Gravina et al. [2023] propose anti-symmetric normalization of the weights, as well as of the propagation operator [Gravina et al., 2025], to obtain non-dissipativity across layers, thereby counteracting over-squashing without altering the graph topology. Similarly, Tortorella and Micheli [2022] rely on reservoir computing to address the exponentially vanishing sensitivity by bounding the spectral radius of the GNN’s weight matrix. These architectural approaches complement rewiring methods by acting on the propagation dynamics of message passing itself, rather than modifying the topology.

In contrast to over-smoothing and over-squashing, **under-reaching** arises when a GNN is too shallow to enable information exchange between nodes that are sufficiently far apart [Alon and Yahav, 2021]. An L -layer GNN can only propagate information up to L hops, so if a task’s radius $r > L$, relevant features simply cannot reach their targets. Formally, when the required interaction range r exceeds the operator’s range L , no parameter set can solve the task without additional layers or non-local connections. Typical remedies expand the effective receptive field without excessively increasing the model’s depth. Examples

include using multi-scale operators that adaptively select neighborhood radii per node [Xu et al., 2018; Abboud et al., 2022], and adding long-range edges or virtual nodes to route information globally [Dwivedi and Bresson, 2021; Ying et al., 2021; Southern et al., 2025].

In practice, handling long-range dependencies usually requires addressing the three issues simultaneously: ensuring a sufficient depth to avoid under-reaching, alleviating bottlenecks to avoid over-squashing, and controlling feature mixing to avoid over-smoothing.

2.3.2 Isotropic vs anisotropic message-passing

A fundamental design choice in MPNNs concerns the nature of the message function; in particular, whether and how it incorporates information about the receiving node. Based on this, message-passing operators can be classified as either *isotropic* or *anisotropic* [Dwivedi et al., 2023].

In **isotropic** message passing, messages from the j -th to the i -th node depend only on the features of the sender node j and potentially on edge attributes. Formally, this corresponds to having a message function $\phi^{(l)}(\mathbf{h}^{j(l)}, \mathbf{e}^{ji})$ that is invariant to the identity or features of the receiver node i . These operations typically rely on a graph shift operator (GSO) $\tilde{\mathbf{A}} \in \mathbb{R}^{N \times N}$, defined as a function of the adjacency matrix, such that entry $\tilde{a}^{ji} \neq 0$ only if $a^{ji} = 0$. A prototypical example is:

$$\mathbf{h}^{i(l+1)} = \xi \left(\mathbf{W}_U^{(l)} \mathbf{h}^{i(l)} + \sum_{j \in \mathcal{N}(i)} \tilde{a}^{ji} \mathbf{W}_M^{(l)} \mathbf{h}^{j(l)} \right), \quad (2.21)$$

where $\mathbf{W}_U^{(l)}, \mathbf{W}_M^{(l)} \in \mathbb{R}^{d \times d}$ are learnable parameters and ξ is a nonlinear activation function, such as ReLU or tanh. Examples of this approach are the diffusion convolutional operator (DCNN) introduced by Atwood and Towsley [2016], where $\tilde{\mathbf{A}} = \mathbf{D}^{-1} \mathbf{A}$, with $\mathbf{D} = \text{diag}(\mathbf{A}^\top \mathbf{1})$ being the graph in-degree matrix, and the GNN operator used by Morris et al. [2019], where simply $\tilde{\mathbf{A}} = \mathbf{A}$. This design is computationally efficient and well-suited for homophile settings, where neighbor similarity is a valid assumption. However, isotropic schemes cannot adapt the message depending on the target node, potentially limiting their expressiveness.

In contrast, **anisotropic** message passing allows the message $\phi^{(l)}(\mathbf{h}^{i(l)}, \mathbf{h}^{j(l)}, \mathbf{e}^{ji})$ to depend on both the sender and receiver's features. This enables richer and more adaptive aggregation schemes where the influence of a neighbor can be calibrated based on the interaction between its own features and those of the

receiving node. A representative example is given by Bresson and Laurent [2017] with the GatedGCN operator:

$$\mathbf{h}^{i(l+1)} = \xi \left(\mathbf{W}_1^{(l)} \mathbf{h}^{i(l)} + \sum_{j \in \mathcal{N}(i)} \alpha^{ji(l)} \odot \mathbf{W}_2^{(l)} \mathbf{h}^{j(l)} \right) \quad (2.22)$$

$$\alpha^{ji(l)} = \sigma \left(\mathbf{w}_3^{(l)\top} \mathbf{h}^{i(l)} + \mathbf{w}_4^{(l)\top} \mathbf{h}^{j(l)} \right) \quad (2.23)$$

where matrices $\mathbf{W}_1^{(l)}, \mathbf{W}_2^{(l)} \in \mathbb{R}^{d \times d}$ and vectors $\mathbf{w}_3^{(l)}, \mathbf{w}_4^{(l)} \in \mathbb{R}^d$ are learnable and σ is the sigmoid function. This formulation enables adaptive edge weights $\alpha^{ji(l)}$ that control how much influence each neighbor exerts on the target node. The distinction between isotropic and anisotropic message passing has important implications for model design.

Isotropic operators can be interpreted as learned filters applied uniformly across the graph, similar to standard convolutions in Euclidean domains. Their simplicity and parameter efficiency make them suitable for settings with strong prior structure and large-scale applications, such as physical sensor networks. However, isotropic filters, especially those relying on Laplacian-based propagation, often act as low-pass filters. While this helps in denoising and smoothing signals, useful in filtering tasks, it can also lead to over-smoothing.

Anisotropic operators, on the other hand, offer greater modeling flexibility and are advantageous in heterogeneous domains or when relationships between nodes vary substantially across the graph. The use of attention mechanisms [Vaswani et al., 2017] is common in anisotropic schemes to enable dynamic weighting and interoperability [Veličković et al., 2018]. Nonetheless, the increased expressiveness comes at a computational cost and higher risk of overfitting.

2.4 Deep learning for temporal processing

Unlike the graph domain, where MPNNs have emerged as the standard framework for processing relational data, the temporal domain lacks a unified framework encompassing all major architectural paradigms [Benidis et al., 2022]. Nonetheless, existing models share the common objective of exploiting temporal dependencies to extract meaningful representations from sequential inputs. Borrowing the terminology from MPNNs, we introduce the term temporal message-passing (TMP) to broadly denote any deep learning operator designed to propagate information along the temporal dimension. We define a generic

temporal processing layer TMP as:

$$\mathbf{h}_{t-T:t}^{(l+1)} = \text{TMP}^{(l)} \left(\mathbf{h}_{t-T:t}^{(l)}, \mathbf{T} \right), \quad (2.24)$$

where $\mathbf{h}_{t-T:t}^{(0)} \in \mathbb{R}^{T \times d}$ is a sequence of initial vector representations and $\mathbf{T}$ is the temporal graph introduced in Section 2.1. Representative examples of TMP operators include recurrent, convolutional, and attention-based neural networks.

2.4.1 Recurrent neural networks

Recurrent networks maintain a latent representation of structured inputs that summarizes causally related information. Similar to Frasconi et al. [1998], we define a general form of such a network in the time domain using a temporal graph $\mathbf{T}$ constrained to be a directed acyclic graph (DAG) to enforce causality. For each $i = 0, \dots, T-1$, the hidden representation $\mathbf{h}_{t-i}^{(l+1)}$ is computed as:

$$\mathbf{h}_{t-i}^{(l+1)} = \rho^{(l)} \left(\mathbf{h}_{t-i}^{(l)}, \left\{ \mathbf{h}_{t-j}^{(l+1)} \mid \mathbf{T}[j, i] \neq 0 \right\} \right) \quad (2.25)$$

where $\rho^{(l)}$ is the state-transition function at layer l . In the typical case of RNNs [Elman, 1990], where $\mathbf{T}$ corresponds to a directed path graph, the update simplifies to the more familiar form:

$$\mathbf{h}_t^{(l+1)} = \rho^{(l)} \left(\mathbf{h}_t^{(l)}, \mathbf{h}_{t-1}^{(l+1)} \right), \quad (2.26)$$

which enables processing regularly sampled sequences of arbitrary length. The vanilla RNN introduced by Elman [1990] applies the following update rule:

$$\mathbf{h}_t^{(l+1)} = \xi \left(\mathbf{W} [\mathbf{h}_t^{(l)} \parallel \mathbf{h}_{t-1}^{(l+1)}] \right), \quad (2.27)$$

where $\mathbf{W} \in \mathbb{R}^{d \times 2d}$ is a learnable weight matrix (bias term omitted), $\parallel$ denotes concatenation over the feature dimension, and ξ is a nonlinearity, typically tanh. Nonetheless, vanilla RNNs suffer from the well-known exploding and vanishing gradient problem, which limits their ability to capture long-range dependencies [Bengio et al., 1994; Pascanu et al., 2013].

To mitigate this issue, modern architectures incorporate memory cells and learnable *gates* that regulate the flow of information through time. Two prominent examples are the Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] and the Gated Recurrent Unit (GRU) [Cho et al., 2014], which

improve stability and learning over long sequences. The update rule of a GRU, in particular, can be described by the following sequence of operations:

$$\mathbf{r}_t^{(l+1)} = \sigma \left(\mathbf{W}_R^{(l)} [\mathbf{h}_t^{(l)} \parallel \mathbf{h}_{t-1}^{(l+1)}] \right), \quad (2.28)$$

$$\mathbf{o}_t^{(l+1)} = \sigma \left(\mathbf{W}_O^{(l)} [\mathbf{h}_t^{(l)} \parallel \mathbf{h}_{t-1}^{(l+1)}] \right), \quad (2.29)$$

$$\mathbf{c}_t^{(l+1)} = \sigma \left(\mathbf{W}_C^{(l)} [\mathbf{h}_t^{(l)} \parallel \mathbf{r}_t^{(l+1)} \odot \mathbf{h}_{t-1}^{(l+1)}] \right), \quad (2.30)$$

$$\mathbf{h}_t^{(l+1)} = \mathbf{o}_t^{(l+1)} \odot \mathbf{h}_{t-1}^{(l+1)} + (1 - \mathbf{o}_t^{(l+1)}) \odot \mathbf{c}_t^{(l+1)}, \quad (2.31)$$

where each weight matrix $\mathbf{W}^{(l)} \in \mathbb{R}^{d \times 2d}$ is learnable. Beyond gating mechanisms, recent work proposed to impose structural constraints in the dynamics to address training instability and memory dissipativity. Examples include antisymmetric weight parameterizations [Chang et al., 2021], the Linear Recurrent Unit by Orvieto et al. [2023], and discretized Hamiltonian formulations with invertible memory [Rusch and Mishra, 2021].

2.4.2 Temporal convolutional networks

Convolutional TMP operators process sequential data by applying learnable filters over fixed-size temporal windows, conditioning each representation on a local context of related time steps [LeCun, Bengio, et al., 1995]. Architectures built on such operators are referred to as temporal convolutional networks (TCNs) [Oord et al., 2016; Borovykh et al., 2017; Bai et al., 2018]. A temporal convolutional layer updates the representation as:

$$\mathbf{h}_{t-T:t}^{(l+1)} = \xi \left(\sum_{k=0}^K (\mathbf{T}^k)^\top \mathbf{h}_{t-T:t}^{(l)} \mathbf{W}_k^{(l)} \right), \quad (2.32)$$

where $K \in \mathbb{N}^+$ is the filter (or kernel) size, each $\mathbf{W}_k^{(l)} \in \mathbb{R}^{d \times d}$ is a learnable weight matrix for the k -th filter position, and ξ denotes a nonlinear activation function.

When the temporal graph $\mathbf{T}$ is a directed path – as is typically the case in TCNs – the layer implements a causal convolution. The update at time t then reduces to:

$$\mathbf{h}_t^{(l+1)} = \xi \left(\sum_{k=0}^K \mathbf{W}_k^{(l)} \mathbf{h}_{t-k}^{(l)} \right), \quad (2.33)$$

ensuring that predictions at t depend only on the current and previous inputs. A major advantage of convolutional TMP operators over recurrent ones is their possibility to be executed in parallel. Indeed, since updates at different time steps are independent, computations can be executed concurrently along the temporal axis, leading to significant efficiency gains.

2.4.3 Attention-based models

The attention mechanism was first introduced by Bahdanau et al. [2014] to overcome the limitations of early encoder-decoder architectures for machine translation, which compressed the entire source sequence into a single fixed-length vector. Known as additive or Bahdanau attention, the mechanism enables the decoder to dynamically focus on relevant parts of the source sequence at each decoding step. It does so by computing an *attention score* between each pair of encoder and decoder hidden states via a nonlinear transformation. Let $\mathbf{h}_{t-T:t} \in \mathbb{R}^{T \times d_h}$ and $\mathbf{z}_{s-S:s} \in \mathbb{R}^{S \times d_z}$ denote the source and target sequences of hidden representations, respectively. For each $k \in [0, T)$ and $j \in [0, S)$, the attention score α_{jk} for the pair $(\mathbf{h}_{t-k}, \mathbf{z}_{s-j})$ is computed as:

$$\beta_{jk} = \mathbf{w}_\beta^\top \tanh(\mathbf{W}_h \mathbf{h}_{t-k} + \mathbf{W}_z \mathbf{z}_{s-j}) \quad (2.34)$$

$$\alpha_{jk} = \text{softmax}_j \{\beta_{ik}\}_{i=0}^{S-1} = \frac{\exp(\beta_{jk})}{\sum_{i=0}^{S-1} \exp(\beta_{ik})} \quad (2.35)$$

where $\mathbf{W}_h \in \mathbb{R}^{d \times d_h}$, $\mathbf{W}_z \in \mathbb{R}^{d \times d_z}$, and $\mathbf{w}_\beta \in \mathbb{R}^d$ are learnable parameters. The resulting attention weights are used to compute a context vector $\mathbf{h}'_{t-k}$ as a weighted sum of the source representations:

$$\mathbf{h}'_{t-k} = \sum_{i=0}^{S-1} \alpha_{ik} \mathbf{z}_{s-i}. \quad (2.36)$$

This formulation inspired a surge of attention-based architectures, culminating in the Transformer model by Vaswani et al. [2017], which introduced the scaled dot-product attention. In this variant, the unnormalized attention scores are computed as:

$$\beta_{jk} = \frac{(\mathbf{W}_h \mathbf{h}_{t-k})^\top (\mathbf{W}_z \mathbf{z}_{s-j})}{\sqrt{d}}, \quad (2.37)$$

where vectors $\mathbf{W}_h \mathbf{h}_{t-k}$ and $\mathbf{W}_z \mathbf{z}_{s-j}$ are called *query* and *key*, borrowing the terminology from database systems.

Attention-based methods are recently gaining traction in time series modeling [Li et al., 2019; Zhou et al., 2021; Nie et al., 2023], trailed by the remarkable success of the Transformer architecture in natural language processing [Radford et al., 2018; Brown et al., 2020]. Like convolutional approaches, these models condition the output representation on a fixed temporal window. This is achieved via *self-attention*, in which the source and target sequences are identical. To ensure causality in forecasting tasks, a causal mask is applied to the attention

matrix, restricting each time step from attending to future inputs. In practice, this masking is implemented by imposing a lower-triangular structure on matrix T . Because attention weights can be computed in parallel across time steps, these models inherit the same computational efficiency as TCNs. Nonetheless, a key limitation of standard self-attention is its quadratic complexity with respect to the sequence length T , which may hinder scalability for long time windows.

2.5 Spatiotemporal graph neural networks

GNNs have also been extended to temporal settings, where the graph topology, node features, or both evolve over time [Kazemi et al., 2020; Barros et al., 2021; Longa et al., 2023]. The field of temporal graph learning studies sequences of events acting on a graph and finds applications in social network analysis and recommender systems [Rossi et al., 2020; Huang et al., 2023]. Other lines of work address sequences of arbitrary graphs without assuming node identification over time [Zambon et al., 2018; Zambon, 2022], or aim to predict changes in graph topology [Zambon et al., 2019; Paassen et al., 2020]. While formally related, these approaches differ substantially from the setting considered in this thesis, where graph nodes are associated with (irregularly sampled) time series.

In contrast, when time series are observed at the nodes of a graph, dedicated hybrid architectures have emerged to integrate message passing with temporal processing, known as spatiotemporal graph neural networks (STGNNs) [Li et al., 2018b; Yu et al., 2018; Cini, Marisca, et al., 2025; Jin et al., 2024; Corradini et al., 2024]. These models have demonstrated strong performance across diverse real-world domains, including weather forecasting [Lam et al., 2023], epidemiology [Kapoor et al., 2020], and biological systems [Pineda et al., 2023]. This section introduces the theoretical foundations and architectural design principles of STGNNs, which form the core modeling framework of this thesis. We begin by defining the notion of *spatiotemporal message-passing*, the cornerstone operator in STGNNs, and then explore how such operators can be designed in practice.

2.5.1 Spatiotemporal message-passing

While the previously introduced MP and TMP operators are limited to acting along a single dimension – either spatial or temporal – the spatiotemporal message-passing (STMP) operator generalizes this capability by enabling joint processing across both dimensions [Cini, Marisca, et al., 2023b]. This allows the model to simultaneously account for spatial dependencies defined by the graph

topology $\mathcal{E}$ and temporal dependencies encoded in T , embedding structural inductive biases directly into the architecture. The STMP layer STMP, which lies at the core of any STGNN, updates representations at layer l via:

$$\mathbf{H}_{t-T:t}^{(l+1)} = \text{STMP}^{(l)} \left(\mathbf{H}_{t-T:t}^{(l)}, \mathcal{E}, T \right). \quad (2.38)$$

The space of possible STMP designs is broad, with most approaches differing in how they integrate temporal and spatial processing. In the next section, we discuss two prototyping paradigms that have emerged as dominant design patterns.

2.5.2 Spatiotemporal message-passing operators

STMP operators can be broadly categorized by how they treat the spatial and temporal dimensions during processing. We refer to them as *disjoint* when processing along the temporal and spatial dimension is decoupled in two distinct phases, and as *joint* when the two dimensions are processed simultaneously within a single operator. Joint operators enable a more expressive modeling of spatiotemporal dependencies, capturing interactions between neighboring nodes and their histories in a tightly coupled fashion. This can be particularly valuable when structural dependencies span space and time simultaneously. In contrast, the disjoint approach offers modularity and simplicity by reusing existing spatial and temporal processing layers, often resulting in easier implementation and improved efficiency, although at the cost of reduced expressivity.

Disjoint STMP

A straightforward yet effective strategy to design an STMP layer is to factorize processing along the two dimensions with a cascade application of MP and TMP functions. Hence, temporal processing in one layer is carried out node-wise, and the output is propagated over the graph through message passing. This enables using existing graph-based and sequence-processing operators, as presented in the previous sections, making the resulting STGNN easy to implement. To simplify, here we discuss operators performing spatial processing downstream to temporal processing, although the other way around is seamlessly possible. Under this assumption and considering a sequence of T node-attribute matrices

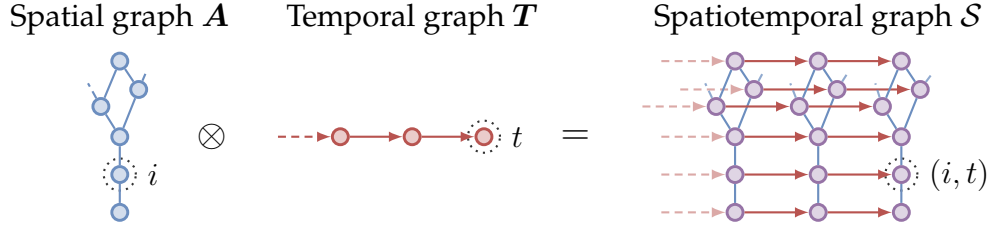


Figure 2.3. Cartesian product of spatial graph $\mathbf{A}$ and temporal graph $\mathbf{T}$.

$\mathbf{H}_{t-T:t}^{(l)}$, we can write the l -th STMP layer as:

$$\mathbf{z}_{t-T:t}^{i(l)} = \text{TMP}_{\times L_T}^{(l)} \left(\mathbf{h}_{t-T:t}^{i(l)}, \mathbf{T} \right) \quad i = 0, \dots, N-1 \quad (2.39)$$

$$\mathbf{H}_{t-k}^{(l+1)} = \text{MP}_{\times L_S}^{(l)} \left(\mathbf{Z}_{t-k}^{(l)}, \mathcal{E} \right) \quad k = 0, \dots, T-1 \quad (2.40)$$

where $\mathbf{z}_{t-T:t}^{i(l)} \in \mathbb{R}^{T \times d}$ is the sequence of intermediate representations resulting from the i -th node temporal encoding, and MP is applied independently (with shared parameters) across time steps. The subscript $\cdot_{\times L_T}$ ($\cdot_{\times L_S}$) concisely denotes a stack of L_T (L_S) functions of the same family – with distinct parameters – each receiving as input representation the output from the preceding function in the stack. Although processing is decoupled within a single STMP layer, the layer’s output effectively incorporates information from both dimensions, including the history of neighboring nodes.

The first architectures adopting this paradigm were message-passing temporal convolutional networks (MPTCNs), i.e., convolutional models where TMP is implemented by a TCN as defined in Equation 2.32 [Yu et al., 2018; Wu et al., 2019]. These architectures can be seen as mirroring the spatial processing on the temporal dimension.

Joint STMP

Joint operators have been tailored to spatiotemporal processing. These operators can be seen as operating on the product graph $\mathcal{S} = \mathbf{A} \otimes \mathbf{T}$ obtained by combining spatial and temporal relationships [Sabbaqi and Isufi, 2023]. We show an example of such a graph in Figure 2.3, where the spatiotemporal graph is the Cartesian product of the spatial and temporal graphs. Message-passing recurrent neural networks (MPRNNs) were among the first operators of this kind to be proposed [Seo et al., 2018; Li et al., 2018b]. The key idea behind is to replace linear projections inside the state-transition function with MPNN. As

an example, by considering a GRU to model dynamics (Equations 2.28 to 2.31), the resulting message-passing GRU (MPGRU) cell updates representations at each time step t as

$$\mathbf{R}_t^{(l+1)} = \sigma \left(\text{MP}_R^{(l)} \left([\mathbf{H}_t^{(l)} \parallel \mathbf{H}_{t-1}^{(l+1)}], \mathcal{E} \right) \right), \quad (2.41)$$

$$\mathbf{O}_t^{(l+1)} = \sigma \left(\text{MP}_O^{(l)} \left([\mathbf{H}_t^{(l)} \parallel \mathbf{H}_{t-1}^{(l+1)}], \mathcal{E} \right) \right), \quad (2.42)$$

$$\mathbf{C}_t^{(l+1)} = \tanh \left(\text{MP}_C^{(l)} \left([\mathbf{H}_t^{(l)} \parallel \mathbf{R}_t^{(l+1)} \odot \mathbf{H}_{t-1}^{(l+1)}], \mathcal{E} \right) \right), \quad (2.43)$$

$$\mathbf{H}_t^{(l+1)} = \mathbf{O}_t^{(l+1)} \odot \mathbf{H}_{t-1}^{(l+1)} + (1 - \mathbf{O}_t^{(l+1)}) \odot \mathbf{C}_t^{(l+1)}, \quad (2.44)$$

where different MP are used to compute the outputs of Equations 2.41 to 2.43. Note that we consider, for each gate, a single MP operation at each l -th layer for conciseness' sake, while a stack of MP layers, as in Equation 2.40, is often adopted in practice.

Other operators can be obtained by extending the MP update rule, as defined in Equation 2.18, to operate on sequences rather than static vectors, i.e.,

$$\mathbf{h}_{t-T:t}^{i(l+1)} = \gamma^{(l)} \left(\mathbf{h}_{t-T:t}^{i(l)}, \text{AGGR}_{j \in \mathcal{N}(i)} \left\{ \phi^{(l)} \left(\mathbf{h}_{t-T:t}^{i(l)}, \mathbf{h}_{t-T:t}^{j(l)}, \mathbf{e}^{ji} \right) \right\} \right). \quad (2.45)$$

As an example, functions $\gamma(\cdot)$ and $\phi(\cdot)$ can be implemented using TCNs as

$$\phi_{t-T:t}^{ji(l)} = \text{TCN}_{\phi}^{(l)} \left([\mathbf{h}_{t-T:t}^{i(l)} \parallel \mathbf{h}_{t-T:t}^{j(l)}] \right) + \mathbf{W}_e^{(l)} \mathbf{e}^{ji}, \quad (2.46)$$

$$\mathbf{h}_{t-T:t}^{i(l+1)} = \text{TCN}_{\gamma}^{(l)} \left([\mathbf{h}_{t-T:t}^{i(l)} \parallel \text{AGGR}_{j \in \mathcal{N}(i)} \left\{ \phi_{t-T:t}^{ji(l)} \right\}] \right), \quad (2.47)$$

where $\text{TCN}_{\phi}^{(l)}$ and $\text{TCN}_{\gamma}^{(l)}$ are different TCNs at the l -th layer defined as in Equation 2.32 and $\mathbf{W}_e^{(l)} \in \mathbb{R}^{d \times d_e}$ is a weight matrix. Note that, differently from MPTCNs where processing is factorized over the two dimensions, messages defined as in Equation 2.46 traverse the two dimensions simultaneously.

2.6 Datasets

To empirically evaluate graph-based spatiotemporal models, we collect a selection of real-world datasets spanning diverse application domains and sampling schemes. These datasets originate from air quality monitoring, traffic sensors, energy analytics, and weather forecasting, each presenting unique challenges in spatial topology, temporal resolution, and data availability. Importantly, two of the datasets – **EngRAD** and **PeakWeather** – have been contributed as part of my doctoral work. Table 2.1 summarizes relevant information about each of them.

Table 2.1. Metadata of real-world datasets used in the experiments throughout the thesis. In **bold**, datasets introduced as part of my research.

Datasets	Type	Nodes	Time steps	% missing	d_x	Sampling
AQI	Undirected	437	8,760	25.67	1	1 hour
AQI-36	Undirected	36	8,760	13.25	1	1 hour
METR-LA	Directed	207	34,272	8.11	1	5 minutes
PEMS-BAY	Directed	325	52,128	0.02	1	5 minutes
PEMS03	Directed	358	26,208	N/A	1	5 minutes
PEMS04	Directed	307	16,992	N/A	1	5 minutes
PEMS07	Directed	883	28,224	N/A	1	5 minutes
PEMS08	Directed	170	17,856	N/A	1	5 minutes
CER-E (Full)	Directed	6,435	25,728	4.59	1	30 minutes
CER-E	Directed	485	25,728	0.04	1	30 minutes
PV-US	Undirected	5,016	52,560	0	1	10 minutes
EngRAD	Undirected	487	26,304	0	5	1 hour
PeakWeather	Undirected	302	433,728	≈ 3	8	10 minutes

Air quality

Air pollution – especially fine particulate matter such as PM 2.5 – poses major risks to public health and the environment worldwide, driving research efforts in monitoring and forecasting urban air quality. The Urban Computing project [Zheng et al., 2014; Zheng et al., 2015] addressed this by releasing a comprehensive benchmark dataset of irregular, multi-city sensor data, tailored to challenge spatiotemporal models in realistic settings. These data have since become a standard benchmark for graph-based methods dealing with variable missingness, noisy readings, and spatial dependencies.

AQI [Zheng et al., 2015]

The dataset contains one year of hourly measurements of several air quality indices from 437 monitoring stations distributed across 43 cities in China from May 2014 to April 2015. We consider only observations related to PM 2.5. Since 25.67% of the measurements are missing, it is a widely used benchmark for irregular spatiotemporal data analytics. Besides air quality readings, the dataset provides geographic coordinates of each monitoring station. We obtain an adjacency matrix from the geographic

distances $\text{dist}(i, j) \in \mathbb{R}$ between each pair of nodes $i, j \in \mathcal{V}$. In particular, the weight a^{ij} of the edge connecting the i -th to the j -th node is computed through a thresholded Gaussian kernel [Shuman et al., 2013] as

$$a^{ij} = \begin{cases} \exp\left(-\frac{\text{dist}(i, j)^2}{\gamma}\right) & \text{dist}(i, j) \leq \delta \\ 0 & \text{otherwise} \end{cases}, \quad (2.48)$$

where γ controls the kernel width and δ is the threshold. We set γ as the standard deviation of the pairwise distances of all the stations in the city of Beijing (see AQI-36 dataset below), and δ such that it corresponds to a distance of approximately 40 km. By doing so, we obtain 2,699 edges, which are 1.41% of the possible connections $\mathcal{V} \times \mathcal{V}$.

AQI-36 [Zheng et al., 2015]

The subset of AQI containing only the 36 sensors in the city of Beijing. This reduced version of the dataset, with 13.25% of data missing, has been considered by several works on time series imputation that do not rely on graphs [Yi et al., 2016; Cao et al., 2018].

Traffic forecasting

The study of traffic networks is key for the development of intelligent transportation systems and a relevant application field for network science [Lippi et al., 2013]. Most works on spatiotemporal deep learning focused on this domain [Li et al., 2018b; Yu et al., 2018; Wu et al., 2019; Shang et al., 2021]. We consider two popular benchmarks introduced by Li et al. [2018b], namely METR-LA and PEMS-BAY datasets. Data has been obtained from the Caltrans Performance Measurement System (PeMS) [Chen et al., 2001] and is collected in real-time from nearly 40,000 (as of 2025) individual detectors divided into districts spanning the freeway system across all major metropolitan areas of California. We further consider the set of PEMS03–08 datasets, introduced by Guo et al. [2022] and collected from the same provider.

METR-LA [Li et al., 2018b]

The dataset contains traffic speed measurements from 207 loop detectors in the Los Angeles County Highway, aggregated in 5-minute intervals over four months between March and June 2012. The total number of time steps is 34,272, and 8.11% of data are missing. A matrix of distances between sensors is provided and used to obtain the adjacency matrix following the same procedure of the air quality datasets (Equation 2.48).

In particular, we use a thresholded Gaussian kernel with kernel width γ equal to the standard deviation of the distances and threshold $\delta = 0.1$. This approach yields 1,515 edges, with an adjacency matrix density of 3.54%.

PEMS-BAY [Li et al., 2018b]

Similar to METR-LA, PEMS-BAY contains traffic readings from 325 sensors located in the San Francisco Bay Area, aggregated in 5-minute intervals over six months from January to June 2017. The total number of time steps is 52,128, and only 0.02% of the data are reported as missing (although others might have been previously imputed). The graph is obtained following the same procedure as METR-LA, providing 2,369 edges (2.24% density).

PEMS03–08 [Guo et al., 2022]

This collection comprises four datasets – PEMS03, PEMS04, PEMS07, and PEMS08 – each recording traffic flow readings at 5-minute intervals. The data is gathered from different districts in California, as indicated by the dataset numbers. For these datasets, the edge set – representing the underlying road network – is included with the data; however, no detailed explanation has been provided regarding how this edge set was constructed. Compared to the graphs for METR-LA and PEMS-BAY, it is significantly sparser, with an adjacency matrix density of less than 1%. The datasets have different sizes:

PEMS03 data collected from 358 traffic sensors from September to November 2018 (26,208 time steps), with 546 edges (0.43% density);

PEMS04 data collected from 307 traffic sensors from January to February 2018 (16,992 time steps), with 340 edges (0.36% density);

PEMS07 data collected from 883 traffic sensors from May to August 2017 (28,224 time steps), with 866 edges (0.11% density);

PEMS08 data collected from 170 traffic sensors from July to August 2016 (17,856 time steps), with 277 edges (0.96% density).

Energy analytics

Energy analytics plays a crucial role in modern urban planning and sustainability, particularly in the context of smart grids and efficient energy management. With the widespread deployment of smart meters across urban areas and solar

plants, vast amounts of spatiotemporal energy consumption and production data are being collected. Architectures that capture both spatial and temporal dependencies are particularly relevant, as they can identify patterns spanning the two dimensions that can guide forecasting of energy consumption and demand, as well as production from renewable sources.

CER-E (Full) [Energy Regulation, 2016]

The dataset contains a collection of energy consumption readings, aggregated into 30-minute intervals, measured from July 14, 2009, to the end of 2010 by 6,435 smart meters at residential and commercial/industrial premises. Data is sourced from the Irish Commission for Energy Regulation (CER) Smart Metering Project [Energy Regulation, 2016]. Although the dataset is not publicly available, it can be accessed free of charge for research purposes upon request. The total number of time steps is 25,728, with 4.59% of data being missing. We build an adjacency matrix by extracting a k -nearest neighbor graph (with $k = 9$) from the similarity matrix built by computing the week-wise correntropy [Liu et al., 2007] among time series. Note that the k -NN reduction makes the resulting graph directed.

CER-E [Energy Regulation, 2016]

This dataset is a subset of CER-E (Full), restricting the data to the 485 smart meters monitoring small and medium-sized enterprises. We follow the same approach as CER-E (Full) to obtain an adjacency matrix, which has a density of 1.86% (4,365 edges).

PV-US [Hummon et al., 2012]

This dataset comprises simulated energy production from 5016 solar plants in the US in 2006, aggregated in 10-minute intervals. Data is provided by the National Renewable Energy Laboratory (NREL)'s Solar Power Data for Integration Studies¹ and contains a total of 52,560 time steps with no missing values. The dataset is divided into two overlapping macro zones, "east" and "west", with 4084 and 1081 plants, respectively. When using the full dataset, duplicated plants are dropped from the "east" zone. A small subset of this dataset – often named "Solar Energy" – with only the 137 plants in Alabama has been used as a multivariate time series forecasting benchmark [Lai et al., 2018]. To obtain an adjacency matrix, we follow the same procedure adopted for METR-LA and PEMS-BAY.

¹<https://www.nrel.gov/grid/solar-power-data.html>

Weather forecasting

Weather forecasting represents a critical and long-standing challenge for both meteorology and machine learning. Accurate prediction of weather variables is crucial for agriculture, disaster prevention, transportation, and energy management. Data-driven approaches to weather forecasting are rapidly emerging as powerful complements to classical Numerical Weather Prediction (NWP) techniques, offering orders-of-magnitude improvements in computational efficiency while approaching – or surpassing – operational models in accuracy. Leading examples include Google DeepMind’s GraphCast [Lam et al., 2023], Nvidia’s FourCastNet [Pathak et al., 2022], and Microsoft’s Aurora [Bodnar et al., 2025], which rely on global reanalysis or model-generated data cast into graph-based or transformer-like deep networks to predict multiple variables over continental to global scales up to 10 days ahead. Complementary to these approaches, an emerging paradigm leverages crowd-sourced sensing data to enhance nowcasting resolution by integrating smartphone sensor data and human observations [Niforatos et al., 2014].

We introduce two complementary datasets that support the development of graph-based forecasting models, but differ in their data generation paradigms and targeted applications. EngRAD [Marisca et al., 2024] is derived from high-resolution NWP model outputs (specifically ECMWF IFS) and provides synthetic yet spatially dense weather variables. In contrast, PeakWeather [Zambon, Cattaneo, Marisca, et al., 2025] comprises real-world ground station measurements from the SwissMetNet network² and represents a prototypical benchmark for Direct Observation Prediction (DOP), where models are trained and evaluated exclusively on measured observations without relying on any numerical simulations. The two datasets therefore support distinct yet complementary research directions.

EngRAD [Marisca et al., 2024]

Released as part of [Marisca et al., 2024], the EngRAD dataset contains hourly measurements of 5 different weather variables collected at 487 grid points in England from 2018 to 2020. Data has been provided by open-meteo.com³ [Zippenfenig, 2023] and licensed under Attribution 4.0 International (CC BY 4.0). The numerical weather prediction model used to generate the data is ECMWF IFS, which has a spatial resolution of 9 km. The measurement locations correspond to cities and are enriched with

²The automatic measurement network of MeteoSwiss.

³<https://open-meteo.com/>

metadata including geographic coordinates, elevation, and administrative region. The variables included are: (1) air temperature at 2 meters above ground ($^{\circ}\text{C}$); (2) relative humidity at 2 meters above ground (%); (3) total hourly precipitation (mm, sum of preceding hour); (4) total cloud cover (%); (5) global horizontal irradiance (W/m^2). The dataset contains no missing values and covers three full years, enabling the study of seasonal patterns. The graph structure is constructed by computing pairwise haversine distances and applying a thresholded Gaussian kernel with threshold $\delta = 0.1$, akin to the procedure adopted in air-quality and traffic datasets. We further cap the maximum number of neighbors to 8 per node and mirror directional edges to obtain an undirected graph. This yields an adjacency matrix with 2,297 edges, corresponding to a density of approximately 0.97%.

PeakWeather [Zambon, Cattaneo, Marisca, et al., 2025]

The dataset contains high-frequency (10 minutes) meteorological observations collected from 302 ground stations across Switzerland between January 1, 2017 and March 31, 2025. Data has been acquired from the SwissMetNet network maintained by MeteoSwiss, and includes eight weather variables sampled every 10 minutes: (1) air temperature at 2 meters above ground ($^{\circ}\text{C}$); (2) relative humidity at 2 meters above ground (%); (3) atmospheric pressure at station level (hPa); (4) sunshine duration (min); (5) precipitation (mm); (6) wind speed (m/s); (7) wind gusts (m/s); (8) wind direction (degrees). Unlike datasets based on satellite or gridded reanalysis, such as EngRAD, PeakWeather comprises ground-level observations from high-quality weather stations, regularly calibrated and quality-checked. Importantly, the dataset is not uniformly complete: different stations are equipped with different sensors, and their configurations evolve over time due to operational constraints. As a result, missing data stems both from temporary gaps (e.g., sensor faults) and from variable-specific sensor availability. For instance, wind-related variables are measured by ≈ 150 stations during the collection period, with an average missingness of $\approx 1.2\%$ over time; cloud-related variables such as sunshine duration are more frequently missing, with up to 13.72% of gaps. Missing values are explicitly marked and accompanied by a binary mask indicating the validity of each measurement across time, station, and variable. In addition to the weather variables, the dataset includes rich topographic descriptors derived from a 50-meter resolution digital height model (DHM25), capturing elevation, slope, terrain aspect, and

gradient characteristics within 2 km and 10 km radii. A baseline forecast is provided using ICON-CH1-EPS, a high-resolution operational NWP model by MeteoSwiss [Zängl et al., 2015]. The graph structure is built using a thresholded Gaussian kernel on haversine distances between stations, capped to $k = 8$ neighbors.

Part I

Imputation

Chapter 3

Imputation of irregular spatiotemporal data

Missing data is a pervasive challenge in real-world applications involving spatiotemporal data, where sensor malfunctions, network outages, or transmission errors often corrupt measurements. These failures can either be sparse and localized – affecting individual sensors or short time intervals – or arise in consecutive blocks – e.g., for faults or blackouts – which significantly degrade the utility of the collected data. Robust imputation methods are thus essential, not only for restoring data completeness but also for enabling downstream tasks such as forecasting and anomaly detection [[Little and Rubin, 2019](#)].

Classical imputation techniques often treat time series individually, relying on linear approaches or autoregressive dynamics without considering spatial context. However, in many applications, sensors are part of a structured system where dependencies across entities play a critical role. Graphs provide a natural abstraction for modeling such relational structure, allowing us to capture spatial and functional relationships among nodes. When combined with deep learning, and in particular with GNNs, this perspective yields expressive architectures capable of jointly modeling temporal and spatial dependencies.

In this chapter, we explore graph-based methods for imputing missing values in irregular spatiotemporal data, focusing on flexible and scalable approaches grounded in modern deep learning. The chapter is organized as follows. We begin by formalizing the imputation task and introducing a general class of graph-based models for reconstructing partially observed time series, and discuss how this formulation enables related tasks such as filtering and virtual

sensing (Section 3.1). In the second part, we detail the design of GRIN [Cini, Marisca, et al., 2022] (Section 3.2), our autoregressive graph-based imputation model. GRIN is a bidirectional recurrent architecture that incorporates spatiotemporal message-passing to produce spatially and temporally coherent imputations. Notably, the same architecture extends naturally to filtering and virtual sensing tasks, highlighting the flexibility of the proposed design. We then discuss related research and state-of-the-art approaches in Section 3.3, including methods used in the experimental analysis. We evaluate our approach across a range of real-world datasets, missing data patterns, and operational regimes, demonstrating state-of-the-art performance and strong generalization capabilities (Section 3.4). We conclude with a discussion of future directions and limitations (Section 3.5), setting the stage for the next chapter, which addresses imputation under sparse observation regimes.

Reference papers

This chapter’s content is partly based on material from the following articles.

Andrea Cini, Ivan Marisca, and Cesare Alippi (Jan. 2022). **Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks**. In: *International Conference on Learning Representations*.

3.1 Graph deep learning for time series imputation

We focus on the task of time series imputation, i.e., given a window of $T \geq 1$ past partial observations $\mathcal{X}_{t-T:t}$, we are interested in estimating the missing value x_{t-k}^i for each node $i \in \mathcal{V}$ and time offset $k < T$ such that $x_{t-k}^i \notin \mathcal{X}_{t-k}$. To this end, we seek to learn a parametric model that approximates the conditional expectation of missing entries, as formalized in Equation 2.14 in Section 2.2.2.

In line with the problem setting introduced in Section 2.1, temporal irregularities caused by asynchrony or acquisition errors are expressed as missing values on a regular time grid. Central to our approach is the assumption that incorporating relational structure, i.e., the graph $\mathcal{E}$, helps constrain the solution space to the most plausible ones and improves imputation accuracy. Under this assumption, we define a family of graph-based imputation models $\mathcal{I}(\cdot)$ as:

$$\begin{aligned}\bar{\mathbf{X}}_{t-T:t} &= \mathcal{I}(\mathcal{X}_{t-T:t}, \mathbf{U}_{t-T:t}, \mathbf{V}, \mathcal{E}; \boldsymbol{\theta}) \\ &\approx \mathbb{E}_p[p(\mathbf{X}_{t-T:t} \mid \mathcal{X}_{\leq t}, \mathbf{U}_{\leq t}, \mathbf{V})],\end{aligned}\quad (3.1)$$

where $\boldsymbol{\theta}$ indicates the model's trainable parameters, and the conditioned $\mathbf{X}_{t-T:t}$ denotes the fully observed data matrix, unavailable in practice. The graph $\mathcal{E}$ acts as a structural prior, enabling localized estimation based on node neighborhoods. For simplicity, we assume temporal dependencies are implicit in the sequence ordering and omit explicit reference to the temporal graph $\mathbf{T}$ at this stage.

To instantiate this modeling framework in practice, we consider imputation architectures that follow a pipeline of three stages: an initial encoding of the input, a stack of spatiotemporal message-passing layers, and a final readout to reconstruct missing values. Formally, the process is defined by:

$$\mathbf{h}_{t-k}^{i(0)} = \text{ENCODER}(\mathbf{x}_{t-k}^i, \mathbf{m}_{t-k}^i, \mathbf{u}_{t-k}^i, \mathbf{v}^i) \quad \text{for } k \in [0, T), i \in \mathcal{V}, \quad (3.2)$$

$$\mathbf{H}_{t-T:t}^{(l+1)} = \text{STMP}^{(l)}(\mathbf{H}_{t-T:t}^{(l)}, \mathcal{E}) \quad \text{for } l \in [0, L), \quad (3.3)$$

$$\bar{\mathbf{x}}_{t-k}^i = \text{READOUT}(\mathbf{h}_{t-k}^{i(L)}) \quad \text{for } \mathbf{m}_{t-k}^i = \mathbf{0}, \quad (3.4)$$

where $\text{ENCODER}(\cdot)$ is an encoder function that processes the raw inputs, and $\text{READOUT}(\cdot)$ is a decoder that maps the final latent representation back to the observation space. Both are differentiable and typically implemented as MLPs applied independently to each node and timestep, thus not propagating information over space and time.

In particular, ENCODER produces an initial representation $\mathbf{h}_{t-k}^{i(0)}$ by combining the available node observation $\mathbf{x}_{t-k}^i$, its associated mask $\mathbf{m}_{t-k}^i$, covariates

$\mathbf{u}_{t-k}^i$, and static features $\mathbf{v}^i$. The mask is crucial for distinguishing between observed and missing entries, which are typically zero-filled. The encoded features are then passed through L stacked STMP layers, which iteratively incorporate information from spatial neighbors and close time steps, enriching each representation with increasingly wider spatiotemporal context. Finally, READOUT projects the final representation $\mathbf{h}_{t-k}^{i(L)}$ back to the input space $\mathbb{R}^{d_x}$ for each location where the original value was missing.

This architectural framework is highly general and can also accommodate standard imputation methods that do not rely on graph structure, either by omitting the spatial processing in Equation 3.3 or resorting to an identity adjacency matrix. Before introducing the specific STGNN architectures proposed in this thesis, we first discuss in Section 3.1.1 related tasks that naturally emerge from the formalization of the imputation problem. Then, in Section 3.1.2, we present alternative imputation strategies suited to different operational settings.

3.1.1 Graph-based filtering and virtual sensing

Although this chapter mainly focuses on imputing missing values, it is worth highlighting that the same modeling principles can be extended to conceptually related tasks. Two such tasks are *filtering*, which aims to denoise observed data, and *virtual sensing*, which seeks to infer entire signals at unobserved nodes.

Filtering

As introduced in Section 2.2.2, filtering refers to the process of producing smooth, denoised estimates for observed entries $\mathbf{x}_{t-k}^i \in \mathcal{X}_{t-k}$, with the objective of improving data quality over a temporal window [Holloway Jr, 1958; Simon, 2006a]. From a probabilistic standpoint, filtering involves computing the conditional expectation of a known value given its surrounding spatiotemporal context. The model leverages this contextual information to improve the reliability and consistency of the raw input. The distinction from imputation lies only in the target – observed entries rather than missing ones.

Graph-based models are particularly well-suited for this task, as their inductive biases promote smoothness across connected nodes and close time steps. As noted in Section 2.3.2, many GNNs inherently act as low-pass filters, performing a form of spatial filtering by design [Kipf and Welling, 2017; Oono and Suzuki, 2020]. In the spatiotemporal setting, this filtering can be extended along both dimensions using spatiotemporal message-passing mechanisms over graph

sequences [Cini, Marisca, et al., 2022]. The interplay between imputation and filtering motivates treating them under a unified modeling framework. As shown later in Section 3.2.3, imputation models can also serve to produce filtered reconstructions of the input time series.

Virtual sensing

Virtual sensing involves estimating signals at spatial locations where no sensor is positioned [Stein, 1999; Liu et al., 2009]. Unlike imputation, which fills in missing values within partially observed time series at existing nodes, virtual sensing aims to reconstruct entire sequences for completely unobserved nodes. This task is particularly relevant in real-world scenarios, such as environmental monitoring, where sensor placement may be limited by cost, accessibility, or physical constraints.

Graph-based models are a natural fit for this problem. A straightforward strategy is to extend an imputation architecture by introducing virtual nodes at the desired locations. When side information is available – such as node-level covariates or static features – these can be seamlessly integrated alongside existing nodes by design. The typical evaluation protocol then involves training on a subset of observed nodes and assessing performance on held-out (virtual) nodes.

This task has recently sparked growing interest in the community [De Felice et al., 2024; Zhao et al., 2025; Zhang et al., 2025]. In particular, De Felice et al. [2024] propose an STGNN that performs virtual sensing from sparse multivariate data using a nested graph structure linking both spatial locations and channels. Notably, although the focus of this thesis is on imputation, the same STGNN architectures extend naturally to virtual sensing by treating unobserved nodes as targets for reconstruction. This extension is empirically validated in the experimental analysis in Section 3.4.5.

Altogether, these advances showcase the broader potential of graph deep learning in irregularly-sampled settings, not only for recovering missing values but also for enabling spatial extrapolation.

3.1.2 Operational settings

The design and evaluation of imputation models depend not only on the learning paradigm and model architecture but also on the specific operational scenario in which the model is deployed. Different assumptions on data availability and

latency constraints give rise to distinct settings, each with its own trade-offs. In this section, we delineate two key operational dimensions relevant to the imputation problem, namely, the imputation regime (fixed-point, fixed-lag, and fixed-interval) and the data partitioning strategy (in-sample and out-of-sample).

