Lagged Spatiotemporal Covariance Neural Networks

LAGGED SPATIOTEMPORAL COVARIANCE NEURAL NETWORKS

Thesis

to obtain the degree of Master in Computer Science with Specialization in Data Science and Technology at Delft University of Technology,
to be publicly defended on Monday, June 23rd 2025 at 15:00

by

Athanasios GEORGOUTSOS
Born in Athens, Greece.

Multimedia Computing Group,
Faculty of Electrical Engineering, Mathematics and Computer Science (Faculteit Elektrotechniek, Wiskunde en Informatica),
Delft University of Technology,
Delft, the Netherlands.

Thesis committee:

Chair and Advisor:	Dr. E. Isufi, Faculty EEMCS, TU Delft
Committee Member:	Dr. M. Gürel, Faculty EEMCS, TU Delft
Advisor:	A. Cavallo, Faculty EEMCS, TU Delft

Keywords: Covariance Neural Networks · Graph Neural Networks · Graph Signal Processing · Multivariate Time Series · Principal Component Analysis

Summary

Multivariate time series arise in a wide range of domains, such as weather forecasting and financial modeling, where multiple interdependent variables evolve simultaneously over time. For instance, temperature readings at one location may have a delayed influence on nearby regions, while currency exchange rates exhibit complex, lagged interactions across global financial markets Effectively modeling these spatiotemporal interactions, particularly in streaming and non-stationary settings, remains a fundamental challenge. Traditional approaches such as Temporal PCA operate on sample covariance matrices but often suffer from stability issues in the estimated eigenvectors, especially in low-data regimes or when the corresponding eigenvalues are close. These covariance estimation errors can propagate into the learned representation and degrade performance in downstream tasks. Recent graph-based learning methods address this limitation by constructing graphs from the sample covariance matrix and learning from its structure. However, these approaches typically consider only lag-zero correlations, thereby limiting their ability to model cross-temporal dependencies and fully capture the spatiotemporal structure inherent in multivariate time series. To overcome this, this thesis proposes the Lagged spatiotemporal coVariance Neural Network (LVNN), a neural network architecture that leverages lagged covariance information to learn representations from multivariate time series in a streaming setting. LVNN constructs a spatiotemporal graph by concatenating consecutive temporal samples, computing their extended sample covariance matrix, and using it as a structural prior for graph convolutions. This design enables the model to capture variable interactions not only within time steps but also across temporal lags. However, the use of a larger spatiotemporal covariance matrix introduces additional computational overhead and spurious correlations. To address this, we introduce two structural modifications to our proposed model. First, we retain only spatial and backward temporal connections, corresponding to the block upper triangular part of the extended covariance matrix. Second, we apply thresholding-based sparsification techniques to prune weak correlations and improve scalability. We begin by proving that LVNN is robust to perturbations in the online estimation of the extended covariance matrix in stationary settings, improving over the stability issues of temporal PCA-based methods. These findings are empirically validated on synthetic stationary datasets. Then, to assess the effectiveness of the learned embeddings, we evaluate LVNN on single-step forecasting tasks using three real-world datasets across different forecasting horizons. The standard LVNN model performs comparably to our baselines, while the LVNN variant with the block upper triangular matrix demonstrates the most consistent performance. Furthermore, applying hard and soft thresholding sparsification techniques to the extended covariance matrix substantially reduces the computational overhead, with only a minor impact on forecasting performance. These results support our hypothesis that cross-temporal covariance terms are a valuable source of inductive bias for representation learning in multivariate time series.

ACKNOWLEDGEMENTS

As this academic journey comes to a close, I would like to take a moment to thank all those who contributed to its completion and made it an unforgettable experience, one that has shaped me both personally and professionally.

First, I would like to thank my supervisor, Elvin, for his guidance and support over the past nine months. Thank you for trusting me with this project from day one, for providing structure to help ensure its timely completion, and for your insightful suggestions that continually pushed me to deliver the best work I could. I would also like to thank my daily supervisor, Andrea, for his unwavering support throughout this project, his meticulous feedback on my writing, and his encouragement of my research ideas. Beyond laying the foundation for this thesis through your own research, our weekly discussions often felt as productive as an entire week of independent effort. Your contribution to this project has been invaluable. Thank you again for your dedication and guidance. Furthermore, I would like to thank Merve Gürel for accepting to be part of my thesis committee.

Next, I would like to thank all my friends who have been with me throughout this journey. First, to the friends I've made here in the Netherlands over the past two years, getting to know such extraordinary people has truly been one of the greatest joys of my life. I'm confident that our connection will extend well beyond these academic years. You made this adventure infinitely more enjoyable, and it simply wouldn't have been the same without you. I would also like to thank my friends from Greece, who have been by my side for most of my life. I cannot put into words how much you mean to me or how profoundly you have influenced my path. A special thanks goes to Giorgos, whom I've known for as long as I can remember and who has always been an integral part of my life, and to Spiros, with whom I've shared the majority of my academic life and his support and friendship have played a major role in shaping the person I am today. Finally, at the intersection of these two groups, I would like to thank my friend Danai, whom I've known for many years in Greece and who preceded me in Delft for her PhD. Thank you for your friendship through the highs and lows, and for patiently listening to me complain for hours about just about everything. These past two years abroad would have felt completely different without you here.

Most importantly, I would like to thank my family, my parents and my sister, for their unconditional love and unwavering belief in me. Words cannot truly express how much you mean to me. I have grown into the person I am today thanks to your tireless support and the countless sacrifices you've made to provide me with everything I could ever need. Whatever I have achieved so far, and whatever I may accomplish in the future, I owe it all to you.

Contents

1 Introduction

High-dimensional multivariate time series data are central to a wide range of scientific and real-world applications, where multiple interdependent variables evolve simultaneously over time. These data are commonly encountered in fields such as neuroscience, where techniques like fMRI capture temporal activity across hundreds of brain regions [129, 128, 127, 125, 124]; in finance, where asset prices, market indicators, and macroeconomic variables can be modeled jointly [85, 51, 141] (Figure 1.1); and in weather forecasting, where spatially distributed sensors record meteorological measurements [65, 82]. The complexity of multivariate time series lies not only in their temporal dimension but also in the intricate and often unknown inter-variable dependencies. Moreover, such data may evolve rapidly, exhibit non-stationary behavior, and become available in a streaming fashion, adding further challenges to modeling. Unlike data with an explicit spatial or relational structure, multivariate time series often lack a clear underlying topology, making it difficult to capture meaningful interactions among variables [19]. This structural uncertainty poses a major challenge for traditional time series models, which typically assume predefined dependencies or rely on simplistic assumptions of variable independence.

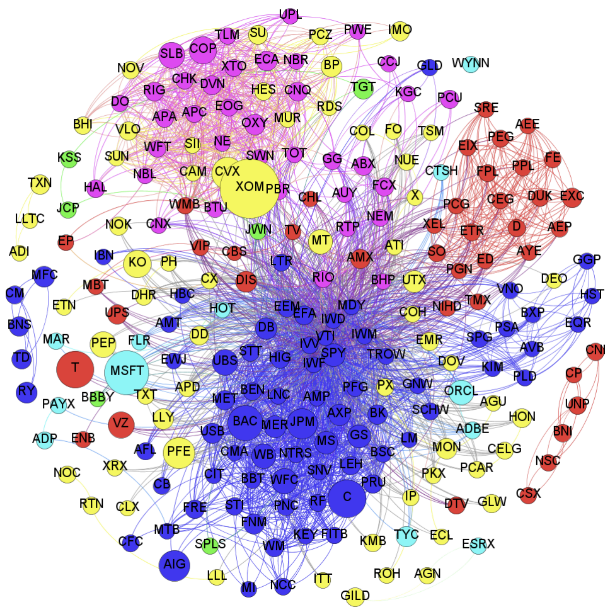

UPL
TLM SU PCZ PWE IMO
SLB COP XTO ECA NBR BP CCJ GLD WYNN
NOV CHK DVN HES CNQ TGT
RIG DO APC EOG RDS KGC PCU
APA OXY MUR SRE
BHI VLO NE COL FO TSM AEE
KSS SUN WFT SII SWN TOT EIX PEG
NBL CAM CVX GG ABX NUE CTSH FPL PPL FE
TXN HAL XOM PBR CHL AUY FCX X
JCP WMB BTU PCG CEG DUK EXC
LLTC NOK CNX NEM XEL ETR D AEP
JWN TV RTP MT
ADI EP VIP CBS ATI SO AYE
IBN KO PH LTR AMX ED GGP
MFC MBT CX DIS RIO BHP UTX PGN DEO
CM ETN DHR HBC VNO HST
UPS HOT EEM EFA MDY COH NIHD TMX BXP
BNS AMT IWD PSA EQR
MAR PEP FLR DD DB IVV TI IWM EMR SPG AVB CNI
TD STT SPY
T MSFT EWJ UBS HIG IWF TROW DOV KIM PLD CP
RY BBBY APD PFG PX GNW ORCL
PAYX TXT MET BEN LNC AMP AXP AGU UNP
VZ LLY BK SCHW HON
ADP ENB USB BAC MER JPM ADBE BNI
AFL PFE MS GS LM MON CELG
NOC XRX CMA WB NTRS BSC LEH NSC
CB CIT BBT WFC SNV PKX PCAR
RTN CLX FRE STI PNC RF C PRU IP DTV GLW CSX
MTB FNM
CFC SPLS WM KEY FITB KMB TYC ECL ESRX
AIG LLL
MI NCC ITT ROH AGN
GILD

Figure 1.1: Visualization of a financial network, where nodes represent stocks and edges are correlations higher than 0.6 of daily price log returns, adapted from [136]

To explicitly capture the inter-dependencies in high-dimensional multivariate time series, a suitable starting point is the sample covariance matrix, which captures pairwise relationships between variables [19, 28, 9]. This matrix provides a data-driven representation of the underlying structure and is commonly used as the basis for dimensionality reduction techniques such as Principal Component Analysis (PCA) [78]. PCA projects high-dimensional data onto a lower-dimensional subspace that preserves maximal variance, yielding compact representations that can facilitate modeling and analysis. However, when applied to time series data, PCA exhibits several important limitations. First, it does not account for temporal dependencies, treating time-indexed observations as independent and identically distributed. Second, PCA typically operates in a batch setting, which restricts its applicability in streaming scenarios. Third, closely spaced eigenvalues can lead to instabilities in the principal components and, subsequently, to unreliable representations and sensitivity to noise [31, 79].

Temporal PCA [78, Chapter 12.2] and online PCA variants [29] have been proposed to address the first two challenges respectively. In particular, Temporal PCA concatenates consecutive temporal samples and projects them on the eigenspace of their extended covariance matrix. In the block structure view of this matrix, the diagonal blocks correspond to lag-zero covariances, while the off-diagonal blocks represent covariances at different time lags. Then, to overcome the instability to close eigenvalues, recent work has drawn on advances in graph neural networks (GNNs), a class of neural networks designed to process data represented as graphs by aggregating and transforming information from a node's local neighborhood. Two key properties of GNNs are their stability to perturbations in the underlying topology [47] and their transferability to graphs of different sizes [115]. The coVariance Neural Networks (VNN) [126], designed for static tabular data, leverage the covariance matrix as a graph representation matrix, showcasing enhanced stability to perturbations in the finite sample covariance matrix in comparison to PCA. Because the sample covariance matrix is typically dense and associated with high computational costs, the VNN paradigm has also been extended with sparsification techniques in [30].

Building upon the VNN framework, the Spatio-Temporal coVariance Neural Network (STVNN) introduces a unified approach for learning from multivariate time series in a streaming setting, extending the stability advantages of VNNs [31]. STVNN performs independent spatial convolutions across variables at each time step using the sample covariance matrix and, then, aggregates the resulting embeddings across time via a temporal sum. The covariance matrix and model parameters are updated online, accounting for the streaming data and possible distribution shifts. This design enables the model to capture spatiotemporal dependencies while maintaining computational efficiency, avoiding the high complexity associated with constructing and decomposing the extended covariance matrices required in approaches like Temporal PCA. However, it may fail to capture dynamic patterns that involve lagged or time-shifted interactions between variables, such as delayed causal effects or cross-temporal covariance terms. In many real-world systems, interactions between variables are not synchronous; neural signals in one brain regions may influence others with a short temporal delay [145], stock prices may respond to market indicators over various time lags [37, 24], and weather patterns often exhibit causal dependencies across hours or days [25]. This limits its expressiveness in domains where the spatiotemporal dynamics are tightly intertwined, motivating the need for models that explicitly encode both spatial and temporal dependencies.