Fixed-point, fixed-lag, and fixed-interval imputation

The classification of temporal imputation regimes finds its roots in the classical literature on optimal smoothing [Simon, 2006a], which distinguishes between strategies based on how much future information is used to estimate past values. Besides being valid for filtering, these notions translate naturally to the imputation setting. Although originally formulated for linear state-space models, the distinction remains conceptually useful to characterize modern deep learning approaches.

In **fixed-point imputation**, the estimate of a missing value at a specific time step x_t^i is progressively refined as new observations become available at future time steps $t + 1, t + 2, \dots$. The index t remains fixed, while the context grows forward in time. This regime allows for dynamic updates of imputed values and may be suitable in streaming applications where one aims to improve past estimates on the fly. However, it requires re-evaluating the same estimate as new data arrives, making it less practical in many offline or batch-processing scenarios.

Conversely, **fixed-lag imputation** introduces a controlled delay $k > 0$ between the occurrence of a missing value and its estimation. That is, a missing value at time $t - k$ is imputed only once, as soon as the model has access to observations up to time t . This regime offers a trade-off between the celerity of fixed-point estimation and the accuracy afforded by future context, making it appropriate for quasi-real-time applications with bounded latency constraints. However, as it assumes a fixed temporal offset, it is less flexible and requires tuning based on the application.

This thesis adopts the **fixed-interval imputation** setting, in which missing values are imputed using all available observations within a window $(t - T, t]$. This approach aligns with fixed-interval smoothing in classical estimation theory – e.g., the Rauch-Tung-Striebel (RTS) smoother [Rauch et al., 1965] – and is best suited for offline scenarios where a full context around the missing entries can be exploited. By accessing both past and future information within the interval, fixed-interval methods generally achieve higher reconstruction accuracy, at the cost of increased latency and computational complexity.

In-sample and out-of-sample imputation

In the context of trainable, parametric imputation methods, we consider two different operational settings that differ in how the input sequences are partitioned for training and evaluation.

In the first setting, referred to as **in-sample imputation**, the model is trained directly on the sequence that will be imputed afterwards. That is, the model is fitted on all the available entries of a given fixed input window $\mathcal{X}_{t-T:t}$, excluding both the naturally missing values and an additional subset that is artificially masked for evaluation purposes. This setting reflects a use case where the practitioner is interested in recovering gaps in a single corrupted sequence by fitting a dedicated model on it.

Conversely, in the **out-of-sample imputation** setting, the model is trained and evaluated on disjoint sequences. During training, it sees only a collection of historical sequences, while at test time it is applied to separate, unseen sequences. This reflects a more general-purpose deployment scenario, where a model trained on past data is reused to impute missing values in new sequences without retraining.

Notably, the statistical assumptions on the target sequences, the ability of the model to generalize, and the resulting imputation accuracy can differ substantially depending on the operational regime. Nonetheless, the model is never exposed to ground-truth values used for evaluation in both settings, ensuring a fair assessment of reconstruction ability.

3.2 Graph-based autoregressive imputation

Time series imputation can be effectively framed as an autoregressive task, where missing entries are predicted based on past and future observations. In multivariate settings, this is extended to incorporate concurrent observations across variables. However, such extensions typically assume fully connected dependencies, limiting their ability to scale or reflect domain-specific relational information – particularly problematic in spatiotemporal contexts where nodes correspond to interconnected entities such as sensors.

To overcome these limitations, we introduce graph-based inductive biases into the autoregressive framework. Specifically, we present the Graph Recurrent Imputation Network (GRIN) [Cini, Marisca, et al., 2022], which integrates bidirectional recurrence with spatial message-passing to model spatiotemporal data

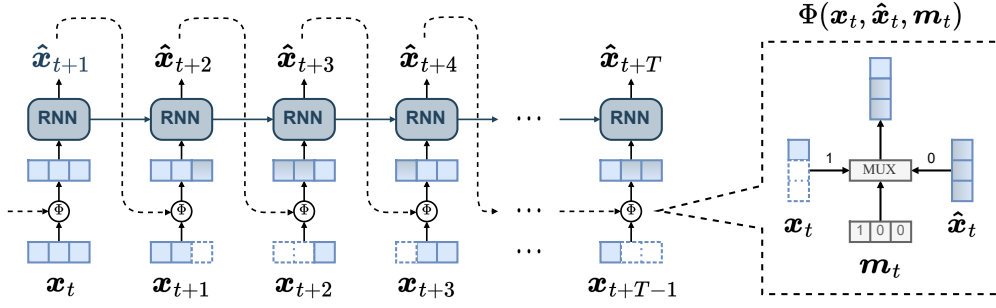


Figure 3.1. The forward component of an autoregressive imputation model implemented by an RNN. In bidirectional architectures, a twin backward component processes the sequence in reverse order.

over irregular domains. This section begins by formalizing the autoregressive imputation approach, and then details the design of GRIN. Section 3.4 provides a comprehensive empirical evaluation demonstrating its effectiveness across diverse benchmarks.

3.2.1 The autoregressive approach

Autoregressive modeling is a well-established paradigm for addressing imputation in temporal data, leveraging sequential dependencies to reconstruct missing values. These methods approximate the posterior distribution in Equation 3.1 by modeling distinct contributions from past and future information. Each of these components corresponds to a symmetrical predictive distribution that is learned independently.

The *forward* component treats the task as next-step prediction, scanning the input sequence sequentially and imputing missing values as they appear. To reconstruct missing observations at time $t - k$, it estimates the conditional expectation

$$\overrightarrow{\mathbf{X}}_{t-k} = \mathbb{E}_p [p(\mathbf{X}_{t-k} | \mathcal{X}_{t-T:t-k-1}, \mathbf{U}_{t-T:t-k}, \mathbf{V})]. \quad (3.5)$$

Figure 3.1 illustrates this strategy using RNNs, a typical choice for sequential processing in deep learning.

The *backward* component flips this processing over the temporal dimension, allowing the model to exploit information from future time steps:

$$\overleftarrow{\mathbf{X}}_{t-k} = \mathbb{E}_p [p(\mathbf{X}_{t-k} | \mathcal{X}_{t-k:t}, \mathbf{U}_{t-k-1:t}, \mathbf{V})]. \quad (3.6)$$

In this formulation, past information contributes a predictive prior, while future observations refine the estimate through a smoothing effect.

Both components are implemented as parametric functions (e.g., RNNs, attention mechanisms), and their outputs are merged to produce the final imputation:

$$\overline{\mathbf{X}}_{t-k} = g_{\theta}(\overline{\mathbf{X}}_{t-k}^{\rightarrow}, \overline{\mathbf{X}}_{t-k}^{\leftarrow}), \quad (3.7)$$

where g_{θ} is a merging function, optionally parameterized by θ . Common strategies include averaging, attention, or a learnable MLP.

While effective, these models treat each time series independently and are agnostic to spatial structure. A first attempt to incorporate relationships between series was proposed by Cao et al. [2018] with the Bidirectional Recurrent Imputation for Time Series (BRITS) architecture. Designed for multivariate time series, BRITS adds a third *concurrent* component in Equation 3.7, which leverages information from other channels at the same time step. For a target variable $\mathbf{X}_{t-k}[i]$, the model estimates

$$\overline{\mathbf{X}}_{t-k}^{\uparrow}[i] = \mathbb{E}_p[p(\mathbf{X}_{t-k}[i] \mid \mathcal{X}_t \setminus \mathbf{X}_{t-k}[i], \mathbf{U}_t, \mathbf{V})], \quad (3.8)$$

where we explicitly remove the target value from the observation set for generality, to account also for the case where it is not missing. This formulation allows the model to exploit concurrent dependencies among variables, but assumes a fully connected multivariate structure in which all time series are treated as equally related. As a result, this approach lacks a principled way to model structured relationships (e.g., spatial proximity, domain knowledge), and scales poorly with the number of time series due to the $\mathcal{O}(N^2)$ number of interactions. Moreover, it struggles to handle settings where the set of nodes varies over time. These limitations are discussed more thoroughly in Chapter 5 (Section 5.1.1), where we examine how such challenges arise more generally in spatiotemporal modeling.

In real-world applications such as sensor networks or traffic systems, incorporating spatial context in a structured fashion is crucial for accurate and scalable imputation. This motivates the extension of autoregressive architectures to spatiotemporal domains by introducing graph-based inductive biases, as we present next.

3.2.2 Graph Recurrent Imputation Network

To address imputation, we introduce GRIN [Cini, Marisca, et al., 2022], a graph-based recurrent architecture (MPRNN) specifically designed for time

series imputation. Given a sequence of partial observations $\mathcal{X}_{t-T:t}$, GRIN learns to reconstruct the missing entries by jointly exploiting temporal and spatial dependencies.

The architecture consists of two unidirectional modules that process the input sequence forward and backward in time. Each module iteratively imputes missing values by incorporating previously imputed observations at every step. Within each time direction, node-level hidden states are updated using a message-passing imputation layer that aggregates information from neighboring nodes. This setup allows the model to interpret each edge as a soft functional dependency, encoding useful relational constraints that bias the model toward spatially coherent imputations.

By leveraging the graph structure in this way, GRIN introduces an inductive bias that aligns well with the nature of the imputation problem. The forward and backward representations are eventually combined using a decoding MLP, which produces a final, refined estimate for each missing value at every node and time step. We begin by describing the architecture of the unidirectional module and then extend it to the full bidirectional model.

Unidirectional GRIN module

We describe here the forward-processing GRIN module; the backward counterpart is obtained by simply reversing the input sequence along the temporal axis. Each unidirectional module is composed of a spatiotemporal encoder followed by a spatial decoder, hence operating in two stages over the input graph sequence.

The spatiotemporal encoder maps the dense input sequence $\mathbf{X}_{t-T:t}$, with missing values indicated by $\mathbf{M}_{t-T:t}$, to a spatiotemporal representation $\mathbf{H}_{t-T:t} \in \mathbb{R}^{T \times N \times d}$ using a tailored recurrent MPNN that captures both temporal dynamics and structural information. An initial imputation is then produced via a linear readout layer, effectively filling in missing values to enable the use of complete inputs in the subsequent decoding stage. Afterwards, the spatial decoder leverages the computed representations to refine these imputations by exploiting the available relational information – i.e., the spatial dependencies – concurrent to the imputation time step. This process is shown graphically in [Figure 3.2](#).

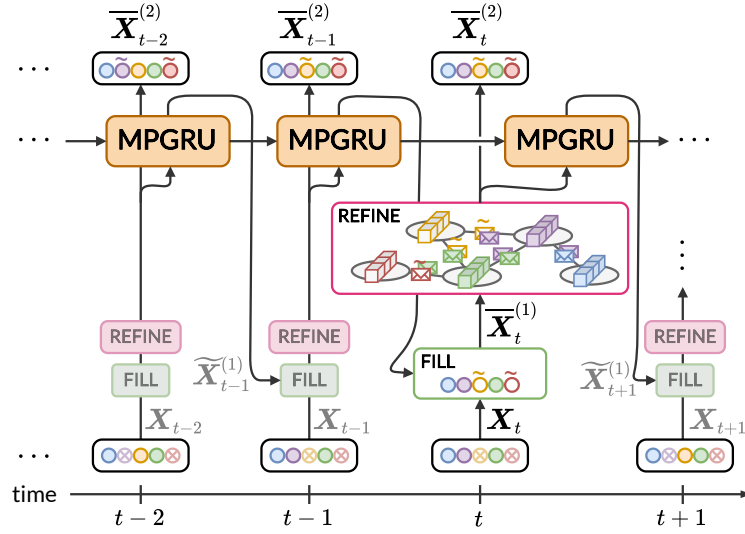


Figure 3.2. The unidirectional GRIN module. The figure focuses on the imputation of the missing values at time t – denoted as $\otimes$ and $\otimes$. A first imputation $\bar{\mathbf{X}}_t^{(1)}$ is obtained as a prediction from the recurrent hidden state. Imputations are later refined through a message-passing operation to incorporate concurrent information at neighboring nodes, obtaining $\bar{\mathbf{X}}_t^{(2)}$.

Spatio-temporal encoder

In the encoder, the input sequence $\mathbf{X}_{t-T:t}$, along with the mask $\mathbf{M}_{t-T:t}$ and covariates $\mathbf{U}_{t-T:t}$, is processed sequentially, one step at a time, by an MPGRU, the graph-based recurrent network presented in Section 2.5.2 (Equations 2.41 to 2.44), following approaches similar to Seo et al. [2018] and Li et al. [2018b]. In general, any message-passing operator can be employed within the recurrent gates; here, MP denotes a generic message-passing neural network, which may consist of one or more layers.

The gates' input is a concatenation of the observations – imputed at the previous processing step – with the concurrent available information, i.e., for a given time step t :

$$\mathbf{R}_t = \sigma \left(\text{MP}_R \left([\bar{\mathbf{X}}_t^{(2)} \parallel \mathbf{M}_t \parallel \mathbf{U}_t \parallel \mathbf{H}_{t-1}], \mathcal{E} \right) \right), \quad (3.9)$$

$$\mathbf{O}_t = \sigma \left(\text{MP}_O \left([\bar{\mathbf{X}}_t^{(2)} \parallel \mathbf{M}_t \parallel \mathbf{U}_t \parallel \mathbf{H}_{t-1}], \mathcal{E} \right) \right), \quad (3.10)$$

$$\mathbf{C}_t = \tanh \left(\text{MP}_C \left([\bar{\mathbf{X}}_t^{(2)} \parallel \mathbf{M}_t \parallel \mathbf{U}_t \parallel \mathbf{R}_t \odot \mathbf{H}_{t-1}], \mathcal{E} \right) \right), \quad (3.11)$$

$$\mathbf{h}_t^i = \mathbf{o}_t^i \odot \mathbf{h}_{t-1}^i + (1 - \mathbf{o}_t^i) \odot \mathbf{c}_t^i. \quad (3.12)$$

Specifically, $\overline{\mathbf{X}}_t^{(2)}$ contains, at each entry, either the corresponding observation (if available) or the imputation from the previous decoding step, as discussed in the next subsection. This approach enables each node to update its hidden state based on both its own input and information propagated from its neighbors. Repeating this process across all nodes and time steps yields the encoded sequence $\mathbf{H}_{t-T:t}$.

Hidden-state initialization When processing a sequence starting at time step t , the initial hidden state $\mathbf{H}_{t-1}$ must be initialized. While simple strategies, e.g., using constants or random noise, are possible, we opt for learnable node-specific parameters randomly initialized and optimized jointly with the rest of the model. This choice allows the encoder to better disambiguate between different time series early in the processing pipeline, enabling a more tailored modeling of local dynamics. A detailed discussion on the impact of such node-specific parameters in STGNN models is provided in [Section 5.3](#).

Spatial decoder

Initially, we generate first-stage imputations as one-step-ahead predictions from the hidden representations of the MPGRU through a linear readout:

$$\widetilde{\mathbf{X}}_t^{(1)} = \mathbf{H}_{t-1} \mathbf{W} + \mathbf{b}, \quad (3.13)$$

$$\overline{\mathbf{X}}_t^{(1)} = \mathbf{M}_t \odot \mathbf{X}_t + (\mathbf{1} - \mathbf{M}_t) \odot \widetilde{\mathbf{X}}_t^{(1)}, \quad (3.14)$$

where $\mathbf{W} \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^d$ are learnable parameters. [Equation 3.14](#) shows that the predictions $\widetilde{\mathbf{X}}_t^{(1)}$ are used to fill in only the missing values in $\mathbf{X}_t$, while observed entries remain unchanged. Notably, $\mathbf{H}_{t-1}$ is computed from past inputs and does not include information from $\mathbf{X}_t$, thus preserving causality.

To further incorporate information concurrent to $\mathbf{x}_t^i$ from the neighbors of node i , we apply a *single-layer* MPNN that refines the imputations by aggregating local context from neighboring nodes. Specifically, we compute a representation $\mathbf{s}_t^i \in \mathbb{R}^d$ as:

$$\mathbf{s}_t^i = \gamma \left(\mathbf{h}_{t-1}^i, \sum_{j \in \mathcal{N}(i) \setminus i} \phi([\overline{\mathbf{x}}_t^{j(1)} \parallel \mathbf{m}_t^j \parallel \mathbf{u}_t^j \parallel \mathbf{h}_{t-1}^j]) \right), \quad (3.15)$$

where $\mathcal{N}(i) \setminus i$ denotes the set of neighboring nodes excluding i itself. This exclusion ensures that the refined representation $\mathbf{s}_t^i$ remains independent of the target node's current input $\mathbf{x}_t^i$ (zero-filled in case of missing value). By

constraining the model to reconstruct the target value using only information from the surrounding subgraph, this design introduces a form of spatial regularization. The use of a single message-passing layer is motivated by this objective, as stacking multiple layers might introduce self-loops via higher-order paths in the graph.

Finally, we combine $\mathbf{H}_{t-1}$ and $\mathbf{S}_t$ through an MLP to generate second-stage imputations $\widetilde{\mathbf{X}}_t^{(2)}$:

$$\widetilde{\mathbf{X}}_t^{(2)} = \text{MLP}([\mathbf{S}_t \parallel \mathbf{H}_{t-1}]), \quad (3.16)$$

$$\overline{\mathbf{X}}_t^{(2)} = \mathbf{M}_t \odot \mathbf{X}_t + (\mathbf{1} - \mathbf{M}_t) \odot \widetilde{\mathbf{X}}_t^{(2)}. \quad (3.17)$$

These updated imputations serve as the input for the next iteration of the encoder MPGRU (Equations 3.9 to 3.12), allowing the model to update its hidden state and continue processing the subsequent input $\mathbf{X}_{t+1}$.

Bidirectional model

Extending GRIN to account for both forward and backward dynamics is straightforward and is achieved by duplicating the architecture described in the previous subsections. The first module processes the sequence in the forward direction (from the beginning to the end), while the second operates in reverse. Once all representations have been computed by both modules, the final imputation is obtained – for each i -th node and time index $k < T$ – through an MLP:

$$\tilde{\mathbf{x}}_{t-k}^i = \text{MLP}([\mathbf{s}_{t-k}^{i,\rightarrow} \parallel \mathbf{h}_{t-k-1}^{i,\rightarrow} \parallel \mathbf{s}_{t-k}^{i,\leftarrow} \parallel \mathbf{h}_{t-k+1}^{i,\leftarrow}]), \quad (3.18)$$

$$\overline{\mathbf{x}}_{t-k}^i = \mathbf{m}_{t-k}^i \odot \mathbf{x}_{t-k}^i + (\mathbf{1} - \mathbf{m}_{t-k}^i) \odot \tilde{\mathbf{x}}_{t-k}^i, \quad (3.19)$$

where the superscripts $\cdot^{\rightarrow}$ and $\cdot^{\leftarrow}$ indicate outputs from the forward and backward modules, respectively.

By design, the model is able to exploit all relevant spatiotemporal information, as the only value explicitly excluded when imputing $\mathbf{x}_t^i$ is the target itself. This guarantees that the reconstruction does not trivially copy the input. Importantly, the model does not act as a simple denoising autoencoder; instead, it follows a predictive approach tailored specifically for the imputation task through its inductive biases. As a result, it produces reconstructions that are structurally consistent across both spatial and temporal dimensions.

3.2.3 Model optimization: a filtering objective

Training GRIN involves minimizing the reconstruction error across all imputation stages in both temporal directions. The overall objective aggregates losses from the final bidirectional output as well as the intermediate predictions from each unidirectional module:

$$\begin{aligned} \tilde{\theta} = \arg \min_{\theta} & \mathcal{L} \left(\widetilde{\mathbf{X}}_{t-T:t}, \mathbf{X}_{t-T:t}, \mathbf{M}_{t-T:t} \right) \\ & + \sum_{l \in \{1,2\}} \mathcal{L} \left(\widetilde{\mathbf{X}}_{t-T:t}^{(l), \rightarrow}, \mathbf{X}_{t-T:t}, \mathbf{M}_{t-T:t} \right) \\ & + \sum_{l \in \{1,2\}} \mathcal{L} \left(\widetilde{\mathbf{X}}_{t-T:t}^{(l), \leftarrow}, \mathbf{X}_{t-T:t}, \mathbf{M}_{t-T:t} \right), \end{aligned} \quad (3.20)$$

where each loss term $\mathcal{L}$ is the filtering loss defined in Equation 2.17, with MAE as entry-wise loss.

Notably, the loss is computed over all observed values in the sequence, leveraging the model's predictive nature. This approach avoids the need to artificially simulate additional missing values during training. To enhance robustness, we still inject synthetic missing data by randomly masking entries with a uniform probability $p = 0.05$ in each batch. These values are excluded from the model input but included in the loss computation, enabling the model to learn from both observed and synthetically masked data, ultimately improving its generalization ability and robustness.

Graph-based filtering An interesting byproduct of the architecture is the generation of intermediate estimates $\widetilde{\mathbf{X}}_{t-k}^{(2)}$ at each time step before accessing the actual observations $\mathbf{X}_{t-k}$. These estimates are produced in a fully predictive fashion, relying solely on the spatiotemporal context captured by the model – specifically, information from previous and subsequent time steps, as well as concurrent inputs from neighboring nodes. Notably, this operation is performed for all entries in the graph, i.e., the model learns to reconstruct each x_{t-k}^i not just when it is missing but also when it is present. By explicitly minimizing the reconstruction loss on both missing and observed values, the model is encouraged to maintain consistency between predicted and actual values across the entire input space. As a result, $\widetilde{\mathbf{X}}_{t-k}^{(2)}$ produced by the same model effectively acts both as imputations of missing observations as well as smoothed estimates of valid ones, achieving the twofold objective of imputation and filtering.

3.3 Related work

Imputing missing values is a long-standing challenge in multivariate time series analysis. Basic heuristics, often favored for their simplicity and low computational overhead, include replacing missing entries with zeros, the last observed value, or summary statistics such as the mean or median computed over available data.

Beyond such elementary strategies and interpolation methods based on polynomial curve fitting and k -nearest neighbors [Troyanskaya et al., 2001; Beretta and Santaniello, 2016], more sophisticated approaches have been developed that leverage classical stochastic frameworks to estimate missing values through likelihood-based inference. Notable examples include methods built on the expectation-maximization algorithm [Ghahramani and Jordan, 1994; Nelwamondo et al., 2007], linear predictors [Box et al., 2015], and state-space models [Durbin and Koopman, 2012; Simon, 2006b]. Within this family of methods, STMVL [Yi et al., 2016] stands out as a linear model that effectively combines temporal and spatial interpolation to address missing data in geospatial time series.

An alternative viewpoint treats imputation as a low-rank matrix completion problem, typically addressed via matrix factorization [Lee and Seung, 1999; Lee and Seung, 2000; Cichocki and Phan, 2009]. This framework has been extended to incorporate spatial [Cai et al., 2010; Rao et al., 2015] and temporal [Yu et al., 2016; Mei et al., 2017] structures by introducing domain-informed priors, commonly implemented through a regularization component.

To model nonlinear dependencies more effectively, deep learning methods [LeCun et al., 2015; Schmidhuber, 2015] have also been adopted. Early efforts focused on autoregressive RNNs, such as GRU-D and BRITS, which capture decay-aware representations or leverage bidirectional recurrence to reconstruct multivariate sequences [Che et al., 2018; Cao et al., 2018; Lipton et al., 2016]. Although these models perform well with a single time series, they often overlook the spatial correlations inherent in spatiotemporal domains.

In parallel, other approaches have reformulated imputation as a generative modeling task. Generative adversarial networks (GANs) [Goodfellow et al., 2014], often combined with RNNs, have been employed to impute missing values by learning to generate subsequences that match the data distribution [Yoon et al., 2018; Fedus et al., 2018; Luo et al., 2018; Luo et al., 2019]. Among these methods, GAIN [Yoon et al., 2018] addresses i.i.d. settings, while the

models by Luo et al. [2018] and Luo et al. [2019] produce realistic synthetic sequences to reconstruct missing data. Building on this idea, SSGAN by Miao et al. [2021] conditions the generator on the predicted label of the incomplete time series. Liu et al. [2019] introduce a multiscale adversarial framework that imputes sparse sequences hierarchically – an idea later refined by Shan et al. [2023]. Although these models yield visually coherent sequences, they are often hindered by training instability and lack explicit mechanisms to incorporate relational inductive biases. Within the generative context, CSDI [Tashiro et al., 2021] leverages score-based diffusion to generate missing values and has recently been extended to account for spatial dependencies in a GRIN-inspired architecture by Liu et al. [2023b].

Some recent efforts have tackled imputation using continuous-time models [Rubanova et al., 2019; Huang et al., 2020], while others have proposed attention-based techniques [Du et al., 2023; Shukla and Marlin, 2020]. However, these approaches typically ignore spatial dependencies. More critically, all the above-cited methods fail to exploit the full expressive power and flexibility afforded by graph-based processing frameworks.

Graph-based methods aim to bridge this gap, with the majority of existing work focused on static graph scenarios [Spinelli et al., 2020; You et al., 2020; Rossi et al., 2022]. For instance, Spinelli et al. [2020] introduce an adversarial training framework for GNNs applied to data reconstruction, while Rossi et al. [2022] propose a strategy grounded in Dirichlet energy minimization (see Section 2.3.1). In more application-specific contexts, graph-based architectures have been employed to impute traffic data [Liang et al., 2021; Ye et al., 2021] and smart grid load profiles [Kuppannagari et al., 2021]. Despite promising results, these models often rely on rigid domain-specific assumptions, which can hinder their applicability in broader settings. More recently, Dimitri et al. [2024] used GNNs to impute missing values in meteorological data by modeling the observations through a horizontal visibility graph [Luque et al., 2009].

For a more comprehensive perspective, Fang and Wang [2020] and Wang et al. [2024] survey deep learning methods for time series imputation, with the latter offering an in-depth examination of graph-based techniques.

Related work on filtering

Filtering can be viewed as an intermediate task between imputation and forecasting, sharing characteristics with both. Like imputation, it operates within

the range of observed data, aiming to improve the quality of already acquired values. At the same time, it resembles forecasting in that it targets observations for which we have – or will have – a ground-truth reference. As a result, many methods originally developed for imputation or forecasting have been successfully adapted to filtering. In the following, we highlight several key approaches specifically designed for smoothing.

Linear filtering methods have been widely applied to multivariate time series [Holloway Jr, 1958]. Moving averages apply uniform weights to observations within a given window, while kernel smoothing generalizes this by assigning smoothly varying weights. An example is the Weierstrass transform, which convolves the signal with a Gaussian function. Infinite impulse response filters, such as exponential smoothing, have also been considered. These methods extend to spatial domains, including both regularly sampled data (e.g., images) and graph-structured data, with Laplacian smoothing enforcing local consistency between neighboring nodes [Herrmann, 1976; Dong et al., 2016].

Smoothing splines and penalized least squares methods aim to balance fidelity to data and smoothness constraints [Wahba, 1990; Eilers and Marx, 1996]. Examples include the Whittaker–Henderson smoother [Whittaker, 1922; Henderson, 1924] and cubic smoothing splines. Recent extensions, such as the spatiotemporal sandwich smoother [French and Kokoszka, 2021; Xiao et al., 2013], further refine these techniques for structured domains. Probabilistic approaches such as the Kalman filter [Kalman, 1960] have been adapted for smoothing [Simon, 2006a], estimating the hidden state distribution when multiple noisy measurements are available [Anderson and Moore, 2005]. As an example, the Rauch-Tung-Striebel (RTS) smoother provides the minimum mean square error estimate for linear state-space models with Gaussian noise [Rauch et al., 1965]. Similarly, spatiotemporal autoregressive models explicitly model dependencies across both dimensions [MacNab and Dean, 2001; Messnarz et al., 2004; Katzfuss and Cressie, 2011]. More recently, partial differential equation (PDE)-regularized methods have emerged as powerful tools for smoothing irregular spatiotemporal data [Ponti et al., 2022; Arnone et al., 2023]. These methods incorporate problem-specific differential equations as constraints, ensuring physically consistent reconstructions. None of these approaches, however, explicitly accounts for relational dependencies using graphs.

3.4 Empirical evaluation

In this section, we present a comprehensive empirical evaluation of the proposed GRIN architecture across diverse imputation scenarios. Our goal is to assess its reconstruction accuracy, robustness to different missingness patterns, and versatility across application domains. We compare GRIN against a wide range of baselines, introduced in [Section 3.4.1](#), including classical statistical methods, matrix factorization techniques, and recent deep learning models, with a particular focus on autoregressive approaches such as BRITS.

Experiments are conducted on four real-world datasets spanning environmental monitoring, traffic analytics, and energy consumption (see [Section 2.6](#)). We simulate realistic missingness regimes and evaluate performance under both in-sample and out-of-sample settings. We begin with air quality datasets ([Section 3.4.2](#)), which feature structured missing blocks and allow direct comparison with prior work. Next, we evaluate performance on traffic and energy benchmarks ([Section 3.4.3](#)), focusing on generalization across domains. We then conduct an ablation study to isolate the contribution of key architectural components in GRIN ([Section 3.4.4](#)). Finally, we explore the model’s capabilities in a virtual sensing setup, testing its ability to extrapolate entire time series at unobserved nodes ([Section 3.4.5](#)).

Throughout, reconstruction accuracy is measured using the mean absolute error (MAE), computed over the imputation windows. All experiments employ DCNN [[Atwood and Towsley, 2016](#)] as the MPNN in GRIN, introduced earlier in [Section 2.3.2](#). The results demonstrate that GRIN consistently achieves state-of-the-art accuracy while maintaining architectural flexibility and scalability.

3.4.1 Imputation baselines

We consider BRITS [[Cao et al., 2018](#)] as the principal competing alternative among non-adversarial deep autoregressive approaches, as it shares architectural similarities with our methods. As additional baselines, we consider:

Mean. Imputation using the node-level average.

KNN. Imputation by averaging values of the $k = 10$ neighboring nodes with the highest weight in the adjacency matrix $\mathbf{A}$.

MICE. The Multiple Imputation using Chained Equations method by White et al. [[2011](#)], limiting the maximum number of iterations to 100 and the

Table 3.1. Imputation error (MAE) on the air quality datasets.

	AQI		AQI-36	
	In-sample	Out-of-sample	In-sample	Out-of-sample
Mean	39.60 $\pm$ 0.00	39.60 $\pm$ 0.00	53.48 $\pm$ 0.00	53.48 $\pm$ 0.00
KNN	34.10 $\pm$ 0.00	34.10 $\pm$ 0.00	30.21 $\pm$ 0.00	30.21 $\pm$ 0.00
MF	26.74 $\pm$ 0.24	N/A	30.54 $\pm$ 0.26	N/A
MICE	26.39 $\pm$ 0.13	26.98 $\pm$ 0.10	29.89 $\pm$ 0.11	30.37 $\pm$ 0.09
VAR	18.13 $\pm$ 0.84	22.95 $\pm$ 0.30	13.16 $\pm$ 0.21	15.64 $\pm$ 0.08
rGAIN	17.69 $\pm$ 0.17	21.78 $\pm$ 0.50	12.23 $\pm$ 0.17	15.37 $\pm$ 0.26
BRITS	17.24 $\pm$ 0.13	20.21 $\pm$ 0.22	12.24 $\pm$ 0.26	14.50 $\pm$ 0.35
MPGRU	15.80 $\pm$ 0.05	18.76 $\pm$ 0.11	12.46 $\pm$ 0.35	16.79 $\pm$ 0.52
GRIN	13.10 $\pm$ 0.08	14.73 $\pm$ 0.15	10.51 $\pm$ 0.28	12.08 $\pm$ 0.47

number of nearest features to 10.

MF. Matrix factorization with rank $r = 10$ [Lee and Seung, 2000].

VAR. A Vector Autoregressive one-step-ahead predictor [Box et al., 2015].

rGAIN. An unsupervised version of SSGAN [Miao et al., 2021], which can be seen as GAIN [Yoon et al., 2018] with bidirectional recurrent encoder and decoder.

MPGRU. A message-passing GRU with MLP predictive readout.

3.4.2 Air quality

We evaluate imputation performance on air quality datasets using the protocol established in prior work [Yi et al., 2016; Cao et al., 2018]. Specifically, following Yi et al. [2016], we define the months of March, June, September, and December as the test set. A data points are held out from training and used as ground-truth label if the observation associated with the corresponding sensor, day, and hour is missing in the following month. This strategy accentuates the presence of block missing patterns (see Section 2.1.3), effectively turning the dataset into a stress test for imputation models.

We assess performance under both in-sample and out-of-sample settings (see Section 3.1.2). In the out-of-sample case, we discard all windows that overlap

with the designated test months. Window lengths are set to $T = 24$ for AQI and $T = 36$ for AQI-36, in alignment with the protocol from Cao et al. [2018].

Results

Table 3.1 summarizes the imputation results on the air quality datasets. For in-sample evaluation, we compute metrics using the value obtained by averaging imputations across all overlapping windows. In the out-of-sample setting, we instead report the average error across windows. GRIN substantially outperforms all competing baselines consistently over all scenarios. In particular, for AQI in the out-of-sample case, it reduces the MAE by over 20% relative to the next best method.

A particularly noteworthy result is that GRIN outperforms BRITS even for sensors associated with isolated nodes, i.e., stations located more than 40 km away from their nearest neighbor. Furthermore, the model maintains superior accuracy on the lower-dimensional AQI-36 dataset, despite the expectation that graph structure would play a less prominent role.

3.4.3 Traffic and energy analytics

For the traffic and energy consumption datasets, we consider only the out-of-sample scenario (except for matrix factorization, which only works in-sample) and use input sequences of 24 steps, corresponding to 2 hours of data for traffic datasets and 12 for CER-E.

We simulate the presence of missing data by considering two different settings. In the **Point** setting, we randomly mask out a fraction $\eta = 0.25$ of the available data. In the **Block-T** setting, in addition to randomly dropping $\eta = 0.05$ of the available data, we simulate for each node and time step a failure with probability $p_f = 0.15\%$ and sample its duration uniformly in the interval $[s_{\min}, s_{\max}]$. We set $s_{\min}$ and $s_{\max}$ as the number of time steps corresponding respectively to 1 and 4 hours in the traffic case and 2 hours and 2 days for CER-E.

Results

Table 3.2 presents the imputation results on the traffic and smart grid datasets. Across all scenarios, GRIN consistently outperforms both BRITS and rGAIN, while requiring significantly fewer parameters. In the traffic datasets, it achieves an average MAE reduction of approximately 29% relative to BRITS. The improve-

Table 3.2. Imputation error (MAE) on the traffic and smart grids datasets.

	Point			Block-T		
	METR-LA	PEMS-BAY	CER-E	METR-LA	PEMS-BAY	CER-E
Mean	7.56 $\pm$ 0.00	5.42 $\pm$ 0.00	1.51 $\pm$ 0.00	7.48 $\pm$ 0.00	5.46 $\pm$ 0.00	1.49 $\pm$ 0.00
KNN	7.88 $\pm$ 0.00	4.30 $\pm$ 0.00	1.22 $\pm$ 0.00	7.79 $\pm$ 0.00	4.30 $\pm$ 0.00	1.15 $\pm$ 0.00
MF	5.56 $\pm$ 0.03	3.29 $\pm$ 0.01	1.01 $\pm$ 0.01	5.46 $\pm$ 0.02	3.28 $\pm$ 0.01	0.97 $\pm$ 0.01
MICE	4.42 $\pm$ 0.07	3.09 $\pm$ 0.02	0.98 $\pm$ 0.00	4.22 $\pm$ 0.05	2.94 $\pm$ 0.02	0.96 $\pm$ 0.01
VAR	2.69 $\pm$ 0.00	1.30 $\pm$ 0.00	0.53 $\pm$ 0.00	3.11 $\pm$ 0.08	2.09 $\pm$ 0.10	0.64 $\pm$ 0.03
rGAIN	2.83 $\pm$ 0.01	1.88 $\pm$ 0.02	0.71 $\pm$ 0.00	2.90 $\pm$ 0.01	2.18 $\pm$ 0.01	0.74 $\pm$ 0.00
BRITS	2.34 $\pm$ 0.00	1.47 $\pm$ 0.00	0.64 $\pm$ 0.00	2.34 $\pm$ 0.01	1.70 $\pm$ 0.01	0.64 $\pm$ 0.00
MPGRU	2.44 $\pm$ 0.00	1.11 $\pm$ 0.00	0.41 $\pm$ 0.00	2.57 $\pm$ 0.01	1.59 $\pm$ 0.00	0.53 $\pm$ 0.00
GRIN	1.91 $\pm$0.00	0.67 $\pm$0.00	0.29 $\pm$0.00	2.03 $\pm$0.00	1.14 $\pm$0.01	0.42 $\pm$0.00

ment is especially pronounced in the **Point** setting of the PEMS-BAY dataset, where the error is effectively halved.

On the CER-E dataset, GRIN continues to outperform all baselines, demonstrating its robustness across application domains. Importantly, this experiment shows that GRIN remains effective even when explicit relational information is not available, highlighting its flexibility beyond settings with clear relational information.

3.4.4 Ablation study

In this subsection, we investigate the contributions of key architectural components of GRIN by means of an ablation study. Specifically, we focus on evaluating the effect of the spatial decoder and the bidirectional design.

Table 3.3 reports the out-of-sample MAE of the ablation study on three datasets: AQI, METR-LA, and CER-E – under both **Point** and **Block-T** settings. We compare GRIN with three baselines to isolate the impact of its core components. The first baseline removes the spatial decoder, hence acting as a bidirectional MPGRU model with an MLP that harmonizes hidden states $\mathbf{h}_{t-1}^{i,\rightarrow}$ and $\mathbf{h}_{t+1}^{i,\leftarrow}$ to produce the imputation $\bar{\mathbf{x}}_t^i$. The second baseline adopts a similar structure, but uses hidden states at time step t itself from both directions, thus acting similarly to a denoising autoencoder. For reference, we also report the performance of a unidirectional MPGRU model.

Table 3.3. Ablation study of GRIN (MAE).

	AQI	METR-LA		CER-E	
		Point	Block-T	Point	Block-T
GRIN	14.73 ± 0.15	1.91 ± 0.00	2.03 ± 0.00	0.29 ± 0.00	0.42 ± 0.00
without spatial decoder	15.40 ± 0.14	2.01 ± 0.00	2.32 ± 0.01	0.29 ± 0.00	0.49 ± 0.00
with denoise decoder	17.23 ± 1.12	2.09 ± 0.02	2.96 ± 0.18	0.32 ± 0.00	0.61 ± 0.02
MPGRU	18.76 ± 0.11	2.44 ± 0.00	2.57 ± 0.01	0.41 ± 0.00	0.53 ± 0.00

The results confirm that each component introduced in GRIN contributes meaningfully to reducing imputation error. Both the spatial decoder and the bidirectional processing prove essential for reconstructing missing values accurately, particularly in realistic scenarios with block-wise missing data. Interestingly, the denoising variant struggles in the **Block-T** setting but performs relatively well in the **Point** case, as might be expected.

3.4.5 Virtual sensing

We conclude our evaluation by assessing the effectiveness of GRIN in a virtual sensing setup, where the goal is to infer the measurements of a sensor for which no data are available. To simulate this, we add a node to the graph with all its observations masked and allow the model to reconstruct the corresponding time series.

This task relies on a few key assumptions: (1) the monitored quantity must be inferable from nearby sensors; and (2) either sensor characteristics are relatively homogeneous (e.g., air quality stations are installed at comparable heights), or relevant node-specific features (e.g., geographical placement) are available to the model. Importantly, due to its relational inductive biases, GRIN reconstructs missing values not only by minimizing the error at the target node, but also by regularizing imputation through spatial consistency with neighboring sensors.

To validate this, we masked all data from the two sensors in AQI-36 with the highest and lowest node degrees: station 1015 (highly connected) and station 1032 (sparsely connected). GRIN was trained on the remaining data as usual. As shown in [Figure 3.3](#), the model accurately reconstructs low-frequency components, e.g., the trend and scale, for both sensors. Quantitatively, it achieves an average MAE of 11.74 on station 1015 and 20.00 on station 1032, emphasizing the positive impact of spatial dependencies.

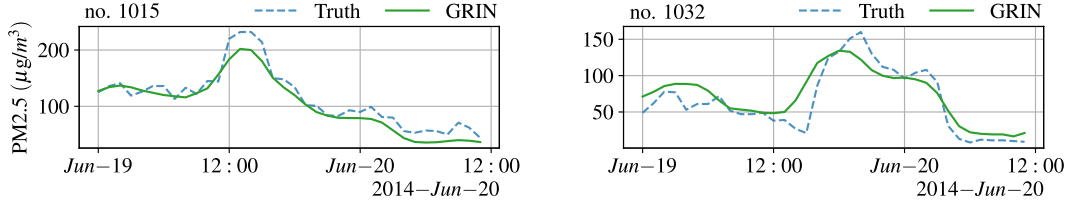


Figure 3.3. Reconstruction of observations at virtual sensors, i.e., nodes completely removed from the training set.

3.5 Discussion and future directions

This chapter addressed the task of imputing missing values in irregular spatiotemporal data using graph-based methods. We began by formalizing the imputation problem and showing how relational priors encoded through graph structures can guide reconstructions in a principled and localized way. Leveraging the flexibility of GNNs, we proposed a general modeling framework based on spatiotemporal message passing, which unifies imputation, filtering, and virtual sensing under a single architectural paradigm. These connections are not only conceptually appealing but also offer practical advantages, as they allow the same model to smooth noisy data or infer entirely unobserved sequences without retraining.

In the second part of the chapter, we introduced GRIN, an autoregressive STGNN architecture for imputation that explicitly incorporates relational information. Built upon recurrent GNNs, GRIN performs bidirectional spatiotemporal processing to reconstruct missing data based on past, future, and concurrent context, while simultaneously enforcing spatial consistency through a message-passing decoder. By design, the model combines predictive encoding with relational regularization, enabling it to produce reconstructions that are not only accurate but also coherent across both space and time. Empirical evaluations across diverse real-world benchmarks demonstrate the effectiveness of our approach. GRIN achieves state-of-the-art performance across multiple datasets and operational settings, including challenging scenarios with blocks of missing data.

The work presented in this chapter opens up several directions for future investigation. From a theoretical standpoint, a deeper understanding of the structural and statistical conditions that guarantee accurate reconstructions remains an open question. Extending the current framework to operate in non-stationary environments – where the graph topology or the underlying data-generating processes evolve over time – is another promising direction.

Moreover, the application of GRIN to virtual and active sensing tasks could be further explored, especially in settings with limited sensor coverage or when optimal sensor placement is of interest.

Finally, while autoregressive architectures like GRIN excel at exploiting local dependencies, they inherently suffer from compounding errors in cases of long blocks of missing data, where predictive uncertainty accumulates over time. In the next chapter, we target precisely this challenging regime and introduce a flexible architecture that makes imputation feasible even under high sparsity.

Chapter 4

Imputation with sparse observations

Autoregressive architectures like GRIN approach the imputation problem by splitting it into three distinct components, each handled by a dedicated module. The forward module captures the predictive distribution $p(\mathbf{x}_{t-k}^i | \mathcal{X}_{<t-k})$ by forecasting from past observations. A backward module mirrors this behavior, modeling $p(\mathbf{x}_{t-k}^i | \mathcal{X}_{t-k:t})$ by running inference on reversed sequences conditioned on future context. A third module focuses on spatial dependencies, estimating $p(\mathbf{x}_{t-k}^i | \{\mathcal{X}_{t-k}\} \setminus \mathbf{x}_{t-k}^i)$ based on concurrent observations at the other nodes. This decomposition is well-suited for cases where missing values are scattered and surrounded by nearby observations, as it leverages strong regularization across both space and time.

However, this design comes with significant limitations. The compartmentalization of the three components can restrict the model’s ability to capture global dependencies. It also introduces extra computational cost, as the three processing lines must be executed independently. More importantly, these models are autoregressive in nature, meaning that predictions are used to condition future steps. As a result, they suffer from error compounding, as prediction errors accumulate over time and across nodes, in particular when multiple consecutive values are missing. This issue becomes critical in sparse scenarios, where observations are few and far apart, making reliable propagation across time and space significantly harder.

To overcome these challenges, we propose an architecture explicitly aligned with the imputation objective. Our model employs a sparse spatiotemporal attention

mechanism grounded in the message-passing paradigm. In this framework, each available observation is treated as a spatiotemporal token. The model then performs a self-organizing propagation process that learns to infer each missing value directly from observed data, without relying on prior imputations. This approach allows the model to learn a predictive representation for each missing value by conditioning only on the observed portion of the data, effectively modeling $p(\mathbf{x}_{t-k}^i | \mathcal{X}_{\leq t})$. By avoiding autoregressive propagation, we eliminate the bias accumulation common in existing approaches. The result is a robust, flexible framework capable of imputing missing values at arbitrary positions in space and time, with particular strength in high-sparsity regimes.

The chapter is organized as follows. [Section 4.1](#) introduces the learning formulation and explains how the imputation task is recast as a point-wise inference problem over spatiotemporal coordinates. [Section 4.2](#) discusses strategies to scale the architecture to large graphs and long sequences, presenting two complementary techniques based on graph subsampling and hierarchical attention. We then explain how to derive or learn the aforementioned spatiotemporal coordinates if missing in [Section 4.3](#). An extensive empirical evaluation is presented in [Section 4.5](#), including benchmarks against state-of-the-art methods, sensitivity analysis under extreme sparsity, an ablation study, and a qualitative assessment of virtual sensing. We conclude the chapter with a discussion of key findings and future research directions.

Reference papers

This chapter’s content is partly based on material from the following articles.

Ivan Marisca, Andrea Cini, and Cesare Alippi (2022). **Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations**. In: *Advances in Neural Information Processing Systems*. Vol. 35, pp. 32069–32082.

4.1 Learning from sparse spatiotemporal data

As discussed above, the autoregressive approach followed by GRIN consisted of targeting the underlying process by decomposing it into three components over the temporal dimension. Here instead, we align directly with the imputation objective of the family of graph-based imputation models $\mathcal{I}(\cdot)$ introduced in [Chapter 3](#) and reported here for clarity:

$$\begin{aligned}\overline{\mathbf{X}}_{t-T:t} &= \mathcal{I}(\mathcal{X}_{t-T:t}, \mathbf{U}_{t-T:t}, \mathbf{V}, \mathcal{E}; \boldsymbol{\theta}) \\ &\approx \mathbb{E}_p [p(\mathbf{X}_{t-T:t} \mid \mathcal{X}_{\leq t}, \mathbf{U}_{\leq t}, \mathbf{V})].\end{aligned}\tag{3.1}$$

Specifically, we assume the sensor-level covariates $\mathbf{U}_t \in \mathbb{R}^{N \times d_u}$ to act as spatiotemporal coordinates that localize a point in time and space (e.g., date/time features and geographic location). Recall that exogenous $\mathbf{u}_t^i$ are assumed available for each node and time step, even when $\mathbf{x}_t^i$ is missing. Later in [Section 4.3](#), we discuss how $\mathbf{U}_t$ can be learned as a spatiotemporal positional encoding if unavailable or insufficient.

Our approach, named Spatiotemporal Point Inference Network (SPIN) [[Marisca et al., 2022](#)], is an attention-based STGNN for imputation, designed to learn representations of discrete points associated with spatiotemporal coordinates. Let $\mathcal{Y}_{t-T:t} = \{\mathbf{u}_{t-k}^i \mid \mathbf{m}_{t-k}^i = \mathbf{0}\}_{k < T}$ be the *target set* collecting the coordinates of the discrete spatiotemporal points for which we want to reconstruct an observation. Note that this set has an element for each missing one in $\mathcal{X}_{t-T:t}$. Then, for all $\mathbf{u}_{t-k}^i \in \mathcal{Y}_{t-T:t}$, we aim at learning a model f as in [Equation 3.1](#) such that

$$f(\mathcal{X}_{t-T:t}, \mathbf{U}_{t-T:t}, \mathbf{V}, \mathcal{E}; \boldsymbol{\theta}, \mathbf{u}_t^i) \approx \mathbb{E}_p [p(\mathbf{x}_{t-k}^i \mid \mathcal{X}_{\leq t}, \mathbf{U}_{\leq t}, \mathbf{V})].\tag{4.1}$$

To this end, we set up a propagation process where each representation $\mathbf{h}_t^i \in \mathbb{R}^d$, associated with the i -th node and time step t , is updated by aggregating information only from available observations at the i -th node and its neighbors, weighted by input-dependent scores. The representation is initialized by encoding all the available information at node i and time t – either $\mathbf{x}_t^i$ and $\mathbf{u}_t^i$ for observed values or only $\mathbf{u}_t^i$ for missing values – and then updated by an ad-hoc spatiotemporal attention mechanism. [Figure 4.1](#) shows an overview of the architecture. In the next subsections, we go through each component explaining it in detail and providing motivations for each design choice; we begin with the sparse spatiotemporal attention mechanism.

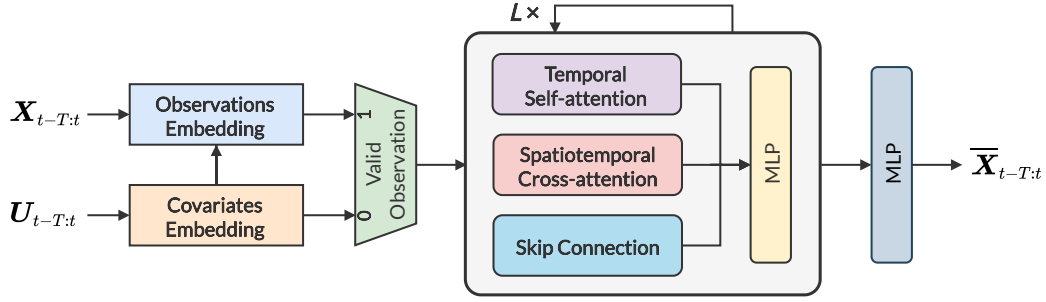


Figure 4.1. The architecture of SPIN. At first, we encode observations $\mathbf{X}_{t-T:t}$ and spatiotemporal coordinates $\mathbf{U}_{t-T:t}$, obtaining initial representations $\mathbf{H}_{t-T:t}^{(0)}$. The representations are updated by a stack of L sparse spatiotemporal attention blocks. Final imputations are obtained from $\mathbf{H}_{t-T:t}^{(L)}$ with an MLP.

4.1.1 Sparse spatiotemporal attention

The core component of SPIN is a sparse spatiotemporal attention layer used to propagate information at the level of single observations. Indeed, leveraging the attention mechanism [Vaswani et al., 2017], we learn representations for each i -th node at each time step t by simultaneously aggregating information from (1) the available observations at the i -th node $\mathbf{x}_{t-k}^i \in \mathcal{X}_{t:t+T}$ and (2) the available observations $\mathbf{x}_{t-k}^j \in \mathcal{X}_{t:t+T}$ at each node $j \in \mathcal{N}(i)$. Figure 4.2 shows a schematic representation of this procedure.

Let $\mathbf{h}_{t-k}^{i(l)} \in \mathbb{R}^d$ be the learned representation for the i -th node and time step $t-k$ at the l -th layer. The representation is initialized in the ENCODER (Equation 3.2) with an MLP with different weights depending on whether the associated observation is available or not, i.e.,

$$\mathbf{h}_{t-k}^{i(0)} = \begin{cases} \text{MLP}_X([\mathbf{x}_{t-k}^i \parallel \mathbf{u}_{t-k}^i]) & \exists \mathbf{x}_{t-k}^i, \\ \text{MLP}_Y(\mathbf{u}_{t-k}^i) & \nexists \mathbf{x}_{t-k}^i. \end{cases} \quad (4.2)$$

The representation is then updated by joint temporal and spatiotemporal attention operations [Vaswani et al., 2017]. The next steps involve computations of spatiotemporal messages, i.e., representations computed to propagate information from one discrete space-time point to another. We indicate the propagation along the temporal dimension from time step $t-s$ to time step $t-k$ with subscripts $\cdot_{s \rightarrow k}$. Similarly, superscripts $\cdot^{j \rightarrow i}$ indicate messages sent from the j -th to the i -th node. To avoid overloading the notation, we omit the layer superscript in the following.

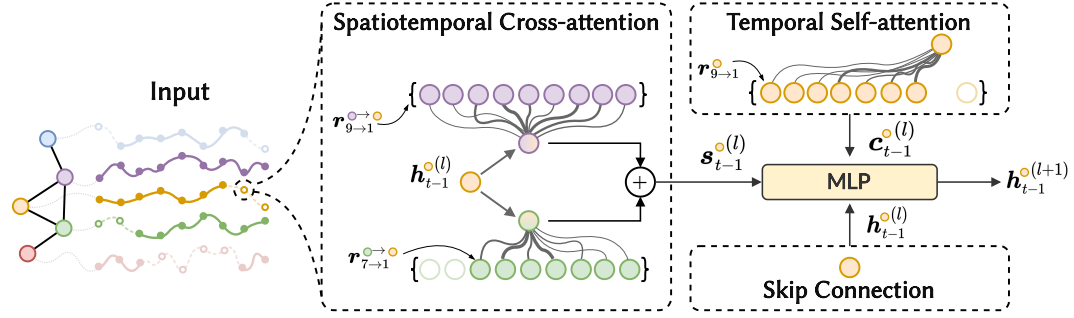


Figure 4.2. The sparse spatiotemporal attention mechanism in SPIN. To compute target representation $h_{t-1}^{(l+1)}$, the layer integrates information from (1) an inter-node spatiotemporal cross-attention, (2) an intra-node temporal self-attention, and (3) a residual connection.

For each available observation, we compute message $r_{s \rightarrow k}^{j \rightarrow i} \in \mathbb{R}^d$ from each j -th node at time step $t-s$ to each i -th node at time step $t-k$ as:

$$r_{s \rightarrow k}^{j \rightarrow i} = \text{MLP}([h_{t-s}^j \parallel h_{t-k}^i]) \in \mathbb{R}^d, \quad (4.3)$$

where, following the adopted terminology of Vaswani et al. [2017], target point representation h_{t-k}^i acts as query, source point representation h_{t-s}^j as key and the computed message $r_{s \rightarrow k}^{j \rightarrow i}$ is the value. Notice that, differently from the dot-product attention mechanism, we additively combine both source and target representations to compute the value, similar to the Bahdanau attention [Bahdanau et al., 2014] presented in Section 2.4.3. This mechanism is used to perform (1) an *intra-node* temporal self-attention to exploit observations at the same node and (2) an *inter-node* spatiotemporal cross-attention to account for spatial information at neighboring nodes. Note that, while the adopted mechanism is the same, we use different parameters for the self-attention and cross-attention modules. Specifically, focusing on a target representation h_{t-k}^i , we use this representation to query separately every encoding in $h_{t-T:t}^i$ and $h_{t-T:t}^j$ for every neighbor $j \in \mathcal{N}(i)$ associated with a valid observation.

We compute and assign a score $\alpha_{s \rightarrow k}^{j \rightarrow i}$ to each message $r_{s \rightarrow k}^{j \rightarrow i}$ through a softmax aggregation as

$$\alpha_{s \rightarrow k}^{j \rightarrow i} = \frac{\exp(\mathbf{w}_\alpha^\top r_{s \rightarrow k}^{j \rightarrow i})}{\sum_{p: \exists x_{t-p}^j \exp(\mathbf{w}_\alpha^\top r_{p \rightarrow k}^{j \rightarrow i})}. \quad (4.4)$$

where $\mathbf{w}_\alpha \in \mathbb{R}^d$ is a vector of learnable parameters and the denominator aggregates all existing messages such that x_{t-p}^j is available. To indicate intra-node messages we simply write $r_{s \rightarrow k}^i = r_{s \rightarrow k}^{i \rightarrow i}$. We weigh and aggregate all

messages $\mathbf{r}_{s \rightarrow k}^i$ going towards $t-k$ to obtain a temporal context vector $\mathbf{c}_{t-k}^i$ as

$$\mathbf{c}_{t-k}^i = \sum_s \alpha_{s \rightarrow k}^i \cdot \mathbf{r}_{s \rightarrow k}^i. \quad (4.5)$$

Similarly, we aggregate temporal messages coming from each j -th node separately, and sum them to obtain a spatial context vector $\mathbf{s}_{t-k}^i$ as

$$\mathbf{s}_{t-k}^i = \sum_{j \in \mathcal{N}(i)} \sum_s \alpha_{s \rightarrow k}^{j \rightarrow i} \cdot \mathbf{r}_{s \rightarrow k}^{j \rightarrow i}, \quad (4.6)$$

which encodes the observed sequences at all neighbors w.r.t. the i -th node and time step $t-k$.

Finally, the encoding $\mathbf{h}_{t-k}^{i(l+1)}$ is computed by combining context vectors $\mathbf{c}_{t-k}^{i(l)}$ and $\mathbf{s}_{t-k}^{i(l)}$, with the previous-layer encoding $\mathbf{h}_{t-k}^{i(l)}$ through an MLP:

$$\mathbf{h}_{t-k}^{i(l+1)} = \text{MLP} \left([\mathbf{h}_{t-k}^{i(l)} \parallel \mathbf{c}_{t-k}^{i(l)} \parallel \mathbf{s}_{t-k}^{i(l)}] \right). \quad (4.7)$$

After L layers, we obtain imputations for each missing value at the i -th node and time step t with a nonlinear readout:

$$\bar{\mathbf{x}}_t^i = \text{MLP} \left(\mathbf{h}_t^{i(L)} \right). \quad (4.8)$$

4.1.2 Two-phase propagation

In the spatiotemporal attention layer, information is propagated exclusively from nodes and time steps associated with valid observations. While this design reinforces the focus on observed data, it also blocks information flow along paths that traverse spatiotemporal locations with missing observations. This can become a limiting factor in settings with high sparsity, where missing values far exceed observed ones. However, it is reasonable to expect that after a few propagation steps, the influence of available observations has already partially diffused into regions with missing data, enabling the associated representations to accumulate enough context to support imputation. Once this occurs, blocked paths can be safely unlocked to allow information – bootstrapped from valid observations – to propagate to higher-order neighborhoods.

To implement this idea, we introduce a hyperparameter π that determines the number of initial layers in which such restrictions are enforced. The propagation process is thus divided into two phases: during the first π layers, information is allowed to flow towards missing points but not from them; in the remaining $L - \pi$ layers, this restriction is relaxed, enabling unrestricted bidirectional flow across space and time.

4.1.3 Model optimization

To train SPIN, we minimize the imputation loss defined in Equation 2.16, computed at each layer of the model:

$$\mathcal{L} = \sum_{l=0}^{L-1} \sum_{k=0}^{T-1} \sum_{i=0}^{N-1} \frac{\left\| \bar{\mathbf{m}}_{t-k}^i \odot \ell \left(\bar{\mathbf{x}}_{t-k}^{i(l)}, \mathbf{x}_{t-k}^i \right) \right\|_1}{\left\| \bar{\mathbf{M}}_{t-T:t} \right\|_1} \quad (4.9)$$

where $\ell(\cdot, \cdot)$ is the absolute error and $\bar{\mathbf{x}}_{t-k}^{i(l)}$ is the imputed value produced by the l -th layer for node i at time step $t-k$. Note that, to provide more supervision to the architecture, the loss is computed and backpropagated not only at the final layer but across all intermediate layers. Importantly, the loss is evaluated only on data points that are not visible to the model during the forward pass. To simulate this, we randomly mask out a subset of observations for each sample in the mini-batch, using a masking probability drawn uniformly from $\{0.2, 0.5, 0.8\}$. These held-out values are then used to compute the imputation loss. Unlike GRIN, which instead adopts a reconstruction loss over the available observations, SPIN relies exclusively on this masked data strategy for supervision.

4.2 Addressing the computational scalability

The proposed spatiotemporal attention mechanism can be conceptually understood as operating over a subgraph product between the spatial graph $\mathcal{E}$ and a temporal path graph of length T , similar to the one shown in Figure 2.3 (Page 32), with the subgraph structure arising due to the pruning of connections associated with missing observations. As a result, the proposed attention mechanism incurs time and memory complexities that scale with $\mathcal{O}((N + |\mathcal{E}|)T^2)$. In scenarios with sparse data, this cost can be partially alleviated by restricting computation to valid observations, which are far fewer than the full NT grid. However, in more general settings with dense or moderately sparse data, the method becomes computationally prohibitive, particularly when scaling to large graphs or long time intervals. To address this limitation and enable broader applicability, we introduce two complementary strategies aimed at reducing the computational burden of spatiotemporal attention.

4.2.1 Graph subsampling

A straightforward strategy to reduce the computational cost of the model is to adopt graph subsampling during training, leveraging established techniques

from the literature (e.g., [Zeng et al., 2020]). Specifically, we sample a k -hop subgraph centered around a set of n target nodes, and restrict both the forward and backward computations to this subgraph. The loss is then computed solely with respect to the selected n target nodes. This approach is particularly well-suited to our architecture due to its inherently node-centric formulation: the model processes and predicts each node’s value independently, conditioned on its local spatiotemporal neighborhood. As such, predictions for a subset of nodes do not require access to the full graph, but only to the subgraph covered by the spatial receptive field. This modular structure enables efficient localized training without loss of accuracy.

In addition to lowering the memory and time complexity by controlling the subgraph size via n and k , this method implicitly introduces a form of regularization. By varying the sampled subgraphs across training iterations, the model learns to generalize under different local contexts, reducing overfitting to specific graph regions [Rong et al., 2020]. However, a key limitation of this strategy is that in deep architectures – where the receptive field spans large portions of the graph – accurately reproducing the predictions of the full model requires subsampling neighborhoods that grow to coincide with the entire graph. As a result, the computational benefits vanish as depth increases, motivating the need for alternative solutions that maintain efficiency regardless of model depth.

4.2.2 Hierarchical attention

An orthogonal and more principled solution to address the scalability bottleneck is to redesign the spatiotemporal attention mechanism to operate hierarchically over the temporal dimension, similar to previous work [Ainslie et al., 2020]. We achieve this by introducing K routers for each node $i \in \mathcal{V}$, which serve as compact summarizers of temporal information. Let $\mathbf{Z}^i \in \mathbb{R}^{K \times d_z}$ represent the K router embeddings for node i , where $\mathbf{z}_k^i = \mathbf{Z}^i[k]$ denotes the k -th router. These routers allow the model to bypass dense attention across all time steps, reducing complexity from quadratic to linear in T . The initial router embeddings at layer $l = 0$ are learnable parameters initialized randomly, which also capture long-term memory associated with the nodes (more details later in Section 5.3).

The hierarchical attention unfolds in two phases. In the first phase, each router $\mathbf{z}_k^i$ is updated by attending to all valid temporal representations $\mathbf{h}_{t-s}^i$ in $\mathbf{H}_{t-T:t}$, resulting in an updated representation $\tilde{\mathbf{z}}_k^i$. This aggregation is computed using the same message and attention structure as in Equations 4.3 to 4.5, but condensed into a temporal context vector $\mathbf{c}_k^i$ for each router rather than time

step. The update operation mirrors [Equation 4.7](#):

$$\tilde{z}_k^i = \text{MLP} \left([z_k^i \parallel c_k^i] \right). \quad (4.10)$$

In the second phase, we update all temporal node encodings h_{t-s}^i by performing standard spatiotemporal attention ([Equations 4.3 to 4.7](#)), but using the router summaries $\tilde{Z}^i$ and $\tilde{Z}^j$ instead of the full sequence of representations for node i and its neighbors $j \in \mathcal{N}(i)$. This drastically reduces the cost of attention, now scaling with $\mathcal{O}((N + |\mathcal{E}|)KT)$, where $K \ll T$, at the expense of introducing an information bottleneck through the routers. Despite the bottleneck, this structure preserves global context efficiently and enables tractable training even on long temporal sequences.

4.3 Spatiotemporal positional encoding

A core requirement of SPIN is the availability of spatiotemporal positional encodings u_t^i that are independent of the input signal x_t^i , and uniquely characterize each location in space and time. These encodings are critical for enabling predictions at arbitrary spatiotemporal locations, including points with missing observations. When no suitable exogenous features are available, they must be learned end-to-end with the model.

We model the spatiotemporal encoding as a composition of a temporal and a spatial component through an MLP:

$$u_t^i = \text{MLP} \left([q_t \parallel v^i] \right), \quad (4.11)$$

where $q_t \in \mathbb{R}^{d_q}$ encodes the temporal position (e.g., time of day, day of week) and $v^i \in \mathbb{R}^{d_v}$ captures the spatial identity of node i .

For temporal encoding, we use sinusoidal functions applied to time t at one or more predefined frequencies to reflect seasonal patterns (e.g., hourly, daily, weekly). In the absence of a known periodic structure, a more flexible alternative is Time2Vec [[Kazemi et al., 2019](#)], which learns temporal embeddings by combining linear and periodic transformations of t . Besides, we also include in q_t the standard positional encoder typical of Transformer architectures to identify the time step position within the input sequence [[Vaswani et al., 2017](#)].

For spatial encoding, we use a learnable embedding vector for each node, optionally combined with static node attributes if available. This design mirrors

the initialization strategy used in GRIN (see [Section 3.2.2](#)) and allows the model to better disambiguate nodes, as later discussed in [Section 5.3](#). Additionally, one can incorporate graph-derived positional encodings – such as Laplacian eigenvectors or other topological embeddings [[Dwivedi et al., 2022](#)] – to inject structure-aware spatial priors.

This flexible formulation accommodates both hand-crafted and learned features, providing a principled mechanism to locate representations in time and space without relying on observations.

4.4 Related work

For related work on spatiotemporal data imputation, we refer to [Section 3.3](#). In the following, we discuss prior research on the interplay between attention mechanisms and spatiotemporal modeling. The flexibility of attention has led to its adoption in both temporal and spatial modeling. For time series forecasting, masked self-attention has been used to restrict attention to past timesteps in autoregressive models [[Zhou et al., 2021](#); [Nie et al., 2023](#)]. In the context of graphs, attention has been incorporated into message-passing operators [[Thekumparampil et al., 2018](#); [Veličković et al., 2018](#); [Shi et al., 2021](#)] and generalized to graph transformers [[Ying et al., 2021](#); [Dwivedi and Bresson, 2021](#); [Müller et al., 2024](#)]. Among STGNNs, attention-based architectures have been exploited in time series forecasting [[Zhang et al., 2018](#); [Zheng et al., 2020](#); [Guo et al., 2019](#); [Wu et al., 2024](#)]. In particular, TraverseNet [[Wu et al., 2024](#)] is closely related to our work, as it relies on spatiotemporal autoregressive attention to compute the messages exchanged between nodes, although its primary focus is on forecasting. One striking example of attention being used successfully to process incomplete data is in pretraining routines for representation learning in natural language processing [[Devlin et al., 2019](#); [Radford et al., 2018](#); [Yang et al., 2019](#); [Brown et al., 2020](#)].

4.5 Empirical evaluation

In this section, we evaluate our method on three real-world datasets and compare the performance against state-of-the-art methods and standard approaches for imputation. As the objective of our approach is to address the imputation problem in highly sparse settings, in a second experiment we assess how performance changes as the percentage of missing values increases.

Table 4.1. Imputation error (MAE) across different datasets and missing data simulations.

	Block-T		Point		Simulated failures	
	PEMS-BAY	METR-LA	PEMS-BAY	METR-LA	AQI-36	AQI
Mean	5.46 $\pm$ 0.00	7.48 $\pm$ 0.00	5.42 $\pm$ 0.00	7.56 $\pm$ 0.00	53.48 $\pm$ 0.00	39.60 $\pm$ 0.00
KNN	4.30 $\pm$ 0.00	7.79 $\pm$ 0.00	4.30 $\pm$ 0.00	7.88 $\pm$ 0.00	30.21 $\pm$ 0.00	34.10 $\pm$ 0.00
MF	3.28 $\pm$ 0.01	5.46 $\pm$ 0.02	3.29 $\pm$ 0.01	5.56 $\pm$ 0.03	30.54 $\pm$ 0.26	26.74 $\pm$ 0.24
MICE	2.94 $\pm$ 0.02	4.22 $\pm$ 0.05	3.09 $\pm$ 0.02	4.42 $\pm$ 0.07	30.37 $\pm$ 0.09	26.98 $\pm$ 0.10
VAR	2.09 $\pm$ 0.10	3.11 $\pm$ 0.08	1.30 $\pm$ 0.00	2.69 $\pm$ 0.00	15.64 $\pm$ 0.08	22.95 $\pm$ 0.30
rGAIN	2.18 $\pm$ 0.01	2.90 $\pm$ 0.01	1.88 $\pm$ 0.02	2.83 $\pm$ 0.01	15.37 $\pm$ 0.26	21.78 $\pm$ 0.50
BRITS	1.70 $\pm$ 0.01	2.34 $\pm$ 0.01	1.47 $\pm$ 0.00	2.34 $\pm$ 0.00	14.50 $\pm$ 0.35	20.21 $\pm$ 0.22
SAITS	1.56 $\pm$ 0.01	2.30 $\pm$ 0.01	1.40 $\pm$ 0.03	2.26 $\pm$ 0.00	18.16 $\pm$ 0.42	21.33 $\pm$ 0.15
Transformer	1.70 $\pm$ 0.02	3.54 $\pm$ 0.00	0.74 $\pm$ 0.00	2.16 $\pm$ 0.00	11.98 $\pm$ 0.53	18.11 $\pm$ 0.25
GRIN	1.14 $\pm$ 0.01	2.03 $\pm$ 0.00	0.67 $\pm$ 0.00	1.91 $\pm$ 0.00	12.08 $\pm$ 0.47	14.73 $\pm$ 0.15
SPIN	1.06 $\pm$ 0.02	1.98 $\pm$ 0.01	0.70 $\pm$ 0.01	1.90 $\pm$ 0.01	11.77 $\pm$ 0.54	13.92 $\pm$ 0.15
SPIN-H	1.05 $\pm$ 0.01	2.05 $\pm$ 0.02	0.73 $\pm$ 0.01	1.96 $\pm$ 0.03	10.89 $\pm$ 0.27	14.41 $\pm$ 0.13

We consider both **SPIN** and the hierarchical version **SPIN-H** (see Section 4.2.2). The figure of merit used is the MAE averaged across imputation windows. We consider only the out-of-sample setting (see Section 3.1.2), in which every parametric model is trained and tested on disjoint sets. Besides GRIN and the baselines considered in Section 3.4.1, we compare our method against the following:

Transformer. A spatiotemporal Transformer, where we alternate temporal and spatial Transformer encoder layers from [Vaswani et al., 2017] and replace missing values with a [MASK] token (as in [Devlin et al., 2019]).

SAITS. An attention-based architecture based on diagonally-masked self-attention [Du et al., 2023].

4.5.1 Imputation benchmarks

In this experiment, we evaluate the performance of SPIN and its lightweight variant SPIN-H using the same experimental setup adopted for GRIN in Section 3.4. The evaluation also includes the additional baselines introduced above.

Table 4.2. Performance (MAE) with increasing data sparsity in the **Point** setting.

	METR-LA			PEMS-BAY			AQI		
	Missing rate			Missing rate			Missing rate		
	50 %	75 %	95 %	50 %	75 %	95 %	50 %	75 %	95 %
BRITS	2.52 $\pm$ 0.00	3.02 $\pm$ 0.00	5.19 $\pm$ 0.02	1.55 $\pm$ 0.00	2.17 $\pm$ 0.00	3.91 $\pm$ 0.02	14.90 $\pm$ 0.03	18.29 $\pm$ 0.03	29.83 $\pm$ 0.07
SAITS	2.48 $\pm$ 0.00	3.74 $\pm$ 0.01	6.72 $\pm$ 0.01	1.50 $\pm$ 0.00	2.96 $\pm$ 0.01	7.40 $\pm$ 0.01	15.36 $\pm$ 0.02	20.64 $\pm$ 0.05	34.57 $\pm$ 0.05
Transf.	2.31 $\pm$ 0.00	2.71 $\pm$ 0.00	5.13 $\pm$ 0.01	0.85 $\pm$ 0.00	1.13 $\pm$ 0.00	2.70 $\pm$ 0.01	9.11 $\pm$ 0.02	12.56 $\pm$ 0.05	25.65 $\pm$ 0.11
GRIN	2.05 $\pm$ 0.00	2.39 $\pm$ 0.00	4.08 $\pm$ 0.02	0.79 $\pm$ 0.00	1.09 $\pm$ 0.00	2.70 $\pm$ 0.01	8.43 $\pm$ 0.01	10.97 $\pm$ 0.02	20.38 $\pm$ 0.10
SPIN	2.02 $\pm$ 0.00	2.24 $\pm$ 0.00	2.89 $\pm$ 0.01	0.79 $\pm$ 0.00	1.00 $\pm$ 0.00	1.71 $\pm$ 0.00	8.15 $\pm$ 0.01	9.96 $\pm$ 0.02	15.51 $\pm$ 0.08
SPIN-H	2.01 $\pm$ 0.00	2.20 $\pm$ 0.00	2.82 $\pm$ 0.00	0.79 $\pm$ 0.00	0.97 $\pm$ 0.00	1.68 $\pm$ 0.00	8.67 $\pm$ 0.02	10.27 $\pm$ 0.02	15.75 $\pm$ 0.07

Table 4.1 summarizes the results. Both SPIN models consistently outperform the baselines across most scenarios. The performance gains are particularly pronounced in settings with block-missing patterns – such as the AQI datasets and the **Block-T** settings – where large contiguous sequences of missing data challenge previous imputation strategies. In these cases, SPIN’s sparse spatiotemporal attention proves particularly effective, enabling long-range information propagation without relying on biased intermediate predictions, as required in autoregressive approaches.

In the **Point** setting, where missing values are fewer and scattered, both SPIN variants achieve performance on par with leading methods. This suggests that while their advantage is more evident in scenarios with sparse observations, they remain competitive in simpler regimes as well. Compared to the spatiotemporal Transformer, SPIN shows superior performance in all datasets except AQI-36, which can be attributed to the ineffectiveness of spatial attention alone in determining dependencies among different spatial locations. Finally, SPIN-H closely matches the performance of SPIN in almost every setting, and even outperforms it on the smaller AQI-36 dataset. This highlights its value as an efficient and scalable alternative, offering strong performance at reduced computational cost.

4.5.2 Sensitivity analysis

To assess the robustness of SPIN and SPIN-H under increasingly sparse conditions, we evaluate both models and the most relevant baselines on the same datasets as before, but under two additional stress-test scenarios with increased data sparsity. The first scenario builds upon the **Point** setting by progressively

Table 4.3. Performance (MAE) with an increasing number of simulated failures in the **Block-T** setting.

	METR-LA			PEMS-BAY			AQI		
	Failure probability			Failure probability			Failure probability		
	5 %	10 %	15 %	5 %	10 %	15 %	5 %	10 %	15 %
BRITS	5.87 $\pm$ 0.03	7.26 $\pm$ 0.06	8.29 $\pm$ 0.07	4.14 $\pm$ 0.05	5.41 $\pm$ 0.08	5.84 $\pm$ 0.04	24.09 $\pm$ 0.30	31.90 $\pm$ 0.26	37.62 $\pm$ 0.42
SAITS	4.73 $\pm$ 0.07	6.66 $\pm$ 0.05	7.27 $\pm$ 0.03	3.88 $\pm$ 0.09	7.62 $\pm$ 0.21	8.01 $\pm$ 0.11	20.78 $\pm$ 0.30	30.16 $\pm$ 0.39	36.34 $\pm$ 0.33
Transf.	6.03 $\pm$ 0.04	7.19 $\pm$ 0.05	8.06 $\pm$ 0.05	3.69 $\pm$ 0.06	5.09 $\pm$ 0.05	6.02 $\pm$ 0.04	29.21 $\pm$ 0.33	33.62 $\pm$ 0.16	37.31 $\pm$ 0.14
GRIN	3.05 $\pm$ 0.02	4.52 $\pm$ 0.05	5.82 $\pm$ 0.06	2.26 $\pm$ 0.03	3.45 $\pm$ 0.06	4.35 $\pm$ 0.04	15.62 $\pm$ 0.24	22.08 $\pm$ 0.39	29.03 $\pm$ 0.42
SPIN	2.71 $\pm$ 0.02	3.32 $\pm$ 0.02	3.87 $\pm$ 0.05	1.78$\pm$0.03	2.15$\pm$0.03	2.41$\pm$0.02	14.29$\pm$0.24	18.71$\pm$0.34	24.34$\pm$0.46
SPIN-H	2.64$\pm$0.02	3.17$\pm$0.02	3.61$\pm$0.04	1.75$\pm$0.04	2.16$\pm$0.03	2.48$\pm$0.02	14.55$\pm$0.26	19.37$\pm$0.36	25.38 $\pm$ 0.37

increasing the missing rate η , i.e., the independent probability of each observation being removed, simulating higher sparsity without structural patterns. The second scenario amplifies the **Block-T** setting by increasing the failure probability p_f of a sensor being unobserved for a random number of future time steps, sampled uniformly from [12, 36]. As p_f increases, longer temporal gaps emerge. For reference, $p_f = 5\%$, 10% , and 15% correspond to missing rates of roughly 70–75%, 90–92%, and 96–97%, respectively. We use the same trained models and test split as in the previous experiments, but replace the evaluation masks to reflect the new missing data patterns. For the traffic datasets, we evaluate using the models pretrained on the **Point** regime.

Tables 4.2 and 4.3 report results in terms of MAE across different missing rates and failure probabilities, averaged over several randomized missing masks. The results highlight three main findings. First, both SPIN and SPIN-H consistently achieve the best performance across all sparsity regimes. Second, they demonstrate stronger robustness: as sparsity increases, their error grows more slowly compared to all baselines. Finally, the performance gap widens notably compared to autoregressive methods like GRIN, which would require much more data to match the performance of SPIN, and attention-based alternatives like SAITS and the Transformer, which lack a structured spatiotemporal propagation. These results confirm that SPIN’s design is particularly well-suited for imputation in high-sparsity regimes, and serves as a complementary alternative to autoregressive methods like GRIN.

Table 4.4. Ablation study to assess the contribution of the single components in the spatiotemporal attention block.

	METR-LA (Point)		AQI-36	
	MAE	MRE (%)	MAE	MRE (%)
SPIN	1.90 ± 0.01	3.29 ± 0.01	11.77 ± 0.54	16.56 ± 0.76
SPIN-H	1.96 ± 0.03	3.40 ± 0.05	10.89 ± 0.27	15.32 ± 0.38
Without cross-attention	2.18 ± 0.01	3.78 ± 0.01	15.47 ± 0.22	21.77 ± 0.31
Without self-attention	2.21 ± 0.08	3.82 ± 0.14	13.76 ± 0.30	19.37 ± 0.42
Transformer	2.16 ± 0.00	3.74 ± 0.01	11.98 ± 0.53	16.87 ± 0.75

4.5.3 Ablation study

Table 4.4 presents the results of an ablation study conducted on the METR-LA dataset under the **Point** setting and on the AQI-36 dataset. Performance is evaluated using both MAE and mean relative error (MRE) [Cao et al., 2018]. We consider two ablated versions of SPIN-H: one with the spatiotemporal cross-attention module removed, and the other without the temporal self-attention component. For reference, we also report the performance of the full SPIN and SPIN-H models, as well as the spatiotemporal Transformer. The results clearly show that both attention components contribute significantly to imputation accuracy. We recall that in METR-LA observations are masked out uniformly at random while the mask in AQI-36 reflects the empirical distribution of missing data in the dataset.

4.5.4 Virtual sensing

The spatiotemporal cross-attention mechanism enables SPIN to leverage valid observations in the network to impute missing values at spatial locations with no available sensors, effectively performing *virtual sensing* (see Section 3.1.1). To evaluate this capability, we assess the performance of SPIN-H on a virtual sensing task in which observations at two nodes are entirely masked during training. The trained model is then used to reconstruct the full temporal sequences at these unseen nodes. For this experiment, we keep the same hyperparameters and use the AQI-36 dataset. Specifically, we mask out the two nodes with the highest (station no. 1015) and lowest (station no. 1032) number of neighbors, following the setup described in Section 3.4.5.

Figure 4.3 displays the qualitative performance of SPIN-H in reconstructing a

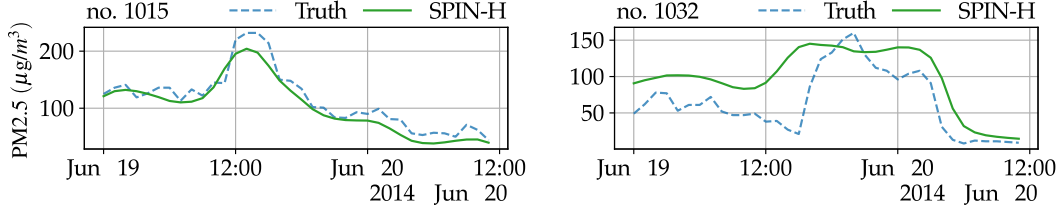


Figure 4.3. Reconstruction of observations at virtual sensors, i.e., nodes completely removed from the training set, by SPIN-H.

window of 36 time steps for the two virtual sensors at test time. The results qualitatively show that the model is able to recover meaningful temporal dynamics for nodes with no observed data, relying only on their position in the network and the surrounding context.

4.6 Discussion and future directions

In this chapter, we introduced SPIN, a graph-based architecture for reconstructing missing observations in sparse spatiotemporal data. To address the key limitations of autoregressive approaches, we proposed a novel sparse spatiotemporal attention mechanism that enables propagation of available observations across discrete points in both time and space in a unified manner. Additionally, we showed that the computational complexity can be significantly reduced by incorporating a hierarchical attention scheme. Empirical results demonstrate that SPIN outperforms state-of-the-art imputation methods by a wide margin, particularly in highly sparse regimes. The method’s design makes it especially suitable for scenarios where long-range dependencies and structured missing data patterns challenge previously proposed models.

Nevertheless, under extremely sparse *training* conditions, i.e., when few and sparse observations are available also at training time, SPIN may experience slower convergence due to limited supervision. A promising direction for future research is to augment the training signal with auxiliary learning objectives that can improve the representation learning process. Other possible extensions could investigate spatiotemporal positional encoding methods, which may further enhance the model’s ability to reason about spatial and temporal structure in data. Overall, the proposed architecture opens new directions for task-aligned and scalable models for spatiotemporal imputation under real-world sparsity constraints.

Part II

Prediction

Chapter 5

Prediction under irregular spatial sampling

Forecasting future values in spatiotemporal data is one of the most fundamental yet challenging tasks in machine learning. Unlike imputation or filtering, which operate within the observed window, forecasting requires extrapolation beyond available data, compounding uncertainty and requiring stronger modeling assumptions. These challenges are exacerbated in settings where spatial sampling is irregular; still, many real-world sensor networks exhibit exactly this structure.

In this chapter, we study how graph-based models can support prediction in such settings. We assume fully observed and synchronous temporal sequences and focus on the irregularity of the spatial domain. In particular, we leverage STGNNs to incorporate structural priors into predictive models, allowing for forecasts localized in the graph neighborhood. Our analysis is centered on two key design dimensions: globality versus locality, which governs how parameters are shared across nodes; and scalability, which determines whether architectures can be trained and deployed efficiently at scale.

Our first contribution is a principled framework to analyze and compare forecasting architectures along these axes. We propose hybrid global-local STGNNs that combine global spatiotemporal processing with node-specific components, implemented via learnable embeddings. These embeddings act as amortized local models and can be regularized to improve transferability and interpretability. This work [Cini, Marisca, et al., 2023b] demonstrates that global-local models can consistently outperform purely global or local alternatives across a range of datasets.

In the second part of the chapter, we address the computational limitations of existing models by introducing a scalable forecasting architecture. Our model [Cini, Marisca, et al., 2023a] decouples encoding and prediction through a training-free spatiotemporal representation built using randomized recurrent networks and graph shift operators. This approach enables fast and memory-efficient training, significantly improving scalability while preserving predictive accuracy.

The remainder of this chapter is organized as follows. We begin by formalizing the forecasting problem and introducing a unified framework for STGNN predictors (Section 5.1), while also exploring the roles of globality and locality and categorizing different spatiotemporal processing paradigms. Building on this foundation, we present our reference models (Section 5.2) and introduce global-local forecasting architectures (Section 5.3). We follow with the scalable graph-based predictor model in Section 5.4. Before turning to the empirical analysis, we review related work on spatiotemporal forecasting (Section 5.5), and later conclude with an empirical evaluation comparing these methods in terms of accuracy, scalability, and flexibility (Section 5.6).

Reference papers

This chapter’s content is partly based on material from the following articles.

Andrea Cini, Ivan Marisca, Filippo Maria Bianchi, and Cesare Alippi (June 2023a). **Scalable Spatiotemporal Graph Neural Networks**. In: *Proceedings of the 37th AAAI Conference on Artificial Intelligence* 37.6, pp. 7218–7226.

Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi (2023b). **Taming Local Effects in Graph-based Spatiotemporal Forecasting**. In: *Advances in Neural Information Processing Systems*. Vol. 36, pp. 55375–55393.

Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi (June 2025). **Graph Deep Learning for Time Series Forecasting**. In: *ACM Computing Surveys* 57.12. ISSN: 0360-0300.

5.1 Graph deep learning for time series forecasting

We address in the following the multi-step-ahead time-series forecasting problem [Benidis et al., 2022], i.e., we are interested in predicting, for each time step t and some forecasting horizon $H \geq 1$, the h step-ahead observations $\mathbf{X}_{t+h}$ for all $h \in [1, H]$ given a window of $T \geq 1$ past observations $\mathbf{X}_{t-T:t}$. We assume that each entry x_{t-k}^i is observed, i.e., not missing, for each $i \in \mathcal{V}$ and $k \in [0, T)$; consequently, we omit the auxiliary mask $\mathbf{M}$ from the input data. We are interested in learning a model p_θ for all $h \in [1, H]$ approximating the unknown conditional probability

$$p_\theta(\mathbf{x}_{t+h}^i \mid \mathbf{X}_{t-T:t}, \mathbf{U}_{t-T:t+h}, \mathbf{V}) \approx p^i(\mathbf{x}_{t+h}^i \mid \mathbf{X}_{\leq t}, \mathbf{U}_{\leq t+h}, \mathbf{V}) \quad \forall i \in \mathcal{V} \quad (5.1)$$

where θ indicates the learnable parameters of the model. Note that not all the exogenous variables $\mathbf{U}_{\leq t+h}$ might be available up to time step $t+h$ in practical applications. As similarly done for imputation (Chapter 3), we focus on point forecasts and study how GNNs can improve forecasting by incorporating relation information described by $\mathcal{E}$. Under such assumptions, we can consider predictive model families $\mathcal{F}(\cdot)$ such that

$$\widehat{\mathbf{X}}_{t:t+H} = \mathcal{F}(\mathbf{X}_{t-T:t}, \mathbf{U}_{t-T:t+H}, \mathbf{V}, \mathcal{E}; \theta) \quad (5.2)$$

$$\approx \mathbb{E}_p[p(\mathbf{X}_{t:t+H} \mid \mathbf{X}_{\leq t}, \mathbf{U}_{\leq t+H}, \mathbf{V})], \quad (5.3)$$

with θ being the vector of model parameters.

Building on the imputation framework defined in Equations 3.2 to 3.4, we consider forecasting architectures composed of an encoding step, followed by a stack of STMP layers, and a final readout that maps node representations to predictions:

$$\mathbf{h}_{t-k}^{i(0)} = \text{ENCODER}(\mathbf{x}_{t-k}^i, \mathbf{u}_{t-k}^i, \mathbf{v}^i) \quad \text{for } k \in [0, T), i \in \mathcal{V}, \quad (5.4)$$

$$\mathbf{H}_{t-T:t}^{(l+1)} = \text{STMP}^{(l)}(\mathbf{H}_{t-T:t}^{(l)}, \mathcal{E}) \quad \text{for } l \in [0, L), \quad (5.5)$$

$$\widehat{\mathbf{x}}_{t:t+H}^i = \text{READOUT}(\mathbf{h}_t^{i(L)}, \mathbf{u}_{t:t+H}^i) \quad \text{for } i \in \mathcal{V}. \quad (5.6)$$

This setup differs from the imputation case in two ways. First, the encoder does not take the auxiliary mask as input, since here we assume complete sequences. Second, and more critically, the readout operates only on the final node representation at time t , making predictions for the entire future horizon based on a single encoded state. This defines a classic encoder-decoder structure in which future outputs are generated from the same latent representation.

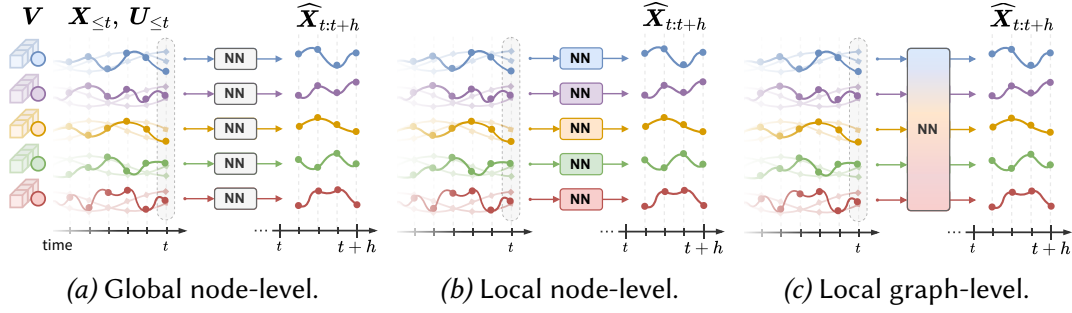


Figure 5.1. Globality and locality in forecasting architectures. Covariates $U_{t:t+h}$ associated with future time steps are here omitted for clarity.

Most existing forecasting STGNNs fit naturally within this template, with their distinctions primarily residing in the specific design of the STMP module.

Before delving into specific architectures, we discuss two complementary axes along which the design of STGNNs can be specialized. The first axis concerns how learnable parameters are shared across nodes. Section 5.1.1 compares *global* predictors, which employ a single parameter set for all nodes, with *local* predictors, which are specialized on individual nodes, and anticipates hybrid variants that combine the two approaches. The second axis, examined in the following subsection (5.1.2), revisits the template defined in Equations 5.4 to 5.6 from the standpoint of *where* spatial and temporal processing take place, distinguishing architectures that interleave or decouple the two dimensions; particular attention is paid to the implications these choices have for computational and memory scalability. Together, these subsections provide a principled taxonomy that will be mirrored in the next sections, where concrete global-local (Section 5.3) and scalable (Section 5.4) graph-based predictors are introduced.

5.1.1 Globality and locality in STGNNs

The terms *global* and *local*, as used in forecasting literature, describe how model parameters are shared across a collection of time series rather than characteristics of the data or the problem themselves [Montero-Manso and Hyndman, 2021]. A forecasting model is said to be *global* when a single set of parameters is learned from, and shared across, a collection of time series; conversely, a *local* model is trained (and used) individually on a single time series. Both classes can address identical prediction tasks, but they differ in how they exploit data, in their inductive capabilities, and in their computational requirements.

Global node-level modeling corresponds to training a single parameter vector θ that is reused for every node i . The resulting predictive distribution is

$$p_{\theta}(\mathbf{x}_{t+h}^i | \mathbf{x}_{t-T:t}^i, \mathbf{u}_{t-T:t+h}^i, \mathbf{v}^i) \approx p^i(\mathbf{x}_{t+h}^i | \mathbf{X}_{\leq t}, \mathbf{U}_{\leq t+h}, \mathbf{V}), \quad \forall i \in \mathcal{V}, \quad (5.7)$$

and is illustrated in [Figure 5.1a](#). Sharing parameters across nodes increases the effective sample size available to the optimizer and enables the use of models with higher capacity without incurring overfitting [[Salinas et al., 2020](#); [Montero-Manso and Hyndman, 2021](#); [Benidis et al., 2022](#)]. Furthermore, because no node-specific parameters are learned, global models naturally handle unseen nodes or even entirely new graphs, enabling deployment in *inductive* settings.

Purely local node-level modeling lies at the opposite extreme. Here, each node owns its parameter set θ^i and is trained in isolation from the others:

$$p_{\theta^i}(\mathbf{x}_{t+h}^i | \mathbf{x}_{t-T:t}^i, \mathbf{u}_{t-T:t+h}^i, \mathbf{v}^i) \approx p^i(\mathbf{x}_{t+h}^i | \mathbf{X}_{\leq t}, \mathbf{U}_{\leq t+h}, \mathbf{V}), \quad \forall i \in \mathcal{V}, \quad (5.8)$$

as depicted in [Figure 5.1b](#). This strategy can capture better node-specific patterns with shorter input sequence lengths and lower capacity compared to the global approach, but it requires fitting N independent models and overlooks any statistical advantage that might derive from pooling information across nodes.

Graph-level modeling, instead, treats the entire collection as a single high-dimensional multivariate series, optimizing one large predictor that outputs predictions for all nodes simultaneously:

$$p_{\theta}(\mathbf{X}_{t+h} | \mathbf{X}_{t-T:t}, \mathbf{U}_{t-T:t+h}, \mathbf{V}) \approx p(\mathbf{X}_{t+h} | \mathbf{X}_{\leq t}, \mathbf{U}_{\leq t+h}, \mathbf{V}), \quad \forall i \in \mathcal{V}, \quad (5.9)$$

as illustrated in [Figure 5.1c](#). While theoretically this approach can model both cross-node dependencies and node-specific behaviors in a unified manner, in practice, scalability issues quickly arise because the input dimensionality and the number of required parameters grow with N . Moreover, the learned model is tied to the training graph, hindering flexibility to inductive settings.

STGNNs as defined earlier in [Equation 5.2](#) belong to the global family. Indeed, the spatiotemporal message-passing layers share parameters over all nodes while conditioning on neighborhood information, thereby accounting for spatial dependencies that would be otherwise ignored by node-level global models (see [Equation 5.7](#)). Such parameter sharing, combined with the permutation-equivariant nature of message passing, confers the inductive capabilities of global models, generalizing naturally to unseen nodes without retraining.

Nonetheless, when important *local effects*, i.e., strong node-specific dynamics, are present, global networks may underfit them or require impractically long

historical windows to disentangle them [Montero-Manso and Hyndman, 2021; Cini, Marisca, et al., 2023b]. These observations motivate the hybrid strategies explored in Section 5.3, where lightweight node-specific components, i.e., *node embeddings*, are coupled with a global STGNN backbone to capture both shared and individual dynamics simultaneously.

5.1.2 Forecasting processing paradigms

The design of STGNNs involves choices not only in the type of message-passing operator or sequence model but also in how spatial and temporal dimensions are integrated. These decisions have direct implications for both the expressivity and the scalability of the resulting architectures. By adopting the terminology of previous works [Gao and Ribeiro, 2022; Cini et al., 2023c; Cini, Marisca, et al., 2023b], we classify STGNNs into three broad processing paradigms, namely, time-and-space (T&S), time-then-space (TTS), and space-then-time (STT), depending on the order and manner in which temporal and spatial processing are applied. Each paradigm makes different trade-offs between modeling flexibility, computational efficiency, and the ability to handle dynamic graph structures.

Time-and-space architectures

T&S architectures are characterized by joint processing over the temporal and spatial dimensions, meaning the two cannot be decoupled into separate steps. At each STMP layer l , node representations are updated through operations that simultaneously propagate information over both time and space. As discussed in Section 2.5.2, this can be achieved either by alternating temporal (TMP) and spatial (MP) operations or through custom operators that process both dimensions in a unified manner.

Well-known examples include MPTCNs, which alternate temporal and spatial convolutions, and MPRNNs, which integrate message passing into the recurrence mechanisms of RNNs [Li et al., 2018b; Zhang et al., 2018; Bai et al., 2020]. Although originally proposed for imputation, both GRIN [Cini, Marisca, et al., 2022] (see Chapter 3) and SPIN [Marisca et al., 2022] (see Chapter 4) fall under this category: GRIN embeds message passing into bidirectional RNNs, while SPIN leverages inter-node attention across temporal sequences.