We argue that a natural next step towards capturing richer spatiotemporal dynamics is to also account for these lagged relationships, which are encoded in the extended sample covariance matrix. Incorporating such information would offer a more complete structural view of the data, enabling models to detect patterns that are distributed not just across variables but also across time. By doing so, one can capture cross-temporal dynamics that would otherwise be lost when relying solely on the lag-zero sample covariance matrix. This motivates the use of the extended sample covariance matrix as a foundation for learning more intricate spatiotemporal embeddings in multivariate time series modeling.

Therefore, in this thesis, we investigate how to learn from multivariate time series data by effectively utilizing the extended sample covariance matrix in the GNN framework, answering the research question:

(RQ) *"How can we learn online meaningful embeddings from multivariate time series by leveraging lagged covariance information as part of the GNN architecture?"*

We propose a Lagged spatiotemporal coVariance Neural Network (LVNN) that builds upon the GNN architecture principles and uses the extended sample covariance matrix as a graph representation matrix. Analogous to STVNN, the extended sample covariance matrix and model parameters are updated in an online manner. Since uncertainties in the online updates could lead to instabilities in the output of LVNN, we first ask the following question:

(RQ1) *"How robust is the proposed model to perturbations introduced by the online estimation of the extended covariance matrix and model parameters?"*

We theoretically derive an upper bound for the stability of LVNN in Section 4.3 and compare it with the corresponding bound for Temporal PCA. In Section 5.2.1, we corroborate our theoretical analysis with stability experiments in stationary and non-stationary settings. After establishing the stability of LVNN, we ask the question:

(RQ2) *"How effective are the learned embeddings from the proposed model when applied to a downstream task, such as single-step forecasting?"*

To answer this question, we begin by analyzing the convolution operations of LVNN on the extended sample covariance matrix in Section 4.2. In Section 5.2.2, we evaluate our model on three real-world datasets: the Molene [56] and NOAA [4] datasets, containing temperature recordings, and the Exchange Rate [156] dataset, with the daily exchange rates of 8 countries' currencies. Finally, given the increased computational cost of processing the extended sample covariance matrix, we ask the question:

(RQ3) *"How can we make the model more computationally efficient and potentially robust to spurious correlations via thresholding sparsification?"*

We adapt hard and soft thresholding strategies from [30] for the online sparsification of the extended sample covariance matrix. In Section 5.2.3, we empirically evaluate the impact of these thresholding techniques on the forecasting performance and computational time of LVNN.

By answering the research questions, this thesis provides the following contributions:

(C1) **Model Architecture.** We introduce LVNN, a spatiotemporal GNN for multivariate time series, which operates with online learning updates and employs the extended sample covariance matrix as a graph shift operator.

(C2) **Stability Analysis.** We theoretically prove that LVNN (with and without thresholding sparsification) provides stability to perturbations in the finite sample estimate of the extended covariance matrix and model parameters in stationary settings.

(C3) **Empirical Evaluation.** We conduct experiments on a synthetic stationary dataset and three real-world datasets, focusing on corroborating the stability of LVNN, evaluating its forecasting performance against a set of baseline models, and quantifying the impact of thresholding sparsification techniques on the model.

The remainder of this thesis proceeds as follows. Chapter 2 covers the necessary background information for the research carried out in this thesis. Chapter 3 discusses the relevant literature for multivariate time series modeling, with a focus on covariance-based approaches. Chapter 4 introduces the proposed LVNN architecture, leveraging the extended sample covariance matrix, and its stability analysis. Chapter 5 contains the numerical experiments and Chapter 6 concludes this thesis and lays down some future research directions.

Notation. Throughout this thesis, the following notation is adopted. Scalars are denoted by plain letters (i.e., a, A), vectors by lowercase boldface letters (i.e., $\mathbf{a}$), and matrices by uppercase boldface letters (i.e., $\mathbf{A}$). The i-th entry of vector $\mathbf{a}$ is denoted by a_i or $[\mathbf{a}]_i$ and, likewise, the (i, j)-th entry of matrix $\mathbf{A}$ is denoted by A_{ij} or $[\mathbf{A}]_{ij}$. $\mathbf{A}^\top$ denotes the transpose of a matrix, $\hat{\mathbf{A}}_t$ denotes the sample estimate of a matrix, where, depending on the context, t could indicate the time step or the number of samples, and $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{C}}$ denote the extended data sample vector and the extended covariance matrix respectively.

2 Background

In this chapter, we introduce the background material necessary for the research carried out in this thesis. Starting in the domain of static data, Section 2.1 presents Principal Component Analysis, elaborating on its role in dimensionality reduction and data decomposition. This acts as the basis for understanding the mechanisms behind coVariance Neural Networks in Section 2.2, a specific Graph Neural Network architecture operating on covariance matrices. Then, we move to the temporal domain, by introducing time series in Section 2.3 and discussing statistical and principal component-based techniques for modeling them in Section 2.4. Finally, we discuss different Spatio-Temporal Graph Neural Network approaches for modeling spatial and temporal dependencies in Section 2.5.

2.1. Principal Component Analysis

Principal Component Analysis (PCA) is a powerful statistical method for dimensionality reduction, feature extraction, and data visualization. It transforms a dataset consisting of correlated variables into a new set of uncorrelated variables called *principal components* [79]. By simplifying high-dimensional data into fewer components, PCA aids in identifying dominant patterns and removing redundancy, making it an invaluable tool in modern data analysis [122].

Consider a N-dimensional random vector $\mathbf{x} \in \mathbb{R}^{N \times 1}$ and its corresponding *true covariance* matrix:

$$\mathbf{C} = \mathbb{E}\left[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{x} - \mathbb{E}[\mathbf{x}])^\top\right] \tag{2.1}$$

In practice, the underlying data model $\mathbf{x}$ and its statistics, including the true covariance matrix $\mathbf{C}$, are not available. At our disposal, we have a dataset $\hat{\mathbf{X}}_n \in \mathbb{R}^{N \times n}$ with n random, independent and identically distributed (i.i.d.) samples of $\mathbf{x}$, given by $\mathbf{x}_i \in \mathbb{R}^{N \times 1}, \forall i \in \{1, ..., n\}$. After shifting its feature columns, so that they exhibit a zero mean, we can use this dataset to compute an estimate of the true covariance matrix, called *sample covariance* matrix:

$$\hat{\mathbf{C}}_n = \frac{1}{n}\hat{\mathbf{X}}_n\hat{\mathbf{X}}_n^\top \tag{2.2}$$

Using the sample covariance matrix, the next step is to obtain the orthogonal matrix $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_N]$, with its columns corresponding to *eigenvectors*, through the eigendecomposition of $\hat{\mathbf{C}}_n$ as:

$$\hat{\mathbf{C}}_n = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top \tag{2.3}$$

where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, ..., \lambda_N)$ is the diagonal matrix of *eigenvalues*, such that $\lambda_1 \geq \lambda_2 ... \geq \lambda_N$. PCA projects the data in the eigenvector space of $\hat{\mathbf{C}}_n$ with the following transformation:

$$\hat{\mathbf{Y}}_n = \mathbf{V}^\top\hat{\mathbf{X}}_n \tag{2.4}$$

where $\hat{\mathbf{Y}}_n$ is the final representation of the n samples in the space determined by the N eigenvectors of $\mathbf{V}$ [78].

The essence of PCA lies in finding a new coordinate system, whose directions would maximize the variance of the data. These directions, or principal components, are represented by the set of pairwise orthogonal eigenvectors. These eigenvectors are ordered in decreasing order, based on the percentage of the original variance they capture, which is described by the corresponding eigenvalues. The application of PCA can be thought of as a rotation of the data to align with these new axes, so that each axis represents a meaningful dimension of variation in the data. By focusing on the first k eigenvectors, we can reduce the dimensionality of the data while preserving its core characteristics. The resulting k-dimensional space would be able to explain a percentage of the original variance corresponding to the cumulative explained variance of the top k eigenvectors. The aim of PCA is to select as few eigenvectors as possible, while retaining as much information as possible [79].

2.2. Covariance-Based Graph Neural Networks

PCA provides a foundation for understanding the structure of multivariate data by analyzing its covariance matrix, revealing dominant modes of variation. Extending this idea, covariance-based methods have been explored in graph-based learning. By representing the N dimensions of the random vector $\mathbf{x}$ as the nodes of an undirected graph, the sample covariance matrix $\hat{\mathbf{C}}_n$ serves as an alternative to traditional adjacency matrices. This perspective naturally leads to coVariance Neural Networks [126], a class of Graph Neural Networks that leverage covariance structures as Graph Shift Operators for learning representations and processing multivariate data.

2.2.1. GRAPH FILTERS AND COVARIANCE FOURIER TRANSFORM

Consider an undirected, unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a node set $\mathcal{V} = \{1, ..., N\}$ and an edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and a graph signal $\mathbf{x} \in \mathbb{R}^N$ formed by the data associated with the graph's nodes. The structure of graph $\mathcal{G}$ can be represented by a *Graph Shift Operator* (GSO) $\mathbf{S} \in \mathbb{R}^{N\times N}$ respecting the graph's topology, meaning that off-diagonal entries for not directly connected nodes are zero. The graph signal $\mathbf{x}$ can be shifted over the nodes via the GSO $\mathbf{S}$, so that the i-th entry of the output $\mathbf{Sx}$, combining information of $\mathbf{x}$ from node i and its neighbors, is:

$$[\mathbf{Sx}]_i = \sum_{j=1}^{N} [\mathbf{S}]_{ij} [\mathbf{x}]_j = \sum_{j \in \mathcal{N}_i} s_{ij} x_j \tag{2.5}$$

where $\mathcal{N}_i$ is the one-hop neighborhood of node i. By introducing a set of real valued parameters $\mathbf{h} = [h_0, ..., h_K]^\top$, the concept of graph signal shifting can be extended to the linear shift-and-sum operation of *graph convolution*, which is defined as:

$$\mathbf{z} = \mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x} \tag{2.6}$$

where $\mathbf{H}(\mathbf{S})$ constitutes a *graph convolutional filter*. Graph convolution linearly combines information from different neighborhoods of the graph by performing K successive shifts of the graph signal with one-hop neighbors, weighted by the corresponding parameters [48]. An example of a graph convolution operation and visualizations of the underlying signal shifts are presented in Figure 2.1.

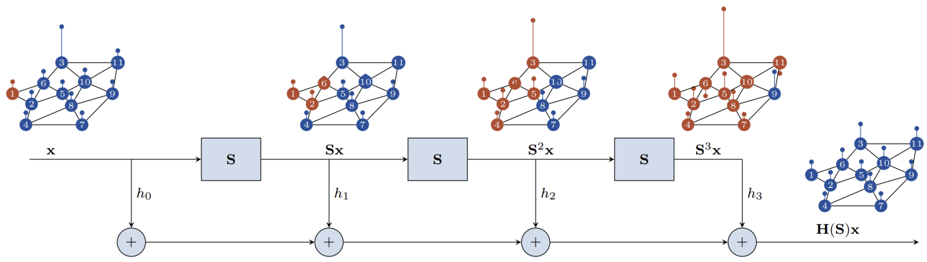


Figure 2.1: Visualization of a graph convolution operation, adapted from [72]

By analyzing graph convolutions in the spectral domain, one may gain additional insights into how the various signal components are processed. Similar to (2.3), the eigendecomposition of the GSO $\mathbf{S} = \mathbf{V\Lambda V}^\top$ leads to the orthogonal eigenvector matrix

$\mathbf{V} \in \mathbb{R}^{N\times N}$ and the diagonal eigenvalue matrix $\mathbf{\Lambda} \in \mathbb{R}^{N\times N}$. In the context of the graph frequency domain, the eigenvalues λ_i can be interpreted as *graph frequencies*, while the corresponding eigenvectors $\mathbf{v}_i$ are signals representing the *graph oscillating modes*, constituting the graph frequency basis [48]. The *Graph Fourier Transform* (GFT) operation projects any signal $\mathbf{x}$ in the eigenvector space of $\mathbf{S}$ through these graph oscillating modes:

$$\bar{\mathbf{x}} = \mathbf{V}^\top \mathbf{x} \tag{2.7}$$

Each entry $\bar{x}_i$ of the signal projection $\bar{\mathbf{x}}$ represents a *Fourier coefficient*, which is associated to graph frequency λ_i and measures the contribution of a graph oscillating mode $\mathbf{v}_i$ to the signal $\mathbf{x}$ [110, 48].