An advantage of this joint design is its granularity, as it enables fine-tuned modeling at each node and time step and naturally accommodates dynamic graphs,

where nodes and edges may change over time. However, this expressiveness comes at a cost: the number of message-passing operations usually grows with the sequence length. For input sequences of length T , with N nodes and $|\mathcal{E}|$ edges, the complexity of T&S models typically scales as $\mathcal{O}(T(N + |\mathcal{E}|))$.

Time-then-space architectures

TTS architectures decouple temporal and spatial processing by first encoding each node's time series into a fixed-length representation, and then applying message passing across the resulting static attributed graph. Regarding the disjoint propagation scheme defined in Equations 2.39 and 2.40 in Section 2.5.2, by adjusting the number of outer layers L and the inner temporal/spatial layers accordingly (while keeping budgets LL_T and LL_S constant), we can control the extent of temporal-spatial processing integration. In this setting, having $L = 1$ provides the TTS design, with message passing being applied only to the final temporal representation:

$$\mathbf{z}_{t-T:t}^i = \text{TMP}_{\times L_T} \left(\mathbf{h}_{t-T:t}^{i(0)}, \mathbf{T} \right) \quad \forall i \in \mathcal{V}, \quad (5.10)$$

$$\mathbf{H}_t^{(L)} = \text{MP}_{\times L_S} (\mathbf{Z}_t, \mathcal{E}), \quad (5.11)$$

where L_T and L_S denote the depth of temporal and spatial processing, respectively. This allows us to perform message-passing operations on a single graph rather than a whole sequence, with evident efficiency advantages.

While relatively underexplored [Gao and Ribeiro, 2022; Satorras et al., 2022; Cini, Marisca, et al., 2023a; Cini, Marisca, et al., 2023b], TTS models are increasingly valued in real-world applications. They allow asynchronous, node-wise temporal encoding prior to any spatial-aware refinement, making them well-suited to distributed and scalable systems [Cini, Marisca, et al., 2023a]. Crucially, unlike T&S models, their computational complexity does not grow with the temporal window size, scaling instead as $\mathcal{O}(NT + |\mathcal{E}|)$. However, when dealing with time-varying topologies, the edge set $\mathcal{E}$ must be carefully defined, e.g., using the union of edge sets across the time horizon, to preserve the available relational information for message passing.

Space-then-time architectures

STT architectures follow the inverse order of the TTS paradigm: spatial processing is performed first at each time step, followed by temporal modeling of the resulting node-level sequences. This can be formalized by applying message

passing to static representations $\mathbf{H}_{t-k}^{(0)}$ at each time step, and then feeding the spatially enriched sequences into a temporal encoder:

$$\mathbf{Z}_{t-k} = \text{MP}_{\times L_S} \left(\mathbf{H}_{t-k}^{(0)}, \mathcal{E} \right) \quad \forall k \in [0, T), \quad (5.12)$$

$$\mathbf{h}_{t-T:t}^{i(L)} = \text{TMP}_{\times L_T} \left(\mathbf{z}_{t-T:t}^i, \mathbf{T} \right) \quad \forall i \in \mathcal{V}. \quad (5.13)$$

The motivation behind this design is to first enrich node observations by integrating spatial information from neighboring nodes, and then model the temporal evolution of these richer signals. Despite being less common [Seo et al., 2018; Pareja et al., 2020; Zhao et al., 2019], STT models retain the same computational complexity as T&S approaches, with cost scaling as $\mathcal{O}(T(N+|\mathcal{E}|))$. Nonetheless, like T&S models, they can flexibly handle time-varying topologies by performing message passing over the appropriate edge set at each time step.

5.2 Reference forecasting STGNNs

In this section, we provide concrete examples on how to build a STGNN for the forecasting task following the template given in Equations 5.4 to 5.6. We introduce 4 simple reference STGNN architectures that vary for the processing paradigm – TTS vs T&S – and the message-passing family adopted – isotropic vs anisotropic (see Section 2.3.2). In all the models, we use as ENCODER a simple linear layer s.t.

$$\mathbf{h}_{t-k}^{i(0)} = \mathbf{W}_{\text{enc}} [\mathbf{x}_{t-k}^i \parallel \mathbf{u}_{t-k}^i] \quad (5.14)$$

and as READOUT a 2-layer module structured as

$$\hat{\mathbf{x}}_{t:t+H}^i = \left[\mathbf{W}_h \xi \left(\mathbf{W}_{\text{dec}} \mathbf{h}_t^{i(L)} \right) \right]_{h=1, \dots, H} \quad (5.15)$$

where $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{d \times (d_x + d_u)}$, $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{d \times d}$, and $\mathbf{W}_1, \dots, \mathbf{W}_H \in \mathbb{R}^{d_x \times d}$ are learnable parameters, and ξ is an activation function. Unless otherwise stated, we consider the Exponential Linear Unit (ELU) [Clevert et al., 2016] as the activation ξ .

For the TTS architectures, we consider STMP modules as defined in Equations 5.10 to 5.11, with TMP layer implemented as a node-wise, shared GRU [Chung et al., 2014] (see Equations 2.28 to 2.31). For T&S models, instead, we use the MPGRU [Li et al., 2018b] presented in Section 2.5.2, where gates are implemented using MP layers (see Equations 2.41 to 2.44).

As MP layer, we consider an isotropic and an anisotropic variants. As isotropic MP operator we rely on the one defined in Equation 2.21 and use as the GSO the

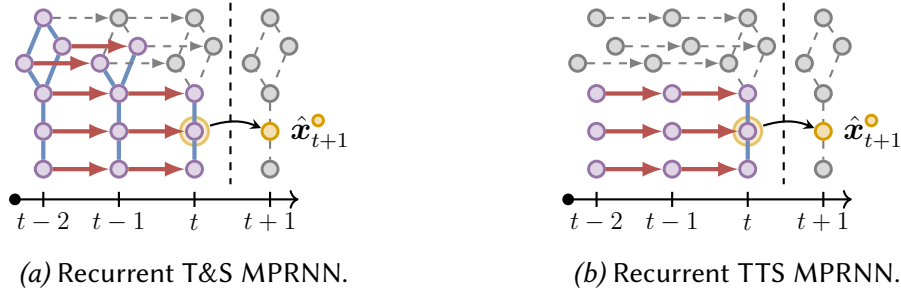


Figure 5.2. Receptive fields in MPRNNs with one MP layer following the T&S (a) and TTS (b) approach. Edge color denotes the type of propagation (— spatial or — temporal). Gray nodes and edges are outside the receptive field for target $\hat{x}_{t+1}$.

adjacency matrix normalized by the in-degree (as illustrated in Section 2.3.2). Conversely, the operator of choice for the anisotropic case corresponds to

$$\mathbf{z}_t^{ji(l)} = \mathbf{W}_2^{(l)} \xi \left(\mathbf{W}_1^{(l)} [\mathbf{h}_t^{i(l)} \parallel \mathbf{h}_t^{j(l)} \parallel \mathbf{e}^{ji}] \right), \quad (5.16)$$

$$\alpha_t^{ji(l)} = \sigma \left(\mathbf{w}_\alpha^{(l)\top} \mathbf{z}_t^{ji(l)} \right), \quad (5.17)$$

$$\mathbf{h}_t^{i,(l)} = \xi \left(\mathbf{W}_3^{(l)} \mathbf{h}_t^{i(l)} + \sum_{j \in \mathcal{N}(i)} \alpha_t^{ji(l)} \mathbf{z}_t^{ji(l)} \right), \quad (5.18)$$

where vector $\mathbf{w}_\alpha^{(l)} \in \mathbb{R}^d$, and matrices $\mathbf{W}_1^{(l)}$, $\mathbf{W}_2^{(l)}$ and $\mathbf{W}_3^{(l)}$ are learnable parameters, and σ is the sigmoid activation function. Note that all the parameters in the STMP blocks are shared among the nodes.

We indicate as **TTS-IMP** and **T&S-IMP** the reference architectures in which MP operators are isotropic and **TTS-AMP** and **T&S-AMP** the ones featuring anisotropic MP. Notably, the different processing paradigms in recurrent architecture result in different receptive fields, i.e., the region of the input space that can influence the computed representation. Indeed, while in the TTS models the full history of each neighbor reached by message passing is considered, in T&S architectures the spatial receptive field grows along with temporal processing, with information distant in space being able to affect the representations only after a certain amount of time. This behavior, graphically shown in Figure 5.2, can induce a form of under-reaching (see Section 2.3.1).

5.3 Global-local graph-based predictors

This section is based on the work of Cini, Marisca, et al. [2023b] and addresses how to design forecasting architectures that balance efficiency and accuracy by incorporating both global and local parameters. The core idea is to combine global graph-based components with node-specific local components to achieve two complementary goals of exploiting relational dependencies and side information through a shared global structure, and improving forecasting accuracy by tailoring part of the model to each individual time series.

By introducing local parameters specific to each node, the model can explicitly capture node-level effects that are often underrepresented in fully global architectures [Bai et al., 2020; Deng and Hooi, 2021]. Although this approach reduces transferability, as local parameters are not immediately reusable across nodes, it frequently results in better task-specific performance.

The family of global-local STGNNs can be formalized as:

$$\widehat{\mathbf{X}}_{t:t+H} = \mathcal{F}(\mathbf{X}_{t-T:t}, \mathbf{U}_{t-T:t+H}, \mathbf{V}, \mathcal{E}; \boldsymbol{\theta}, \{\boldsymbol{\omega}^i\}_{i=1}^N) \quad (5.19)$$

where $\mathcal{F}(\cdot)$ is the forecasting function shared across all nodes with global parameters $\boldsymbol{\theta}$, and $\{\boldsymbol{\omega}^i\}_{i=1}^N$ are local parameters specific to each node's time series. A simple instantiation of the global-local framework implements $\mathcal{F}(\cdot)$ as the sum of two models: a shared global model $\mathcal{F}_G$ and a node-specific local model f_L :

$$\widehat{\mathbf{X}}_{t:t+H}^{(1)} = \mathcal{F}_G(\mathbf{X}_{t-T:t}, \mathbf{U}_{t-T:t+H}, \mathbf{V}, \mathcal{E}; \boldsymbol{\theta}), \quad (5.20)$$

$$\hat{\mathbf{x}}_{t:t+H}^{i,(2)} = f_L(\mathbf{x}_{t-T:t}^i, \mathbf{u}_{t-T:t+H}^i; \boldsymbol{\omega}^i), \quad (5.21)$$

$$\hat{\mathbf{x}}_{t:t+H}^i = \hat{\mathbf{x}}_{t:t+H}^{i,(1)} + \hat{\mathbf{x}}_{t:t+H}^{i,(2)}. \quad (5.22)$$

An alternative, more integrated approach introduces local parameters directly into the encoding and/or decoding operations (see Equations 5.4 and 5.6):

$$\mathbf{h}_{t-k}^{i(0)} = \text{ENCODER}(\mathbf{x}_{t-k}^i, \mathbf{u}_{t-k}^i, \mathbf{v}^i; \boldsymbol{\omega}^i), \quad k \in [0, T), \quad (5.23)$$

$$\hat{\mathbf{x}}_{t:t+H}^i = \text{READOUT}(\mathbf{h}_t^{i(L)}, \mathbf{u}_{t:t+H}^i; \boldsymbol{\omega}^i). \quad (5.24)$$

While these approaches enable stronger specialization, they reduce generalization to unseen nodes and increase the number of model parameters – often significantly – thus affecting scalability. A compromise is to assign each node a learnable embedding, which serves as a compact, amortized representation of local effects. This strategy helps preserve transferability and limits the parameter overhead.

5.3.1 Learnable node embeddings

In some cases, static node features V can help distinguish individual time series, reducing the need for explicitly specialized model components. However, such features are often unavailable or lack the granularity needed for effective node identification. Learnable node embeddings offer a practical solution: by introducing a trainable parameter matrix $N_\theta \in \mathbb{R}^{N \times d_n}$, each node can be associated with a compact vector that implicitly captures its unique behavior [Cini, Marisca, et al., 2023b]. These embeddings effectively amortize the cost of learning local models, allowing node-specific effects to be captured within a shared global forecasting architecture.

This idea can be integrated into the encoder-decoder pipeline by modifying the input to both modules:

$$h_{t-k}^{i(0)} = \text{ENCODER} \left(x_{t-k}^i, u_{t-k}^i, v^i, n_\theta^i \right), \quad (5.25)$$

$$\hat{x}_{t:t+H}^i = \text{READOUT} \left(h_t^{i(L)}, u_{t:t+H}^i, n_\theta^i \right), \quad (5.26)$$

where n_θ^i denotes the learnable embedding for node i . These equations can be viewed as lightweight analogs of the local models defined in Equations 5.23 and 5.24. Within the reference architectures introduced in Section 5.2, this corresponds to concatenating n_θ^i to the layer inputs in both Equations 5.14 and 5.15. Importantly, including the embedding at the encoding stage (Equation 5.25) allows the model to inject node identity early, enabling node identification during message passing throughout the architecture.

In addition to conditioning the encoder and decoder, many state-of-the-art STGNNs use node embeddings directly into the model’s processing layers. These embeddings are often used as positional encodings [Marisca et al., 2022; Shao et al., 2022] or to parameterize a factorized weighted adjacency matrix [Wu et al., 2019; Wu et al., 2020; Bai et al., 2020]. Such hybrid strategies have shown remarkable empirical effectiveness [Cini, Marisca, et al., 2023b], leading to substantial gains in forecasting accuracy over fully global models. They are now widely adopted in transductive scenarios, where the node set remains fixed during training and inference.

Altogether, globality and locality are fundamental considerations in the design of STGNNs. While hybrid global-local architectures often yield better performance in practice, they do so at the cost of reduced flexibility, particularly in inductive settings that require generalization to unseen nodes. In the next subsection, we

see how structuring the embedding space at training time helps in fitting the corresponding embeddings for new nodes.

5.3.2 Structuring the embedding space

One of the key advantages of global GNN-based forecasting models is their inductive capability: they can operate on unseen node sets, support graphs of varying size, and adapt to dynamic topologies. This property is crucial in real-world applications, where new sensors may be added over time, and zero-shot generalization becomes essential. However, introducing node-specific components into the architecture typically undermines this flexibility. While node embeddings are lightweight, easy to adapt, and enable per-node specialization with relatively few additional parameters, ensuring that embeddings generalize to new nodes still requires additional care. Without constraints, new embeddings might fall outside the region of the latent space where the model has learned to operate effectively, with potential performance degradation.

To address this limitation, it is beneficial to impose structure on the embedding space, which improves interpretability, transferability, and robustness. Regularization plays a key role here. For instance, when domain knowledge suggests that neighboring nodes exhibit similar behavior, Laplacian regularization can be applied to encourage smoothness in the learned space [Zhou et al., 2003; Kipf and Welling, 2017]. Such constraints must be designed carefully to preserve the embeddings' ability to specialize forecasts. In what follows, we introduce two complementary strategies for structuring the latent space: one grounded in variational inference, and the other based on a clustering loss. As we demonstrate later, these approaches not only facilitate model transfer but also enhance our understanding of the features captured by the learned embeddings.

Variational regularization

A first approach we consider follows a probabilistic framework based on variational inference. The idea is to treat each node embedding not as a fixed vector but as a random variable drawn from an approximate posterior distribution conditioned on the training data. Specifically, we model each embedding as:

$$\mathbf{n}_\theta^i \sim q^i(\mathbf{n}_\theta^i | \mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}_i, \text{diag}(\boldsymbol{\sigma}_i^2)),$$

where $(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)$ are learnable parameters specific to node i . The posterior is regularized with respect to a standard Gaussian prior $P = \mathcal{N}(\mathbf{0}, \mathbb{I})$, and

optimized using the reparameterization trick [Kingma and Welling, 2014]. Training minimizes a variational objective over randomly sampled sequences indexed by t :

$$\delta_t = \mathbb{E}_{\mathbf{N}_\theta \sim Q} \left[\left\| \widehat{\mathbf{X}}_{t:t+H} - \mathbf{X}_{t:t+H} \right\|_2^2 \right] + \beta D_{\text{KL}}(Q \mid P), \quad (5.27)$$

where the first term measures forecasting accuracy and the second – the Kullback-Leibler divergence D_{KL} – penalizes divergence from the prior, with β controlling the trade-off. This regularization induces a smooth latent space, facilitating interpolation between node embeddings and improving the model’s ability to generalize to new nodes in a principled manner.

Clustering regularization

A complementary strategy for structuring the latent space involves encouraging node embeddings to organize into clusters. This self-organization can reveal higher-order structures among time series and improve generalization. To this end, we introduce a clustering-based regularization inspired by deep K -means methods [Yang et al., 2017]. Alongside the embedding matrix $\mathbf{N}_\theta \in \mathbb{R}^{N \times d_n}$, we define $K \ll N$ learnable centroids $\mathbf{C} \in \mathbb{R}^{K \times d_n}$, and a cluster-assignment matrix $\mathbf{S} \in \mathbb{R}^{N \times K}$, representing unnormalized affinities between each node and each cluster. These scores act as logits of a Boltzmann (categorical) distribution, inducing a soft assignment $\tilde{\mathbf{S}}$ used to compute the regularization loss:

$$\mathcal{L}_{\text{clu}} = \mathbb{E}_{\tilde{\mathbf{S}}} \left[\left\| \mathbf{N}_\theta - \tilde{\mathbf{S}} \mathbf{C} \right\|_2 \right], \quad p(\tilde{\mathbf{S}}_{ij} = 1) = \frac{e^{\mathbf{S}_{ij}/\tau}}{\sum_k e^{\mathbf{S}_{ik}/\tau}}, \quad (5.28)$$

where τ is a temperature parameter that controls the assignment. We minimize $\mathcal{L}_{\text{clu}}$ jointly with the forecasting loss, using the Gumbel-softmax trick [Maddison et al., 2017] for differentiable sampling of cluster assignments. Like variational regularization, this approach promotes structure in the embedding space, offering interpretability and insight into recurring patterns across time series.

5.3.3 Empirical validation

In this subsection, we empirically assess how the proposed regularization strategies support the transferability of forecasting models to new node sets.

Table 5.1. Forecasting error (MAE) in the transfer learning setting on all PEMS datasets. Results refer to a 1-week fine-tuning set size, and **bold** formatting indicates the best result (on average) in each dataset.

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

Table 5.2. Forecasting error (MAE) on PEMS04 in the transfer learning setting by varying fine-tuning set size. **Bold** formatting indicates the best result in each setting.

Model	Training set size			
TTS-IMP	2 weeks	1 week	3 days	1 day
Global	20.86 $\pm$ 0.03	21.59 $\pm$ 0.11	21.84 $\pm$ 0.06	22.26 $\pm$ 0.10
Embeddings	19.96 $\pm$ 0.08	20.27 $\pm$ 0.11	21.03 $\pm$ 0.14	21.99 $\pm$ 0.13
– Variational	19.94 $\pm$ 0.08	20.19 $\pm$ 0.05	20.71 $\pm$ 0.12	21.20 $\pm$ 0.15
– Clustering	19.69 $\pm$ 0.06	19.91 $\pm$ 0.11	20.48 $\pm$ 0.09	21.91 $\pm$ 0.21

Transferability of graph-based predictors

In our first experiment, we simulate a realistic transfer scenario in traffic forecasting: a model is trained on data from multiple networks and evaluated on a disjoint set of sensors from the same region. We use the **PEMS03–08** datasets [Guo et al., 2022], which contain measurements from four distinct districts in California (see Section 2.6). We train models on three of these datasets, then fine-tune them on one week of data from the held-out district, validate on the following week, and test on the week thereafter. The task is to predict the next $H = 12$ steps given a window of $T = 12$ time steps. We compare variants of TTS-IMP with and without node embeddings at the encoder and decoder. In addition to standard (unconstrained) embeddings, we evaluate both variational and clustering-based regularizations.

During fine-tuning, the fully global model updates all of its parameters. In contrast, hybrid global-local models retain shared parameters and adapt only

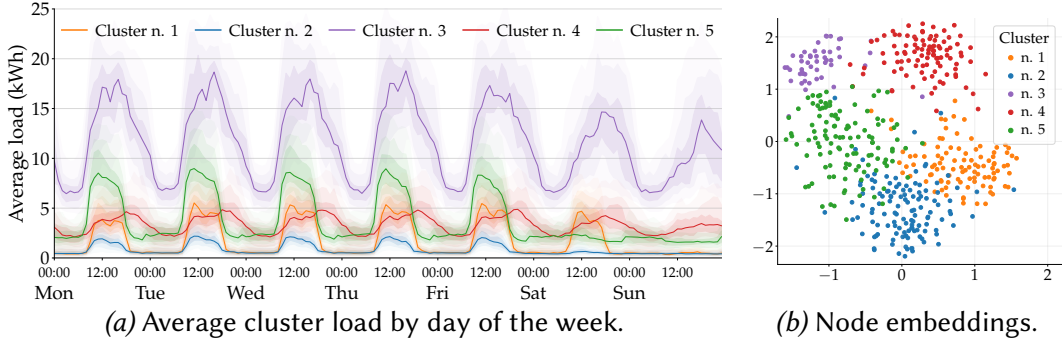


Figure 5.3. Time series clusters in CER-E, obtained by regularizing the embedding space. **(a)** Average load for each clusters. **(b)** t-SNE plot of the corresponding node embeddings.

the node embeddings, resulting in faster transfer. Table 5.1 reports results across all target regions. Hybrid models consistently outperform the fully global baseline, demonstrating improved adaptability with minimal retraining. Additionally, regularization strategies further enhance transferability, helping new node embeddings align better with the pretrained space. These findings are confirmed by Table 5.2, which shows a clear reduction in forecasting error on PEMS04 as the fine-tuning window increases.

Time series clustering

To investigate whether structuring the embedding space provides interpretable insights into node-level effects, we analyze the output of the clustering regularization strategy on the reference TTS-IMP model trained on the CER-E dataset. We consider $K = 5$ learnable centroids and train the model using the clustering loss alongside the forecasting objective. After training, we inspect the cluster assignments by analyzing intra-cluster energy consumption patterns. For each node, we compute its weekly average load profile, and then, for each hour of the day, we evaluate the distribution of energy consumption within each cluster. Figure 5.3a reports the median load curve for each cluster, with shaded bands representing 10% quantile intervals. The analysis reveals that each cluster corresponds to a distinct consumption pattern. To further explore the embedding space, Figure 5.3b shows a 2D t-SNE projection of the learned node embeddings. This visualization illustrates how the clustering regularization effectively promotes structure in the latent space and reflects the different behaviors across premises.

5.4 Scalable graph-based predictors

Despite their effectiveness, most graph-based architectures for spatiotemporal forecasting face limited adoption in real-world applications due to their substantial computational requirements. The necessity to process both spatial and temporal dimensions simultaneously results in operations that scale with the product of the number of edges and the sequence length, significantly impacting model training and deployment. Addressing this computational bottleneck is crucial for translating academic research into practical implementations.

While research to improve the scalability of models for large static graphs has been very prolific [Hamilton et al., 2017; Chiang et al., 2019; Zeng et al., 2020; Frasca et al., 2020], little attention has been paid to the additional challenges encountered when time series are associated with the graph nodes. Several of the existing scalable training techniques rely on subsampling graphs to reduce the computational requirements of the training procedure, e.g., [Hamilton et al., 2017; Zeng et al., 2020]. However, as discussed in Section 4.2.1, sampling the node-level observations as if they were i.i.d. can break relational (spatial) dependencies in static graphs, and it is even more problematic in the dynamic case, as dependencies occur also across the temporal dimension. As an alternative, precomputing aggregated features over the graph allows for factoring out spatial propagation from the training phase in certain architectures [Zambon et al., 2020; Frasca et al., 2020]. However, similarly to the subsampling approach, extending this method to the spatiotemporal case is not trivial as the preprocessing step must also account for the temporal dependencies besides the graph topology.

We introduce Scalable Graph Predictor (SGP) [Cini, Marisca, et al., 2023a], a scalable encoder-decoder STGNN relying on a training-free spatiotemporal encoding. First, it exploits a deep randomized recurrent neural network [Jaeger, 2001; Gallicchio et al., 2017] to encode the history of each sequence in a high-dimensional vector embedding; then, it uses powers of the graph adjacency matrix to build informative node representations of the spatiotemporal dynamics at different scales. According to the downstream task at hand, the trainable *decoder* maps the node representations into the desired output, e.g., the future values of the time series associated with each node. To improve efficiency, we exploit the structure of the extracted embedding to design the decoder so that it acts as a collection of filters localized at different spatiotemporal scales. Before delving into the technical explanation of SGP’s components, the following subsection provides preliminary notions on echo state networks, the temporal-processing backbone of our approach.

5.4.1 Preliminaries: echo state networks

Echo state networks (ESNs) [Jaeger, 2001; Lukoševičius and Jaeger, 2009] are a class of randomized architectures that consist of recurrent neural networks with random connections that encode the history of input signals into a high-dimensional state representation to be used as input to a (trainable) readout layer. Given input $\mathbf{x}_t \in \mathbb{R}^{d_x}$, state $\mathbf{h}_t \in \mathbb{R}^d$, and output $\mathbf{y}_t \in \mathbb{R}^{d_y}$, an ESN performs the following operations:

$$\mathbf{h}_t = \xi(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b}), \quad (5.29)$$

$$\mathbf{y}_t = \text{READOUT}_\theta(\mathbf{h}_t), \quad (5.30)$$

where $\mathbf{W}_x \in \mathbb{R}^{d \times d_x}$ and $\mathbf{W}_h \in \mathbb{R}^{d \times d}$ are the random matrices defining the connectivity pattern in the reservoir, $\mathbf{b} \in \mathbb{R}^d$ is a randomly initialized bias, and ξ is a nonlinear activation function (usually $\tanh$). Notably, only the decoder READOUT_θ has trainable parameters θ . The main idea is to feed an input signal into a high-dimensional, randomized, and nonlinear reservoir (Equation 5.29), whose internal state can be used as an embedding of the input dynamics.

If the random matrices are defined properly, the reservoir will extract a rich pool of dynamics characterizing the system underlying the input time series $\mathbf{x}_t$ and, thus, the reservoir states $\mathbf{h}_{t-T:t}$ become informative embeddings of $\mathbf{x}_{t-T:t}$ [Lukoševičius and Jaeger, 2009]. Specifically, to ensure stable, input-driven dynamics, the encoder must satisfy the *echo-state property*. Informally, this property states that the reservoir state is uniquely determined by the input history, and, hence, not by the initial conditions. A common and sufficient condition is to have all eigenvalues of the state-transition matrix $\mathbf{W}_h$ within the unit circle, i.e., to have spectral radius $\rho(\mathbf{W}_h) < 1$.

This is the original and most basic formulation of an ESN. However, several new approaches have been proposed. Jaeger et al. [2007] introduced leaky integrator neurons, for a finer control of the state update equation (5.29):

$$\mathbf{h}_t = (1 - \lambda)\mathbf{h}_{t-1} + \lambda \cdot \xi(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b}), \quad (5.31)$$

where $\lambda \in [0, 1]$ is the leaking factor. More recently, Gallicchio et al. [2017] introduced DeepESN, which consists of a stack of ESNs in the encoder, allowing for a more structured state space organization with multi-scale temporal dynamics. The l -th encoding layer follows a similar processing to Equation 5.31:

$$\mathbf{h}_t^{(l+1)} = (1 - \lambda^{(l)})\mathbf{h}_{t-1}^{(l+1)} + \lambda^{(l)} \cdot \xi(\mathbf{W}_x^{(l)} \mathbf{h}_t^{(l)} + \mathbf{W}_h^{(l)} \mathbf{h}_{t-1}^{(l+1)} + \mathbf{b}^{(l)}), \quad (5.32)$$

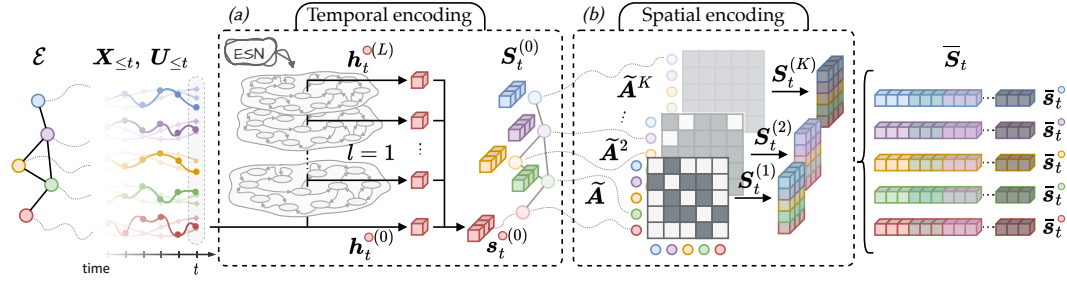


Figure 5.4. Overview of the SGP encoder. Input time series are fed individually into an ESN and embedded into a multi-scale vector representation. A GSO is used to gather spatial information of different orders, which is then concatenated to obtain a final embedding.

where each layer has different parameters and $h_t^{(0)} = x_t$. By varying the parameters at each layer, e.g., by changing the leaking factor, we can extract a rich pool of multi-scale temporal dynamics [Gallicchio et al., 2017].

5.4.2 Randomized spatiotemporal encoding

The encoder first constructs representations of the time series observed at each node by using a reservoir that accounts for dynamics at different time scales. Representations are further processed to account for spatial dynamics described by the graph structure. The final embedding hence contains a rich encoding of both spatial and temporal features. The encoder does not need any training and, once computed, the embeddings can be uniformly sampled over time and space at training time.

For the temporal encoding, we adopt a DeepESN as defined in Equation 5.32, with $h_t^{i(0)} = [x_t^i \parallel u_t^i]$ and L layers. We obtain the node-level temporal encoding $s_t^{i(0)}$ as the concatenation of the states from each layer as

$$s_t^{i(0)} = [h_t^{i(0)} \parallel h_t^{i(1)} \parallel \dots \parallel h_t^{i(L)}]. \quad (5.33)$$

Notably, by decreasing the leaking rate $\lambda^{(l)}$ progressively at each layer – specifically, we set $\lambda^{(l+1)} = \lambda^{(l)} - 0.1 - s_t^{i(0)}$ ends up containing features extracted focusing on L different temporal dynamics. The extraction of the node-level temporal embeddings is depicted in Figure 5.4a, where, to simplify the drawing, we report the detailed processing only for a single node.

The next step is to propagate information along the spatial dimension. We use powers of a GSO $\tilde{A}$ to propagate and aggregate node representations at different

scales. By using a notation similar to Equation 5.33, we obtain spatiotemporal encodings as

$$\mathbf{S}_t^{(k)} = \tilde{\mathbf{A}} \mathbf{S}_t^{(k-1)} = [\tilde{\mathbf{A}}^k \mathbf{H}_t^{(0)} \parallel \tilde{\mathbf{A}}^k \mathbf{H}_t^{(1)} \parallel \dots \parallel \tilde{\mathbf{A}}^k \mathbf{H}_t^{(L)}], \quad (5.34)$$

$$\bar{\mathbf{S}}_t = [\mathbf{S}_t^{(0)} \parallel \mathbf{S}_t^{(1)} \parallel \dots \parallel \mathbf{S}_t^{(K)}], \quad (5.35)$$

where integer K governs the spatial receptive field of the output representation. In practice, by indicating with $\mathbf{D} = \text{diag}(\mathbf{A}\mathbf{1})$ the graph degree matrix, we use $\tilde{\mathbf{A}} = \mathbf{D}^{-1} \mathbf{A}$ in the case of a directed graph, similar to DCNN [Atwood and Towsley, 2016], and the symmetrically normalized adjacency $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$ in the undirected case. Furthermore, for directed graphs we optionally increase the number of representations to $2K + 1$ to account for bidirectional dynamics, i.e., we repeat the encoding process w.r.t. the transpose adjacency matrix similar to Li et al. [2018b]. Intuitively, each propagation step $\tilde{\mathbf{A}} \mathbf{S}_t^{(k-1)}$ weighs and aggregates features between nodes connected by paths of length k in the graph. As shown in Equation 5.34, features corresponding to each order k can be computed recursively with K sparse matrix multiplications. Alternatively, each matrix $\tilde{\mathbf{A}}^k$ can be precomputed, and the computation of the different blocks of matrix $\bar{\mathbf{S}}_t$ can be distributed in a parallel fashion, as shown in Figure 5.4b. Even in the case of extremely large graphs, features $\bar{\mathbf{S}}_t$ can be computed offline by exploiting distributed computing as they do not need to be loaded on the GPU memory.

5.4.3 Trainable multi-scale decoder

The role of the decoder is that of selecting and weighing from the pool of the (possibly redundant) features extracted by the spatiotemporal encoder and mapping them to the desired output. Representations $\bar{\mathbf{S}}_t$ can be fed into an MLP that performs node-wise predictions. Since the representations are large vectors, a naïve implementation of the MLP results in many parameters that hinder scalability. Therefore, we replace the first MLP layer with a more efficient implementation that exploits the structure of the embeddings. In fact, $\bar{\mathbf{S}}_t$ is the concatenation of the representations corresponding to different spatial propagation steps which, in turn, are obtained from the concatenation of multi-scale temporal features. To exploit this structure, we design the first layer

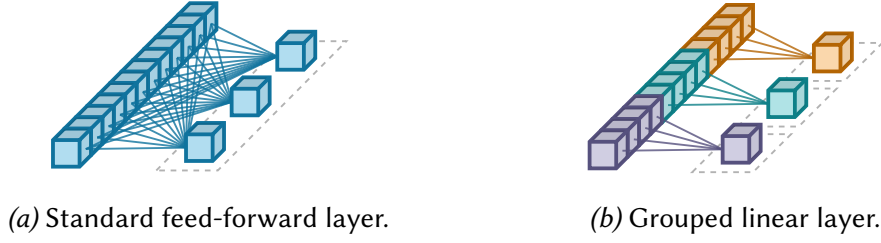


Figure 5.5. Comparison between standard feed-forward layer (a) and the grouped implementation adopted in the SGP decoder (b). The latter can be efficiently computed by exploiting tensor-tensor multiplications or grouped 1D convolutions to parallelize computation on GPUs.

of the decoder with a sparse connectivity pattern to learn representations $\bar{\mathbf{Z}}_t$ s.t.

$$\mathbf{Z}_t^{(k)} = \xi \left([\tilde{\mathbf{A}}^k \mathbf{H}_t^{(0)} \mathbf{W}^{(0,k)} \parallel \dots \parallel \tilde{\mathbf{A}}^k \mathbf{H}_t^{(L)} \mathbf{W}^{(L,k)}] \right) \quad (5.36)$$

$$= \xi \left(\mathbf{S}_t^{(k)} \begin{bmatrix} \mathbf{W}^{(0,k)} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{W}^{(L,k)} \end{bmatrix} \right), \quad (5.37)$$

$$\bar{\mathbf{Z}}_t = [\mathbf{Z}_t^{(0)} \parallel \mathbf{Z}_t^{(1)} \parallel \dots \parallel \mathbf{Z}_t^{(K)}], \quad (5.38)$$

where $\mathbf{W}^{(l,k)} \in \mathbb{R}^{d^{(l)} \times d}$ are the learnable parameters associated with the k -th graph propagation and l -th temporal layer ($d^{(l)}$ being the layer's number of features), and ξ is an activation function. Each projection $\mathbf{W}^{(l,k)}$ is localized on the dynamics of spatial order k and temporal scale l , and the joint output can be seen as a concatenation of the results of $L(K+1)$ graph convolutions of different orders (including the identity). Besides, the number of parameters is reduced precisely by a factor of $L(K+1)$ (or $L(2K+1)$ for directed graphs) compared to a standard fully-connected layer, as shown in Figure 5.5.

Finally, the obtained representations are fed into a standard MLP that predicts the next H observations as

$$\hat{\mathbf{x}}_{t:t+H}^i = \text{MLP}([\bar{\mathbf{z}}_t^i \parallel \mathbf{v}^i]), \quad (5.39)$$

which also includes static node-level attributes $\mathbf{v}^i$. As discussed in Section 5.3, these features can also be enriched by concatenating a node embedding, i.e., a vector of learnable parameters, to ease node identification.

5.4.4 Computational complexity

Since the spatiotemporal encoder requires neither training nor supervision, the representation of each node and time step can be computed in a preprocessing stage, without the constraints that come from online training on GPUs with limited memory. Indeed, representations $\bar{\mathbf{S}}_t$ capture both temporal and spatial dependencies among the history of graph observations $\mathbf{X}_{\leq t}$. Consequently, each sample $\bar{\mathbf{s}}_t^i$ can be processed individually. This allows for training the decoder with stochastic gradient descent by uniformly and independently sampling mini-batches of data points $\bar{\mathbf{s}}_t^i$ of a size proportional to the hardware capacity.

This is the key property that makes the training procedure extremely scalable and drastically reduces the lower bound on the computational complexity required for the training compared to standard STGNNs. In fact, state-of-the-art architectures have computational and memory costs that scales as $\mathcal{O}(T|\mathcal{E}|)$, or $\mathcal{O}(T^2|\mathcal{E}|)$ in attention-based architectures [Wu et al., 2024; Marisca et al., 2022]. Conversely, in our approach, mini-batches can be sampled disregarding the length of the sequence and size of the graph, thus making scalability in training constant, i.e., $\mathcal{O}(1)$, w.r.t. the spatiotemporal dimension of the problem.

5.5 Related work

The application of statistical methods to forecast spatiotemporal data has a rich history, tracing back several decades [Cressie and Wikle, 2011; Box et al., 2015]. Initial applications emerged in the mid-1970s, primarily in healthcare, economics, and environmental sciences [Granger, 1975; Cliff and Ord, 1975]. The early 2000s saw a surge in spatial analysis techniques, driven by the growing adoption of Geographic Information Systems (GIS), which fostered research in spatiotemporal data analysis [Shekhar et al., 2015; Atluri et al., 2018].

In parallel, kernel-based and probabilistic approaches such as Gaussian processes [Rasmussen and Williams, 2005] have been studied for graph-structured data [Smola and Kondor, 2003; Zhi et al., 2023], and more recently extended to spatiotemporal settings [Hamelijnck et al., 2021]. These methods offer uncertainty estimates based on data availability and often deliver good performance with few training data, making them particularly valuable in applications with sparse and irregularly sampled data. However, the cubic scalability with the number of observations for exact inference in Gaussian processes makes it challenging to apply them to large datasets without approximation.

In recent years, deep learning has gained traction for modeling complex spatiotemporal dependencies [Wang et al., 2020]. One of the earliest efforts in this direction was by Graves and Schmidhuber [2008], who introduced multi-dimensional RNNs to process spatial and temporal axes simultaneously, primarily for structured grid data like images. Extending this idea, Jain et al. [2016] proposed Structural-RNN, which factorizes spatiotemporal graphs into node- and edge-specific RNNs connected according to graph topology, allowing the model to capture entity interactions over time. Jia et al. [2017] later developed a dual-memory LSTM for land-cover classification in satellite imagery. Similarly, Shi et al. [2015] combined convolutions within RNNs to jointly process spatial and temporal dependencies simultaneously. While these models incorporate structural and temporal priors, they lack the flexibility of GNNs, which support general message passing over arbitrary topologies. To the best of our knowledge, Seo et al. [2018] and Li et al. [2018b] were the first to generalize the spatial processing within recurrent networks to graph-structured data using GNNs. The following works by Yu et al. [2018] and Wu et al. [2019]; Wu et al. [2020] introduced the first MPTCN architectures that alternate between temporal and spatial convolutions. More recently, Wu et al. [2022] leveraged attention mechanisms to enable information propagation across both spatial and temporal dimensions. Modern architectures incorporate relational inductive biases with full Transformer-style attention [Vaswani et al., 2017] along the spatial dimension [Zhang et al., 2018; Zheng et al., 2020; Oreshkin et al., 2021; Satorras et al., 2022], although this often comes at the cost of quadratic computational complexity on the number of nodes. A more in-depth view of state-of-the-art approaches in spatiotemporal data modeling is provided by Jin et al. [2024].

Global-local modeling Several methods have addressed the problem of learning node-specific dynamics in spatiotemporal forecasting by combining global and local modeling. For instance, Bai et al. [2020] propose AGCRN, which factorizes recurrent weight matrices to adapt shared representations to each node through learnable embeddings. In contrast, Chen et al. [2021] build on the architecture introduced by Wang et al. [2019], employing a global GNN alongside a local model conditioned on neighboring information. Node embeddings have also been used to amortize graph structure learning – e.g., in adaptive adjacency methods [Wu et al., 2019; Shang et al., 2021; Deng and Hooi, 2021] – and as positional encodings in attention-based architectures [Satorras et al., 2022; Marisca et al., 2022; Zheng et al., 2020]. Shao et al. [2022] show

that adding explicit spatiotemporal identifiers to a more standard forecasting architecture can outperform various state-of-the-art STGNNs, while Yin et al. [2022] introduce a cluster-based regularization to adapt AGCRN-like models across datasets. However, previous works have not systematically addressed the interplay between global and local modeling in STGNNs, nor have they offered a unified framework for incorporating learnable node embeddings across different settings and architectures. Beyond STGNNs, hybrid time series forecasting models have tackled similar challenges. For example, Wang et al. [2019] combine K global latent factor models with local probabilistic predictors, while Sen et al. [2019] use matrix factorization with a TCN to learn a multivariate predictor used to condition a second local model.

Scalable graph-based predictors Research on scalable models for discrete-time dynamic graphs remains relatively limited. Most approaches extend techniques developed for static graphs, such as node-wise sampling (e.g., GraphSAGE [Hamilton et al., 2017]) or subgraph-based methods like ClusterGCN [Chiang et al., 2019] and GraphSAINT [Zeng et al., 2020]; a few examples are provided by Wu et al. [2020]; Gandhi et al. [2021]; Wu et al. [2021]. Among scalable GNNs for static graphs, SIGN [Frasca et al., 2020] is particularly relevant to SGP. SIGN performs spatial propagation as a preprocessing step by applying multiple GSOs to capture multi-hop neighborhoods, followed by an MLP for learning. However, it is restricted to static graphs and propagates raw node features without temporal modeling. More closely aligned with our work, DynGESN [Micheli and Tortorella, 2022] addresses dynamic graphs using a graph-based randomized recurrent architecture (i.e., an ESN). However, it adopts a T&S strategy for processing temporal and spatial information jointly, in contrast to the TTS separation employed by SGP, and uses a simpler linear readout.

5.6 Empirical evaluation

This section reports an excerpt of the most relevant results of an extensive empirical analysis of STGNNs in spatiotemporal forecasting benchmarks and different problem settings. Complete results of these analyses can be found in the corresponding works [Cini, Marisca, et al., 2023a; Cini, Marisca, et al., 2023b; Cini, Marisca, et al., 2025]. Performance is measured in terms of MAE.

5.6.1 Forecasting baselines

Besides the reference STGNNs introduced in Section 5.2 and SGP [Cini, Marisca, et al., 2023a] (Section 5.4), we consider the following baselines and popular state-of-the-art architectures.

Naïve. For naïve forecasting baselines, we simply repeat a previously observed value from the time series. Specifically, **Last** predicts the most recent value, i.e., $\hat{x}_{t+h}^i = x_t^i$ for each $h \geq 1$. The **Day** and **Week** variants instead use the value observed at the same time in the previous day or week, respectively. Formally, they set $\hat{x}_{t:t+H}^i = x_{t-k:t-k+H}^i$, with $k \in \mathbb{N}$ being the seasonality lag (e.g., 24 or 168 hours).

GRU. A global GRU sharing the same parameters across the time series (as in Figure 5.1a). The GRU is preceded and followed by the same encoding and decoding schemes adopted by the reference architectures (Equations 5.14 and 5.15), respectively.

LocalGRUs. Similar to GRU, but contains a set of local GRUs with different parameters for each time series (as in Figure 5.1b).

FC-GRU. A multivariate GRU taking as input the time series as if they were a multivariate one (as in Figure 5.1c). Also for this model, encoding and decoding are performed similarly to the reference STGNNs, but graph-wise rather than node-wise.

DCRNN. The global recurrent T&S model with isotropic message passing from [Li et al., 2018b]. It follows the MPGRU framework (Equations 2.41 to 2.44) and uses DCNN by Atwood and Towsley [2016] for message passing. This operator enables message propagation across both incoming and outgoing edges simultaneously, up to the K -hop neighborhood ($K=2$), with separate parameters for each edge direction and diffusion step.

AGCRN. The T&S global-local Adaptive Graph Convolutional Recurrent Network by Bai et al. [2020].

Graph WaveNet. The deep T&S MPTCN from Wu et al. [2019]. It interleaves dilated causal convolutions [Oord et al., 2016] with the same diffusion operator used in DCRNN. In addition to the provided graph structure, it introduces a learnable adjacency matrix derived from cosine similarities between node embeddings, which are optimized end-to-end. This auxiliary graph is used to propagate messages with distinct weights from the ones used along with the original topology.

Table 5.3. Forecasting MAE for global, local, and global-local models across real-world datasets. In **bold**, the best-performing model within each group (local, global, global-local).

	METR-LA	PEMS-BAY	CER-E	AQI	EngRAD
Local models					
FC-GRU	3.56 ± 0.03	2.32 ± 0.01	713.01 ± 8.27	18.24 ± 0.07	55.37 ± 1.57
LocalGRUs	3.69 ± 0.00	1.91 ± 0.00	508.95 ± 1.48	14.75 ± 0.02	58.80 ± 0.31
Global models					
GRU	3.54 ± 0.00	1.77 ± 0.00	456.98 ± 0.61	14.02 ± 0.04	47.41 ± 0.57
TTS-IMP	3.34 ± 0.01	1.72 ± 0.00	439.13 ± 0.51	12.74 ± 0.02	44.48 ± 0.24
TTS-AMP	3.24 ± 0.01	1.66 ± 0.00	431.33 ± 0.68	12.30 ± 0.02	44.62 ± 0.35
T&S-IMP	3.35 ± 0.01	1.70 ± 0.01	443.85 ± 0.99	12.87 ± 0.02	45.55 ± 0.33
T&S-AMP	3.22 ± 0.02	1.65 ± 0.00	456.72 ± 3.91	12.29 ± 0.02	43.93 ± 0.55
Global-local models (with node embeddings)					
GRU	3.15 ± 0.03	1.59 ± 0.00	421.50 ± 1.78	13.73 ± 0.04	46.83 ± 0.19
TTS-IMP	3.08 ± 0.01	1.58 ± 0.00	412.44 ± 2.80	12.33 ± 0.02	43.96 ± 0.42
TTS-AMP	3.06 ± 0.01	1.58 ± 0.01	412.95 ± 1.28	12.15 ± 0.02	43.70 ± 0.33
T&S-IMP	3.10 ± 0.01	1.59 ± 0.00	417.71 ± 1.28	12.48 ± 0.03	44.90 ± 0.33
T&S-AMP	3.07 ± 0.02	1.59 ± 0.00	416.74 ± 1.57	12.17 ± 0.05	43.14 ± 0.19

5.6.2 Global, local, and global-local architectures

This experiment investigates the trade-offs between global and local modeling strategies and explores the benefits of hybrid designs that integrate both. Specifically, we evaluate the performance of purely global, purely local, and global-local architectures to better understand their respective roles and interactions. The evaluation spans real-world datasets from four domains: traffic networks (METR-LA and PEMS-BAY), energy consumption analytics (CER-E), air quality monitoring (AQI), and weather forecasting (EngRAD). Dataset characteristics are detailed in [Section 2.6](#).

For consistency, we adopt the same data splits and pre-processing pipeline used in [\[Cini, Marisca, et al., 2022; Marisca et al., 2022\]](#). We split datasets into windows of T time steps, and train the models to predict the next H observations. We set $T = 12$, $H = 12$ for the traffic datasets, $T = 48$, $H = 6$ for CER-E, $T = 24$, $H = 3$ for AQI, and $T = 24$, $H = 6$ for EngRAD. Then, for all datasets except for AQI

and EngRAD, we divide the obtained windows sequentially into 70%/10%/20% splits for training, validation, and testing, respectively. For AQI, we use as the test set the months of March, June, September, and December, following Yi et al. [2016]. For EngRAD, containing 3 years of data, we use 2020 for testing and one week per month in 2019 for validation; all other samples that do not overlap with the validation and test sets are used to train the models.

Table 5.3 summarizes the forecasting results, reporting the MAE averaged over the prediction horizon for each model and dataset. Overall, graph-based architectures consistently outperform both local and global baselines across all settings. The performance gap is especially striking when comparing against fully global models, which struggle in diverse environments. As expected, local models scale poorly and deliver subpar performance – particularly in the EngRAD dataset, where the homogeneity of data-generating processes reduces the benefits of local modeling. In contrast, hybrid global-local models demonstrate strong performance improvements in all other scenarios, confirming the advantages of integrating local information within a global framework. Nevertheless, as discussed in **subsection 5.1.1**, these architectures are less flexible in inductive settings where unseen nodes are introduced at test time.

Additional trends emerge when examining architectural variants: anisotropic message-passing generally outperforms isotropic schemes, and TTS models tend to match or exceed the performance of T&S architectures. Notably, while purely global or local RNNs fall short, their global-local counterpart ranks among the top-performing models, often surpassing graph-based baselines that do not leverage node embeddings.

5.6.3 Large-scale forecasting

This experiment evaluates how well different architectures scale under realistic memory and time constraints, shifting the focus from pure accuracy to computational feasibility. We use six months of data – four for training, one for validation, and one for testing – and train models to forecast simultaneously at horizons of {00:30, 07:30, 11:00} hours.

We consider the CER-E (Full) dataset, which contains smart metering data from 6,435 nodes (see **Section 2.6**). To probe scalability concerning graph connectivity, we test two configurations: a sparse graph where each node has at most 100 neighbors (yielding 639,369 edges, 1.54% density), and a dense graph with no limit on node degree (3,186,369 edges, 7.69% density). We impose a fixed

Table 5.4. Forecasting error and computational performance on large-scale datasets. We report MAE over H -step-ahead predictions, $H = \{30\text{m}, 7\text{h}30\text{m}, 11\text{h}\}$, together with timings and memory consumption. We indicate in **bold** the best performance in each setting. * Subsampling was needed to comply with the memory constraints.

		CER-E (Full)					
		MAE			Resource utilization		
		30 m	7 h 30 m	11 h	Batch/s	Memory	Batch size
100-NN	DCRNN	0.22 $\pm$.00	0.28$\pm$.00	0.29$\pm$.00	1.43 $\pm$.02	11.10 GB	2
	Graph WaveNet	0.23 $\pm$.00	0.36 $\pm$.01	0.36 $\pm$.01	2.41 $\pm$.03	8.39 GB	1
	SGP	0.21$\pm$.00	0.30 $\pm$.00	0.31 $\pm$.01	117.32$\pm$8.36	2.21 GB	4096
Full graph	DCRNN	0.23 $\pm$.00	0.29$\pm$.00	0.29$\pm$.00	1.13 $\pm$.01	11.10 GB	1*
	Graph WaveNet	0.25 $\pm$.01	0.38 $\pm$.03	0.37 $\pm$.01	1.26 $\pm$.01	8.58 GB	1
	SGP	0.21$\pm$.00	0.30 $\pm$.00	0.31 $\pm$.01	115.85$\pm$1.60	2.21 GB	4096

GPU memory budget of 12 GB and adapt the batch size accordingly. When a batch size of one exceeds this limit, we subsample the graph edges to reduce memory usage. SGP follows a different protocol, first computing spatiotemporal embeddings offline and then training the decoder on uniformly sampled node representations. Each model is trained for one hour, selecting the weights corresponding to the lowest training error for final evaluation. To stay within the time budget, we omit validation during training, as this would otherwise consume a high amount of time for the baselines.

Table 5.4 presents the results of our scalability analysis. One key distinction lies in how batch size is defined across models. For SGP, each training sample corresponds to a single preprocessed spatiotemporal embedding $\bar{S}_t$. In contrast, baselines process full temporal windows $\langle \mathbf{X}_{t-T:t}, \mathbf{U}_{t-T:t}, \mathbf{A} \rangle$, with edge subsampling applied as needed to meet memory limits – losses are computed in both cases over all observations in the batch. The results highlight SGP’s efficiency under tight resource constraints: it achieves up to two orders of magnitude higher batch processing rates compared to other models, while maintaining stable GPU memory usage across both sparse and dense graph regimes. This contrasts with most baselines, which require dropping edges from a single-loaded graph to operate within the same 12 GB budget. Importantly, SGP also delivers competitive forecasting accuracy across all settings, confirming that its efficiency does not come at the expense of predictive performance.

Table 5.5. Comparison of forecasting architectures across five real-world datasets, measured by MAE. Best model performance for each dataset reported in **bold**.

		METR-LA	PEMS-BAY	CER-E	AQI	EngRAD
Naïve	Last	4.24±0.00	2.18±0.00	749.83±0.00	14.93±0.00	174.63±0.00
	Day	5.84±0.00	3.37±0.00	764.01±0.00	39.58±0.00	84.68±0.00
	Week	5.55±0.00	2.82±0.00	682.19±0.00	48.29±0.00	111.77±0.00
	FC-GRU	3.56±0.03	2.32±0.01	713.01±8.27	18.24±0.07	55.37±1.57
	GRU	3.54±0.00	1.77±0.00	456.98±0.61	14.02±0.04	47.41±0.57
Global-local	GRU-E	3.15±0.03	1.59±0.00	421.50±1.78	13.73±0.04	46.83±0.19
	TTS-IMP	3.08±0.01	1.58±0.00	412.44±2.80	12.33±0.02	43.96±0.42
	TTS-AMP	3.06±0.01	1.58±0.01	412.95±1.28	12.15±0.02	43.70±0.33
	T&S-IMP	3.10±0.01	1.59±0.00	417.71±1.28	12.48±0.03	44.90±0.33
	T&S-AMP	3.07±0.02	1.59±0.00	416.74±1.57	12.17±0.05	43.14±0.19
	DCRNN	3.22±0.01	1.64±0.00	428.36±1.23	12.96±0.03	45.29±0.26
	Graph WaveNet	3.05±0.03	1.56±0.01	397.17±0.67	12.08±0.11	40.50±0.27
	AGCRN	3.16±0.01	1.61±0.00	444.80±1.25	13.33±0.02	42.90±0.32
	SGP	3.00±0.00	1.54±0.00	459.95±1.97	12.64±0.19	42.94±0.43

5.6.4 Forecasting benchmarks

We conclude our evaluation by benchmarking the global-local STGNNs and SGP against leading STGNN models and standard baselines. The experimental setup mirrors that of [Section 5.6.2](#), with the aim of assessing how our models compare to complex, state-of-the-art alternatives. Among the baselines, DCRNN operates as a purely global model. In contrast, both Graph WaveNet and AGCRN make use of node embeddings to obtain a latent adjacency matrix for message passing. Notably, AGCRN goes further by adapting its convolutional filters to the identity of each node via these embeddings.

[Table 5.5](#) summarizes the benchmarking results. The top portion of the table lists baseline models, including the Naïve approach and univariate/multivariate GRU variants. The relative performance of different seasonal Naïve baselines offers insight into the underlying structure of the data. SGP emerges as the top-performing model across both traffic datasets, outperforming all baselines and state-of-the-art STGNNs. Notably, in the METR-LA dataset, it achieves a substantial improvement over Graph WaveNet, the strongest STGNN baseline.

Surprisingly, the simple TTS-IMP model equipped with node embeddings delivers performance on par with complex STGNNs, despite its streamlined design and significantly lower parameter count. This result underscores the strength of local modeling and spatial encoding even in minimalist architectures.

5.7 Discussion and future directions

This chapter investigated the use of STGNNs for time series forecasting under irregular spatial sampling, where temporal observations are complete but the spatial structure is non-regular. We focused on two key design dimensions: global-local modeling and scalability. In the first part, we examined how the interplay between globality and locality impacts forecasting performance and proposed hybrid architectures that combine global spatiotemporal message passing with lightweight, node-specific components. Our contribution showed that learnable node embeddings can efficiently capture local effects while easing application in inductive settings, especially when regularized to promote structure and transferability [Cini, Marisca, et al., 2023b]. In the second part, we addressed the scalability bottleneck by introducing SGP [Cini, Marisca, et al., 2023a], a forecasting model that decouples spatiotemporal encoding from training. By relying on randomized temporal encoders and graph-based spatial diffusion, SGP achieves strong predictive performance at a fraction of the computational cost of standard STGNNs.

Altogether, this chapter identifies globality and scalability as two central challenges in spatiotemporal forecasting with graph-based models. Future work can build on these results in several directions. From a modeling perspective, designing even more transferable approaches for local adaptation remains an open challenge, especially in nonstationary or dynamic environments. Further investigation is needed to understand how structured priors and architectural choices impact generalization across time, space, and graph domains. Complementary lines of research should also explore the theoretical foundations of these models. In [Chapter 7](#), we begin addressing this by studying spatiotemporal over-squashing and its effect on information propagation. Finally, while this chapter assumed complete and synchronous time series, real-world forecasting often involves missing or asynchronous data. In [Chapter 6](#), we extend our investigation to such settings and study how to perform accurate forecasting when irregularities occur in both spatial and temporal dimensions.

Chapter 6

Prediction under irregular spatiotemporal sampling

In [Chapter 5](#), we showed how graph deep learning can be leveraged to forecast complete multivariate time series with spatial dependencies. This setting has received the most attention in prior research, leading most existing STGNNs to assume that input data are complete and regularly sampled. However, data from spatiotemporal domains are not rarely affected by irregularities in both space and time. Real-world sensor networks, for example, are subject to faults and communication failures, often resulting in missing values across the collected time series. In our formulation, missing values also include those not acquired at a given sensor due to asynchrony with respect to other sensors.

When missing data are sparse and occur randomly, the localized processing enforced by the inductive biases in STGNNs serves as an effective regularizer, allowing the model to interpolate missing values by exploiting nearby observations, as demonstrated by GRIN [[Cini, Marisca, et al., 2022](#)] in [Chapter 3](#). However, when data are missing in larger and contiguous blocks, retrieving relevant information from distant observations becomes critical for capturing the underlying dynamics. Addressing these cases typically requires additional structured processing layers to extend the receptive field, as shown by SPIN [[Marisca et al., 2022](#)] in [Chapter 4](#). Still, since inductive biases act as regularizers, deeper models may risk attenuating higher-frequency dynamics, potentially reducing the ability to exploit local information when present [[Rusch et al., 2023](#)]. Ideally, the receptive field should adapt to the availability of the observed data, expanding only when necessary. This calls for adaptive processing strategies

that align with the temporal and spatial structure of missingness in the data.

In this chapter, we present a graph-based forecasting framework specifically designed to operate under missing data conditions. The framework, named HD-TTS [Marisca et al., 2024], extracts multiscale spatiotemporal representations by employing a hierarchical downsampling strategy. In time, downsampling is interleaved with temporal message passing to isolate dynamics at different resolutions. In space, we resort to graph pooling [Grattarola et al., 2024] to build a hierarchy of coarsened graphs, which gradually distill the global information necessary for compensating localized gaps. The outputs from each scale are then adaptively weighted based on the input observations and the specific missing data pattern. The architecture ensures scalability through a TTS factorization of processing and pre-computed downsampling operators, making it suitable also for large-scale settings.

The rest of the chapter is organized as follows. [Section 6.1](#) introduces the key operations required to perform hierarchical downsampling in time and space. [Section 6.2](#) presents the proposed HD-TTS architecture, detailing how multiscale spatiotemporal representations are extracted and dynamically reweighted by an attention-based decoder to generate the final predictions, posing a particular focus on scalability. Related work is discussed in [Section 6.3](#), while [Section 6.4](#) reports an extensive empirical evaluation on synthetic and real-world benchmarks, comparing HD-TTS against state-of-the-art models under different missing data regimes. We further explore model interpretability in [Section 6.5](#), where we analyze how attention weights evolve across scales in response to missingness and dynamics. Finally, [Section 6.6](#) concludes the chapter by summarizing key insights and identifying directions for future research.

Reference papers

This chapter’s content is partly based on material from the following articles.

Ivan Marisca, Cesare Alippi, and Filippo Maria Bianchi (2024). [Graph-based Forecasting with Missing Data through Spatiotemporal Downsampling](#). In: *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. Proceedings of Machine Learning Research, pp. 34846–34865.

6.1 Spatiotemporal downsampling

Downsampling in temporal data is a common operation, often used to reduce the sample complexity or filter out noisy measurements [Harris, 2022]. In classical signal processing, it is implemented by applying a low-pass filter and then keeping only 1-every- k samples, with k being the downsampling factor [Strang and Nguyen, 1996]. In deep learning, this approach is replicated by strided or dilated operations in TMP operators [Yu and Koltun, 2016; Oord et al., 2016; Chang et al., 2017], which exploit the structural regularity of temporal data. More generally, downsampling a sequence from T_l to T_{l+1} time steps can be conveniently expressed by a *temporal downsampling matrix* $\mathbf{D}_l \in \mathbb{R}^{T_{l+1} \times T_l}$, which defines how original time steps are aggregated to produce the reduced sequence. For example, matrix $\mathbf{D}_l$ obtained by keeping only every k -th row in the $T_l \times T_l$ identity matrix corresponds to the standard 1-every- k downsampling presented above.

Being non-Euclidean structures, the concept of downsampling for graphs is less straightforward and tied to the procedure for graph coarsening. In the GNNs literature, the latter is known as *graph pooling*. Let $\mathbf{A}^{(k)} \in \mathbb{R}_{\geq 0}^{N_k \times N_k}$ be a graph with node features $\mathbf{X}^{(k)} \in \mathbb{R}^{N_k \times d_x}$, with superscript index $\cdot^{(k)}$ indicating the k -th spatial scale. The Select-Reduce-Connect (SRC) framework by Grattarola et al. [2024] expresses a graph pooling operator $\text{POOL} : (\mathbf{A}^{(k)}, \mathbf{X}^{(k)}) \mapsto (\mathbf{A}^{(k+1)}, \mathbf{X}^{(k+1)})$ as the combination of three functions:

SEL : $(\mathbf{A}^{(k)}, \mathbf{X}^{(k)}) \mapsto \mathbf{S}_k \in \mathbb{R}^{N_{k+1} \times N_k}$. Defines how to aggregate the N_k nodes in the input graph into N_{k+1} *supernodes* in the coarsened graph.

RED : $(\mathbf{X}^{(k)}, \mathbf{S}_k) \mapsto \mathbf{X}^{(k+1)} \in \mathbb{R}^{N_{k+1} \times d_x}$. Creates the supernode features by combining the features of the nodes assigned to the same supernode. A typical reduction operation is $\mathbf{X}^{(k+1)} = \mathbf{S}_k \mathbf{X}^{(k)}$.

CON : $(\mathbf{A}^{(k)}, \mathbf{S}_k) \mapsto \mathbf{A}^{(k+1)} \in \mathbb{R}_{\geq 0}^{N_{k+1} \times N_{k+1}}$. Generates the edges (and, potentially, the edge features) by connecting the supernodes. A typical CON is $\mathbf{A}^{(k+1)} = \mathbf{S}_k \mathbf{A}^{(k)} \mathbf{S}_k^\top$, with the transpose $\mathbf{S}_k^\top$ sometimes replaced with the pseudo-inverse $\mathbf{S}_k^+$.

A fourth function is used to *lift*, i.e., upsample, supernode features to the associated nodes in the original graph:

LFT : $(\mathbf{X}^{(k+1)}, \mathbf{S}_k) \mapsto \widetilde{\mathbf{X}}^{(k)} \in \mathbb{R}^{N_k \times d_x}$. Can be implemented as $\widetilde{\mathbf{X}}^{(k)} = \mathbf{S}_k^\top \mathbf{X}^{(k+1)}$ (or $\widetilde{\mathbf{X}}^{(k)} = \mathbf{S}_k^+ \mathbf{X}^{(k+1)}$).

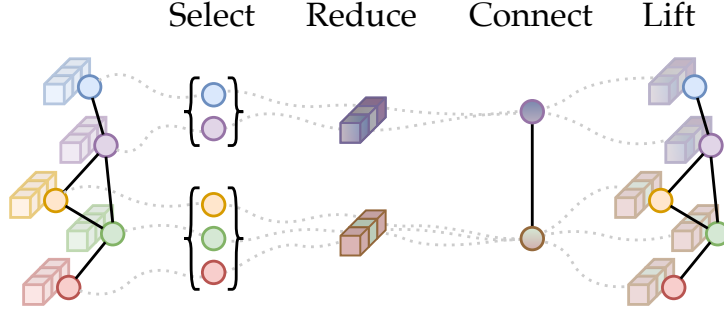


Figure 6.1. Schematic representation of the SRC pooling framework by Grattarola et al. [2024]. Nodes are first grouped in clusters (called supernodes) and then reconnected, forming a coarsened graph. Node features are aggregated through a reduction operation, and possibly propagated back to the original graph nodes through a lifting operation.

Figure 6.1 shows a schematic representation of these functions within the SRC framework. As for TMP and MP, there is a strong analogy also between downsampling in time and space. Notably, the selection matrix S_k , hereinafter called *spatial downsampling matrix*, plays the same role as D_l . Both matrices, indeed, reduce the input dimensionality conditioned on the underlying structure of the data, expressed in the temporal domain through the T matrix. However, irregularities in arbitrary graphs make it challenging to define concepts like “1-every- k ”, failing to properly mirror the temporal dimension. An interpretation is given by the k -MIS method [Bacciu et al., 2023], which relies on maximal k -independent sets to keep the nodes that cover uniformly the k -th power graph in the pooled graph.

k -MIS pooling k -MIS pooling [Bacciu et al., 2023] provides a graph-based analogue to evenly spaced coarsening in regular data, with theoretical guarantees on path distortion and topological preservation. It selects approximately equidistant nodes by computing maximal k -independent sets, which are then used as vertices of the reduced graph. Specifically, supernodes are chosen as centroids of the sets via a greedy procedure guided by a ranking vector π , which can be computed from node features. To minimize computational cost, this work adopts the non-trainable variant, which uses a fixed ranking ($\pi = 1$) across nodes. The trainable alternative, while more expressive, may be preferable when efficiency is not a concern. As the method operates on undirected graphs, directed graphs are symmetrized. Notably, k -MIS is an expressive pooling operator, meaning that distinguishable input graphs remain distinguishable

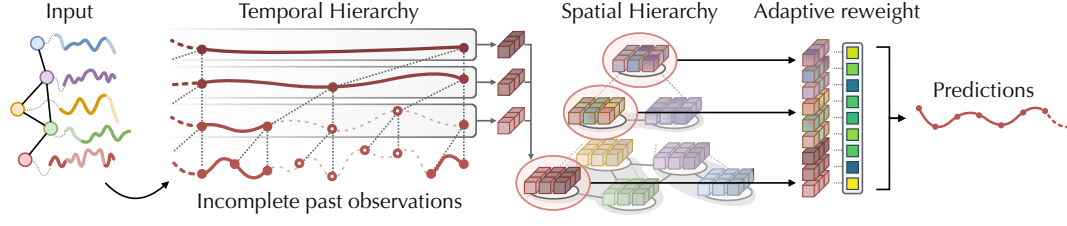


Figure 6.2. Overview of the proposed framework. The hierarchical design allows us to learn a pool of multi-scale spatiotemporal representations. Conditioned on the data and the missing value pattern, the attention mechanism dynamically combines representations from different scales to compute the predictions.

after pooling [Bianchi and Lachi, 2023].

6.2 Scalable hierarchical downsampling

In this section, we introduce Hierarchical Downsampling Time-Then-Space (HD-TTS), an architecture for graph-based forecasting of spatiotemporal data with missing values following arbitrary patterns. To learn spatiotemporal representations at different scales, we rely on hierarchical downsampling, which progressively reduces the size of the input and enables efficient learning of long-range dependencies. In time, HD-TTS learns multiple yet limited representations from the input sequences, each at a different temporal scale. Similarly, we process the spatial dimension by propagating messages along a hierarchy of pre-computed coarsened graphs. Notably, our approach combines the advantages of the T&S and TTS paradigms, enabling adaptive expansion of the receptive field while keeping the computational and memory complexity under control.

HD-TTS follows the forecasting framework defined in Equations 5.4 to 5.6, with the key components being (1) an **input encoder**, (2) a **temporal processing** module, (3) a **spatial processing** module, and (4) an **adaptive decoder**. The first three blocks extract hierarchical representations of the input at different spatial and temporal scales, while the last block reweights the representations and outputs the predictions. The whole architecture is trained end-to-end to minimize the forecasting error over the prediction horizon (as defined in Equation 2.9). Notably, our approach does not require missing data imputation as a pre-processing step. In contrast, HD-TTS handles missing data by automatically reweighing the different spatiotemporal dynamics present in the data while

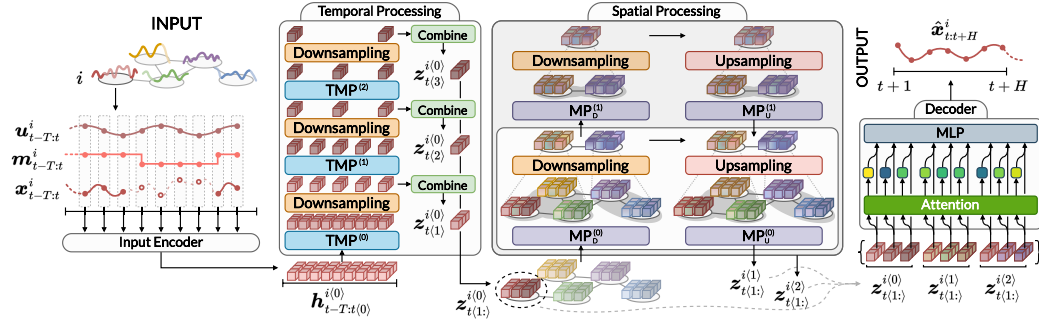


Figure 6.3. Architectural scheme of HD-TTS. Information associated with every i -th node and time step $t-k$ is encoded in vector $h_{t-k(0)}^{i(0)}$, then processed node-wise along the temporal dimension by alternating TMP and downsampling. Each l -th layer output is embedded in a single vector $z_{t(l)}^{i(0)}$, then processed along the spatial dimension by alternating MP, pooling, and lifting operations, obtaining $z_{t(l)}^{i(k)}$. The $L(K+1)$ encodings $z_{t(1)}^{i(0)}$ are adaptively combined through an attention-based decoder to obtain the predictions.

learning to solve the prediction task. Figure 6.3 shows an overview of the architecture. In the figure and the following, we use the superscript and subscript indices $\cdot^{(k)}_{(l)}$ to refer to the k -th spatial and l -th temporal scale, respectively, with $k = l = 0$ corresponding to the input scales.

6.2.1 Learning hierarchical representations

Consider a window of $T \geq 1$ past observations $\mathbf{X}_{t-T:t}$ with missing values indicated by mask $\mathbf{M}_{t-T:t}$ and paired with exogenous $\mathbf{U}_{t-T:t}$. In the encoding phase, we process these sequences to extract a pool of $L(K+1)$ embeddings $\{z_{t(l)}^{(k)}\}_{l=1, \dots, L}^{k=0, \dots, K}$ are associated with L temporal and k spatial scales.

Input encoder

The input encoder combines the information associated with the i -th node at time step $t-k$, in an embedding vector $h_{t(0)}^{i(0)} \in \mathbb{R}^d$ with an MLP:

$$h_{t(0)}^{i(0)} = \text{MLP}([x_t^i \parallel m_t^i \parallel u_t^i \parallel n_\theta^i]), \quad (6.1)$$

where, n_θ^i is a vector of node-specific trainable parameters that facilitates node identification [Cini, Marisca, et al., 2023b], as discussed in Section 5.3. At this stage, missing values are replaced using the last observed value. Obtained

representations are then refined by downstream components that leverage temporal and spatial dependencies.

Temporal processing

The temporal processing module acts on $\mathbf{h}_{t-T:t\langle 0 \rangle}^{i\langle 0 \rangle}$ to generate, for every i -th node, L representations $\{\mathbf{z}_{t\langle l \rangle}^{i\langle 0 \rangle}\}_{l=1,\dots,L}$, each associated with a different temporal scale but the same initial spatial scale. In particular, the l -th processing layer takes as input the sequence $\mathbf{h}_{t-T_l:t\langle l \rangle}^{i\langle 0 \rangle}$ and outputs a new sequence of updated encodings decimated by a factor of δ . Hence, the output sequence has length $T_{l+1} = \lceil \frac{T_l}{\delta} \rceil$, with $T_0 = T$ being the length of the original time series. We can write the operations performed by layer $l = 0, \dots, L-1$ as

$$\mathbf{h}_{t-T_{l+1}:t\langle l+1 \rangle}^{i\langle 0 \rangle} = \mathbf{D}_l \left(\text{TMP}^{(l)} \left(\mathbf{h}_{t-T_l:t\langle l \rangle}^{i\langle 0 \rangle} \right) \right), \quad (6.2)$$

where $\text{TMP}^{(l)}$ is any TMP operator (as defined in Equation 2.24) acting at temporal scale l and $\mathbf{D}_l \in \mathbb{R}^{T_{l+1} \times T_l}$ is the temporal downsampling matrix defined in Section 6.1. The downsampling operation progressively expands the temporal receptive field, allowing us to capture increasingly slower temporal dynamics in the input.

We then combine the sequences $\mathbf{h}_{t-T_l:t\langle l \rangle}^{i\langle 0 \rangle}$ at each layer l into a single representation $\mathbf{z}_{t\langle l \rangle}^{i\langle 0 \rangle} \in \mathbb{R}^d$, i.e.,

$$\mathbf{z}_{t\langle l \rangle}^{i\langle 0 \rangle} = \text{COMBINE} \left(\mathbf{h}_{t-T_l:t\langle l \rangle}^{i\langle 0 \rangle} \right). \quad (6.3)$$

A simple implementation is to return the encoding associated with the last time step, i.e., $\mathbf{z}_{t\langle l \rangle}^{i\langle 0 \rangle} = \mathbf{h}_{t\langle l \rangle}^{i\langle 0 \rangle}$.

Spatial processing

The set of multi-scale temporal encodings $\{\mathbf{z}_{t\langle l \rangle}^{i\langle 0 \rangle}\}_{l=1,\dots,L}$ is processed spatially with an analogous hierarchical approach. In particular, representations associated with the k -th spatial scale are obtained by applying k times a combination of an MP layer followed by the graph pooling operation described in (Section 6.1):

$$\mathbf{R}_{t\langle l \rangle}^{(k+1)} = \mathbf{S}_k \left(\text{MP}_D^{(k)} \left(\mathbf{R}_{t\langle l \rangle}^{(k)}, \mathbf{A}^{(k)} \right) \right), \quad (6.4)$$

where $\mathbf{S}_k \in \mathbb{R}^{N_{k+1} \times N_k}$ is the spatial downsampling matrix generated by the pooling's SEL function, $\text{MP}_D^{(k)}$ is the MP operator applied at k -th layer before downsampling, $\mathbf{A}^{(0)} = \mathbf{A}$ and $\mathbf{R}_{t\langle l \rangle}^{(0)} = \mathbf{Z}_{t\langle l \rangle}^{(0)}$. Note that, at each spatial resolution

level k , the encodings $\mathbf{R}_{t\langle l \rangle}^{(k)} \in \mathbb{R}^{N_k \times d}$ are associated with supernodes rather than the nodes in the original graph.

Since the predictions must ultimately be computed for the nodes in the original graph, the coarsened spatial representations must be brought back to the initial spatial resolution. For this reason, we mirror the operations carried out in Equation 6.4 and recursively lift supernodes' encodings back to the associated nodes in previous layers. Starting from the coarsened representation $\tilde{\mathbf{R}}_{t\langle l \rangle}^{(k,k)} = \mathbf{R}_{t\langle l \rangle}^{(k)}$, the lifting step from layer $j + 1$ to j can be described as

$$\tilde{\mathbf{R}}_{t\langle l \rangle}^{(k,j)} = \text{MP}_{\mathbf{U}}^{(j)} \left(\mathbf{S}_j^+ \left(\tilde{\mathbf{R}}_{t\langle l \rangle}^{(k,j+1)} \right), \mathbf{A}^{(j)} \right). \quad (6.5)$$

After k recursive application of Equation 6.5, the representations associated with the k -th spatial scale $\mathbf{R}_{t\langle l \rangle}^{(k)}$ are propagated to the original graph and we assign $\mathbf{Z}_{t\langle l \rangle}^{(k)} = \tilde{\mathbf{R}}_{t\langle l \rangle}^{(k,0)}$. The details are depicted in Figure 6.4.

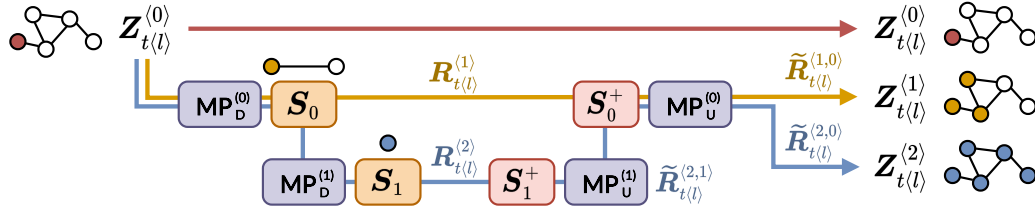


Figure 6.4. Details of the spatial processing procedure.

This hierarchical processing is inspired by the design of filterbanks in digital and graph signal processing [Strang and Nguyen, 1996; Tremblay and Borgnat, 2016; Tremblay et al., 2018]. Graph filterbanks often rely on specific node decimation operators that discard approximately half of the nodes at each application [Shuman et al., 2015]. Despite completely removing some nodes, when such techniques are paired with MP, the original graph signal can be reconstructed without losses from the pooled one. This approach has been adapted to the graph pooling framework by Bianchi et al. [2020] and represents an alternative to the adopted k -MIS method.

6.2.2 Adaptive decoder

The $L(K + 1)$ representations $\left\{ \mathbf{Z}_{t\langle l \rangle}^{(k)} \right\}_{l=1, \dots, L}^{k=0, \dots, K}$ obtained in previous steps are associated with different temporal (l) and spatial (k) scales. To condition the

importance of a representation to the prediction based on the input dynamics and the missing data pattern, we first compute an adaptive weight $\alpha_{t\langle l \rangle}^{i\langle k \rangle} \in [0, 1]$ for each representation as

$$\left\{ \alpha_{t\langle l \rangle}^{i\langle k \rangle} \right\} = \underset{l,k}{\text{softmax}} \left\{ \mathbf{w}_\alpha^\top \mathbf{z}_{t\langle l \rangle}^{i\langle k \rangle} \right\}_{l=1,\dots,L}^{k=0,\dots,K} \quad (6.6)$$

where $\mathbf{w}_\alpha \in \mathbb{R}^d$ is a vector of learnable parameters. The weights can be interpreted as soft attention coefficients that select the appropriate spatiotemporal scale for computing the predictions. We use the weights to weigh and combine the representations, followed by a multi-step MLP decoder to obtain the predictions individually for each i -th node:

$$\hat{\mathbf{x}}_{t:t+H}^i = \text{MLP} \left(\sum_{l=1}^L \sum_{k=0}^K \alpha_{t\langle l \rangle}^{i\langle k \rangle} \mathbf{z}_{t\langle l \rangle}^{i\langle k \rangle} \right). \quad (6.7)$$

This adaptive approach allows the model to focus on different scales based on the available observations, effectively bypassing blocks of missing data by relying on coarser-scale representations when necessary.

Multi-step decoder In some applications, the different dynamics captured by the multi-scale representations might be more or less relevant to the prediction depending on the distance of the target time step from the last observations. The structured multi-scale encodings, paired with the adaptive weighting mechanism, allow us to easily compute H different score sets $\left\{ \alpha_{t+h\langle l \rangle}^{i\langle k \rangle} \right\}_{h=0,\dots,H-1}$ for each temporal and spatial scales l and k , respectively. Hence, the H predictions are obtained, $\forall h \in [0, H)$, from H different aggregated representations as

$$\hat{\mathbf{x}}_{t+h}^i = \text{MLP} \left(\sum_{l=1}^L \sum_{k=0}^K \alpha_{t+h\langle l \rangle}^{i\langle k \rangle} \mathbf{z}_{t\langle l \rangle}^{i\langle k \rangle} \right). \quad (6.8)$$

The dynamically computed scores offer a tool to analyze how the model focuses on different scales according to the dynamics and observability of the input. Moreover, when the multi-step decoding strategy is adopted, they provide further insights in response to the input dynamics. In [Section 6.5](#), we show this empirically by inspecting the differences in the scores computed for the first and last time steps in the forecasting horizon in a controlled environment.

6.2.3 Computational scalability

Maintaining multiple representations adds a computational burden to the already expensive operations performed by STGNNs for standard spatial and temporal processing. For this reason, we adopted a series of architectural design choices to reduce the computational and memory complexity.

As previously discussed, we follow the TTS paradigm (Section 5.1.2), which enables performing message passing on a single static graph, regardless of the length of the sequence being processed. Additionally, by integrating downsampling operations, the amount of computation reduces progressively with the layers, as the number of time steps and nodes is decimated. Temporal processing, for instance, depends on the dilation factor chosen rather than the number of layers, scaling as $\mathcal{O}(NT(\delta/(\delta-1)))$ in contrast to $\mathcal{O}(NTL)$. Notably, this strategy has a positive impact also on the spatial and temporal receptive fields, which grow exponentially faster than standard TTS architectures, *ceteris paribus*.

The chosen k -MIS pooling operator belongs to the class of *sparse* and *non-trainable* pooling methods, which have a low memory footprint and do not require additional computation. A pooling operator is said to be sparse if every node is assigned to at most one supernode. Additionally, pooling operators whose selection matrix S_k is learned together with the model's weights are called trainable (non-trainable otherwise). The use of non-trainable methods allows us to compute the downsampling matrices once in a pre-processing step, making the spatial downsampling computationally comparable to the temporal one. Nonetheless, while it can also act as a regularization for large models with high capacity, this lack of flexibility might introduce a too restrictive bias for its strong reliance on graph topology.

6.3 Related work

Numerous techniques exist for handling missing data in multivariate time series [Li and Marlin, 2020; Shukla and Marlin, 2019]. GRU-D [Che et al., 2018] modifies the standard GRU recurrent cell (Equations 2.28 to 2.31) to impute missing values encountered while processing the sequence; the learned representations serve as sequence encodings and can be used for different downstream tasks, including imputation and prediction. A similar idea is later proposed by Tang et al. [2020], who instead use an LSTM [Hochreiter and Schmidhuber, 1997] to model temporal dynamics. Building on the work by

You et al. [2020], Yalavarthi et al. [2024] propose a graph-based architecture tailored to forecasting multivariate time series rather than spatiotemporal data, where graph edges connect channels with timestamps for which they have an observation, forming a bipartite graph. Other approaches have been proposed for classification [Shukla and Marlin, 2020] and anomaly detection [Bianchi et al., 2019]. Finally, models based on neural ODEs [Chen et al., 2018] have been proposed to handle continuous-time time series, and hence can fit irregular sampling and missing data [Rubanova et al., 2019; De Brouwer et al., 2019; Biloš et al., 2021]. Yet, none of these methods explicitly models the spatial dependencies in the data.

As discussed in previous chapters, prominent representatives of STGNNs for forecasting include MPRNNs, which integrate MP within temporal processing [Seo et al., 2018; Li et al., 2018b; Zhang et al., 2018; Yu et al., 2019; Bai et al., 2020], and convolutional approaches, e.g., MPTCNs, which apply convolutions or attention mechanisms across temporal and spatial dimensions [Yu et al., 2018; Wu et al., 2019; Wu et al., 2022]. Remarkably, none of these methods explicitly consider missing data in the input. GRIN [Cini, Marisca, et al., 2022], SPIN [Cini, Marisca, et al., 2022], and other STGNNs [Wu et al., 2021; Liu et al., 2023b] specifically address missing data in time series with relational information but focus on imputation. Zhang et al. [2022] targets irregularly sampled multivariate time series through GNNs without intermediate imputation, although it is designed for classification. Forecasting with missing data, instead, is explored by a limited number of works on STGNNs in the context of traffic analytics [Zhong et al., 2021; Wang et al., 2023]. Finally, Gravina et al. [2024b] introduce TG-ODE to learn spatial and temporal dynamics of irregularly-sampled sequences of graphs in continuous-time settings.

Pyramidal approaches to process inputs with missing (or noisy) data have been extensively studied in graph [Shuman et al., 2015; Tremblay and Borgnat, 2016] and discrete-time [Strang and Nguyen, 1996] signal processing. The idea of using a hierarchy of either temporal or spatial representations has also been adopted by deep learning models for time series classification [Cui et al., 2016], forecasting [Wang and Jing, 2022; Chen et al., 2023], and forecast reconciliation [Rangapuram et al., 2021; Rangapuram et al., 2023]. In particular, Cini et al. [2024] propose an approach for hierarchical time series forecasting where learnable graph pooling operations are used to cluster the time series.

6.4 Empirical evaluation

In this section, we report the results of the empirical analysis of our approach in different synthetic and real-world settings. We use MAE as the figure of merit, averaged over only valid observations.

HD-TTS implementation The proposed framework is general and allows for various choices of TMP, MP, and downsampling operators; we describe the specific configuration adopted in our model. As TMP, we use a GRU [Cho et al., 2014] combined with the standard 1-every- k temporal downsampling, which results in something similar to a dilated RNN [Chang et al., 2017]. For MP, we consider the same isotropic and anisotropic variants presented in Section 5.2. We refer to the models resulting from these two variants as HD-TTS (IMP) and HD-TTS (AMP), respectively. We rely on the k -MIS pooling method (presented in Section 6.1) to obtain spatial downsampling matrices $\mathbf{S}_k \in [0, 1]^{N_{k+1} \times N_k}$ that assign each node to exactly one supernode. For RED, CON and LFT, we use, respectively,

$$\mathbf{X}^{\langle k+1 \rangle} = \mathbf{S}_k \mathbf{X}^{\langle k \rangle}, \quad \mathbf{A}^{\langle k+1 \rangle} = \mathbf{S}_k \mathbf{A}^{\langle k \rangle} \mathbf{S}_k^\top, \quad \widetilde{\mathbf{X}}^{\langle k \rangle} = \mathbf{S}_k^+ \mathbf{X}^{\langle k+1 \rangle}. \quad (6.9)$$

Baselines We consider as baselines key methods among those introduced in Section 5.6.1. Since they do not handle missing values in input, we impute them node-wise using the last observed value and concatenate the mask to the input as an exogenous variable. Note that this processing is the same followed by our approach. In addition, we consider approaches that are specifically designed to process time series with missing data:

GRU-I. A GRU model that imputes missing values with 1-step-ahead predictions during processing.

GRU-D. The GRU-based model with state decay by Che et al. [2018].

GRIN. A forecasting variant of GRIN [Cini, Marisca, et al., 2022] with an MLP predictive decoder.

Importantly, given the similarity of TTS-IMP and TTS-AMP with our respective models, except for the hierarchical processing, they represent an ablation study about the effectiveness of learning hierarchical representations in TTS architectures.

Simulated missing data patterns To evaluate model performance in relation to the structure of missing data, we simulate realistic missingness patterns by masking observations according to the distributions introduced in Section 2.1.3. In addition to the **Point** and **Block-T** settings already tested in Sections 3.4 and 4.5, we also consider the more complex **Block-ST** setting, defined in Equation 2.5. This pattern extends the temporal block missingness by simulating faults that affect groups of spatially correlated sensors within a localized region. Specifically, once a fault occurs at a given node, it propagates to all k -hop neighbors with probability $p_g[k]$, where $\mathbf{p}_g \in [0, 1]^K$ is a vector of Bernoulli means and K denotes the maximum propagation radius. In all cases, the parameters of the missing data process are kept constant across nodes and time steps, enforcing stationarity in the missing data distribution. For multivariate sensor observations, masking is applied independently across channels, potentially resulting in partially observed features for a given node and time step. During training, the generated masks are applied to both the input and the ground-truth sequences, ensuring the model is supervised only on held-out data. At test time, instead, the mask is applied only to the input, and forecasting metrics are computed over all available observations.

6.4.1 Controlled environment

In this experiment, we assess the performance of HD-TTS architectures on a synthetic environment we designed specifically for graph-based predictors and described in the following.

GraphMSO Inspired by a popular benchmark in time series forecasting [Bianchi et al., 2017], we introduced a new controlled environment for spatiotemporal data analysis called Graph Multiple Superimposed Oscillators (GraphMSO) [Marisca et al., 2024]. Given a graph with binary adjacency matrix $\mathbf{A} \in [0, 1]^{N \times N}$, we assign all nodes sinusoids with incommensurable frequencies $\sigma_t^i = \sin\left(\frac{t}{e^{i/N}}\right)$ between each other. We then propagate the signals over the graph as

$$\mathbf{X}_t = \mathbf{O}_t + \tilde{\mathbf{A}}_K^\top \mathbf{O}_t, \quad (6.10)$$

where $\tilde{\mathbf{A}}_K$ is obtained by randomly sampling 5 among all neighbors with at most K -hop distance, such that the i -th column in $\tilde{\mathbf{A}}_K$ has only 5 nonzero entries randomly selected from the i -th column of $\sum_{k=1}^K \mathbf{A}^k$. This choice makes the task more difficult as only the original graph $\mathbf{A}$ is provided to the models. The resulting time series are almost aperiodic, and, thus, difficult to predict unless

Table 6.1. Forecasting error (MAE) on GraphMSO with different missing data distributions. **Bold** formatting indicates the best result in each setting. [†]Models without spatial message passing.

Model	Point (5%)	Block-T	Block-ST
GRU-D [†]	0.385 $\pm$ 0.012	0.670 $\pm$ 0.020	1.081 $\pm$ 0.003
GRU-I [†]	0.322 $\pm$ 0.016	0.619 $\pm$ 0.011	1.064 $\pm$ 0.003
GRIN	0.163 $\pm$ 0.008	0.392 $\pm$ 0.031	0.895 $\pm$ 0.012
GRU [†]	0.346 $\pm$ 0.027	0.639 $\pm$ 0.011	1.137 $\pm$ 0.008
DCRNN	0.291 $\pm$ 0.277	0.645 $\pm$ 0.510	1.103 $\pm$ 0.001
AGCRN	0.067 $\pm$ 0.004	0.366 $\pm$ 0.013	1.056 $\pm$ 0.012
Graph WaveNet	0.089 $\pm$ 0.002	0.340 $\pm$ 0.001	0.955 $\pm$ 0.012
T&S-IMP	0.118 $\pm$ 0.009	0.323 $\pm$ 0.011	0.935 $\pm$ 0.005
T&S-AMP	0.063 $\pm$ 0.003	0.293 $\pm$ 0.020	0.868 $\pm$ 0.006
TTS-IMP	0.113 $\pm$ 0.008	0.271 $\pm$ 0.007	0.697 $\pm$ 0.005
TTS-AMP	0.096 $\pm$ 0.004	0.251 $\pm$ 0.004	0.669 $\pm$ 0.013
HD-TTS (IMP)	0.058 $\pm$ 0.004	0.247 $\pm$ 0.002	0.651 $\pm$ 0.023
HD-TTS (IMP)	0.062 $\pm$ 0.002	0.261 $\pm$ 0.009	0.679 $\pm$ 0.005

the graph structure is used to recover the initial sinusoids. As such, GraphMSO is a very effective benchmark to test the performance of spatiotemporal models. **Figure 6.5** illustrates the procedure for generating the GraphMSO dataset. For illustration purposes, we show a graph with a ring topology and propagate the sinusoids simply using the first-order adjacency matrix.

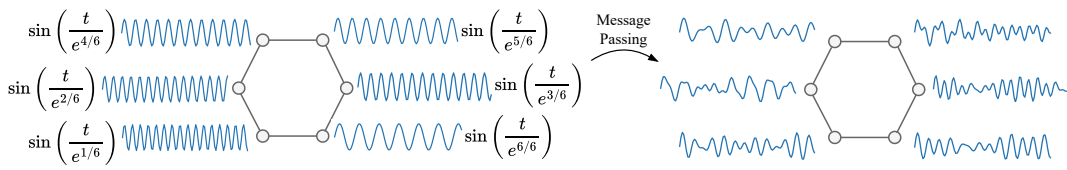


Figure 6.5. Generation of the GraphMSO dataset.

For our experiment, we generate a graph such that each node has exactly 3 incoming edges chosen at random. The number of nodes in the graph is $N = 100$, each time series has a length of 10,000 time steps, and the spatial order of GraphMSO is $K = 3$. The parameters used to inject the missing values are summarized in **Table 6.2**.

Table 6.2. Parameters used to generate the missing data masks in the different settings.

Parameter	GraphMSO			AQI	EngRAD / PV-US		
	Point	Block-T	Block-ST	Point	Block-T	Block-ST	
η	0.05	0.05	0.05	0.25	0.05	0.075	
p_f	0	0.01	0.005	0	0.01	0.003	
$s_{\min}$	–	8	8	–	4	6	
$s_{\max}$	–	48	48	–	12	21	
p_g	0	0	[1]	0	0	[0.33 0.15 0.05]	

Table 6.1 reports the results of the analysis. Notably, HD-TTS consistently outperforms all baselines across all evaluated settings. The performance gap is particularly pronounced when compared to more complex STGNNs, such as Graph WaveNet (GWNNet) and AGCRN. In principle, these models can reach all the necessary spatial information within a single message-passing layer, since they learn a dense connectivity matrix jointly with the model parameters. Interestingly, the strong performance of HD-TTS in the **Point** scenario suggests that learning dedicated representations across multiple spatiotemporal scales can also be beneficial when the missing information is less structured and more easily recoverable. This highlights the generality of the proposed approach, even in settings where the need for long-range context is less critical.

6.4.2 Forecasting benchmarks

In this experiment, we consider three of the real-world benchmarks presented in Section 2.6 following the experimental setup in Section 5.6 with these additional settings:

AQI Given the high number of missing values (25.67%), often occurring in blocks, we test the models on both the original data and in a **Point** setting with an additional $\eta = 0.25$ portion of data masked out uniformly at random.

PV-US We consider the subset of 1081 western farms and aggregate measurements at 20-minute intervals. We consider the **Block-T** and **Block-ST** settings.

EngRAD Introduced together with HD-TTS in [Marisca et al., 2024], this dataset

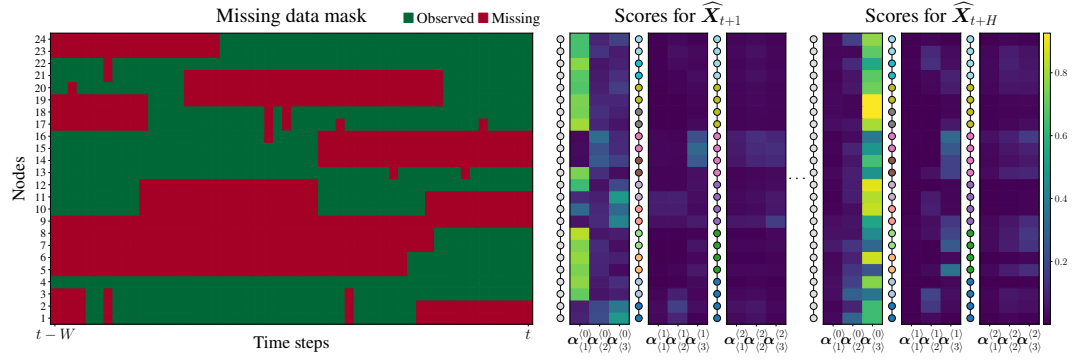
Table 6.3. Forecasting error (MAE) on real-world datasets with different missing data distributions. We use **bold** and underline formatting for best and second-best results, respectively, and N/A for runs exceeding resource capacity. [†]Models without spatial message passing.