Now, consider the special setting where $\mathbf{S} = \hat{\mathbf{C}}_n$, so that the sample covariance matrix $\hat{\mathbf{C}}_n$ of the data $\hat{\mathbf{X}}_n$ acts as the GSO of $\mathcal{G}$. Then, the graph frequencies obtained by GFT are equivalent to the eigenvalues of the sample covariance matrix, and the GFT projection into the eigenspace of the GSO, in (2.7), is equivalent to the PCA projection into the eigenspace of the sample covariance matrix, in (2.4) [110]. This relation between PCA and GFT, with the sample covariance matrix as the GSO, is formalized with the notion of *coVariance Fourier Transform* (VFT), which is defined as:

$$\bar{\mathbf{x}} = \mathbf{V}^\top \mathbf{x} \tag{2.8}$$

This equation expresses the projection of a random sample $\mathbf{x}$ in the eigenspace of $\hat{\mathbf{C}}_n$, where entry $\bar{x}_i$ represents the i-th Fourier coefficient and is associated with eigenvalue w_i [126].

2.2.2. COVARIANCE FILTERS

Analogous to the definition of VFT, the filtering through the graph convolution operation in (2.6) can be extended to *coVariance Filters* (VF), polynomial functions applied to a covariance matrix [126]. Given a set of parameters $\mathbf{h} = [h_0, \ldots, h_K]^\top$, a VF for the sample covariance matrix $\hat{\mathbf{C}}_n$ is expressed as:

$$\mathbf{H}(\hat{\mathbf{C}}_n) = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}_n^k \tag{2.9}$$

Similarly to graph convolutional filters, VFs combine information in different neighborhoods of the graph in a weighted manner through their parameter set $\mathbf{h}$ [48]. For an input signal $\mathbf{x}$, the output of a VF is:

$$\mathbf{z} = \mathbf{H}(\hat{\mathbf{C}}_n)\mathbf{x} = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}_n^k \mathbf{x} \tag{2.10}$$

By leveraging the eigendecomposition of $\hat{\mathbf{C}}_n$ in (2.3), the orthonormality of eigenvectors, and the definition of VFT in (2.8), the VF output can be projected in the eigenspace of $\hat{\mathbf{C}}_n$ as:

$$\bar{\mathbf{z}} = \sum_{k=0}^{K} h_k \mathbf{\Lambda}^k \bar{\mathbf{x}} \tag{2.11}$$

This leads to the definition of a coVariance filter's frequency response over $\hat{\mathbf{C}}_n$:

$$h(\lambda_i) = \sum_{k=0}^{K} h_k \lambda_i^k \tag{2.12}$$

which determines how each eigenvalue λ_i is weighted, effectively shaping how different principal components contribute to the filtered signal. Because of the equivalence between graph frequencies and principal components, one can construct a narrowband VF, with a frequency response $h(\lambda) = 1$ if $\lambda = \lambda_i$ and $h(\lambda) = 0$ otherwise, that recovers the score for the projection of a data sample $\mathbf{x}$ on eigenvector $\mathbf{v}_i$, associated with the i-th principal component of $\hat{\mathbf{C}}_n$. By extending this idea, for a given covariance matrix $\hat{\mathbf{C}}_n$, it is possible to select a filter bank of narrowband VFs, with carefully selected filter coefficients, which can recover the PCA transformation (*Theorem 1* in [126]).

A significant advantage of using coVariance filters over direct PCA computation is their robustness to perturbations in the sample covariance matrix $\hat{\mathbf{C}}_n$. Since $\hat{\mathbf{C}}_n$ is estimated from a finite dataset, it is subject to statistical uncertainties, leading to deviations from the true covariance matrix $\mathbf{C}$. To study the stability of VFs under the presence of this type of perturbations, given a random data sample $\mathbf{x}$, consider the comparison of the outputs $\mathbf{H}(\mathbf{C})\mathbf{x}$ and $\mathbf{H}(\hat{\mathbf{C}}_n)\mathbf{x}$. It is proven that the distance $\left\|\mathbf{H}(\hat{\mathbf{C}}_n) - \mathbf{H}(\mathbf{C})\right\|$ between the two filters decays with the number of samples n in the dataset, at least at the rate of $1/n^{\frac{1}{2}-\epsilon}$, where $\epsilon \in (0, 1/2]$ (*Theorem 2* in [126]), as:

$$\left\|\mathbf{H}(\hat{\mathbf{C}}_n) - \mathbf{H}(\mathbf{C})\right\| = \frac{M}{n^{\frac{1}{2}-\epsilon}} \cdot \mathcal{O}(\sqrt{N} + \frac{\|\mathbf{C}\| \sqrt{\log N n}}{k_{min} n^{2\epsilon}}), \tag{2.13}$$

where k_{min} is the minimum among the kurtoses of the distribution of $\mathbf{x}$ in the direction of each eigenvector in $\mathbf{V}$, for some constant $M > 0$. This means that the stability of coVariance filters increases as n increases, since the estimate $\hat{\mathbf{C}}_n$ is expected to converge to $\mathbf{C}$, in accordance with the law of large numbers [126].

This observation about the stability of a coVariance filter $\mathbf{H}(\mathbf{C})$ relies on the assumption that its frequency response satisfies the following relation:

$$\left|h(\lambda_i) - h(\lambda_j)\right| \leq M \frac{\left|\lambda_i - \lambda_j\right|}{k_i} \tag{2.14}$$

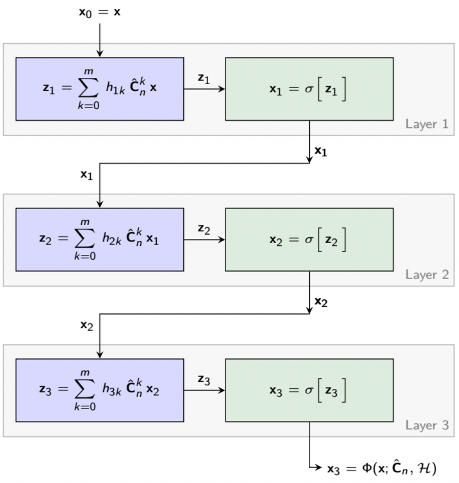
where k_i is a measure of kurtosis of the distribution of $\mathbf{x}$ in the direction of $\mathbf{v}_i$, and $\lambda_i \neq \lambda_j$ are non-zero eigenvalues of $\mathbf{C}$. From perturbation theory of eigenvectors and eigenvalues of sample covariance matrices, it is known that estimates of the eigenspaces become harder to distinguish as $\left|\lambda_i - \lambda_j\right|$ decreases [100]. Based on (2.14), the distance between the frequency responses of two VFs for non-zero eigenvalues λ_i and λ_j is limited by the corresponding eigengap $\left|\lambda_i - \lambda_j\right|$. Therefore, a VF sacrifices discriminability between close eigenvalues by restraining the corresponding frequency responses to be close as well, in order to gain stability with respect to the statistical uncertainty of the sample covariance matrix. Then, this assumption also accounts for the fact that distributions with higher kurtosis k_i have heavier tails and more outliers, impeding the estimation of the corresponding eigenvalue λ_i and eigenvector $\mathbf{v}_i$ [100]. Observing the upper bound in (2.14), a higher kurtosis k_i further restricts the expressivity of the VF's frequency response for eigenvalue λ_i, while a smaller kurtosis provides more flexibility, since it is associated with a faster decaying tail and a more accurate estimation.

2.2.3. COVARIANCE NEURAL NETWORKS

Building on the concept of coVariance filters, a *coVariance Neural Network* (VNN) provides an end-to-end, non-linear, parametric mapping from input data $\mathbf{x}$ to a specified objective $\mathbf{r}$, defined as:

$$\mathbf{r} = \Phi(\mathbf{x}; \hat{\mathbf{C}}_n, \mathcal{H}) \tag{2.15}$$

for a sample covariance matrix $\hat{\mathbf{C}}_n$ and a set of filter coefficients $\mathcal{H}$ [126]. VNNs leverage structured filtering operations to extract meaningful representations. A VNN consists of multiple layers, each containing two key components: (i) a filter bank composed of coVariance filters and (ii) a pointwise non-linear activation function. The architecture resembles that of graph neural networks, with the sample covariance matrix serving as a shift operator.



$x_0 = x$
$z_1 = \sum_{k=0}^{m} h_{1k} \hat{C}_n^k x$
z_1
$x_1 = \sigma[z_1]$
Layer 1
x_1
x_1
$z_2 = \sum_{k=0}^{m} h_{2k} \hat{C}_n^k x_1$
z_2
$x_2 = \sigma[z_2]$
Layer 2
x_2
x_2
$z_3 = \sum_{k=0}^{m} h_{3k} \hat{C}_n^k x_2$
z_3
$x_3 = \sigma[z_3]$
Layer 3
$x_3 = \Phi(x; \hat{C}_n, \mathcal{H})$

Figure 2.2: A 3-layer VNN architecture, adapted from [126]

The fundamental building block of a VNN is the *coVariance perceptron* [126], defined as a single-layer transformation applied to an input signal $\mathbf{x}$:

$$\Phi(\mathbf{x}; \hat{\mathbf{C}}_n, \mathcal{H}) = \sigma(\mathbf{H}(\hat{\mathbf{C}}_n)\mathbf{x}) \tag{2.16}$$

By stacking multiple perceptron layers, a deeper VNN is constructed, enabling more expressive representations. An example of a 3-layer VNN architecture can be seen in Figure 2.2. To further enhance representational power, VNNs incorporate filter banks, allowing multiple parallel inputs and outputs per layer. For a layer with F_{in} input channels and F_{out} output channels, each input is processed by F_{out} coVariance filters in parallel, producing multiple filtered outputs that undergo non-linear transformations. This leads to a layer containing $F_{in} \times F_{out}$ coVariance filters in total, with increased flexibility in capturing complex dependencies in the data.

Following from the stability of coVariance filters, it is desirable that VNNs are also stable, regarding perturbations in the sample covariance matrix. The stability of a VNN extends from the stability properties of coVariance filters [126] and the stability properties of GNNs [48, 47], ensuring robustness to small changes in the input (*Theorem 3* in [126]):

$$\left\| \Phi(\mathbf{x}; \hat{\mathbf{C}}_n, \mathcal{H}) - \Phi(\mathbf{x}; \mathbf{C}, \mathcal{H}) \right\| \leq L F^{L-1} \alpha_n \tag{2.17}$$

where L is the number of layers, F is the number of inputs and outputs per layer, and $\alpha_n > 0$ represents the finite sample effect on the perturbations in $\hat{\mathbf{C}}_n$ with respect to $\mathbf{C}$. As the number of layers L and the number of input-output channels per layer F increase, the stability of VNNs decreases and their sensitivity to perturbations may grow. For a coVariance perceptron, the computational cost scales as $\mathcal{O}(N^2 K F_{in} F_{out})$. In high-dimensional settings, enforcing sparsity in the covariance structure significantly reduces computational demands, making VNNs more scalable [11].

2.3. INTRODUCTION TO TIME SERIES

A discrete time series is a sequence of data points indexed by time, typically recorded at regular intervals [32]. Formally, a *univariate time series* can be defined as a sequence of observations:

$$\mathbf{x} = \{x_t : t = 1, 2, \ldots, T\} \in \mathbb{R}^T \tag{2.18}$$

where T is the total number of time steps and $x_t \in \mathbb{R}$ represents the value of the time series at time t. This definition can be naturally extended in the presence of multiple variables. A *multivariate time series* consists of observations of multiple interdependent variables over the same time period:

$$\mathbf{X} = \{\mathbf{x}_t : t = 1, 2, \ldots, T\} \in \mathbb{R}^{N \times T} \tag{2.19}$$

where $\mathbf{x}_t = [x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(N)}] \in \mathbb{R}^N$ represents an N-dimensional vector with the values of observed variables at time t. Unlike independent data samples, time series exhibit temporal dependencies, meaning that observations are inherently linked across time, making their analysis and modeling distinct from traditional data-driven tasks [17].

The main goal of time series modeling is to extract meaningful insights and represent the underlying structure of temporal data, enabling tasks such as time series forecasting, anomaly detection, imputation and classification [75]. *Forecasting* involves predicting future values of a time series based on its historical observations. The forecasting horizon $\tau \in \mathbb{N}$, determining the number of steps into the future for which predictions are made, distinguishes short-term forecasts (i.e. for scheduling needs), medium-term forecasts (i.e. for future resource requirements), and long-term forecasts (i.e. for strategic planning) [68]. *Anomaly detection* identifies outliers or other types of irregularities in time series data, often used for fraud detection [46] or fault monitoring in industrial systems [106]. *Data Imputation* focuses on reconstructing missing time series values while preserving temporal consistency, allowing the utilization of initially incomplete time series. *Classification* assigns labels to entire time series or sequences of observations based on underlying patterns and characteristics [75], aiding in tasks such as disease detection from physiological signals [114].