Model	AQI		EngRAD		PV-US			
	Original	+ Point	Block-T	Block-ST	Block-T	Block-ST	Batch/s	GPU RAM
GRU-D [†]	18.26 $\pm$ 0.09	19.23 $\pm$ 0.08	5.29 $\pm$ 0.05	5.41 $\pm$ 0.01	4.04 $\pm$ 0.03	4.27 $\pm$ 0.01	5.07 $\pm$ 0.00	9.33 GB
GRU-I [†]	18.12 $\pm$ 0.03	19.07 $\pm$ 0.01	5.24 $\pm$ 0.04	5.39 $\pm$ 0.00	4.05 $\pm$ 0.02	4.29 $\pm$ 0.02	2.90 $\pm$ 0.00	10.28 GB
GRIN	16.85 $\pm$ 0.05	17.59 $\pm$ 0.06	4.91 $\pm$ 0.04	5.05 $\pm$ 0.00	3.62 $\pm$ 0.02	3.85 $\pm$ 0.07	1.52 $\pm$ 0.00	17.28 GB
GRU [†]	18.17 $\pm$ 0.03	19.19 $\pm$ 0.06	5.30 $\pm$ 0.03	5.42 $\pm$ 0.02	3.98 $\pm$ 0.02	4.14 $\pm$ 0.02	11.59 $\pm$ 0.04	12.01 GB
DCRNN	16.99 $\pm$ 0.09	17.51 $\pm$ 0.08	5.14 $\pm$ 0.06	5.33 $\pm$ 0.05	3.54 $\pm$ 0.01	3.76 $\pm$ 0.00	1.36 $\pm$ 0.01	19.72 GB
AGCRN	17.19 $\pm$ 0.06	17.92 $\pm$ 0.05	4.84 $\pm$ 0.01	5.10 $\pm$ 0.06	4.06 $\pm$ 0.01	4.20 $\pm$ 0.04	1.15 $\pm$ 0.01	23.40 GB
GWNet	15.89 $\pm$ 0.04	16.39 $\pm$ 0.14	4.59 $\pm$ 0.04	4.76 $\pm$ 0.03	3.48 $\pm$ 0.05	3.71 $\pm$ 0.03	2.12 $\pm$ 0.00	16.02 GB
T&S-IMP	16.54 $\pm$ 0.03	17.13 $\pm$ 0.05	4.98 $\pm$ 0.01	5.15 $\pm$ 0.03	3.60 $\pm$ 0.02	3.82 $\pm$ 0.03	2.68 $\pm$ 0.00	7.03 GB
T&S-AMP	16.15 $\pm$ 0.02	16.58 $\pm$ 0.10	4.93 $\pm$ 0.02	5.11 $\pm$ 0.05	N/A	N/A	N/A	N/A
TTS-IMP	16.25 $\pm$ 0.01	16.90 $\pm$ 0.26	4.81 $\pm$ 0.07	5.08 $\pm$ 0.04	3.50 $\pm$ 0.01	3.66 $\pm$ 0.02	18.84 $\pm$ 0.14	12.81 GB
TTS-AMP	15.63 $\pm$ 0.06	16.15 $\pm$ 0.05	4.70 $\pm$ 0.00	4.81 $\pm$ 0.06	3.46 $\pm$ 0.03	3.65 $\pm$ 0.05	14.26 $\pm$ 0.08	12.81 GB
HD-TTS	<u>15.50</u> $\pm$ 0.07	<u>15.94</u> $\pm$ 0.10	4.48 $\pm$ 0.01	4.64 $\pm$ 0.03	<u>3.47</u> $\pm$ 0.01	<u>3.62</u> $\pm$ 0.02	7.11 $\pm$ 0.03	10.86 GB
HD-TTS	15.35 $\pm$ 0.01	15.76 $\pm$ 0.07	<u>4.53</u> $\pm$ 0.03	<u>4.65</u> $\pm$ 0.04	<u>3.47</u> $\pm$ 0.02	3.61 $\pm$ 0.02	6.21 $\pm$ 0.02	10.86 GB

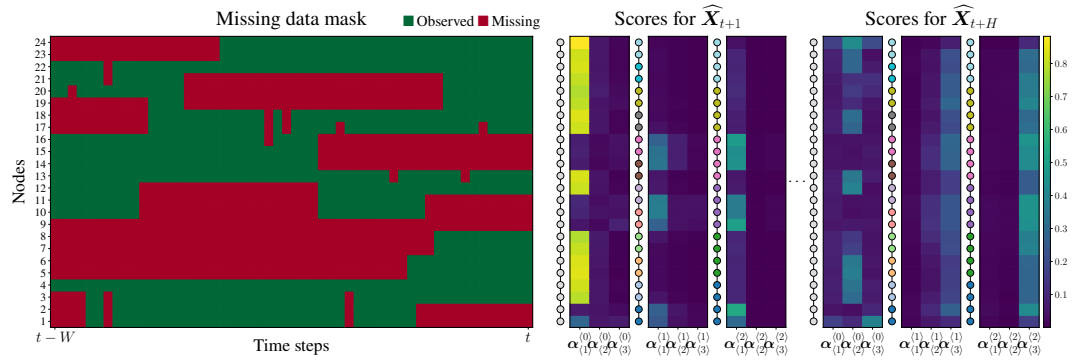
contains weather variables collected from a simulation model; as such, it has no missing values. We inject them through **Block-T** and **Block-ST** strategies.

Table 6.2 reports the parameters used to inject missing data, while evaluation results across all datasets and missingness patterns are in **Table 6.3**. Our proposed HD-TTS models consistently rank among the best-performing methods in all tested scenarios. Notably, the largest performance gains occur under **Block-ST** conditions, where missing values span both time and space. In this setting, HD-TTS significantly outperforms more complex STGNNs and both the IMP and AMP variants of the corresponding TTS model.

These findings underscore the benefits of combining hierarchical representations across multiple spatiotemporal scales, particularly in scenarios with sparse observations. On the AQI dataset – which contains a high proportion of missing observations – HD-TTS proves particularly effective, demonstrating its practical relevance in contexts where sampling irregularities are a real concern. Finally, results on training speed and memory utilization on PV-US highlight the scalability advantages of our approach compared to competing STGNNs, while also delivering higher forecasting accuracy.



(a) GraphMSO (different time series across nodes).



(b) Synthetic dataset with the same time series associated with each node.

Figure 6.6. Decoder weights in synthetic datasets with the same missing data pattern but different node signals as input. The graph used to produce this plot is the undirected line graph shown close to the scores associated with the first spatial resolution $k = 0$. Node colors in the graphs associated with higher spatial scales show how nodes are clustered at each scale k .

6.5 Interpretability of decoder weights

As previously discussed, the learned attention scores can be used to interpret which spatial and temporal scales the model relies on under varying input dynamics and missing data patterns. **Figure 6.6** visualizes the input missing data mask $M_{t-T:t}$ and the corresponding decoder scores. For clarity, we use an undirected line graph where adjacent nodes are placed consecutively, so that the associated scores align as adjacent rows in the score matrices. We adopt the **Block-ST** setting and use the original adjacency matrix A as the GSO $\tilde{A}$.

The decoder is multi-step, producing different attention scores for each forecast

horizon. We report scores for the first and last predicted steps, $\widehat{\mathbf{X}}_{t+1}$ and $\widehat{\mathbf{X}}_{t+H}$. Scores are grouped into $K + 1$ spatial blocks (with $K = 2$), each containing $L = 3$ temporal scales. For example, the leftmost block in the matrix corresponds to scores $\left[\alpha_{\langle 1 \rangle}^{(0)}, \alpha_{\langle 2 \rangle}^{(0)}, \alpha_{\langle 3 \rangle}^{(0)}\right]$ from the finest spatial scale.

We consider two input signals. In [Figure 6.6a](#), time series are generated as in the GraphMSO dataset. When recent data is available, the model focuses on fine-grained temporal scales. If a node has missing data, attention shifts to higher spatial levels, indicating reliance on neighboring nodes. Predicting further into the future (e.g., $\widehat{\mathbf{X}}_{t+H}$) naturally increases focus on coarser spatiotemporal scales. Interestingly, the most relevant scales vary even among nodes with similar missing data patterns, reflecting the influence of the specific dynamics of the input signal.

To isolate this effect, we repeat the experiment using identical chaotic signals at each node ([Figure 6.6b](#)). The model consistently assigns lower weight to representations from spatiotemporal regions with missing input, and the contrast between high and low scores becomes even more pronounced, highlighting the model’s ability to adaptively route attention based on observed data.

6.6 Discussion and future directions

In this chapter, we presented a novel framework for graph-based forecasting with missing data based on hierarchical spatiotemporal downsampling. Our model learns representations at different spatiotemporal scales, which are then combined by an interpretable attention mechanism to generate the forecasts. Thanks to this design, it can dynamically adapt the receptive field and handle different types of missing data patterns in a scalable fashion. Empirical comparisons against state-of-the-art methods on both synthetic and real-world datasets confirmed the effectiveness of the proposed approach, both in terms of forecasting accuracy and computational efficiency. Improvements are even more evident in the challenging settings with blocks of missing values in space and time, where other STGNNs models struggle.

The framework we proposed is very general and flexible. In this work, we explored only a subset of architectural choices for the temporal and spatial components. However, the modular nature of the framework readily supports alternative operators, including more expressive message-passing or sequence models. Similarly, while we adopted k -MIS pooling due to its alignment with

temporal downsampling and low computational cost, this choice may not be optimal in all domains. For example, in traffic networks, where signals can change abruptly between adjacent nodes (e.g., at crossroads), graph coarsening may discard useful fine-grained information. In contrast, environmental monitoring networks – with smoother spatial correlations – are more naturally suited to hierarchical pooling. This suggests that graph topology alone may not be sufficient to define effective downsampling, and incorporating data-driven or task-adaptive pooling strategies could be beneficial.

Chapter 7

Spatiotemporal over-squashing

While the strong empirical performance of STGNNs has largely driven their development, much less attention has been devoted to understanding their theoretical capabilities and limitations. This contrasts with the literature on static GNNs, which includes extensive analyses on expressivity [Xu et al., 2019; Bianchi and Lachi, 2023] and training dynamics, including over-smoothing [Cai and Wang, 2020; Di Giovanni et al., 2023b; Rusch et al., 2023] and over-squashing [Alon and Yahav, 2021; Topping et al., 2022; Di Giovanni et al., 2023a] (see Section 2.3.1). In particular, over-squashing – where information from distant nodes is compressed through graph bottlenecks – has emerged as a central limitation of message-passing architectures. However, these insights do not directly extend to STGNNs, in which information is propagated not only across the *graph* but also along the *time* axis.

The temporal axis represents an additional challenge, typically handled by imposing a locality assumption: adjacent time steps are presumed to be more correlated than distant ones. Under this assumption, the sequence behaves like a second, directed graph whose edges encode temporal order. As we stack local filters – or simply process longer sequences – increasing volumes of information are stored into fixed-width embeddings, a limitation already noted for recurrent architectures [Bengio et al., 1994; Pascanu et al., 2013]. When this temporal compression meets the constraints of spatial message passing, it creates a compound bottleneck we call *spatiotemporal over-squashing*: messages must cross rapidly expanding receptive fields in both space and time, exceeding the capacity of intermediate representations.

In this chapter, we investigate how the interplay between temporal and spatial

processing in STGNNs shapes learned representations, and how this process is limited by spatiotemporal over-squashing. We analyze information propagation patterns in existing STGNN designs by tracing the sensitivity of each embedding to the input features contained at neighboring nodes and time steps. Specifically, our study focuses on MPTCNs, whose temporal component is implemented through a shift operator that exchanges messages between adjacent time steps [Bai et al., 2018] – the time-domain analogue of message-passing on graphs.

Our findings reveal that architectures leveraging causal convolutions are, counterintuitively, more sensitive to information far apart in time, and we outline architectural modifications that mitigate this imbalance when required by the task. Furthermore, we demonstrate that spatiotemporal over-squashing affects T&S and TTS paradigms to the same degree. Thus, the computational benefits of TTS models come at no extra cost in terms of information bottlenecks, providing theoretical support for scalable designs.

The chapter is organized as follows. In [Section 7.1](#), we review spatial over-squashing in static GNNs and introduce the sensitivity analysis framework. Then, in [Section 7.2](#), we generalize this framework to spatiotemporal settings and define spatiotemporal over-squashing. [Section 7.3](#) studies temporal over-squashing in isolation through theoretical and empirical analysis of TCNs. In [Section 7.4](#), we extend this analysis to MPTCNs, providing theoretical guarantees, validated in both controlled and real-world settings, that clarify the role of architecture and topology. We conclude the chapter with final remarks and suggestions for future directions ([Section 7.6](#)).

Reference papers

This chapter’s content is partly based on material from the following articles.

Ivan Marisca, Jacob Bamberger, Cesare Alippi, and Michael M. Bronstein (2025). [Over-squashing in Spatiotemporal Graph Neural Networks](#). In: *arXiv preprint arXiv:2506.15507*.

7.1 Spatial over-squashing and sensitivity analysis

As introduced in Section 2.3.1, the over-squashing phenomenon in GNNs [Alon and Yahav, 2021; Topping et al., 2022] occurs when a large amount of information, resulting from the exponential expansion of the receptive field, gets compressed or “squashed” into a fixed-size representation vector during message passing, potentially resulting in information losses.

A widely adopted method to assess over-squashing, first proposed by Topping et al. [2022], is to analyze how much the initial features of a source node j affect the final representation of a target node i . This influence is quantified through a sensitivity analysis using the Jacobian:

$$\nabla_j \mathbf{h}^{i(L)} = \frac{\partial \mathbf{h}^{i(L)}}{\partial \mathbf{h}^{j(0)}} \in \mathbb{R}^{d \times d}, \quad (7.1)$$

where the *spectral* norm $\|\nabla_j \mathbf{h}^{i(L)}\|$ serves as a proxy of how strongly the j -th node’s initial features impact the representation of node i after L GNN layers [Topping et al., 2022; Di Giovanni et al., 2023a]. Relatively low values of this sensitivity indicate that node j has limited influence on node i , suggesting that information is being lost during message propagation.

Di Giovanni et al. [2024] studied over-squashing through sensitivity analysis for a broad class of MPNNs that fall within the following family:

$$\mathbf{h}^{i(l+1)} = \xi \left(\Theta_U^{(l)} \mathbf{h}^{i(l)} + \Theta_M^{(l)} \sum_{j \in \mathcal{N}(i)} \tilde{a}^{ji} \phi^{(l)}(\mathbf{h}^{i(l)}, \mathbf{h}^{j(l)}) \right) \quad (7.2)$$

where $\Theta_U^{(l)}, \Theta_M^{(l)} \in \mathbb{R}^{d \times d}$ are learnable weight matrices, $\phi^{(l)}$ is a $\mathcal{C}^1$ function, and ξ is a point-wise nonlinear activation function. This class includes common convolutional MPNNs, including GCN [Kipf and Welling, 2017], DCNN [Atwood and Towsley, 2016], and more expressive models like GIN [Xu et al., 2019] and GatedGCN [Bresson and Laurent, 2017] (presented in Section 2.3.2). Within this family, Di Giovanni et al. [2024] derived a bound that isolates the respective contributions of the graph topology and the architecture to the emergence of over-squashing. To simplify notation, throughout the chapter we use $(\cdot)_{ij} = \cdot [i, j]$ to indicate the element positioned at the i -th row and j -th column of a matrix.

Theorem C.1 (from Di Giovanni et al. [2024]). *Consider an MPNN with L layers defined as in Equation 7.2, with c_ξ being the Lipschitz constant of the update function after activation ξ , and $\theta_m \geq \|\Theta_M^{(l)}\|$ and $\theta_u \geq \|\Theta_U^{(l)}\|$ being the maximal norms over all*

weight matrices across layers in the message and update functions, respectively. For $i, j \in V$ we have:

$$\|\nabla_j \mathbf{h}^{i(L)}\| \leq \underbrace{(c_\xi \theta_m)^L}_{\text{model}} \underbrace{(\mathbf{S}^L)_{ji}}_{\text{topology}},$$

where $\mathbf{S} = \frac{\theta_u}{\theta_m} \mathbf{I} + c_1 \text{diag}(\tilde{\mathbf{A}}^\top \mathbf{1}) + c_2 \tilde{\mathbf{A}} \in \mathbb{R}^{N \times N}$, is the message-passing matrix such that the Jacobian of the message function $\phi^{(l)}$ w.r.t. the target (i) and neighbor (j) node features has bounded norms c_1 and c_2 , respectively.

This bound reveals two separate contributions to the emergence of over-squashing: the first term, $(c_\xi \theta_m)^L$, depends solely on the model's parameters and activation functions, reflecting the role of network depth and weight norms. The second term, $(\mathbf{S}^L)_{ji}$, quantifies how easily information can propagate from node j to node i over L message-passing layers.

7.2 Spatiotemporal over-squashing

Sensitivity analysis has become a widely used method for diagnosing over-squashing in static GNNs [Topping et al., 2022; Di Giovanni et al., 2023a; Di Giovanni et al., 2024; Gravina et al., 2025]. However, these analyses are limited to models with static node features and do not account for temporal dynamics. This is a significant limitation for STGNNs, where node attributes evolve over time and both spatial and temporal dependencies contribute to the learning process. Ignoring temporal propagation overlooks a critical dimension along which information can be squashed.

In this section, we extend sensitivity-based over-squashing analysis to the spatiotemporal setting. Our goal is to understand how node representations evolve under both temporal and spatial message passing, and to quantify how the influence of past node features is preserved or lost across layers of a forecasting STGNN. To isolate the role of spatiotemporal propagation, we exclude exogenous inputs $\mathbf{U}$ and static node attributes $\mathbf{V}$, focusing solely on the core input observations $\mathbf{X}$. For similar reasons, we assume all relational information is contained in $\mathbf{A}$, hence discarding edge attributes and adopting the GSO $\tilde{\mathbf{A}}$ as the message-passing matrix (see Section 2.3.2). We follow a

simplified version of the architecture family defined in [Section 5.1](#):

$$\mathbf{h}_{t-p}^{i(0)} = \text{ENCODER}(\mathbf{x}_{t-p}^i) \quad \text{for } p \in [0, T], i \in \mathcal{V}, \quad (7.3)$$

$$\mathbf{H}_{t-T:t}^{(l+1)} = \text{STMP}^{(l)}(\mathbf{H}_{t-T:t}^{(l)}, \tilde{\mathbf{A}}, \mathbf{T}) \quad \text{for } l \in [0, L), \quad (7.4)$$

$$\hat{\mathbf{y}}_t^i = \text{READOUT}(\mathbf{h}_t^{i(L)}) \quad \text{for } i \in \mathcal{V}, \quad (7.5)$$

where $\mathbf{T}$ is the temporal graph defined in [Section 2.1](#) and $\hat{\mathbf{y}}_t^i$ is the estimate for task label $\mathbf{y}_t^i$ (e.g., $\mathbf{y}_t^i = \mathbf{x}_{t:t+H}^i$ in the forecasting task).

Since both encoder and decoder operate independently across time and nodes, we have:

$$\frac{\partial \mathbf{h}_{t-p}^{i(0)}}{\partial \mathbf{x}_{t-p}^j} = \mathbf{0}, \quad \frac{\partial \hat{\mathbf{y}}_t^i}{\partial \mathbf{h}_t^{j(L)}} = \mathbf{0}, \quad \text{for } i \neq j, p \neq q.$$

Therefore, to assess information propagation, we analyze the Jacobian between node features before and after the stack of L STMP layers:

$$\nabla_p \mathbf{h}_t^{i(L)} = \frac{\partial \mathbf{h}_t^{i(L)}}{\partial \mathbf{h}_{t-p}^{j(0)}} \in \mathbb{R}^{d \times d}. \quad (7.6)$$

Unlike the static case ([Equation 7.1](#)), this spatiotemporal Jacobian captures the combined effect of temporal and spatial message passing, allowing us to investigate information flow across both dimensions.

Spatiotemporal processing, as abstractly defined in [Equation 7.4](#), can vary substantially across architectures, making it difficult to analyze a unified formulation that applies to all STGNNs. To enable a more principled analysis, we narrow our focus to architectures that interleave temporal and spatial message-passing layers, following the decoupled structure introduced in [Equations 2.39 to 2.40](#). This choice is motivated by two key factors. First, it allows us to isolate and study the interactions between existing spatial and temporal operators, leveraging theoretical results already established in the literature. Second, by adjusting the number of inner temporal and spatial layers, as well as the depth of the outer stack that contains them, we can flexibly analyze the characteristics of both T&S and TTS paradigms, as detailed in [Section 5.1.2](#).

In these architectures, information flows strictly along separate dimensions, as the STMP operation in [Equation 7.4](#) is decomposed into:

$$\mathbf{z}_{t-T:t}^{i(l)} = \text{TMP}_{\times L_T}^{(l)}(\mathbf{h}_{t-T:t}^{i(l)}, \mathbf{T}) \quad \forall i \in \mathcal{V}, \quad (7.7)$$

$$\mathbf{H}_{t-p}^{(l+1)} = \text{MP}_{\times L_S}^{(l)}(\mathbf{z}_{t-p}^{(l)}, \tilde{\mathbf{A}}) \quad \forall p \in [0, T), \quad (7.8)$$

where $\mathbf{z}_{t-T:t}^{i(l)}$ denotes the sequence of intermediate representations obtained from the temporal encoding of node i at layer l . The spatial operator MP then acts independently at each time step, preventing cross-dimensional interactions.

This decoupled formulation has important implications for sensitivity analysis. Given an STGNN with L layers, for each pair of nodes $i, j \in \mathcal{V}$, and time indices $p, q < T$, the Jacobian of the l -th STMP layer factorizes as:

$$\frac{\partial \mathbf{h}_{t-q}^{i(l+1)}}{\partial \mathbf{h}_{t-p}^{j(l)}} = \underbrace{\frac{\partial \mathbf{h}_{t-q}^{i(l+1)}}{\partial \mathbf{z}_{t-q}^{j(l)}}}_{\text{space}} \underbrace{\frac{\partial \mathbf{z}_{t-q}^{j(l)}}{\partial \mathbf{h}_{t-p}^{j(l)}}}_{\text{time}}. \quad (7.9)$$

This decomposition cleanly separates the contributions of spatial and temporal processing to representation learning. Notably, in the TTS setting with $L = 1$, this factorization provides a direct analytical tool to investigate $\nabla_j \mathbf{h}_t^{i(L)}$, and thus quantify how initial node features influence the network output at different nodes.

To deepen our analysis, we narrow our focus to a specific subfamily of STGNNs so as to enable further investigation of the factorized Jacobian in Equation 7.9. Specifically, we consider message-passing temporal convolutional networks (MPTCNs) [Yu et al., 2018; Wu et al., 2019], which combine TCNs [Oord et al., 2016; Bai et al., 2018] for temporal message passing with MPNNs [Gilmer et al., 2017] for spatial message passing, as presented in Section 2.5.2. This choice is driven by the architectural parallels between GNNs and TCNs, which allows for a more natural extension of over-squashing analysis to the spatiotemporal domain and facilitates drawing analogies between temporal and graph-based representation learning.

While the spatial aspects of over-squashing have been extensively studied in the GNN literature, less is known about how temporal processing may induce analogous bottlenecks in TCNs – what we refer to as *temporal over-squashing*. In what follows, we adopt an incremental approach: we first isolate and study the impact of temporal processing on sensitivity within TCNs only, and then extend our analysis to the full spatiotemporal case of MPTCNs in Section 7.4.

7.3 Over-squashing in TCNs

To isolate the temporal dynamics, we focus exclusively on the temporal processing component by analyzing encoder-decoder networks constructed from L_T

successive TCN layers positioned between encoding and readout operations, similar to [Equations 7.3 and 7.5](#). We consider the standard causal prior for temporal dependencies, resulting in $\mathbf{T}$ being a directed path graph connecting each time step with the following in time. Hence, we consider causal convolutions of a nonlinear filter with K elements over the temporal dimension [[LeCun, Bengio, et al., 1995](#); [Bai et al., 2018](#); [Oord et al., 2016](#)], as defined in [Equation 2.32](#). In the next sections, we analyze how information propagates over the temporal dimension and how this relates to temporal over-squashing.

7.3.1 Information propagation in TCNs

Causal convolutions in TCNs can be conveniently expressed using a lower-triangular Toeplitz matrix formulation, which facilitates the analysis of temporal propagation dynamics. Given a filter $\boldsymbol{\tau} = [\tau_0, \tau_1, \dots, \tau_{K-1}]^\top$ of size K and a univariate input sequence $\mathbf{h}_{t-T:t}^{(l)} \in \mathbb{R}^T$, we define the causal convolution operator $\ast_{\leftarrow}$ as:

$$\mathbf{h}_{t-T:t}^{(l+1)} = \boldsymbol{\tau} \ast_{\leftarrow} [\mathbf{0}_{K-1} \parallel \mathbf{h}_{t-T:t}^{(l)}], \quad (7.10)$$

where $\mathbf{0}_{K-1} \in \mathbb{R}^{K-1}$ is used to prepend $K-1$ zeros to the input to preserve the same length T in the output, and, for each p ,

$$\mathbf{h}_{t-p}^{(l+1)} = \sum_{k=0}^{K-1} \tau_k \cdot \mathbf{h}_{t-p-k}^{(l)}. \quad (7.11)$$

This operation can be equivalently represented as matrix multiplication, where the filter $\boldsymbol{\tau}$ is arranged into a lower-triangular Toeplitz matrix:

$$\mathbf{R} = \begin{bmatrix} \tau_0 & 0 & \cdots & 0 & 0 \\ \tau_1 & \tau_0 & \cdots & 0 & 0 \\ \vdots & \ddots & \cdots & \vdots & \vdots \\ \tau_{K-1} & \tau_{K-2} & \cdots & 0 & 0 \\ 0 & \tau_{K-1} & \cdots & \vdots & \vdots \\ \vdots & \ddots & \cdots & \tau_0 & 0 \\ 0 & 0 & \cdots & \tau_1 & \tau_0 \end{bmatrix} \in \mathbb{R}^{T \times T}, \quad (7.12)$$

so that the convolution becomes simply $\mathbf{h}_{t-T:t}^{(l+1)} = \mathbf{R}^\top \mathbf{h}_{t-T:t}^{(l)}$. This formulation extends naturally to the multivariate case by applying independent filters over each input channel.

In TCNs, the filtering operation is effectively decoupled into a backshift operator $\mathbf{T}$ that shifts the sequence forward in time and a separate linear transformation applied over the feature dimension. The matrix $\mathbf{R} \in \{0, 1\}^{T \times T}$, viewed in its binary form, captures the temporal dependency structure: $(\mathbf{R})_{pq} = \mathbf{R}[p, q] = 1$ if the input at time step $t - p$ influences the output at time step $t - q$.

This matrix plays a role analogous to the graph shift operator $\tilde{\mathbf{A}}$ in MPNNs, encoding a temporal topology similar to how $\tilde{\mathbf{A}}$ encodes spatial relationships. In fact, when using a filter of size K , the temporal topology matrix takes the form:

$$\mathbf{R} = \sum_{k=0}^{K-1} \mathbf{T}^k.$$

We can hence re-write the l -th layer of a TCN defined in Equation 2.32 as:

$$\mathbf{h}_{t-T:t}^{(l+1)} = \xi \left(\sum_{k=0}^K \text{diag}_k(\mathbf{R})^\top \mathbf{h}_{t-T:t}^{(l)} \mathbf{W}_k^{(l)} \right), \quad (7.13)$$

where each $\mathbf{W}_k^{(l)} \in \mathbb{R}^{d \times d}$ is a matrix of learnable weights, ξ is an element-wise activation function, and $\text{diag}_k(\mathbf{R})$ is the matrix obtained by zeroing all entries of $\mathbf{R}$ except those on its k -th lower diagonal. This formulation reveals the structural parallels between temporal and spatial message passing, both conditioned on specialized operators ($\mathbf{R}$ and $\tilde{\mathbf{A}}$, respectively) that encode the underlying topology. This parallel sets the stage for analyzing temporal over-squashing using tools similar to those developed for spatial propagation in GNNs.

7.3.2 Temporal over-squashing

In TCNs, each processing layer not only increases the model's expressivity but also expands its temporal receptive field, enabling the model to capture progressively longer-range dependencies in the input sequence. This hierarchical processing creates an information flow pattern that can be precisely characterized through the powers of the temporal topology matrix $\mathbf{R}$. Specifically, for a stack of L TCN layers (as defined in Equation 7.13) that share the same temporal topology matrix $\mathbf{R}$, the matrix power $\mathbf{R}^l$ represents the temporal receptive field at the l -th layer. In fact, entry $(\mathbf{R}^l)_{pq}$ counts the number of distinct paths by which information from time step $t - p$ can influence the representation at time step $t - q$ after l layers.

Analyzing how information flows along these paths is critical to understanding whether the model can effectively leverage all temporal context, or whether

such information vanishes. To formalize this, we study how the output representations are influenced by perturbations in the input at preceding time steps. The following theorem establishes a bound on this sensitivity, revealing how the temporal topology governs information flow and drawing analogies with the over-squashing behavior observed in the graph domain.

Theorem 7.1. *Consider a TCN with L_T successive TCN layers as in Equation 7.13, all with kernel size K , and assume that $\|\mathbf{W}_k^{(l)}\| \leq w$ for all $k < K$ and $l \leq L_T$, and that $|\sigma'| \leq c_\sigma$. For each nonnegative $p, q < T$, we have:*

$$\left\| \frac{\partial \mathbf{h}_{t-q}^{(L_T)}}{\partial \mathbf{h}_{t-p}^{(0)}} \right\| \leq \underbrace{(c_\sigma w)^{L_T}}_{\text{model}} \underbrace{(\mathbf{R}^{L_T})_{pq}}_{\text{temporal topology}}.$$

Proof. We fix the number of stacked TCN layers to be $l = L_T$ and prove the bound by induction on l .

Let $\tilde{\mathbf{h}}_t^{(1)}$ be the pre-activation output of the TCN layer, such that $\mathbf{h}_t^{(1)} = \sigma(\tilde{\mathbf{h}}_t^{(1)})$. Since $\mathbf{R}$ is lower-triangular and Toeplitz and has lower bandwidth $K - 1$, i.e., $(\mathbf{R})_{pq} = r_{p-q}$ for $0 \leq p - q < K$, for indices $p, q \in [0, T)$, we have

$$\frac{\partial \tilde{\mathbf{h}}_{t-q}^{(1)}}{\partial \mathbf{h}_{t-p}^{(0)}} = \begin{cases} r_{p-q} \mathbf{W}_{p-q}, & 0 \leq p - q < P, \\ 0, & \text{otherwise.} \end{cases}$$

Applying the chain rule through the pointwise non-linearity σ gives

$$\frac{\partial \mathbf{h}_{t-q}^{(1)}}{\partial \mathbf{h}_{t-p}^{(0)}} = \text{diag} \left(\sigma'(\tilde{\mathbf{h}}_{t-q}^{(1)}) \right) \frac{\partial \tilde{\mathbf{h}}_{t-q}^{(1)}}{\partial \mathbf{h}_{t-p}^{(0)}}.$$

Applying the sub-multiplicative property of the spectral norm together with the bounds $|\sigma'| \leq c_\sigma$ and $\|\mathbf{W}_k\| \leq w$ yields

$$\begin{aligned} \left\| \frac{\partial \mathbf{h}_{t-q}^{(1)}}{\partial \mathbf{h}_{t-p}^{(0)}} \right\| &\leq \left\| \text{diag} \left(\sigma'(\tilde{\mathbf{h}}_{t-q}^{(1)}) \right) \right\| \left\| \frac{\partial \tilde{\mathbf{h}}_{t-q}^{(1)}}{\partial \mathbf{h}_{t-p}^{(0)}} \right\| \\ &\leq c_\sigma w |r_{p-q}| \\ &= c_\sigma w (\mathbf{R})_{pq}, \end{aligned} \tag{7.1.1}$$

which proves the theorem for the base case $l = 1$.

Suppose the bound holds for $l - 1$:

$$\left\| \frac{\partial \mathbf{h}_{t-r}^{(l-1)}}{\partial \mathbf{h}_{t-p}^{(0)}} \right\| \leq (c_\sigma w)^{l-1} (\mathbf{R}^{l-1})_{pr} \quad \forall p, r. \tag{7.1.2}$$

For l layers, the chain rule gives

$$\left\| \frac{\partial \mathbf{h}_{t-q}^{(l)}}{\partial \mathbf{h}_{t-p}^{(0)}} \right\| = \left\| \sum_{r=0}^{T-1} \frac{\partial \mathbf{h}_{t-q}^{(l)}}{\partial \mathbf{h}_{t-r}^{(l-1)}} \frac{\partial \mathbf{h}_{t-r}^{(l-1)}}{\partial \mathbf{h}_{t-p}^{(0)}} \right\| \leq \sum_{k=0}^{T-1} \left\| \frac{\partial \mathbf{h}_{t-q}^{(l)}}{\partial \mathbf{h}_{t-r}^{(l-1)}} \right\| \left\| \frac{\partial \mathbf{h}_{t-r}^{(l-1)}}{\partial \mathbf{h}_{t-p}^{(0)}} \right\|,$$

where the first factor is the base case of a single layer (7.1.1), and the second one is the induction hypothesis (7.1.2). Substituting these bounds gives

$$\begin{aligned} \left\| \frac{\partial \mathbf{h}_{t-q}^{(l)}}{\partial \mathbf{h}_{t-p}^{(0)}} \right\| &\leq \sum_{r=0}^{T-1} ((c_\sigma \mathbf{w})(\mathbf{R})_{rq} (c_\sigma \mathbf{w})^{l-1} (\mathbf{R}^{l-1})_{pr}) \\ &= (c_\sigma \mathbf{w})^l \sum_{r=0}^{T-1} (\mathbf{R}^{l-1})_{pr} (\mathbf{R})_{rq} \\ &= (c_\sigma \mathbf{w})^l (\mathbf{R}^l)_{pq}, \end{aligned}$$

which proves the induction. $\square$

As in the spatial case shown in [Theorem C.1](#), the bound on sensitivity derived for TCNs decomposes into two components: one dependent on model parameters and another on the temporal topology, here encoded by $\mathbf{R}$. However, unlike spatial graphs – which may exhibit arbitrary and irregular connectivity – the temporal structure has a fixed, well-defined form that lends itself to deeper theoretical analysis. In particular, the lower-triangular Toeplitz structure of $\mathbf{R}$ ensures that its powers $\mathbf{R}^l$ retain the same structural form [[Krim et al., 2022](#)]. Moreover, as l increases, the number of distinct paths from earlier time steps to a given output grows significantly faster than those originating from more recent steps. This results in a natural *amplification* of long-range temporal dependencies in deeper layers, as formalized in the following proposition:

Proposition 7.2. *Let $\mathbf{R} \in \mathbb{R}^{T \times T}$ be a real, lower-triangular, Toeplitz band matrix with lower bandwidth $K - 1$, i.e., with $(\mathbf{R})_{pq} = r_{p-q}$ for $0 \leq p - q < K$, and $K \geq 2$, $r_1 \neq 0$, and $r_0 \neq 0$. Then for any $p > q$ we have $\left| \frac{(\mathbf{R}^l)_{q0}}{(\mathbf{R}^l)_{p0}} \right| \rightarrow 0$ as $l \rightarrow \infty$. In fact $\left| \frac{(\mathbf{R}^l)_{q0}}{(\mathbf{R}^l)_{p0}} \right| = \mathcal{O}(l^{-(p-q)})$. Informally, this means that the final token receives considerably more influence from tokens positioned earlier.*

Proof. Write $\mathbf{R} = r_0 \mathbf{I} + \mathbf{N}$, where $\mathbf{N} = \mathbf{R} - r_0 \mathbf{I}$ is strictly lower-triangular. Because $\mathbf{I}$ and $\mathbf{N}$ commute, the binomial expansion for commuting matrices gives

$$\mathbf{R}^l = r_0^l \mathbf{I} + \sum_{\tau=1}^{T-1} \binom{l}{\tau} r_0^{l-\tau} \mathbf{N}^\tau, \quad l \in \mathbb{N},$$

the sum truncating at $T - 1$ since $\mathbf{N}^T = \mathbf{0}$. For a fixed row index $p \geq 1$, the strictly lower-triangular structure implies $(\mathbf{N}^\tau)_{p0} = 0$ whenever $\tau > p$, thus

$$(\mathbf{R}^l)_{p0} = \sum_{\tau=1}^p \binom{l}{\tau} r_0^{l-\tau} (\mathbf{N}^\tau)_{p0}.$$

With τ fixed, $\binom{l}{\tau} \sim l^\tau / \tau!$ as $l \rightarrow \infty$ since $\lim_{l \rightarrow \infty} \frac{\binom{l}{\tau}}{l^\tau} = \frac{1}{\tau!}$, so with some abuse of notation

$$(\mathbf{R}^l)_{p0} \sim \sum_{\tau=1}^p \frac{l^\tau}{\tau!} r_0^{l-\tau} (\mathbf{N}^\tau)_{p0} = r_0^l \sum_{\tau=1}^p \frac{l^\tau}{\tau! r_0^\tau} (\mathbf{N}^\tau)_{p0}, \quad l \rightarrow \infty.$$

Note that it is the product of r_0^l and a degree p polynomial in l , therefore in order to see the behaviour as $l \rightarrow \infty$ it suffices to study the leading term $\frac{l^p}{p! r_0^p} (\mathbf{N}^p)_{p0}$ and the factor of r_0^l . Additionally, due to the strictly lower triangular structure of $\mathbf{N}$, we have $(\mathbf{N}^p)_{p0} = r_1^p$. This follows from the fact that, in a directed acyclic graph as the one described by $\mathbf{N}$, there exists a unique directed path of length p from the p -th node to the sink node 0. Each edge along this path contributes a multiplicative factor of r_1 , resulting in a total weight of r_1^p for the path. Applying the same reasoning with p replaced by $q < p$, we can study the ratio

$$\left| \frac{(\mathbf{R}^l)_{q0}}{(\mathbf{R}^l)_{p0}} \right| \sim \frac{\frac{l^q}{q!} r_0^{l-q} r_1^q}{\frac{l^p}{p!} r_0^{l-p} r_1^p} = \frac{1}{l^{p-q}} \frac{p! r_0^{p-q}}{q! r_1^{p-q}} = \mathcal{O} \left(\frac{1}{l^{p-q}} \right) \xrightarrow{l \rightarrow \infty} 0,$$

which proves the proposition. $\square$

Proposition 7.2 reveals a critical insight: **causal convolutions progressively diminish sensitivity to recent information while amplifying the influence of temporally distant inputs**. This creates a form of temporal over-squashing that *inverts* the pattern observed in MPNNs, where distant nodes suffer from reduced influence [Alon and Yahav, 2021]. When a model's receptive field

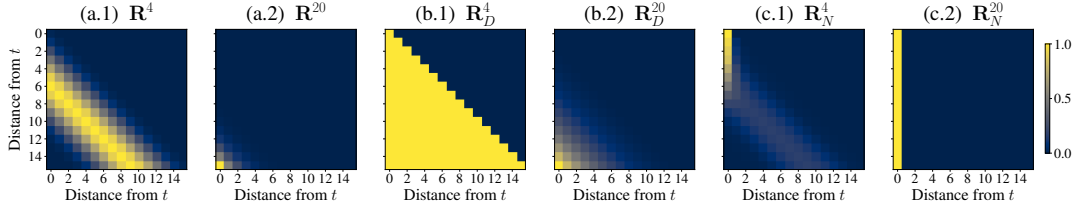


Figure 7.1. Evolution of temporal receptive field for standard ($\mathbf{R}$), dilated ($\mathbf{R}_D$), and row-normalized ($\mathbf{R}_N$) convolutions. For $\mathbf{R}$ and $\mathbf{R}_D$, we observe a shift towards the initial time step, while for $\mathbf{R}_N$ we observe a shift to a uniform distribution on the first column. Values normalized matrix-wise in $[0, 1]$.

exceeds the sequence length, i.e., $(K - 1)L_T \gg T$, the earliest time step exerts disproportionate influence on the final one compared to any intermediate time step, mirroring the recently investigated *attention sink* effect in Transformers [Xiao et al., 2024; Barbero et al., 2024a], as shown in Figure 7.1a.2. This behavior directly undermines the locality bias that causal convolutions are designed to enforce, particularly in time series applications, where recent observations typically carry greater relevance. To achieve a more balanced receptive field that preserves local information while incorporating broader context, we can act by modifying $\mathbf{R}$, effectively implementing *temporal graph rewiring* analogous to techniques used to address over-squashing in MPNNs [Topping et al., 2022; Di Giovanni et al., 2023a].

7.3.3 Temporal graph rewiring

The bottleneck highlighted in Proposition 7.2 stems from the structured processing enforced by causal convolutions, which exploit the inherent locality of temporal data. A natural way to mitigate this phenomenon is to relax the underlying temporal topology, either by adopting alternative designs or by learning a different temporal connectivity matrix. However, by significantly altering the original connections, we might eliminate the inductive bias that recent observations should prevail over those further back in time. The rewiring strategies proposed in this section should therefore be viewed as controlled modifications, aiming to alleviate over-squashing in favor of retaining, rather than discarding, an appropriate notion of temporal locality. Notably, Proposition 7.2 outlines that the influence from recent time steps progressively vanishes when the temporal topology (1) remains fixed across layers and (2) maintains a lower-triangular Toeplitz structure. We propose two rewiring approaches targeting these assumptions separately.

Our first approach addresses the fixed topology assumption by employing different matrices $\mathbf{R}^{(l)}$ at each layer (each maintaining a lower-triangular Toeplitz structure). This modifies the Jacobian bound in [Theorem 7.1](#) to depend on $\prod_{l=1}^{L_T} \mathbf{R}^{(l)}$ rather than $\mathbf{R}^{L_T}$. Dilated convolutions naturally implement this approach, applying filters with progressively increasing gaps $\delta^{(l)}$ (dilation rates) between elements [[Oord et al., 2016](#)]. These convolutions produce matrices $\mathbf{R}_D^{(l)}$ with nonzero entries $(\mathbf{R}_D^{(l)})_{pq} = r_{p-q}$ only when $p - q = k\delta^{(l)}$ for $k \in [0, K)$. When $\delta^{(l)} = K^{l-1}$, the receptive field expands exponentially while ensuring $(\prod_l \mathbf{R}_D^{(l)})_{k0} = 1$ for each $k \leq K^l$, hence distributing influence equally across all time steps in the receptive field ([Figure 7.1b.1](#)). Thus, besides efficiency, dilated convolutions have the advantage of preserving local information better than standard convolutions. However, this strategy results in the trivial identity $\mathbf{R}_D^{(l)} = \mathbf{I}_T$ for $l > \log_K T$, with deeper architectures required to reset the dilation rate every m layers, i.e., $\delta^{(l)} = K^{(l-1) \bmod m}$. While effective in practice, these resets reintroduce over-squashing patterns, as shown in [Figure 7.1b.2](#).

Our second approach targets the Toeplitz assumption by row-normalizing $\mathbf{R}$ to create $\mathbf{R}_N = \text{diag}(\mathbf{R}\mathbf{1})^{-1}\mathbf{R}$, where each entry $(\mathbf{R}_N)_{pq}$ is normalized by the number of edges from time step $t - p$ for any q . This normalization maintains stronger influence from recent time steps while expanding the receptive field, with $(\mathbf{R}_N^l)_{p0}$ converging to 1 for all $p \in [0, T)$ as $l \rightarrow \infty$. We show this graphically [Figure 7.1c.1–2](#) and prove the claim in the following proposition.

Proposition 7.3. *Let $\mathbf{R} \in \mathbb{R}_{\geq 0}^{T \times T}$ be a positive, real, lower-triangular Toeplitz band matrix with lower bandwidth $K - 1$, i.e., with $(\mathbf{R})_{pq} = r_{p-q}$ for $0 \leq p - q < K$, and let $K \geq 2$, $r_1 \neq 0$, and $r_0 \neq 0$. We define $\mathbf{R}_N = \text{diag}(\mathbf{R}\mathbf{1})^{-1}\mathbf{R}$ to be the row-normalized matrix $\mathbf{R}$ ($\mathbf{1}$ is the all-ones vector). Then*

$$\lim_{l \rightarrow \infty} \mathbf{R}_N^l = \mathbf{1}e_0^\top,$$

i.e. every row of $\mathbf{R}_N^l$ converges to $e_0^\top = (1, 0, \dots, 0)$.

Proof. Since $\mathbf{R}_N$ is a stochastic matrix (every row sum to 1 by construction), we can interpret $\mathbf{R}_N$ as the transition matrix of a Markov chain on the finite state space $\{0, 1, \dots, T - 1\}$. Because $\mathbf{R}$ is lower-triangular, we have that state 0 is *absorbing*, i.e. $(\mathbf{R}_N)_{00} = r_0/r_0 = 1$, hence once the chain reaches 0 it never leaves. Moreover, since $r_1 \neq 0$, for every $p > 0$ and $q = p - 1$ there

is a positive probability of moving one step closer to 0, i.e.,

$$(\mathbf{R}_N)_{pq} = \frac{r_1}{\sum_{k=0}^{\min\{K-1,p\}} r_k} > 0,$$

hence all states $1, 2, \dots, T-1$ are *transient*. As shown in Theorem 11.3 of Grinstead and Snell [2006], for a finite absorbing Markov chain with a single absorbing state $\{0\}$, every state eventually reaches the absorbing state 0 with probability 1. Thus,

$$\lim_{l \rightarrow \infty} \mathbf{R}_N^l = \mathbf{1} \mathbf{e}_0^\top,$$

where the p -th row of the limit contains the absorption probabilities starting from state p (i.e., time $t-p$) and are all equal to 1 for state 0 and to 0 for the others. $\square$

While this normalization strategy violates the Toeplitz structure, it remains computationally efficient: it can still be implemented as a temporal convolution by simply scaling the input at time step $t-p$ by $\min(p+1, K)^{-1}$.

Importantly, this normalization primarily enhances sensitivity at the final time step, making it particularly well-suited for the architectural framework considered (see Equation 7.4). In the following subsection, we empirically validate the theoretical claims presented above and demonstrate how the proposed rewiring schemes effectively mitigate the temporal over-squashing phenomenon.

7.3.4 Empirical validation

To validate our claims and the effects of our proposed temporal convolution modifications, we design two synthetic tasks specifically aimed at testing how well different architectures preserve temporal information after processing:

COPYFIRST We generate sequences $\mathbf{x}_{t-T:t}$ consisting of $T = 16$ time steps with values sampled uniformly from the $[0, 1]$ interval. The objective is to predict the first-observed element in the sequence, i.e., $\mathbf{x}_{t-T+1}$. We generate 20,000 sequences for training, 320 for validation, and 500 for testing. Since a constant prediction of 0.5 yields an MSE of approximately 0.083, we consider the task solved when the test MSE is below 0.001 and use success rate across multiple runs as primary evaluation metric.

COPYLAST This task mirrors the COPYFIRST setup, but the target is instead the

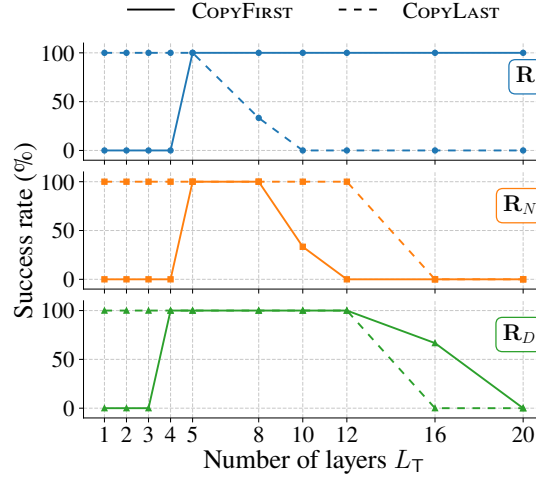


Figure 7.2. Success rate (%) on the tasks of copying the first or last observed value across different temporal topologies and number of layers L_T .

last element in the sequence, i.e., x_t . This shift in objective allows us to test whether models can retain recent inputs after processing.

We compare three TCNs variants, each composed of L_T temporal layers but differing in the design of the convolution operation:

R A standard causal convolution with receptive field size $K = 4$.

R_N A row-normalized version of the above, defined as $\mathbf{R}_N = \text{diag}(\mathbf{R}\mathbf{1})^{-1}\mathbf{R}$, also with $K = 4$.

R_D A dilated convolution with $K = 2$ and dilation rate $\delta^{(l)} = K^{(l-1) \bmod m}$, using a reset cap of $m = 4$ to control dilation growth.

We vary the number of temporal layers L_T from 1 to 20 and report results in Figure 7.2, which shows the success rate for both tasks as a function of model depth and temporal topology. For small L_T , performance on COPYFIRST remains poor, as the target lies outside the receptive field, due to under-reaching (see Section 2.3.1). In contrast, models generally succeed on COPYLAST at shallow depths, reflecting the natural bias of causal convolutions toward recent inputs. However, as the number of layers increases, standard causal convolutions begin to fail on COPYLAST, particularly beyond $L_T > 5$. This failure is ascribable to the *sink effect*, where the repeated causal convolutions over-emphasizes older time steps and attenuates recent inputs, a direct manifestation of temporal over-squashing.

Conversely, performance on COPYFIRST improves with depth, in particular for the standard causal convolution, reinforcing the idea that deeper models favor inputs distant in time. While this behavior may be beneficial in some tasks, it reveals a structural bias that can hinder learning in general-purpose forecasting scenarios. Normalization ($\mathbf{R}_N$) and dilated convolutions ($\mathbf{R}_D$) partially mitigate this effect, especially in preserving sensitivity to recent inputs. Overall, these findings empirically confirm the theoretical observations of the previous section.

7.4 Over-squashing in MPTCNs

Having analyzed over-squashing in the temporal domain, and relying on previous results on the spatial domain, we now integrate these insights to investigate information flow in MPTCNs, in which temporal and spatial processing are interleaved across layers. For our analysis, we consider the same class of message-passing functions employed by Di Giovanni et al. [2024] (defined in Equation 7.2) for which Theorem C.1 holds.

Following the space-time factorization established in Equation 7.9, we observe that within a single layer, the sensitivity is simply the product of the spatial bound from Theorem C.1 and the temporal bound from Theorem 7.1. For TTS architectures with $L = 1$, this yields:

$$\left\| \nabla_j \mathbf{h}_t^{i(L)} \right\| \leq \underbrace{(c_\xi \theta_m)^{L_S} (\mathbf{S}^{L_S})_{ji}}_{\text{space}} \underbrace{(c_\sigma \mathbf{w})^{L_T} (\mathbf{R}^{L_T})_{p0}}_{\text{time}}, \quad (7.14)$$

This bound directly measures how input features influence the final representations used by the readout layer. However, this result only addresses the case where $L = 1$ (the TTS approach). The question remains: how does information propagate when temporal and spatial processing alternate multiple times ($L > 1$) as in the T&S approach? To answer this question, we derive the following theorem that characterizes sensitivity across MPTCNs with any number of alternating processing blocks.

Theorem 7.4. *Consider an MPTCN with L STMP layers, each consisting of L_T temporal (TMP) and L_S spatial (MP) layers as defined in Equations 2.39 to 2.40. Assume that each TMP layer satisfies the conditions of Theorem 7.1, and each MP layer satisfies the assumptions in Theorem C.1. Then, for any $i, j \in \mathcal{V}$ and $p, q \in [0, T)$, the*

following holds:

$$\left\| \frac{\partial \mathbf{h}_{t-q}^{i(L)}}{\partial \mathbf{h}_{t-p}^{j(0)}} \right\| \leq \underbrace{(c_\xi \theta_m)^{LL_S} (c_\sigma \mathbf{w})^{LL_\tau}}_{\text{model}} \underbrace{(\mathbf{S}^{LL_S})_{ji} (\mathbf{R}^{LL_\tau})_{pq}}_{\text{spatiotemporal topology}}.$$

Proof. We fix $i, j \in \mathcal{V}$, $p, q \in [0, T)$, and the number of STMP layers to be $l = L$ and prove the bound by induction on l .

For the base case $l = 1$, one STMP layer first applies the L_τ TMP layers node-wise, and then the L_S MP layers time-wise. Thus, for the chain rule, we have

$$\frac{\partial \mathbf{h}_{t-q}^{i(1)}}{\partial \mathbf{h}_{t-p}^{j(0)}} = \sum_{w \in \mathcal{V}} \sum_{\tau=0}^{T-1} \frac{\partial \mathbf{h}_{t-q}^{i(1)}}{\partial \mathbf{z}_{t-\tau}^{w(L_\tau)}} \frac{\partial \mathbf{z}_{t-\tau}^{w(L_\tau)}}{\partial \mathbf{h}_{t-p}^{j(0)}}. \quad (7.4.1)$$

Every MP layer processes each time step separately, hence $\partial \mathbf{h}_{t-q}^{i(1)} / \partial \mathbf{z}_{t-\tau}^{w(L_\tau)} = \mathbf{0}$ unless $\tau = q$. Similarly, every TMP layer processes each node separately, hence $\partial \mathbf{z}_{t-q}^{w(L_\tau)} / \partial \mathbf{h}_{t-p}^{j(0)} = \mathbf{0}$ unless $w = j$. Both sums in Equation 7.4.1 therefore collapse, giving

$$\frac{\partial \mathbf{h}_{t-q}^{i(1)}}{\partial \mathbf{h}_{t-p}^{j(0)}} = \frac{\partial \mathbf{h}_{t-q}^{i(1)}}{\partial \mathbf{z}_{t-q}^{j(L_\tau)}} \frac{\partial \mathbf{z}_{t-q}^{j(L_\tau)}}{\partial \mathbf{h}_{t-p}^{j(0)}}.$$

Using sub-multiplicativity of the spectral norm, and considering the bounds from Theorem C.1 for MP layers and Theorem 7.1 for TCN layers, we have

$$\begin{aligned} \left\| \frac{\partial \mathbf{h}_{t-q}^{i(1)}}{\partial \mathbf{h}_{t-p}^{j(0)}} \right\| &\leq \left\| \frac{\partial \mathbf{h}_{t-q}^{i(1)}}{\partial \mathbf{z}_{t-q}^{j(L_\tau)}} \right\| \left\| \frac{\partial \mathbf{z}_{t-q}^{j(L_\tau)}}{\partial \mathbf{h}_{t-p}^{j(0)}} \right\| \\ &\leq (c_\xi \theta_m)^{L_S} (\mathbf{S}^{L_S})_{ji} (c_\sigma \mathbf{w})^{L_\tau} (\mathbf{R}^{L_\tau})_{pq}, \end{aligned} \quad (7.4.2)$$

which proves the bound for $l = 1$.

We assume the bound holds for $l - 1$ stacked STMP blocks, for all $j, w \in \mathcal{V}$ and $p, \tau \in [0, T)$:

$$\left\| \frac{\partial \mathbf{h}_{t-\tau}^{w(l-1)}}{\partial \mathbf{h}_{t-p}^{j(0)}} \right\| \leq (c_\xi \theta_m)^{(l-1)L_S} (c_\sigma \mathbf{w})^{(l-1)L_\tau} (\mathbf{S}^{(l-1)L_S})_{jw} (\mathbf{R}^{(l-1)L_\tau})_{pk}. \quad (7.4.3)$$

For the l -th layer, we apply the chain rule, the triangle inequality, and sub-multiplicativity:

$$\left\| \frac{\partial \mathbf{h}_{t-q}^{i(l)}}{\partial \mathbf{h}_{t-p}^{j(0)}} \right\| \leq \sum_{w \in \mathcal{V}} \sum_{\tau=0}^{T-1} \left\| \frac{\partial \mathbf{h}_{t-q}^{i(l)}}{\partial \mathbf{h}_{t-\tau}^{w(l-1)}} \right\| \left\| \frac{\partial \mathbf{h}_{t-\tau}^{w(l-1)}}{\partial \mathbf{h}_{t-p}^{j(0)}} \right\|.$$

The first term is the base case of a single outer layer (7.4.2), and the right term is our induction hypothesis (7.4.3), which combined with the triangular inequality proves the induction:

$$\begin{aligned}
\left\| \frac{\partial \mathbf{h}_{t-q}^{i(l)}}{\partial \mathbf{h}_{t-p}^{j(0)}} \right\| &\leq \sum_{w \in \mathcal{V}} \sum_{\tau=0}^{T-1} \left((c_\xi \theta_m)^{L_S} (c_\sigma \mathbf{w})^{L_\tau} (\mathbf{S}^{L_S})_{wi} (\mathbf{R}^{L_\tau})_{\tau q} \right. \\
&\quad \left. (c_\xi \theta_m)^{(l-1)L_S} (c_\sigma \mathbf{w})^{(l-1)L_\tau} (\mathbf{S}^{(l-1)L_S})_{jw} (\mathbf{R}^{(l-1)L_\tau})_{p\tau} \right) \\
&= (c_\xi \theta_m)^{lL_S} (c_\sigma \mathbf{w})^{lL_\tau} \sum_{w \in \mathcal{V}} \sum_{\tau=0}^{T-1} (\mathbf{S}^{L_S})_{wi} (\mathbf{S}^{(l-1)L_S})_{jw} (\mathbf{R}^{L_\tau})_{\tau q} (\mathbf{R}^{(l-1)L_\tau})_{p\tau} \\
&= (c_\xi \theta_m)^{lL_S} (c_\sigma \mathbf{w})^{lL_\tau} (\mathbf{S}^{(l-1)L_S} \mathbf{S}^{L_S})_{ji} (\mathbf{R}^{(l-1)L_\tau} \mathbf{R}^{L_\tau})_{pq} \\
&= (c_\xi \theta_m)^{lL_S} (c_\sigma \mathbf{w})^{lL_\tau} (\mathbf{S}^{lL_S})_{ji} (\mathbf{R}^{lL_\tau})_{pq}.
\end{aligned}$$

Induction on l establishes the inequality for every $l \leq L$; in particular, for $l = L$, which is exactly the claimed bound. $\square$

This result bounds the influence of input features $\mathbf{h}_{t-p}^{j(0)}$ on output representation $\mathbf{h}_{t-q}^{i(L)}$, revealing a clean separation between model parameters and topological factors, as well as between spatial and temporal components. Two key implications emerge from this factorization.

First, the bound's multiplicative structure across space and time dimensions persists regardless of how many STMP layers are used. This means that redistributing the computational budget among outer layers L and inner layers L_τ and L_S does not alter the bound's characteristics. Therefore, **from the perspective of information propagation, TTS architectures ($L = 1$) are not limited compared to T&S architectures ($L > 1$)**. While this does not guarantee equivalence in expressivity or optimization dynamics, it provides a principled justification for adopting more computationally efficient TTS designs without compromising how information flows.

Second, the theorem reveals that **spatiotemporal over-squashing in MPTCNs arises from the combined effects of spatial and temporal over-squashing**. This is evident in the spatiotemporal topology component, where both the spatial distance between nodes j and i and the temporal distance between time steps $t - p$ and $t - q$ contribute equally to potential bottlenecks. This insight carries significant practical implications: addressing over-squashing effectively requires

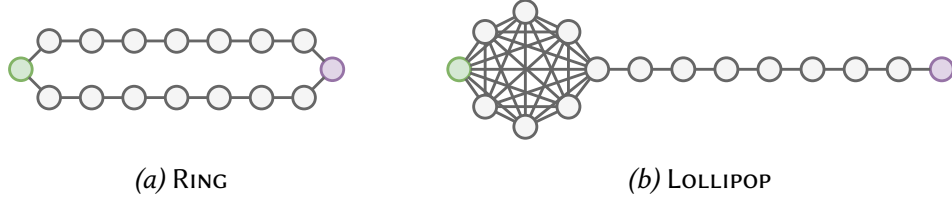


Figure 7.3. RING and LOLLIPOP graphs used in the synthetic experiments. We highlight in green the target node $\textcolor{green}{\bullet}$, and show an example of a source node $\textcolor{purple}{\bullet}$ when the spatial distance d is equal to the graph diameter.

targeting both dimensions simultaneously. Improving only one component, through either spatial or temporal graph rewiring alone, will prove insufficient if bottlenecks persist in the other dimension. In the following subsection, we empirically assess these claims in both a controlled environment and real-world benchmarks.

7.4.1 Empirical validation

Controlled environment

For this empirical analysis, we introduce a new synthetic benchmark specifically designed to test both spatial and temporal over-squashing in STGNNs:

ROCKETMAN We construct a graph with a fixed topology and $N = 16$ nodes. Each node is associated with a sequence of $T = 9$ time steps, where values are sampled uniformly from the $[0, 1]$ interval. The prediction task is designed to test long-range spatiotemporal propagation. For a given target node i , the model must predict the average of the values at time step $t - p$ for all nodes located exactly d hops away from i . More formally, let $\mathcal{N}_d(i)$ denote the set of nodes at shortest-path distance d from i , with $\mathcal{N}_0(i) = \{i\}$. The label for node i is defined as:

$$\mathbf{y}_t^i = \frac{1}{|\mathcal{N}_d(i)|} \sum_{j \in \mathcal{N}_d(i)} \mathbf{x}_{t-p}^j,$$

while labels for all other nodes are masked out. The task thus requires information to travel both d hops in space and p steps in time to reach the decoder at node i . As in previous synthetic benchmarks, we generate 20,000 training samples, 320 for validation, and 500 for testing. To evaluate the effect of spatial over-squashing, we use two graph topologies with

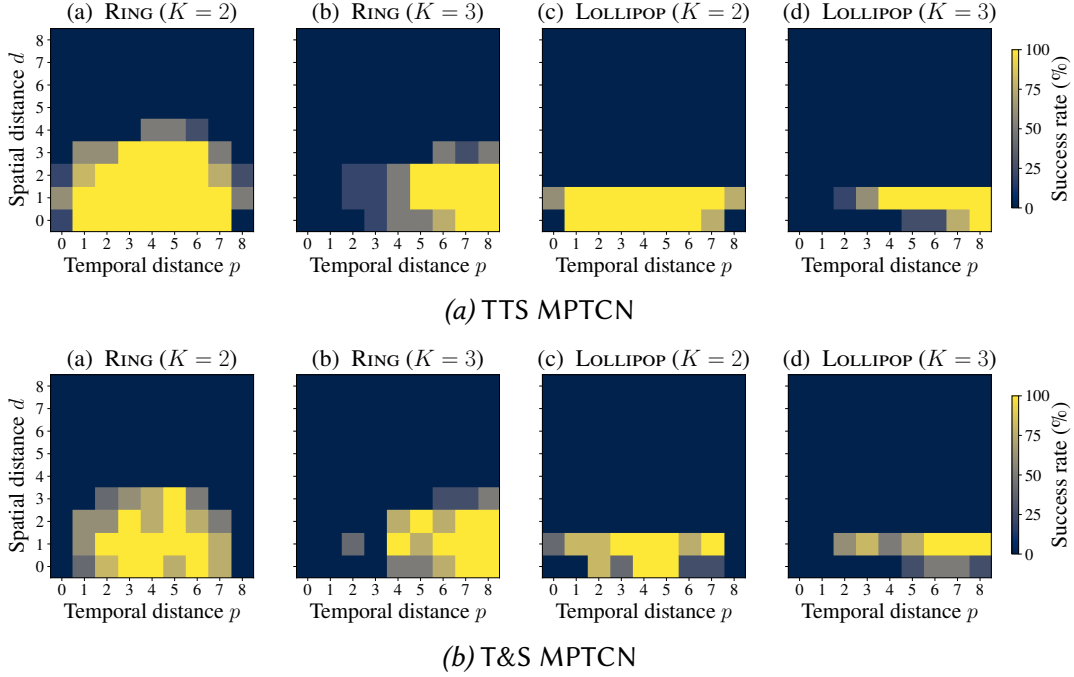


Figure 7.4. Success rate (%) of MPTCNs on the ROCKETMAN dataset. The tasks vary for the type of graph used (RING or LOLLIPOP) and kernel size K (2 or 3). The plot axes show the source neighbors distance d and the temporal distance p .

different properties: the RING and the LOLLIPOP graphs, both visualized in Figure 7.3.

We evaluate both TTS and T&S variants of MPTCNs, equipping them with just enough temporal and spatial layers to span the entire input domain (i.e., full spatial and temporal coverage). The success rate across multiple random seeds is reported in Figure 7.4, with separate plots for TTS (7.4a) and T&S (7.4b) models.

The results show several trends consistent with our theory. First, the task is consistently more difficult on the LOLLIPOP graph than on the RING, despite both having similar size. This aligns with prior findings on spatial over-squashing, where LOLLIPOP graphs are known to promote over-squashing due to their highly unbalanced topology [Di Giovanni et al., 2024]. Second, the impact of temporal over-squashing becomes apparent as we vary the convolutional filter size K . For small temporal distances (i.e., low p), performance deteriorates with increasing K . This confirms our theoretical claim that deeper stacks of convolutions favor early inputs, making it harder to recover information from more recent time

Table 7.1. Forecasting error (MAE) of MPTCNs with fixed budget in real-world benchmarks.

Models		L	METR-LA	PEMS-BAY	EngRAD
MPTCN	$\mathbf{R}$	6	$3.19_{\pm 0.02}$	$1.66_{\pm 0.00}$	$44.43_{\pm 0.41}$
		3	$3.19_{\pm 0.01}$	$1.65_{\pm 0.01}$	$43.83_{\pm 0.03}$
		1	$3.14_{\pm 0.02}$	$1.63_{\pm 0.01}$	$44.47_{\pm 0.42}$
	$\mathbf{R}_N$	6	$3.17_{\pm 0.02}$	$1.65_{\pm 0.01}$	$41.82_{\pm 0.38}$
		3	$3.17_{\pm 0.01}$	$1.65_{\pm 0.00}$	$41.78_{\pm 0.09}$
		1	$3.16_{\pm 0.01}$	$1.65_{\pm 0.01}$	$40.38_{\pm 0.08}$
Graph WaveNet (orig.)			$3.02_{\pm 0.02}$	$1.55_{\pm 0.01}$	$40.50_{\pm 0.27}$
Graph WaveNet (TTS)			$3.00_{\pm 0.01}$	$1.57_{\pm 0.00}$	$40.64_{\pm 0.29}$

steps. Interestingly, this effect is visible in both TTS and T&S architectures, although it is more pronounced in the latter.

Forecasting benchmarks

To verify that our theoretical insights extend to practical applications, we evaluated various MPTCNs of fixed spatial and temporal budgets LL_S and LL_T , respectively, on three standard spatiotemporal forecasting benchmarks, namely METR-LA, PEMS-BAY, and EngRAD (see Section 2.6). We follow the same experimental setup of Section 5.6 and report forecasting error measured by MAE in Table 7.1.

The results offer several insights that support our theoretical analysis. First, T&S and TTS approaches perform comparably on average, with the more efficient TTS models outperforming in the majority of cases. This empirical finding supports our theoretical conclusion that the TTS design is not inherently limited from an information propagation perspective, although additional factors may also influence comparative performance. Second, in the EngRAD dataset, which requires larger filter sizes to cover the input sequence, the row-normalization approach consistently improves performance. This hints that temporal over-squashing affects standard convolutional models in practical scenarios where longer temporal contexts are needed, as our theory suggests. Finally, we compare Graph WaveNet [Wu et al., 2019] – a widely-used and more complex T&S architecture – against its TTS counterpart. Results show that our findings remain valid even when more sophisticated architectural components

are involved.

7.5 Related work

The issue of over-squashing in GNNs was first highlighted by Alon and Yahav [2021], who showed that GNNs struggle to capture long-range dependencies in graphs with structural bottlenecks. Building on this, Topping et al. [2022] introduced a sensitivity-based framework to identify over-squashing, extended in following works [Di Giovanni et al., 2023a; Di Giovanni et al., 2024]. Two main strategies have emerged to alleviate this problem: graph re-wiring to enhance connectivity [Gasteiger et al., 2019; Topping et al., 2022; Karhadkar et al., 2023; Barbero et al., 2024b], and architectural modifications to stabilize gradients [Chen et al., 2020a; Tortorella and Micheli, 2022; Gravina et al., 2023; Gravina et al., 2024a; Gravina et al., 2025]. Models such as graph transformers [Ying et al., 2021; Müller et al., 2024] or those based on virtual nodes [Southern et al., 2025] use global mechanisms that enable modeling long-range interactions directly, bypassing the topological bottleneck. Notably, these efforts largely overlook the role of time-evolving node features.

Similar challenges in modeling long-range dependencies have been studied in temporal architectures like RNNs, where vanishing and exploding gradients hinder effective learning [Bengio et al., 1994; Pascanu et al., 2013]. Solutions include enforcing orthogonality [Vorontsov et al., 2017] or antisymmetry [Chang et al., 2021] in weight matrices, as well as designing specialized stable architectures [Rusch and Mishra, 2021]. Recently, the attention sink effect [Xiao et al., 2024] has revealed a bias in Transformers towards early tokens [Wu et al., 2025], with Barbero et al. [2024a] linking this phenomenon to over-squashing in language models. None of these approaches, however, consider the compounding challenge proper of graph-based sequence processing operators.

In the spatiotemporal domain, Gao and Ribeiro [2022] analyzed the expressive power of STGNNs, offering theoretical insights into their ability to distinguish non-isomorphic graphs in both T&S and TTS paradigms. Interestingly, they proved that TTS architectures are more expressive than T&S counterparts. Yet, to our knowledge, no prior work has directly tackled the problem of spatiotemporal over-squashing in STGNNs, where the interaction between spatial and temporal dimensions introduces unique challenges for information propagation.

7.6 Discussion and future directions

In this chapter, we have formally characterized spatiotemporal over-squashing in STGNNs, demonstrating its distinctions from the static case. Our analysis reveals that convolutional STGNNs counterintuitively favor information propagation from temporally distant points, offering key insights into their behavior. Despite their structural differences, we proved that both T&S and TTS paradigms equally suffer from this phenomenon, providing theoretical justification for computationally efficient implementations. Experiments on synthetic and real-world datasets confirm our theoretical framework’s practical relevance. The insights from this study directly impact the design of effective spatiotemporal architectures. By bridging theory and practice, we contribute to a deeper understanding of STGNNs and provide principled guidance for their implementation.

Our focus on factorized convolutional approaches aligns with established GNN research while extending existing theoretical results to the spatiotemporal domain. While this choice reflects popular design patterns, our analysis does not directly include recurrent architectures or models with joint space-time filters, which present opportunities for future work. Finally, our sensitivity bounds are worst-case and therefore potentially conservative; deriving tighter, data-dependent estimates and validating them empirically remain open directions.

Chapter 8

Conclusion

8.1 Summary of contributions

This thesis advanced the field of graph deep learning for irregular spatiotemporal data by developing novel methodologies and theoretical insights, with a focus on the practically relevant tasks of imputation and prediction. We summarize the main contributions below.

In the first part of the thesis, we focused on missing-data imputation in spatiotemporal settings by leveraging the relational inductive biases offered by graph neural networks. We introduced the GRIN [Cini, Marisca, et al., 2022] architecture (Chapter 3), a bidirectional message-passing recurrent neural network that integrates spatial dependencies via graph convolutions within an RNN. Thanks to its inductive biases, GRIN promotes imputations that are consistent with the spatiotemporal context and yields smoothed reconstructions as a byproduct, thereby also serving as a filtering model. We extended this approach to regimes with high data sparsity in Chapter 4, where we proposed the SPIN model [Marisca et al., 2022]. SPIN employs a novel sparse spatiotemporal attention mechanism, avoiding autoregressive processing and directly gathering available observations across both dimensions. The model demonstrates strong performance in settings with extremely sparse observations, where autoregressive models fail.

In the second part of the thesis, we addressed the task of forecasting future values from irregularly sampled spatiotemporal inputs. We began by considering the simpler case with only spatial irregularity while assuming temporally complete

sequences ([Chapter 5](#)). We formalized the design space of spatiotemporal graph neural networks for forecasting, contrasting different processing paradigms – including the time-and-space (T&S) and time-then-space (TTS) approaches [[Cini, Marisca, et al., 2025](#)]. We introduced *global-local* forecasting architectures [[Cini, Marisca, et al., 2023b](#)] that combine node-specific embeddings with shared spatiotemporal message-passing layers, balancing specialization and generalization to improve prediction accuracy. To address the scalability challenge, we developed the SGP model [[Cini, Marisca, et al., 2023a](#)], which decouples encoding and decoding via randomized spatiotemporal embeddings using echo state networks and precomputed graph propagation. SGP achieves competitive forecasting performance while drastically reducing training and inference cost.

In [Chapter 6](#), we addressed the more general setting where data are incomplete and irregular across both space and time, and developed architectures that directly predict future values from partial inputs. We introduced a framework based on spatiotemporal hierarchical downsampling, HD-TTS [[Marisca et al., 2024](#)], that extracts representations at multiple spatiotemporal resolutions via interleaved downsampling and message passing over both space and time dimensions. By adaptively combining representations across scales, the model learns to route attention away from regions with missing observations while preserving local information where available. We showed that HD-TTS not only achieves state-of-the-art forecasting accuracy under varying missing data patterns but also enables interpretable analysis of model behavior through the attention scores.

A key theoretical contribution of this thesis is the formalization and study of spatiotemporal over-squashing [[Marisca et al., 2025](#)] – a phenomenon where information struggles to propagate between two points due to graph and temporal structure, limiting the power of STGNN models ([Chapter 7](#)). We extended the concept of over-squashing from static graphs to the spatiotemporal domain, showing that both temporal and spatial topologies can restrict information propagation. We demonstrated analytically and empirically that in convolutional STGNNs recent information is unexpectedly attenuated in favor of information distant in time. Moreover, we observe that this behavior equally affects TTS and T&S models, supporting the adoption of computationally efficient implementations with theoretical results. We provided design guidelines to mitigate this form of over-squashing via temporal graph rewiring, offering new directions for robust long-range propagation in (spatio)temporal networks.

Together, these contributions establish a general-purpose, theoretically grounded,

and empirically validated framework for graph-based modeling of irregular spatiotemporal data. Each component is designed to handle key real-world challenges such as irregular sampling, sparsity, and scalability, contributing to the development of more effective and efficient learning systems.

8.2 Future directions

Several promising research directions remain open. Most models in this thesis assume the spatial graph structure to be fixed or slowly evolving. While this assumption is valid in many real-world scenarios, it may become limiting in applications where the topology changes rapidly – such as social or communication networks. Learning dynamic or task-specific topologies end-to-end remains an open challenge, particularly under irregular or sparse observations. Recent advances in both discrete and continuous graph structure learning offer promising tools, and active research is ongoing in this area [Dong et al., 2019; Franceschi et al., 2019; Shang et al., 2021; Cini et al., 2023c; Manenti et al., 2025]. In addition, continuous-time modeling of dynamic graphs [Rubanova et al., 2019; Chamberlain et al., 2021; Gravina et al., 2024a; Gravina et al., 2024b] is well-aligned with the irregular settings addressed in this thesis and offers an interesting extension for future work.

The global-local STGNNs introduced in this thesis demonstrate strong performance by balancing generalization and specialization. However, extending these architectures to dynamic or inductive scenarios – where new nodes, edges, or subgraphs may appear at test time – remains a challenging problem. How to effectively specialize and adapt learned representations in such evolving environments is still an open question. Promising directions include local fine-tuning, meta-learning approaches, and embedding space regularization [Yin et al., 2022; Butera et al., 2025].

Despite growing empirical progress, the theoretical foundations of STGNNs – particularly under irregular sampling – still remain underdeveloped. While this thesis analyzed spatiotemporal over-squashing as a key limitation, other important aspects such as model expressivity [Gao and Ribeiro, 2022] and stability to missing data still require in-depth investigation. Bridging the gap between theoretical understanding and practical model design is crucial for advancing the field and designing more robust architectures.

8.3 Final remarks

This thesis explored the intersection of graph deep learning and irregular spatiotemporal data, contributing a principled and flexible methodological framework that combines task-specific design with theoretical insight. From reconstructing sparse observations to forecasting future values under incomplete data, the work presented here consistently demonstrates the power of graph-based representations for modeling complex, real-world phenomena. By explicitly incorporating spatial and temporal structure throughout the modeling pipeline, the presented results showed that it is possible to move beyond the assumptions of regular sampling and complete data, a critical step toward making machine learning models more aligned with how data is actually acquired in practice.

Looking ahead, the growing availability of distributed sensor networks – and spatiotemporal data in general – calls for models that are not only accurate but also efficient, scalable, and robust to irregular inputs. With data-driven approaches becoming increasingly central to modern scientific workflows, I hope the ideas and tools developed in this thesis can support and accelerate discovery, especially in fields like climate science and healthcare, where dealing with incomplete yet structured data is common, and where progress can ultimately lead to meaningful benefits for society.

Glossary

COPYFIRST	148
Synthetic sequence task, contains univariate sequences of length $T=16$ with values randomly sampled in $[0, 1]$. The model must predict the first element in the sequence. Includes 20,000 training, 320 validation, and 500 test samples.	
COPYLAST	148
Synthetic sequence task identical to COPYFIRST, but the target is the last element in the input sequence.	
AGCRN	107
The Adaptive Graph Convolutional Recurrent Network by Bai et al. [2020] .	
AQI	34
Urban air quality benchmark introduced by Zheng et al. [2015] , contains hourly PM 2.5 measurements from 437 stations across 43 cities in China over one year.	
AQI-36	34
Subset of AQI from Zheng et al. [2015] , contains hourly PM 2.5 measurements from 36 stations in Beijing over one year.	
Block-S	18
The spatial block missing data pattern.	
Block-ST	18
The spatiotemporal block missing data pattern.	
Block-T	17
The temporal block missing data pattern.	
CER-E	34
Subset of the CER-E (Full) dataset [Energy Regulation, 2016] containing	

energy consumption readings from 485 smart meters monitoring small and medium-sized enterprises.

CER-E (Full)	34
Smart meter dataset from the Irish Commission for Energy Regulation [Energy Regulation, 2016], contains 30-minute electricity consumption readings from 6,435 residential and commercial users over ≈ 18 months (25,728 time steps).	
DCNN	22
The diffusion convolutional operator by Atwood and Towsley [2016].	
DCRNN	108
The Diffusion Convolutional Recurrent Neural Network by Li et al. [2018b].	
DeepESN	101
The Deep Echo State Network by Gallicchio et al. [2017].	
EngRAD	8
Weather forecasting dataset introduced by Marisca et al. [2024], contains hourly measurements of 5 variables from 487 grid points across England over 3 years.	
FC-GRU	108
A multivariate GRU model taking as input a set of time series as if they were a single multivariate one.	
Graph WaveNet	108
The deep spatiotemporal convolutional network by Wu et al. [2019].	
GraphMSO	127
The Graph Multiple Superimposed Oscillators synthetic dataset from [Marisca et al., 2024].	
GRU-D	57
The GRU-based model with state decay by Che et al. [2018].	
GRU-I	126
A GRU model that imputes missing values with one-step-ahead predictions during processing.	
HD-TTS (AMP)	126
The Hierarchical Downsampling Time-Then-Space architecture by Marisca et al. [2024] with anisotropic message-passing layers.	

HD-TTS (IMP)	126
The Hierarchical Downsampling Time-Then-Space architecture by Marisca et al. [2024] with isotropic message-passing layers.	
LocalGRUs	108
A set of local GRU models, each with different parameters and associated with a different time series.	
METR-LA	34
The traffic forecasting benchmarks introduced by Li et al. [2018b], contains 4 months of 5-minute speed measurements acquired from 207 traffic sensors in the Los Angeles County Highway.	
PeakWeather	8
Ground-station weather dataset introduced by Zambon, Cattaneo, Marisca, et al. [2025], contains 10-minute measurements of 8 variables from 302 Swiss stations over more than 8 years.	
PEMS-BAY	34
The traffic forecasting benchmarks introduced by Li et al. [2018b], contains 6 months of 5-minute speed measurements acquired from 325 traffic sensors in the San Francisco Bay Area.	
PEMS03	34
Traffic flow dataset from Guo et al. [2022], contains 5-minute readings from 358 sensors in the third district in California over 3 months.	
PEMS04	34
Traffic flow dataset from Guo et al. [2022], contains 5-minute readings from 307 sensors in the fourth district in California over 2 months.	
PEMS07	34
Traffic flow dataset from Guo et al. [2022], contains 5-minute readings from 883 sensors in the seventh district in California over 4 months.	
PEMS08	34
Traffic flow dataset from Guo et al. [2022], contains 5-minute readings from 170 sensors in the eighth district in California over 2 months.	
Point	17
The point missing data pattern.	
PV-US	34
Simulated solar energy production dataset from the National Renewable	

Energy Laboratory (NREL) [Hummon et al., 2012], contains 10-minute resolution data for the year 2006 from 5,016 solar plants across the United States.

- Pytorch Geometric** 8
 An open-source Python library built upon PyTorch to implement and train Graph Neural Networks [Fey and Lenssen, 2019].
 📖 <https://pytorch-geometric.readthedocs.io/>
- SPIN-H** 77
 The hierarchical variant of SPIN by Marisca et al. [2022] for reduced computational complexity.
- T&S-AMP** 93
 The time-and-space recurrent architecture with anisotropic message passing by Cini, Marisca, et al. [2023b].
- T&S-IMP** 93
 The time-and-space recurrent architecture with isotropic message passing by Cini, Marisca, et al. [2023b].
- Torch Geometric Pool** 8
 An open-source Python library providing a suite of graph pooling operators compatible with Pytorch Geometric [Bianchi and Marisca, 2025].
 📖 <https://torch-geometric-pool.readthedocs.io/>
- Torch Spatiotemporal** 8
 An open-source Python library based on PyTorch for deep learning on spatiotemporal data [Cini and Marisca, 2022].
 📖 <https://torch-spatiotemporal.readthedocs.io/>
- TTS-AMP** 93
 The time-then-space recurrent architecture with anisotropic message passing by Cini, Marisca, et al. [2023b].
- TTS-IMP** 93
 The time-then-space recurrent architecture with isotropic message passing by Cini, Marisca, et al. [2023b].

Bibliography

- Ralph Abboud, Radoslav Dimitrov, and Ismail Ilkan Ceylan (Dec. 2022). “Shortest Path Networks for Graph Property Prediction”. In: *Proceedings of the First Learning on Graphs Conference*. Ed. by Bastian Rieck and Razvan Pascanu. Vol. 198. Proceedings of Machine Learning Research. PMLR, 5:1–5:25. URL: <https://proceedings.mlr.press/v198/abboud22a.html>.
- Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang (Nov. 2020). “ETC: Encoding Long and Structured Inputs in Transformers”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Association for Computational Linguistics, pp. 268–284. DOI: [10.18653/v1/2020.emnlp-main.19](https://doi.org/10.18653/v1/2020.emnlp-main.19). URL: <https://aclanthology.org/2020.emnlp-main.19/>.
- Uri Alon and Eran Yahav (2021). “On the bottleneck of graph neural networks and its practical implications”. In: *International Conference on Learning Representations*.
- Brian DO Anderson and John B Moore (2005). Optimal filtering. Courier Corporation.
- Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid (2021). “ViViT: A Video Vision Transformer”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6816–6826. DOI: [10.1109/ICCV48922.2021.00676](https://doi.org/10.1109/ICCV48922.2021.00676).
- Eleonora Arnone, Laura M Sangalli, and Andrea Vicini (2023). “Smoothing spatio-temporal data with complex missing data patterns”. In: *Statistical Modelling* 23.4, pp. 327–356.
- Kevin Ashton et al. (2009). “That ‘internet of things’ thing”. In: *RFID journal* 22.7, pp. 97–114.

- Gowtham Atluri, Anuj Karpatne, and Vipin Kumar (Aug. 2018). "Spatio-Temporal Data Mining: A Survey of Problems and Methods". In: *ACM Computing Surveys* 51.4, 83:1–83:41. ISSN: 0360-0300. DOI: [10.1145/3161602](https://doi.org/10.1145/3161602).
- James Atwood and Don Towsley (2016). "Diffusion-Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc.
- Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda (2020). "A gentle introduction to deep learning for graphs". In: *Neural Networks* 129, pp. 203–221.
- Davide Bacciu, Alessio Conte, and Francesco Landolfi (2023). "Generalizing downsampling from regular data to graphs". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37, pp. 6718–6727.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural machine translation by jointly learning to align and translate". In: *International Conference on Learning Representations*.
- Shaojie Bai, J Zico Kolter, and Vladlen Koltun (2018). "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling". In: *arXiv preprint arXiv:1803.01271*.
- Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang (2020). "Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 17804–17815.
- Claudio DT Barros, Matheus RF Mendonça, Alex B Vieira, and Artur Ziviani (2021). "A survey on embedding dynamic graphs". In: *ACM Computing Surveys (CSUR)* 55.1, pp. 1–37.
- Federico Barbero, Andrea Banino, Steven Kapturowski, Dharshan Kumaran, João Madeira Araújo, Oleksandr Vitvitskyi, Razvan Pascanu, and Petar Veličković (2024a). "Transformers need glasses! information over-squashing in language tasks". In: *Advances in Neural Information Processing Systems* 37, pp. 98111–98142.
- Federico Barbero, Ameya Velingker, Amin Saberi, Michael M. Bronstein, and Francesco Di Giovanni (2024b). "Locality-Aware Graph Rewiring in GNNs". In: *The Twelfth International Conference on Learning Representations*.
- Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo,

- Adam Santoro, Ryan Faulkner, et al. (2018). “Relational inductive biases, deep learning, and graph networks”. In: *arXiv preprint arXiv:1806.01261*.
- Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, François-Xavier Aubet, Laurent Callot, and Tim Januschowski (Dec. 2022). “Deep Learning for Time Series Forecasting: Tutorial and Literature Survey”. In: *ACM Comput. Surv.* 55.6, pp. 1–36. ISSN: 0360-0300. DOI: [10.1145/3533382](https://doi.org/10.1145/3533382). URL: <https://doi.org/10.1145/3533382>.
- Yoshua Bengio, Patrice Simard, and Paolo Frasconi (1994). “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE transactions on neural networks* 5.2, pp. 157–166.
- Lorenzo Beretta and Alessandro Santaniello (2016). “Nearest neighbor imputation algorithms: a critical evaluation”. In: *BMC medical informatics and decision making* 16.3, pp. 197–208.
- Gedas Bertasius, Heng Wang, and Lorenzo Torresani (July 2021). “Is Space-Time Attention All You Need for Video Understanding?” In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 813–824. URL: <https://proceedings.mlr.press/v139/bertasius21a.html>.
- Filippo Maria Bianchi, Enrico Maiorino, Michael C Kampffmeyer, Antonello Rizzi, and Robert Jenssen (2017). Recurrent neural networks for short-term load forecasting: an overview and comparative analysis. Springer Briefs in Computer Science.
- Filippo Maria Bianchi, Lorenzo Livi, Karl Øyvind Mikalsen, Michael Kampffmeyer, and Robert Jenssen (2019). “Learning representations of multivariate time series with missing data”. In: *Pattern Recognition* 96, p. 106973.
- Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi (2020). “Hierarchical Representation Learning in Graph Neural Networks with Node Decimation Pooling”. In: *IEEE Transaction on Neural Networks and Learning Systems*.
- Filippo Maria Bianchi and Veronica Lachi (2023). “The expressive power of pooling in Graph Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine. Vol. 36. Curran Associates, Inc., pp. 71603–71618.

- Filippo Maria Bianchi and Ivan Marisca (Mar. 2025). *Torch Geometric Pool*. URL: <https://github.com/tgp-team/tgp>.
- Marin Biloš, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann (2021). “Neural Flows: Efficient Alternative to Neural ODEs”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 21325–21337. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/b21f9f98829dea9a48fd8aaddc1f159d-Paper.pdf.
- Cristian Bodnar, Wessel P. Bruinsma, Ana Lucic, Megan Stanley, Anna Allen, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A. Weyn, Haiyu Dong, Jayesh K. Gupta, Kit Thambiratnam, Alexander T. Archibald, Chun-Chieh Wu, Elizabeth Heider, Max Welling, Richard E. Turner, and Paris Perdikaris (May 2025). “A Foundation Model for the Earth System”. In: *Nature* 641.8065, pp. 1180–1187. ISSN: 1476-4687. DOI: [10.1038/s41586-025-09005-y](https://doi.org/10.1038/s41586-025-09005-y).
- Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee (2017). “Conditional time series forecasting with convolutional neural networks”. In: *arXiv preprint arXiv:1703.04691*.
- George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.
- Xavier Bresson and Thomas Laurent (2017). “Residual gated graph convnets”. In: *arXiv preprint arXiv:1711.07553*.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020). “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf.

- Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković (2021). “Geometric deep learning: Grids, groups, graphs, geodesics, and gauges”. In: *arXiv preprint arXiv:2104.13478*.
- Luca Butera, Giovanni De Felice, Andrea Cini, and Cesare Alippi (2025). “On the Regularization of Learnable Embeddings for Time Series Forecasting”. In: *Transactions on Machine Learning Research*. ISSN: 2835-8856. URL: <https://openreview.net/forum?id=F5ALCh3GWG>.
- Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang (2010). “Graph regularized nonnegative matrix factorization for data representation”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.8, pp. 1548–1560.
- Chen Cai and Yusu Wang (2020). “A note on over-smoothing for graph neural networks”. In: *arXiv preprint arXiv:2006.13318*.
- Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li (2018). “BRITS: Bidirectional Recurrent Imputation for Time Series”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc.
- Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang (2017). “Dilated recurrent neural networks”. In: *Advances in neural information processing systems* 30.
- Ben Chamberlain, James Rowbottom, Maria I. Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi (July 2021). “GRAND: Graph Neural Diffusion”. In: *Proceedings of the 38th International Conference on Machine Learning*. PMLR, pp. 1407–1418.
- Bo Chang, Minmin Chen, Eldad Haber, and Ed H Chi (2021). “Antisymmetric-cRNN: A dynamical system view on recurrent neural networks”. In: *International Conference on Learning Representations*.
- Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia (2001). “Freeway performance measurement system: mining loop detector data”. In: *Transportation Research Record* 1748.1, pp. 96–102.
- Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu (2018). “Recurrent neural networks for multivariate time series with missing values”. In: *Scientific reports* 8.1, pp. 1–12.
- Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud (2018). “Neural Ordinary Differential Equations”. In: 31. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett.

- Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun (Apr. 2020a). "Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04, pp. 3438–3445. doi: [10.1609/aaai.v34i04.5747](https://doi.org/10.1609/aaai.v34i04.5747).
- Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li (July 2020b). "Simple and Deep Graph Convolutional Networks". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 1725–1735. URL: <https://proceedings.mlr.press/v119/chen20v.html>.
- Hongjie Chen, Ryan A Rossi, Kanak Mahadik, Sungchul Kim, and Hoda Eldardiry (2021). "Graph Deep Factors for Forecasting with Applications to Cloud Resource Allocation". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 106–116.
- Ling Chen, Donghui Chen, Zongjiang Shang, Binqing Wu, Cen Zheng, Bo Wen, and Wei Zhang (2023). "Multi-scale adaptive graph neural network for multivariate time series forecasting". In: *IEEE Transactions on Knowledge and Data Engineering*.
- Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh (2019). "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 257–266.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: pp. 1724–1734.
- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555*.
- Andrzej Cichocki and Anh-Huy Phan (2009). "Fast local algorithms for large scale nonnegative matrix and tensor factorizations". In: *IEICE transactions on fundamentals of electronics, communications and computer sciences* 92.3, pp. 708–721.
- Andrea Cini and Ivan Marisca (Mar. 2022). [Torch Spatiotemporal](https://github.com/TorchSpatiotemporal/tsl). URL: <https://github.com/TorchSpatiotemporal/tsl>.

- Andrea Cini, Ivan Marisca, and Cesare Alippi (Jan. 2022). “Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks”. In: *International Conference on Learning Representations*. URL: <https://arxiv.org/abs/2108.00298>.
- Andrea Cini, Ivan Marisca, Filippo Maria Bianchi, and Cesare Alippi (June 2023a). “Scalable Spatiotemporal Graph Neural Networks”. In: *Proceedings of the 37th AAAI Conference on Artificial Intelligence* 37.6, pp. 7218–7226. doi: [10.1609/aaai.v37i6.25880](https://doi.org/10.1609/aaai.v37i6.25880).
- Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi (2023b). “Taming Local Effects in Graph-based Spatiotemporal Forecasting”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine. Vol. 36. Curran Associates, Inc., pp. 55375–55393. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/ad58c61c71efd5436134a3ecc87da6ea-Abstract-Conference.html.
- Andrea Cini, Daniele Zambon, and Cesare Alippi (2023c). “Sparse Graph Learning from Spatiotemporal Time Series”. In: *Journal of Machine Learning Research* 24.242, pp. 1–36.
- Andrea Cini, Danilo Mandic, and Cesare Alippi (July 2024). “Graph-based Time Series Clustering for End-to-End Hierarchical Forecasting”. In: *Proceedings of the 41st International Conference on Machine Learning*. Ed. by Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp. Vol. 235. Proceedings of Machine Learning Research. PMLR, pp. 8985–8999.
- Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi (June 2025). “Graph Deep Learning for Time Series Forecasting”. In: *ACM Computing Surveys* 57.12. ISSN: 0360-0300. doi: [10.1145/3742784](https://doi.org/10.1145/3742784).
- Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter (2016). “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”. In: *International Conference on Learning Representations*.
- AD Cliff and John K Ord (1975). “Space-time modelling with an application to regional forecasting”. In: *Transactions of the Institute of British Geographers*, pp. 119–128.
- Riccardo Corrias, Martin Gjoreski, and Marc Langheinrich (2023). “Exploring Transformer and Graph Convolutional Networks for Human Mobility

- Modeling". In: *Sensors* 23.10. ISSN: 1424-8220. DOI: [10.3390/s23104803](https://doi.org/10.3390/s23104803). URL: <https://www.mdpi.com/1424-8220/23/10/4803>.
- Flavio Corradini, Flavio Gerosa, Marco Gori, Carlo Lucheroni, Marco Piangerelli, and Martina Zannotti (2024). "A systematic literature review of spatio-temporal graph neural network models for time series forecasting and classification". In: *arXiv preprint arXiv:2410.22377*.
- Noel Cressie and Christopher K Wikle (2011). *Statistics for spatio-temporal data*. John Wiley & Sons.
- Zhicheng Cui, Wenlin Chen, and Yixin Chen (2016). "Multi-scale convolutional neural networks for time series classification". In: *arXiv preprint arXiv:1603.06995*.
- Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau (2019). "GRU-ODE-Bayes: Continuous Modeling of Sporadically-Observed Time Series". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/455cb2657aaa59e32fad80cb0b65b9dc-Paper.pdf.
- Giovanni De Felice, Andrea Cini, Daniele Zambon, Vladimir Gusev, and Cesare Alippi (2024). "Graph-based Virtual Sensing from Sparse and Partial Multivariate Observations". In: *International Conference on Learning Representations*.
- Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst (2016). "Convolutional neural networks on graphs with fast localized spectral filtering". In: 29, pp. 3844–3852.
- Min Deng, Wentao Yang, Qiliang Liu, Rui Jin, Feng Xu, and Yunfei Zhang (2018). "Heterogeneous Space–Time Artificial Neural Networks for Space–Time Series Prediction". In: *Transactions in GIS* 22.1, pp. 183–201. DOI: [10.1111/tgis.12302](https://doi.org/10.1111/tgis.12302).
- Ailin Deng and Bryan Hooi (2021). "Graph neural network-based anomaly detection in multivariate time series". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 5, pp. 4027–4035.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and*

- Short Papers*). Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, pp. 4171–4186. DOI: [10.18653/V1/N19-1423](https://doi.org/10.18653/V1/N19-1423). URL: <https://doi.org/10.18653/v1/n19-1423>.
- Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M Bronstein (2023a). “On over-squashing in message passing neural networks: The impact of width, depth, and topology”. In: *International Conference on Machine Learning*. PMLR, pp. 7865–7885.
- Francesco Di Giovanni, James Rowbottom, Benjamin Paul Chamberlain, Thomas Markovich, and Michael M. Bronstein (2023b). “Understanding convolution on graphs via energies”. In: *Transactions on Machine Learning Research*. ISSN: 2835-8856.
- Francesco Di Giovanni, T. Konstantin Rusch, Michael Bronstein, Andreea Deac, Marc Lackenby, Siddhartha Mishra, and Petar Veličković (2024). “How does over-squashing affect the power of GNNs?” In: *Transactions on Machine Learning Research*. ISSN: 2835-8856.
- Giovanna Maria Dimitri, Irene Cappelli, Franco Scarselli, Ada Fort, and Marco Gori (2024). “Graph Neural Networks for Missing Data Imputation in Time Series from Meteorological Sensors”. In: *2024 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRaine)*, pp. 1242–1247. DOI: [10.1109/MetroXRaine62247.2024.10796616](https://doi.org/10.1109/MetroXRaine62247.2024.10796616).
- Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst (2016). “Learning Laplacian matrix in smooth graph signal representations”. In: *IEEE Transactions on Signal Processing* 64.23, pp. 6160–6173.
- Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard (2019). “Learning Graphs From Data: A Signal Representation Perspective”. In: *IEEE Signal Processing Magazine* 36.3, pp. 44–63. DOI: [10.1109/MSP.2018.2887284](https://doi.org/10.1109/MSP.2018.2887284).
- Xiaowen Dong, Dorina Thanou, Laura Toni, Michael Bronstein, and Pascal Frossard (2020). “Graph signal processing for machine learning: A review and new perspectives”. In: *IEEE Signal processing magazine* 37.6, pp. 117–127.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby (2021). “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=YicbFdNTTy>.