Despite its importance, time series modeling presents several challenges. Many time series exhibit non-stationarity, where statistical properties such as mean and variance change over time, requiring specialized methods to handle trends and seasonality. Noise and random fluctuations can obscure meaningful patterns, complicating analysis. In the multivariate setting, capturing dependencies between multiple variables adds an additional layer of complexity, as relationships can evolve dynamically over time. Furthermore, practical challenges such as missing values, irregular sampling rates, and data sparsity often arise, necessitating advanced techniques to ensure reliable modeling [68].

To address these challenges, a variety of methods have been developed. Statistical models rely on mathematical formulations to identify underlying patterns, offering interpretable and computationally efficient solutions [68]. Machine learning techniques provide greater flexibility by learning complex, non-linear relationships with minimal manual feature engineering [50]. Graph-based approaches further extend time series modeling by leveraging structured representations to capture both temporal dynamics and interdependencies between variables, making them particularly useful for multivariate and spatiotemporal data [75]. These diverse methodologies enable robust time series modeling, supporting decision-making across a wide range of applications.

2.4. Time Series Modeling

Modeling time series data is essential for capturing temporal patterns and extracting meaningful patterns for prediction and analysis. Unlike conventional datasets, time series exhibit ordered dependencies, meaning that observations are not independent but influenced by past values. Effective models must account for characteristics such as trends, seasonality, and autocorrelations, while also addressing challenges like missing values, noise, and structural shifts over time. Statistical approaches remain widely used due to their interpretability and strong theoretical foundation, providing insights into underlying data dynamics. These models serve as the basis for more advanced machine learning and deep learning techniques, which aim to capture complex temporal relationships beyond linear dependencies.

One of the most widely used statistical techniques for time series modeling is *Exponential Smoothing*, which assumes that future values are weighted averages of past observations, with exponentially decreasing weights assigned more distant observations [20]. Variants such as Holt's method [67] extend this approach to capture trends, while Holt-Winters method [67, 152] additionally incorporates seasonal components. Another fundamental class of models is the *AutoRegressive Integrated Moving Average* (ARIMA) family, which focuses on capturing linear relationships within the data, using autoregression (AR) terms, moving average (MA) terms, and differencing operations to ensure stationarity [15]. The Vector ARIMA (VARIMA) model extends ARIMA to multivariate time series by capturing interactions among multiple variables. Due to their interpretability and effectiveness, these statistical models remain widely used in settings with stationary or well-structured time series. However, they may struggle with highly nonlinear time series or long-range dependencies [18].

2.4.1. TEMPORAL PCA

As discussed in Section 2.1, PCA is a widely used dimensionality reduction technique that projects high-dimensional data onto a lower-dimensional subspace while preserving the maximum variance. However, traditional PCA assumes that the available dataset consists of i.i.d. samples of the underlying data distribution. For settings with non-independent data, such as time series with inherent temporal dependencies, the PCA framework has been extended to *Temporal PCA* (TPCA) [78]. Instead of focusing solely on spatial relationships between the variables, TPCA also captures temporal and spatio-temporal relationships, accounting for the special nature of time series.

At time t, consider the extended data sample $\tilde{\mathbf{X}}_t = [\mathbf{x}_t^\top, \mathbf{x}_{t-1}^\top, \ldots, \mathbf{x}_{t-T+1}^\top]^\top \in \mathbb{R}^{N \times T}$, given by the concatenation of the latest T samples of the N-dimensional time series, under a stationary setting. Then, based on this extended data sample, we can compute the extended sample covariance matrix:

$$\tilde{\mathbf{C}} = \mathbb{E}\left[(\tilde{\mathbf{X}}_t - \tilde{\mu})(\tilde{\mathbf{X}}_t - \tilde{\mu})^\top\right] \tag{2.20}$$

where $\tilde{\mu}$ is the sample mean of the extended data sample. Analogous to PCA, TPCA projects the data $\tilde{\mathbf{X}}_t$ in the eigenvector space of $\tilde{\mathbf{C}}$. To gain a better understanding of this eigenvector space, we can view the extended covariance matrix in a block structure:

$$\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C} & \mathbf{C}_1 & \ldots & \mathbf{C}_{T-1} \\ \mathbf{C}_1 & \mathbf{C} & \ldots & \mathbf{C}_{T-2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_{T-1} & \mathbf{C}_{T-2} & \ldots & \mathbf{C} \end{bmatrix} \tag{2.21}$$

under the assumption of stationarity. The diagonal matrices $\mathbf{C}$ are lag-zero covariance matrices, representing the covariance between variables at the same time point. The off-diagonal matrices $\mathbf{C}_\tau$ are lagged covariance matrices, capturing the covariance between variables at different times with a time lag τ:

$$\mathbf{C}_\tau = \mathbb{E}\left[(\mathbf{x}_t - \mu)(\mathbf{x}_{t-\tau} - \mu)^\top\right] \tag{2.22}$$

We expect the covariance matrices with the same time lag τ (including $\tau = 0$) to be equivalent, assuming stationarity, which is reflected in the block structure of the extended sample covariance matrix above. However, if the time series are not necessarily stationary, all the covariance matrices could be different, leading to this generic form of the extended sample covariance matrix:

$$\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C}_{0,0} & \mathbf{C}_{0,1} & \ldots & \mathbf{C}_{0,T-1} \\ \mathbf{C}_{1,0} & \mathbf{C}_{1,1} & \ldots & \mathbf{C}_{1,T-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_{T-1,0} & \mathbf{C}_{T-1,1} & \ldots & \mathbf{C}_{T-1,T-1} \end{bmatrix} \tag{2.23}$$

where the covariance matrix $\mathbf{C}_{k,l}$ captures the covariance between the variables at time k and time l.

A non-parametric method for performing TPCA is *Singular Spectrum Analysis* (SSA), which decomposes and analyzes the autocovariance of a 1-dimensional time series [78]. Initially, considering L lagged copies of the series, SSA constructs a trajectory matrix:

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \dots & x_{T-L+1} \\ x_2 & x_3 & \dots & x_{T-L+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_L & x_{L+1} & \dots & x_T \end{bmatrix} \tag{2.24}$$

Assuming stationarity, the (i, j)-th element of the corresponding covariance matrix depends only on $\tau = |i - j|$, and represents the autocovariance γ_τ of the series for a time lag τ. After the eigendecomposition of this covariance matrix, SSA reconstructs the series by selecting the most dominant eigenvalues, which represent main patterns of the series, such as trends and cycles. Analogously, *Multivariate Singular Spectrum Analysis* (MSSA) extends the SSA paradigm in the multivariate setting by capturing spatiotemporal relationships across multiple variables [78].