- Wenjie Du, David Côté, and Yan Liu (2023). "SAITS: Self-attention-based imputation for time series". In: *Expert Systems with Applications* 219, p. 119619. ISSN: 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2023.119619>. URL: <https://www.sciencedirect.com/science/article/pii/S0957417423001203>.
- James Durbin and Siem Jan Koopman (2012). *Time series analysis by state space methods*. Oxford university press.
- Vijay Prakash Dwivedi and Xavier Bresson (2021). "A Generalization of Transformer Networks to Graphs". In: *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*.
- Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson (2022). "Graph Neural Networks with Learnable Structural and Positional Representations". In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=wTTjnvGphYj>.
- Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson (2023). "Benchmarking Graph Neural Networks". In: *Journal of Machine Learning Research* 24.43, pp. 1–48. URL: <http://jmlr.org/papers/v24/22-0567.html>.
- Paul HC Eilers and Brian D Marx (1996). "Flexible smoothing with B-splines and penalties". In: *Statistical science* 11.2, pp. 89–121.
- Jeffrey L Elman (1990). "Finding structure in time". In: *Cognitive science* 14.2, pp. 179–211.
- Commission for Energy Regulation (2016). "CER Smart Metering Project - Electricity Customer Behaviour Trial, 2009-2010 [dataset]". In: *Irish Social Science Data Archive*. SN: 0012-00. URL: <https://www.ucd.ie/issda/data/commissionforenergyregulationcer>.
- Chenguang Fang and Chen Wang (2020). "Time series data imputation: A survey on deep learning approaches". In: *arXiv preprint arXiv:2011.11347*.
- William Fedus, Ian Goodfellow, and Andrew M Dai (2018). "MaskGAN: Better Text Generation via Filling in the _". In: *International Conference on Learning Representations*.
- Matthias Fey and Jan E. Lenssen (2019). "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
- Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He (June 2019). "Learning Discrete Structures for Graph Neural Networks". In: *Proceedings*

- of the 36th International Conference on Machine Learning. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 1972–1982. URL: <https://proceedings.mlr.press/v97/franceschi19a.html>.
- Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Benjamin Chamberlain, Michael Bronstein, and Federico Monti (2020). “SIGN: Scalable Inception Graph Neural Networks”. In: *ICML 2020 Workshop on Graph Representation Learning and Beyond*.
- Paolo Frasconi, Marco Gori, and Alessandro Sperduti (1998). “A general framework for adaptive processing of data structures”. In: *IEEE transactions on Neural Networks* 9.5, pp. 768–786.
- Joshua P French and Piotr S Kokoszka (2021). “A sandwich smoother for spatio-temporal functional data”. In: *Spatial Statistics* 42, p. 100413.
- Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli (2017). “Deep reservoir computing: A critical experimental analysis”. In: *Neurocomputing* 268, pp. 87–99. ISSN: 0925-2312. DOI: [10.1016/j.neucom.2016.12.089](https://doi.org/10.1016/j.neucom.2016.12.089).
- Ankit Gandhi, Sivaramakrishnan Kaveri, Vineet Chaoji, et al. (2021). “Spatio-Temporal Multi-graph Networks for Demand Forecasting in Online Marketplaces”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 187–203.
- Jianfei Gao and Bruno Ribeiro (July 2022). “On the Equivalence Between Temporal and Static Equivariant Graph Representations”. In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 7052–7076.
- Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann (2019). “Diffusion Improves Graph Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc.
- Zoubin Ghahramani and Michael Jordan (1994). “Supervised learning from incomplete data via an EM approach”. In: *Advances in Neural Information Processing Systems*. Ed. by J. Cowan, G. Tesauro, and J. Alspector. Vol. 6. Morgan-Kaufmann.
- Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl (2017). “Neural message passing for quantum chemistry”. In: *International Conference on Machine Learning*. PMLR, pp. 1263–1272.

- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). “Generative adversarial nets”. In: *Advances in neural information processing systems* 27.
- M. Gori, G. Monfardini, and F. Scarselli (2005). “A new model for learning in graph domains”. In: *Proceedings. 2005 IEEE international joint conference on neural networks, 2005*. Vol. 2. IEEE, pp. 729–734. DOI: [10.1109/IJCNN.2005.1555942](https://doi.org/10.1109/IJCNN.2005.1555942).
- Alex Graves and Jürgen Schmidhuber (2008). “Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou. Vol. 21. Curran Associates, Inc.
- Alessio Gravina, Davide Bacciu, and Claudio Gallicchio (2023). “Anti-Symmetric DGN: a stable architecture for Deep Graph Networks”. In: *The Eleventh International Conference on Learning Representations*.
- Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi (2024). “Understanding Pooling in Graph Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* 35.2, pp. 2708–2718.
- Alessio Gravina, Giulio Lovisotto, Claudio Gallicchio, Davide Bacciu, and Claas Grohnfeldt (July 2024a). “Long Range Propagation on Continuous-Time Dynamic Graphs”. In: *Proceedings of the 41st International Conference on Machine Learning*. Ed. by Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp. Vol. 235. Proceedings of Machine Learning Research. PMLR, pp. 16206–16225. URL: <https://proceedings.mlr.press/v235/gravina24a.html>.
- Alessio Gravina, Daniele Zambon, Davide Bacciu, and Cesare Alippi (Aug. 2024b). “Temporal Graph ODEs for Irregularly-Sampled Time Series”. In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*. International Joint Conferences on Artificial Intelligence Organization, pp. 4025–4034. DOI: [10.24963/ijcai.2024/445](https://doi.org/10.24963/ijcai.2024/445). URL: <https://doi.org/10.24963/ijcai.2024/445>.
- Alessio Gravina, Moshe Eliasof, Claudio Gallicchio, Davide Bacciu, and Carola-Bibiane Schönlieb (Apr. 2025). “On Oversquashing in Graph Neural Networks Through the Lens of Dynamical Systems”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 39.16, pp. 16906–16914. DOI: [10.1609/aaai.v39i16.33858](https://doi.org/10.1609/aaai.v39i16.33858).

- Clive WJ Granger (1969). "Investigating causal relations by econometric models and cross-spectral methods". In: *Econometrica: journal of the Econometric Society* 37.3, pp. 424–438.
- C. W. J. Granger (1975). "Aspects of the Analysis and Interpretation of Temporal and Spatial Data". In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 24.3, pp. 197–210. (Visited on 04/04/2025).
- Charles Miller Grinstead and James Laurie Snell (2006). Grinstead and Snell's introduction to probability. Chance Project.
- Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan (2019). "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 922–929.
- Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong (2022). "Learning Dynamics and Heterogeneity of Spatial-Temporal Graph Data for Traffic Forecasting". In: *IEEE Transactions on Knowledge and Data Engineering* 34.11, pp. 5415–5428. DOI: [10.1109/TKDE.2021.3056502](https://doi.org/10.1109/TKDE.2021.3056502).
- Benjamin Gutteridge, Xiaowen Dong, Michael M. Bronstein, and Francesco Di Giovanni (July 2023). "DRew: Dynamically Rewired Message Passing with Delay". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, pp. 12252–12267.
- Will Hamilton, Zhitao Ying, and Jure Leskovec (2017). "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc.
- Oliver Hamelijnck, William Wilkinson, Niki Loppi, Arno Solin, and Theodoros Damoulas (2021). "Spatio-Temporal Variational Gaussian Processes". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 23621–23633.
- Fredric J Harris (2022). Multirate signal processing for communication systems. CRC Press.
- Robert Henderson (1924). "A new method of graduation". In: *Transactions of the Actuarial Society of America* 25, pp. 29–40.

- Leonard R Herrmann (1976). "Laplacian-isoparametric grid generation scheme". In: *Journal of the Engineering Mechanics Division* 102.5, pp. 749–756.
- Sepp Hochreiter and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.
- J Leith Holloway Jr (1958). "Smoothing and filtering of time series and space fields". In: *Advances in geophysics*. Vol. 4. Elsevier, pp. 351–389.
- Zijie Huang, Yizhou Sun, and Wei Wang (2020). "Learning continuous system dynamics from irregularly-sampled partial observations". In: *Advances in Neural Information Processing Systems* 33, pp. 16177–16187.
- Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany (2023). "Temporal Graph Benchmark for Machine Learning on Temporal Graphs". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine. Vol. 36. Curran Associates, Inc., pp. 2056–2073.
URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/066b98e63313162f6562b35962671288-Paper-Datasets_and-Benchmarks.pdf.
- Marissa Hummon, Eduardo Ibanez, Gregory Brinkman, and Debra Lew (2012). Sub-hour solar data for power system modeling from static spatial variability analysis. Tech. rep. National Renewable Energy Lab.(NREL), Golden, CO (United States).
- Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert (2007). "Optimization and Applications of Echo State Networks with Leaky-Integrator Neurons". In: *Neural Networks* 20.3, pp. 335–352.
- Herbert Jaeger (2001). "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note". In: *Bonn, Germany: German national research center for information technology gmd technical report* 148.34, p. 13.
- Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena (2016). "Structural-rnn: Deep learning on spatio-temporal graphs". In: *Proceedings of the ieee conference on computer vision and pattern recognition*, pp. 5308–5317.
- Xiaowei Jia, Ankush Khandelwal, Guruprasad Nayak, James Gerber, Kimberly Carlson, Paul West, and Vipin Kumar (2017). "Incremental dual-memory lstm in land cover prediction". In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 867–876.

- Ming Jin, Huan Yee Koh, Qingsong Wen, Daniele Zambon, Cesare Alippi, Geoffrey I Webb, Irwin King, and Shirui Pan (2024). "A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Rudolph Emil Kalman (1960). "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME-Journal of Basic Engineering* 82.Series D, pp. 35–45.
- Amol Kapoor, Xue Ben, Luyang Liu, Bryan Perozzi, Matt Barnes, Martin Blais, and Shawn O'Banion (2020). "Examining COVID-19 Forecasting using Spatio-Temporal GNNs". In: *Proceedings of the 16th International Workshop on Mining and Learning with Graphs (MLG)*.
- Kedar Karhadkar, Pradeep Kr. Banerjee, and Guido Montufar (2023). "FoSR: First-order spectral rewiring for addressing oversquashing in GNNs". In: *The Eleventh International Conference on Learning Representations*.
- Matthias Katzfuss and Noel Cressie (2011). "Spatio-temporal smoothing and EM estimation for massive remote-sensing data sets". In: *Journal of Time Series Analysis* 32.4, pp. 430–446.
- Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker (2019). "Time2vec: Learning a vector representation of time". In: *arXiv preprint arXiv:1907.05321*.
- Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart (2020). "Representation Learning for Dynamic Graphs: A Survey". In: *Journal of Machine Learning Research* 21.70, pp. 1–73.
- Diederik P Kingma and Max Welling (2014). "Auto-encoding variational bayes". In: *International Conference on Learning Representations*.
- Thomas N. Kipf and Max Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=SJU4ayYgl>.
- Ismaiel Krim, Mohamed Tahar Mezeddek, and Abderrahmane Smail (2022). "On powers and roots of triangular Toeplitz matrices". In: *Applied Mathematics E-Notes* 22, pp. 322–330.
- Sanmukh R. Kuppannagari, Yao Fu, Chung Ming Chueng, and Viktor K. Prasanna (2021). "Spatio-Temporal Missing Data Imputation for Smart Power Grids". In: *Proceedings of the Twelfth ACM International Conference on Future En-*

- ergy Systems*. e-Energy '21. New York, NY, USA: Association for Computing Machinery, pp. 458–465. ISBN: 9781450383332. DOI: [10.1145/3447555.3466586](https://doi.org/10.1145/3447555.3466586). URL: <https://doi.org/10.1145/3447555.3466586>.
- Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu (2018). “Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks”. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '18. Ann Arbor, MI, USA: Association for Computing Machinery, pp. 95–104. ISBN: 9781450356572. DOI: [10.1145/3209978.3210006](https://doi.org/10.1145/3209978.3210006). URL: <https://doi.org/10.1145/3209978.3210006>.
- Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. (2023). “Learning skillful medium-range global weather forecasting”. In: *Science* 382.6677, pp. 1416–1421.
- Marc Langheinrich (2001). “Privacy by design—principles of privacy-aware ubiquitous systems”. In: *International conference on ubiquitous computing*. Springer, pp. 273–291.
- Marine Le Morvan, Julie Josse, Erwan Scornet, and Gael Varoquaux (2021). “What’s a good imputation to predict with missing values?” In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 11530–11540. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/5fe8fdc79ce292c39c5f209d734b7206-Paper.pdf.
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton (2015). “Deep learning”. In: *nature* 521.7553, pp. 436–444.
- Yann LeCun, Yoshua Bengio, et al. (1995). “Convolutional networks for images, speech, and time series”. In: *The handbook of brain theory and neural networks* 3361.10, p. 1995.
- Yann LeCun and Yoshua Bengio (1998). “Convolutional Networks for Images, Speech, and Time Series”. In: *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA, USA: MIT Press, pp. 255–258. ISBN: 0262511029.
- Daniel Lee and H. Sebastian Seung (2000). “Algorithms for Non-negative Matrix Factorization”. In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press. URL: https://proceedings.neurips.cc/paper_files/paper/2000/file/f9d1152547c0bde01830b7e8bd60024c-Paper.pdf.

- Daniel D. Lee and H. Sebastian Seung (Oct. 1999). "Learning the Parts of Objects by Non-Negative Matrix Factorization". In: *Nature* 401.6755, pp. 788–791. ISSN: 1476-4687. DOI: [10.1038/44565](https://doi.org/10.1038/44565).
- Geert Leus, Antonio G Marques, José MF Moura, Antonio Ortega, and David I Shuman (2023). "Graph Signal Processing: History, development, impact, and outlook". In: *IEEE Signal Processing Magazine* 40.4, pp. 49–60.
- Qimai Li, Zhichao Han, and Xiao-ming Wu (Apr. 2018a). "Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1. DOI: [10.1609/aaai.v32i1.11604](https://doi.org/10.1609/aaai.v32i1.11604).
- Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu (2018b). "Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting". In: *International Conference on Learning Representations*.
- Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyong Zhou, Wenhui Chen, Yu-Xiang Wang, and Xifeng Yan (2019). "Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc.
- Steven Cheng-Xian Li and Benjamin Marlin (July 2020). "Learning from Irregularly-Sampled Time Series: A Missing Data Perspective". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 5937–5946. URL: <https://proceedings.mlr.press/v119/li20k.html>.
- Guohao Li, Matthias Müller, Guocheng Qian, Itzel C. Delgadillo, Abdulellah Abualshour, Ali Thabet, and Bernard Ghanem (2023). "DeepGCNs: Making GCNs Go as Deep as CNNs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.6, pp. 6923–6939. DOI: [10.1109/TPAMI.2021.3074057](https://doi.org/10.1109/TPAMI.2021.3074057).
- Yuebing Liang, Zhan Zhao, and Lijun Sun (2021). "Dynamic spatiotemporal graph convolutional neural networks for traffic data imputation with complex missing patterns". In: *arXiv preprint arXiv:2109.08357*.
- Marco Lippi, Matteo Bertini, and Paolo Frasconi (2013). "Short-Term Traffic Flow Forecasting: An Experimental Comparison of Time-Series Analysis and Supervised Learning". In: *IEEE Transactions on Intelligent Transportation Systems* 14.2, pp. 871–882. DOI: [10.1109/TITS.2013.2247040](https://doi.org/10.1109/TITS.2013.2247040).

- Zachary C Lipton, David Kale, and Randall Wetzel (Aug. 2016). "Directly Modeling Missing Data in Sequences with RNNs: Improved Classification of Clinical Time Series". In: *Proceedings of the 1st Machine Learning for Healthcare Conference*. Ed. by Finale Doshi-Velez, Jim Fackler, David Kale, Byron Wallace, and Jenna Wiens. Vol. 56. Proceedings of Machine Learning Research. PMLR, pp. 253–270.
- Roderick JA Little and Donald B Rubin (2019). *Statistical analysis with missing data*. Vol. 793. John Wiley & Sons.
- Weifeng Liu, Puskal P Pokharel, and Jose C Principe (2007). "Correntropy: Properties and applications in non-Gaussian signal processing". In: *IEEE Transactions on Signal Processing* 55.11, pp. 5286–5298.
- Lichuan Liu, Sen M. Kuo, and MengChu Zhou (2009). "Virtual sensing techniques and their applications". In: *2009 International Conference on Networking, Sensing and Control*, pp. 31–36. DOI: [10.1109/ICNSC.2009.4919241](https://doi.org/10.1109/ICNSC.2009.4919241).
- Yukai Liu, Rose Yu, Stephan Zheng, Eric Zhan, and Yisong Yue (2019). "NAOMI: Non-Autoregressive Multiresolution Sequence Imputation". In: *Advances in Neural Information Processing Systems* 32, pp. 11238–11248.
- Gary Liu, Denise B Catacutan, Khushi Rathod, Kyle Swanson, Wengong Jin, Jody C Mohammed, Anush Chiappino-Pepe, Saad A Syed, Meghan Fragis, Kenneth Rachwalski, et al. (2023a). "Deep learning-guided discovery of an antibiotic targeting *Acinetobacter baumannii*". In: *Nature Chemical Biology* 19.11, pp. 1342–1350.
- Mingzhe Liu, Han Huang, Hao Feng, Leilei Sun, Bowen Du, and Yanjie Fu (2023b). "PriSTI: A Conditional Diffusion Framework for Spatiotemporal Imputation". In: *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pp. 1927–1939. DOI: [10.1109/ICDE55515.2023.00150](https://doi.org/10.1109/ICDE55515.2023.00150).
- Antonio Longa, Veronica Lachi, Gabriele Santin, Monica Bianchini, Bruno Lepri, Pietro Lio, Franco Scarselli, and Andrea Passerini (2023). "Graph Neural Networks for Temporal Graphs: State of the Art, Open Challenges, and Opportunities". In: *Transactions on Machine Learning Research*. ISSN: 2835-8856. URL: <https://openreview.net/forum?id=pHCdMat0gI>.
- Mantas Lukoševičius and Herbert Jaeger (2009). "Reservoir Computing Approaches to Recurrent Neural Network Training". In: *Computer Science Review* 3.3, pp. 127–149.
- Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, and Yuan Xiaojie (2018). "Multivariate Time Series Imputation with Generative Adversarial Networks".

- In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc.
- Yonghong Luo, Ying Zhang, Xiangrui Cai, and Xiaojie Yuan (July 2019). “E²GAN: End-to-End Generative Adversarial Network for Multivariate Time Series Imputation”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, pp. 3094–3100.
- B. Luque, L. Lacasa, F. Ballesteros, and J. Luque (Oct. 2009). “Horizontal visibility graphs: Exact results for random time series”. In: *Phys. Rev. E* 80 (4), p. 046103. DOI: [10.1103/PhysRevE.80.046103](https://doi.org/10.1103/PhysRevE.80.046103). URL: <https://link.aps.org/doi/10.1103/PhysRevE.80.046103>.
- Ying C MacNab and CB Dean (2001). “Autoregressive spatial smoothing and temporal spline smoothing for mapping rates”. In: *Biometrics* 57.3, pp. 949–956.
- C Maddison, A Mnih, and Y Teh (2017). “The concrete distribution: A continuous relaxation of discrete random variables”. In: *International Conference on Learning Representations*.
- Alessandro Manenti, Daniele Zambon, and Cesare Alippi (2025). “Learning Latent Graph Structures and their Uncertainty”. In: *Forty-second International Conference on Machine Learning*. URL: <https://openreview.net/forum?id=TMRh3ScSCb>.
- Ivan Marisca, Andrea Cini, and Cesare Alippi (2022). “Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., pp. 32069–32082. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/cf70320e93c08b39b1b29a348097a376-Paper-Conference.pdf.
- Ivan Marisca, Cesare Alippi, and Filippo Maria Bianchi (2024). “Graph-based Forecasting with Missing Data through Spatiotemporal Downsampling”. In: *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. Proceedings of Machine Learning Research. PMLR, pp. 34846–34865. URL: <https://proceedings.mlr.press/v235/marisca24a.html>.
- Ivan Marisca, Jacob Bamberger, Cesare Alippi, and Michael M. Bronstein (2025). “Over-squashing in Spatiotemporal Graph Neural Networks”. In: *arXiv preprint arXiv:2506.15507*. URL: <https://arxiv.org/abs/2506.15507>.

- Jiali Mei, Yohann De Castro, Yannig Goude, and Georges Hébrail (2017). “Non-negative matrix factorization for time series recovery from a few temporal aggregates”. In: *International Conference on Machine Learning*. PMLR, pp. 2382–2390.
- Amil Merchant, Simon Batzner, Samuel S Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk (2023). “Scaling deep learning for materials discovery”. In: *Nature* 624.7990, pp. 80–85.
- Bernd Messnarz, Bernhard Tilg, Robert Modre, Gerald Fischer, and Friedrich Hanser (2004). “A new spatiotemporal regularization approach for reconstruction of cardiac transmembrane potential patterns”. In: *IEEE transactions on Biomedical Engineering* 51.2, pp. 273–281.
- Xiaoye Miao, Yangyang Wu, Jun Wang, Yunjun Gao, Xudong Mao, and Jianwei Yin (2021). “Generative Semi-supervised Learning for Multivariate Time Series Imputation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 10, pp. 8983–8991.
- Alessio Micheli and Domenico Tortorella (2022). “Discrete-time dynamic graph echo state networks”. In: *Neurocomputing* 496, pp. 85–95. ISSN: 0925-2312. DOI: [10.1016/j.neucom.2022.05.001](https://doi.org/10.1016/j.neucom.2022.05.001).
- Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein (2019). “Fake news detection on social media using geometric deep learning”. In: *arXiv preprint arXiv:1902.06673*.
- Pablo Montero-Manso and Rob J Hyndman (2021). “Principles and algorithms for forecasting groups of time series: Locality and globality”. In: *International Journal of Forecasting* 37.4, pp. 1632–1653.
- Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe (July 2019). “Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01, pp. 4602–4609. DOI: [10.1609/aaai.v33i01.33014602](https://doi.org/10.1609/aaai.v33i01.33014602). URL: <https://ojs.aaai.org/index.php/AAAI/article/view/4384>.
- Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampásek (2024). “Attending to Graph Transformers”. In: *Transactions on Machine Learning Research*. ISSN: 2835-8856. URL: <https://openreview.net/forum?id=HhbqHBBrfZ>.

- Fulufhelo V Nelwamondo, Shakir Mohamed, and Tshilidzi Marwala (2007). "Missing data: A comparison of neural network and expectation maximization techniques". In: *Current Science*, pp. 1514–1521.
- Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam (2023). "A Time Series is Worth 64 Words: Long-term Forecasting with Transformers". In: *International Conference on Learning Representations*.
- Evangelos Niforatos, Athanasios Vourvopoulos, Marc Langheinrich, Pedro Campos, and Andre Doria (2014). "Atmos: a hybrid crowdsourcing approach to weather estimation". In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. UbiComp '14 Adjunct. Seattle, Washington: Association for Computing Machinery, pp. 135–138. ISBN: 9781450330473. DOI: [10.1145/2638728.2638780](https://doi.org/10.1145/2638728.2638780). URL: <https://doi.org/10.1145/2638728.2638780>.
- Kenta Oono and Taiji Suzuki (2020). "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification". In: *International Conference on Learning Representations*.
- Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu (2016). "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499*.
- Boris N Oreshkin, Arezou Amini, Lucy Coyle, and Mark Coates (2021). "FC-GAGA: Fully connected gated graph architecture for spatio-temporal traffic forecasting". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 10, pp. 9233–9241.
- Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vanderghelynst (2018). "Graph signal processing: Overview, challenges, and applications". In: *Proceedings of the IEEE* 106.5, pp. 808–828.
- Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De (July 2023). "Resurrecting Recurrent Neural Networks for Long Sequences". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, pp. 26670–26698. URL: <https://proceedings.mlr.press/v202/orvieto23a.html>.

- Benjamin Paassen, Daniele Grattarola, Daniele Zambon, Cesare Alippi, and Barbara Eva Hammer (2020). “Graph Edit Networks”. In: *International Conference on Learning Representations*.
- Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson (2020). “EvolveGCN: Evolving graph convolutional networks for dynamic graphs”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04, pp. 5363–5370.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio (June 2013). “On the difficulty of training recurrent neural networks”. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, pp. 1310–1318. URL: <https://proceedings.mlr.press/v28/pascanu13.html>.
- Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. (2022). “FourCastNet: A global data-driven high-resolution weather model using adaptive fourier neural operators”. In: *arXiv preprint arXiv:2202.11214*.
- Jesús Pineda, Benjamin Midtvedt, Harshith Bachimanchi, Sergio Noé, Daniel Midtvedt, Giovanni Volpe, and Carlo Manzo (2023). “Geometric deep learning reveals the spatiotemporal features of microscopic motion”. In: *Nature Machine Intelligence* 5.1, pp. 71–82. DOI: [10.1038/s42256-022-00595-0](https://doi.org/10.1038/s42256-022-00595-0).
- Luca Ponti, Simona Perotto, and Laura M Sangalli (2022). “A PDE-regularized smoothing method for space–time data over manifolds with application to medical data”. In: *International Journal for Numerical Methods in Biomedical Engineering* 38.12, e3650.
- Alain Pulfer, Diego Ulisse Pizzagalli, Paolo Armando Gagliardi, Lucien Hinderling, Paul Lopez, Romaniya Zayats, Pau Carrillo-Barberà, Paola Antonello, Miguel Palomino-Segura, Benjamin Grädel, Mariaclaudia Nicolai, Alessandro Giusti, Marcus Thelen, Luca Maria Gambardella, Thomas T Murooka, Olivier Pertz, Rolf Krause, and Santiago Fernandez Gonzalez (Mar. 2024). “Transformer-based spatial–temporal detection of apoptotic cell death in live-cell imaging”. In: *eLife* 12. Ed. by Vitaly Ryu and Detlef Weigel, RP90502. ISSN: 2050-084X. DOI: [10.7554/eLife.90502](https://doi.org/10.7554/eLife.90502). URL: <https://doi.org/10.7554/eLife.90502>.

- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (June 2018). "Improving Language Understanding by Generative Pre-Training". In: *OpenAI Technical Report*. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.
- Syama Sundar Rangapuram, Lucien D Werner, Konstantinos Benidis, Pedro Mercado, Jan Gasthaus, and Tim Januschowski (2021). "End-to-end learning of coherent probabilistic forecasts for hierarchical time series". In: *International Conference on Machine Learning*. PMLR, pp. 8832–8843.
- Syama Sundar Rangapuram, Shubham Kapoor, Rajbir Singh Nirwan, Pedro Mercado, Tim Januschowski, Yuyang Wang, and Michael Bohlke-Schneider (2023). "Coherent probabilistic forecasting of temporal hierarchies". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 9362–9376.
- Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon (2015). "Collaborative Filtering with Graph Information: Consistency and Scalable Methods". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc.
- Carl Edward Rasmussen and Christopher K. I. Williams (Nov. 2005). *Gaussian Processes for Machine Learning*. The MIT Press. ISBN: 9780262256834. DOI: [10.7551/mitpress/3206.001.0001](https://doi.org/10.7551/mitpress/3206.001.0001). URL: <https://doi.org/10.7551/mitpress/3206.001.0001>.
- Herbert E Rauch, F Tung, and Charlotte T Striebel (1965). "Maximum likelihood estimates of linear dynamic systems". In: *AIAA journal* 3.8, pp. 1445–1450.
- Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang (2020). "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification". In: *International Conference on Learning Representations*.
- Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein (2020). "Temporal graph networks for deep learning on dynamic graphs". In: *arXiv preprint arXiv:2006.10637*.
- Emanuele Rossi, Henry Kenlay, Maria I Gorinova, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein (2022). "On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features". In: *Learning on Graphs Conference*. PMLR, pp. 11–1.
- Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud (2019). "Latent Ordinary Differential Equations for Irregularly-Sampled Time Series". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H.

- Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc.
- Donald B Rubin (1976). "Inference and missing data". In: *Biometrika* 63.3, pp. 581–592.
- Luana Ruiz, Luiz Chamon, and Alejandro Ribeiro (2020). "Graphon Neural Networks and the Transferability of Graph Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 1702–1712.
- T. Konstantin Rusch and Siddhartha Mishra (July 2021). "UnICORNN: A recurrent model for learning very long time dependencies". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 9168–9178.
- T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra (2023). "A survey on oversmoothing in graph neural networks". In: *arXiv preprint arXiv:2303.10993*.
- Mohammad Sabbaqi and Elvin Isufi (2023). "Graph-Time Convolutional Neural Networks: Architecture and Theoretical Analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.12, pp. 14625–14638. doi: [10.1109/TPAMI.2023.3311912](https://doi.org/10.1109/TPAMI.2023.3311912).
- David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski (2020). "DeepAR: Probabilistic forecasting with autoregressive recurrent networks". In: *International Journal of Forecasting* 36.3, pp. 1181–1191.
- Victor Garcia Satorras, Syama Sundar Rangapuram, and Tim Januschowski (2022). "Multivariate time series forecasting with latent graph inference". In: *arXiv preprint arXiv:2203.03423*.
- Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini (2008). "The graph neural network model". In: *IEEE transactions on neural networks* 20.1, pp. 61–80.
- Jürgen Schmidhuber (2015). "Deep learning in neural networks: An overview". In: *Neural networks* 61, pp. 85–117.
- Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon (2019). "Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett.

- Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/3a0844cee4fcf57de0c71e9ad3035478-Paper.pdf.
- Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson (2018). “Structured sequence modeling with graph convolutional recurrent networks”. In: *International Conference on Neural Information Processing*. Springer, pp. 362–373.
- Chao Shang, Jie Chen, and Jinbo Bi (2021). “Discrete Graph Structure Learning for Forecasting Multiple Time Series”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=WEHSLH5m0k>.
- ZeZhi Shao, Zhao Zhang, Fei Wang, Wei Wei, and Yongjun Xu (2022). “Spatial-Temporal Identity: A Simple yet Effective Baseline for Multivariate Time Series Forecasting”. In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. Atlanta, GA, USA, pp. 4454–4458.
- Siyuan Shan, Yang Li, and Junier B. Oliva (2023). “NRTSI: Non-Recurrent Time Series Imputation”. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. doi: [10.1109/ICASSP49357.2023.10095054](https://doi.org/10.1109/ICASSP49357.2023.10095054).
- Shashi Shekhar, Zhe Jiang, Reem Y Ali, Emre Eftelioglu, Xun Tang, Venkata MV Gunturi, and Xun Zhou (2015). “Spatiotemporal data mining: A computational perspective”. In: *ISPRS International Journal of Geo-Information* 4.4, pp. 2306–2338.
- Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo (2015). “Convolutional LSTM network: A machine learning approach for precipitation nowcasting”. In: *Advances in Neural Information Processing Systems* 28.
- Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun (2021). “Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1548–1554.
- David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst (2013). “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains”. In: *IEEE Signal Processing Magazine* 30.3, pp. 83–98.

- David I Shuman, Mohammad Javad Faraji, and Pierre Vandergheynst (2015). "A multiscale pyramid transform for graph signals". In: *IEEE Transactions on Signal Processing* 64.8, pp. 2119–2134.
- Satya Narayan Shukla and Benjamin Marlin (2019). "Interpolation-Prediction Networks for Irregularly Sampled Time Series". In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=r1efr3C9Ym>.
- Satya Narayan Shukla and Benjamin Marlin (2020). "Multi-Time Attention Networks for Irregularly Sampled Time Series". In: *International Conference on Learning Representations*.
- Dan Simon (2006a). "Optimal smoothing". In: *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. John Wiley & Sons, Ltd. Chap. 9, pp. 263–296. ISBN: 9780470045343. DOI: <https://doi.org/10.1002/0470045345.ch9>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/0470045345.ch9>.
- Dan Simon (2006b). *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. John Wiley & Sons, Ltd. ISBN: 9780470045343. DOI: <https://doi.org/10.1002/0470045345>. URL: <https://onlinelibrary.wiley.com/doi/book/10.1002/0470045345>.
- Alexander J. Smola and Risi Kondor (2003). "Kernels and Regularization on Graphs". In: *Learning Theory and Kernel Machines*. Ed. by Bernhard Schölkopf and Manfred K. Warmuth. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 144–158. ISBN: 978-3-540-45167-9.
- Joshua Southern, Francesco Di Giovanni, Michael M. Bronstein, and Johannes F. Lutzeyer (2025). "Understanding Virtual Nodes: Oversquashing and Node Heterogeneity". In: *The Thirteenth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=Nmc0AwRyH5>.
- A. Sperduti and A. Starita (1997). "Supervised neural networks for the classification of structures". In: *IEEE Transactions on Neural Networks* 8.3, pp. 714–735. DOI: [10.1109/72.572108](https://doi.org/10.1109/72.572108).
- Indro Spinelli, Simone Scardapane, and Aurelio Uncini (2020). "Missing data imputation with adversarially-trained graph convolutional networks". In: *Neural Networks* 129, pp. 249–260.
- Michael L Stein (1999). *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media.

- Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. (2020). "A deep learning approach to antibiotic discovery". In: *Cell* 180.4, pp. 688–702.
- Gilbert Strang and Truong Nguyen (1996). *Wavelets and filter banks*. SIAM.
- Nathan J Szymanski, Bernardus Rendy, Yuxing Fei, Rishi E Kumar, Tanjin He, David Milsted, Matthew J McDermott, Max Gallant, Ekin Dogus Cubuk, Amil Merchant, et al. (2023). "An autonomous laboratory for the accelerated synthesis of novel materials". In: *Nature* 624.7990, pp. 86–91.
- Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang (2020). "Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34, pp. 5956–5963.
- Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon (2021). "CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation". In: *Advances in Neural Information Processing Systems* 34.
- Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li (2018). "Attention-based graph neural network for semi-supervised learning". In: *arXiv preprint arXiv:1803.03735*.
- Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein (2022). "Understanding over-squashing and bottlenecks on graphs via curvature". In: *International Conference on Learning Representations*.
- Domenico Tortorella and Alessio Micheli (2022). "Leave Graphs Alone: Addressing Over-Squashing without Rewiring". In: *The First Learning on Graphs Conference*.
- Minh Ha Tran, Ofelia Gomez, and Baowei Fei (2023). "A video transformer network for thyroid cancer detection on hyperspectral histologic images". In: *Medical Imaging 2023: Digital and Computational Pathology*. Vol. 12471. SPIE, pp. 32–41.
- Nicolas Tremblay and Pierre Borgnat (2016). "Subgraph-based filterbanks for graph signals". In: *IEEE Transactions on Signal Processing* 64.15, pp. 3827–3840.
- Nicolas Tremblay, Paulo Gonçalves, and Pierre Borgnat (2018). "Design of graph filters and filterbanks". In: *Cooperative and Graph Signal Processing*. Elsevier, pp. 299–324.

- Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman (2001). "Missing value estimation methods for DNA microarrays". In: *Bioinformatics* 17.6, pp. 520–525.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need". In: *Advances in Neural Information Processing Systems*. Vol. 30, pp. 5998–6008.
- Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio (2018). "Graph Attention Networks". In: *International Conference on Learning Representations*.
- Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal (Aug. 2017). "On orthogonality and learning recurrent networks with long term dependencies". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 3570–3578.
- Grace Wahba (1990). Spline models for observational data. SIAM.
- Yuyang Wang, Alex Smola, Danielle Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski (June 2019). "Deep Factors for Forecasting". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 6607–6617. URL: <https://proceedings.mlr.press/v97/wang19k.html>.
- Senzhang Wang, Jiannong Cao, and S Yu Philip (2020). "Deep learning for spatio-temporal data mining: A survey". In: *IEEE transactions on knowledge and data engineering* 34.8, pp. 3681–3700.
- Yi Wang and Changfeng Jing (2022). "Spatiotemporal graph convolutional network for multi-scale traffic forecasting". In: *ISPRS International Journal of Geo-Information* 11.2, p. 102.
- Ao Wang, Yongchao Ye, Xiaozhuang Song, Shiyao Zhang, and JQ James (2023). "Traffic Prediction With Missing Data: A Multi-Task Learning Approach". In: *IEEE Transactions on Intelligent Transportation Systems* 24.4, pp. 4189–4202.
- Jun Wang, Wenjie Du, Yiyuan Yang, Linglong Qian, Wei Cao, Keli Zhang, Wenjia Wang, Yuxuan Liang, and Qingsong Wen (2024). "Deep learning for multi-variate time series imputation: A survey". In: *arXiv preprint arXiv:2402.04059*.

- Ian R White, Patrick Royston, and Angela M Wood (2011). "Multiple imputation using chained equations: issues and guidance for practice". In: *Statistics in medicine* 30.4, pp. 377–399.
- Edmund T Whittaker (1922). "On a new method of graduation". In: *Proceedings of the Edinburgh Mathematical Society* 41, pp. 63–75.
- Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang (July 2019). "Graph WaveNet for Deep Spatial-Temporal Graph Modeling". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, pp. 1907–1913. DOI: [10.24963/ijcai.2019/264](https://doi.org/10.24963/ijcai.2019/264). URL: <https://doi.org/10.24963/ijcai.2019/264>.
- Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang (2020). "Connecting the dots: Multivariate time series forecasting with graph neural networks". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA: Association for Computing Machinery, pp. 753–763.
- Yuankai Wu, Dingyi Zhuang, Aurelie Labbe, and Lijun Sun (2021). "Inductive Graph Neural Networks for Spatiotemporal Kriging". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 5, pp. 4478–4485.
- Zonghan Wu, Da Zheng, Shirui Pan, Quan Gan, Guodong Long, and George Karypis (2022). "Traversenet: Unifying space and time in message passing for traffic forecasting". In: *IEEE Transactions on Neural Networks and Learning Systems*.
- Zonghan Wu, Da Zheng, Shirui Pan, Quan Gan, Guodong Long, and George Karypis (2024). "TraverseNet: Unifying Space and Time in Message Passing for Traffic Forecasting". In: *IEEE Transactions on Neural Networks and Learning Systems* 35.2, pp. 2003–2013. DOI: [10.1109/TNNLS.2022.3186103](https://doi.org/10.1109/TNNLS.2022.3186103).
- Xinyi Wu, Yifei Wang, Stefanie Jegelka, and Ali Jadbabaie (2025). "On the Emergence of Position Bias in Transformers". In: *arXiv preprint arXiv:2502.01951*.
- Luo Xiao, Yingxing Li, and David Ruppert (2013). "Fast bivariate P-splines: the sandwich smoother". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 75.3, pp. 577–599.
- Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis (2024). "Efficient streaming language models with attention sinks". In: *International Conference on Learning Representations*.

- Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka (July 2018). "Representation Learning on Graphs with Jumping Knowledge Networks". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 5453–5462. URL: <https://proceedings.mlr.press/v80/xu18c.html>.
- Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka (2019). "How powerful are graph neural networks?" In: *International Conference on Learning Representations*.
- Vijaya Krishna Yalavarthi, Kiran Madhusudhanan, Randolph Scholz, Nourhan Ahmed, Johannes Burchert, Shayan Jawed, Stefan Born, and Lars Schmidt-Thieme (Mar. 2024). "GraFITi: Graphs for Forecasting Irregularly Sampled Time Series". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 38.15, pp. 16255–16263. DOI: [10.1609/aaai.v38i15.29560](https://doi.org/10.1609/aaai.v38i15.29560). URL: <https://ojs.aaai.org/index.php/AAAI/article/view/29560>.
- Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong (Aug. 2017). "Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 3861–3870. URL: <https://proceedings.mlr.press/v70/yang17b.html>.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le (2019). "Xlnet: Generalized autoregressive pretraining for language understanding". In: *Advances in neural information processing systems* 32.
- Yongchao Ye, Shiyao Zhang, and James JQ Yu (2021). "Spatial-temporal traffic data imputation via graph attention convolutional network". In: *International Conference on Artificial Neural Networks*. Springer, pp. 241–252.
- Xiuwen Yi, Yu Zheng, Junbo Zhang, and Tianrui Li (2016). "ST-MVL: Filling Missing Values in Geo-Sensory Time Series Data". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. IJCAI'16. AAAI Press, pp. 2704–2710. ISBN: 9781577357704.
- Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu (2021). "Do Transformers Really Perform Badly for Graph Representation?" In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and

- J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 28877–28888. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/f1c1592588411002af340cbaedd6fc33-Paper.pdf.
- Xueyan Yin, Feifan Li, Yanming Shen, Heng Qi, and Baocai Yin (2022). “Node-Trans: A Graph Transfer Learning Approach for Traffic Prediction”. In: *arXiv preprint arXiv:2207.01301*.
- Jinsung Yoon, James Jordon, and Mihaela van der Schaar (July 2018). “GAIN: Missing Data Imputation using Generative Adversarial Nets”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 5689–5698.
- Jiaxuan You, Xiaobai Ma, Yi Ding, Mykel J Kochenderfer, and Jure Leskovec (2020). “Handling Missing Data with Graph Representation Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 19075–19087.
- Fisher Yu and Vladlen Koltun (2016). “Multi-scale context aggregation by dilated convolutions”. In: *International Conference on Learning Representations*.
- Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon (2016). “Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction”. In: 29. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett.
- Bing Yu, Haoteng Yin, and Zhanxing Zhu (July 2018). “Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting”. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, pp. 3634–3640. doi: [10.24963/ijcai.2018/505](https://doi.org/10.24963/ijcai.2018/505).
- Bing Yu, Haoteng Yin, and Zhanxing Zhu (2019). “ST-UNet: A spatio-temporal U-network for graph-structured time series modeling”. In: *arXiv preprint arXiv:1903.05631*.
- Daniele Zambon, Cesare Alippi, and Lorenzo Livi (2018). “Concept Drift and Anomaly Detection in Graph Streams”. In: *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14. doi: [10.1109/TNNLS.2018.2804443](https://doi.org/10.1109/TNNLS.2018.2804443).
- Daniele Zambon, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi (2019). “Autoregressive Models for Sequences of Graphs”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8.

- Daniele Zambon, Cesare Alippi, and Lorenzo Livi (July 2020). "Graph Random Neural Features for Distance-Preserving Graph Representations". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 10968–10977. URL: <https://proceedings.mlr.press/v119/zambon20a.html>.
- Daniele Zambon, Michele Cattaneo, Ivan Marisca, Jonas Bhend, Daniele Nerini, and Cesare Alippi (2025). "PeakWeather: MeteoSwiss Weather Station Measurements for Spatiotemporal Deep Learning". In: *arXiv preprint arXiv:2506.13652*. URL: <https://arxiv.org/abs/2506.13652>.
- Daniele Zambon (2022). "Anomaly and Change Detection in Sequences of Graphs". PhD thesis. Università della Svizzera italiana.
- Günther Zängl, Daniel Reinert, Pilar Rípodas, and Michael Baldauf (2015). "The ICON (ICOsahedral Non-hydrostatic) modelling framework of DWD and MPI-M: Description of the non-hydrostatic dynamical core". In: *Quarterly Journal of the Royal Meteorological Society* 141.687, pp. 563–579. doi: <https://doi.org/10.1002/qj.2378>.
- Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna (2020). "GraphSAINT: Graph Sampling Based Inductive Learning Method". In: *International Conference on Learning Representations*.
- Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit Yan Yeung (2018). "GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs". In: *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*.
- Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li (2019). "T-gcn: A temporal graph convolutional network for traffic prediction". In: *IEEE transactions on intelligent transportation systems* 21.9, pp. 3848–3858.
- Xiang Zhang, Marko Zeman, Theodoros Tsiligkaridis, and Marinka Zitnik (2022). "Graph-Guided Network for Irregularly Sampled Multivariate Time Series". In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=Kwm8I7dU-15>.
- Yusong Zhang, Zhenyu Liu, Guodong Sa, Jiacheng Sun, Mingjie Hou, Yougen Huang, and Jianrong Tan (2025). "Virtual-Real Spatial-Temporal Dual Layer Transformer for virtual sensor state perception". In: *Computers in Industry* 169, p. 104288. ISSN: 0166-3615. doi: <https://doi.org/10.1016/>

- j . compind . 2025 . 104288. URL: <https://www.sciencedirect.com/science/article/pii/S0166361525000533>.
- Mengjie Zhao, Cees Taal, Stephan Baggerohr, and Olga Fink (2025). "Graph neural networks for virtual sensing in complex systems: Addressing heterogeneous temporal dynamics". In: *Mechanical Systems and Signal Processing* 230, p. 112544.
- Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang (2014). "Urban computing: concepts, methodologies, and applications". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 5.3, pp. 1–55.
- Yu Zheng, Xiuwen Yi, Ming Li, Ruiyuan Li, Zhangqing Shan, Eric Chang, and Tianrui Li (2015). "Forecasting fine-grained air quality based on big data". In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 2267–2276.
- Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi (2020). "GMAN: A graph multi-attention network for traffic prediction". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34, pp. 1234–1241.
- Yin-Cong Zhi, Yin Cheng Ng, and Xiaowen Dong (2023). "Gaussian Processes on Graphs Via Spectral Kernel Learning". In: *IEEE Transactions on Signal and Information Processing over Networks* 9, pp. 304–314. doi: [10.1109/TSIPN.2023.3265160](https://doi.org/10.1109/TSIPN.2023.3265160).
- Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf (2003). "Learning with Local and Global Consistency". In: *Advances in Neural Information Processing Systems*. Ed. by S. Thrun, L. Saul, and B. Schölkopf. Vol. 16. MIT Press. URL: https://proceedings.neurips.cc/paper_files/paper/2003/file/87682805257e619d49b8e0dfdc14affa-Paper.pdf.
- Weida Zhong, Qiuling Suo, Xiaowei Jia, Aidong Zhang, and Lu Su (2021). "Heterogeneous spatio-temporal graph convolution network for traffic forecasting with missing values". In: *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, pp. 707–717.
- Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang (2021). "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting". In: *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*. Vol. 35. 12. AAAI Press, pp. 11106–11115.

Patrick Zippenfenig (2023). [Open-Meteo.com Weather API](#). DOI: [10.5281/zenodo.7970649](#). URL: <https://open-meteo.com/>.