The aforementioned temporal principal component-based methods are valuable for uncovering long-term spatiotemporal dependencies and reducing data dimensionality. However, they face several challenges, especially in real-world applications. A key limitation is the assumption of stationarity, as these methods rely on covariance structures that may change over time, making them less effective for highly dynamic or non-stationary time series. Additionally, time lag selection significantly impacts performance, requiring careful tuning to balance capturing meaningful dependencies while avoiding excessive noise. Computational complexity also increases with longer time horizons and higher-dimensional data, as these methods involve large covariance or trajectory matrices that require eigendecomposition, leading to scalability issues [78]. Nevertheless, their ability to capture intricate patterns makes them highly effective when paired with forecasting models, enhancing their ability to handle complex datasets.

2.4.2. ONLINE PCA

In dynamic environments, where data streams are generated continuously, traditional batch PCA methods become impractical due to their computational and memory constraints. This challenge has led to the development of *Online PCA*, which updates the principal components incrementally as new data arrives [14]. Unlike standard PCA, which processes all data at once, Online PCA accommodates streaming data by iteratively refining estimates of the principal components, making it particularly well-suited for applications in real-time systems, adaptive forecasting, and high-dimensional data analysis [14].

Online PCA typically relies on optimization techniques for approximating principal components without storing the entire data set. Given a sequence of data vectors $\{\mathbf{x}_t : t = 1, 2, \dots, T\}$, Online PCA aims to iteratively estimate the leading eigenvectors of the covariance matrix. At each time step t, the algorithm updates the principal components by incorporating the new data sample $\mathbf{x}_t$, while adjusting the current estimates based on the selected learning rates and convergence criteria. Several algorithms have been proposed for Online PCA, each balancing computational efficiency, convergence speed, and accuracy differently. One key method is *Oja's Rule*, a gradient-based method that iteratively updates the principal component matrix by minimizing the reconstruction error as new data arrives [108, 107]. Another approach is *Incremental PCA* (IPCA), which maintains a low-rank approximation of the data matrix and updates it with small batches of new data vectors arriving [5].

Online PCA methods, rooted in the aforementioned approaches, have inspired numerous variants tailored to specific applications and computational constraints. These methods are crucial for processing large-scale and streaming datasets, enabling dynamic updates of principal components without requiring full data storage [29]. Despite their utility, key challenges include maintaining computational efficiency, ensuring robustness to noise, and effectively adapting to non-stationary data distributions [103, 109]. Addressing these challenges remains an active area of research, highlighting the evolving importance of online PCA in modern data-driven tasks.

2.5. Spatio-Temporal Graph Neural Networks

By incorporating spatial dependencies alongside temporal dynamics, *Spatio-Temporal Graph Neural Networks* (STGNNs) extend statistical and machine learning techniques for time series modeling. This is achieved by leveraging the underlying graph structure that characterizes the different time series, allowing the STGNNs to also capture complex inter-variable relationships. This more expressive representation of multivariate temporal data makes them well-suited for applications where both spatial and temporal correlations are crucial.

2.5.1. Modeling Spatial and Temporal Dependencies

To effectively capture inter-variable relationships between different time series, STGNNs incorporate spatial modeling techniques utilized by GNNs on static graphs [75]. By leveraging the underlying graph structures that represent how the time series are interconnected, these methods ensure that relevant spatial patterns are preserved in the subsequent analysis. Broadly, there are 3 main approaches for spatial modeling:

- **Spectral GNN-based Approaches**. These methods stem from spectral graph theory, capturing relationships in the graph frequency domain using the GSO [76, 123].
- **Spatial GNNs-based Approaches**. Instead of working in the frequency domain, these methods design filters directly over the neighborhood of each node, modeling spatial dependencies through message passing [55] or graph diffusion [54].
- **Hybrid Approaches**. These models interwine spectral and spatial GNN techniques, aiming to effectively combine their strengths.

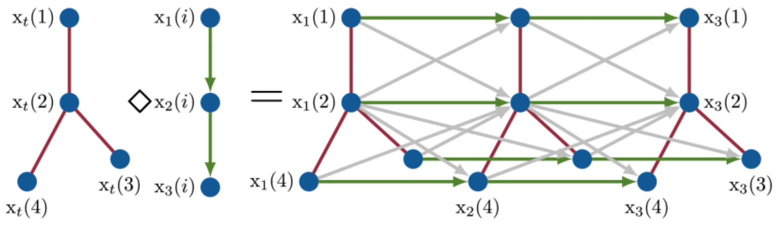
Besides spatial dependencies, STGNNs must also account for how data evolves over time. Temporal modeling is essential for understanding sequential patterns within the time series. To achieve this, STGNNs incorporate temporal modules that process time-dependent relationships, either in the time domain or frequency domain [75]. By learning meaningful temporal relationships, alongside the aforementioned spatial modules, STGNNs are capable of modeling complex spatio-temporal patterns. Operating in the time domain, there are 4 categories of models:

- **Recurrence-based Approaches**. These methods rely on standard recurrent-based architectures, like Recurrent Neural Networks (RNNs) [38] or variants, to model inter-temporal relationships in the time domain.
- **Convolution-based Approaches**. Inspired by the efficient learning of CNNs through parallel processing, these methods extract temporal patterns through temporal convolutions [140]. Even though most proposed models operate in the time domain, there are also works focusing on the frequency domain.
- **Attention-based Approaches**. These methodologies focus on attention mechanisms, such as transformers [147], to either model temporal dependencies within univariate time series, or account for spatial and temporal features at the same time, thus directly capturing spatiotemporal patterns.
- **Hybrid Approaches**. Again, the aforementioned approaches can be effectively combined, resulting in more complex hybrid models that exploit the strengths of each of these.

2.5.2. Architectural Paradigms

Effectively combining spatial and temporal modules is crucial for constructing powerful STGNN models. Different architectural paradigms determine how spatial and temporal dependencies interact within the model, impacting its ability to capture complex patterns. The following methods focus on processing static graphs, with a fixed spatial structure and the data dynamically changing across time. An initial approach involves concatenating distinct learning modules that handle separately the spatial and temporal dependencies. These *disjoint models* typically consist of a combination of graph-based learning algorithms for spatial feature extraction and sequence-based models to capture temporal patterns. The two models can be ordered in any way, with the proper adjustments, leading to a variety of different structures. For example, a GNN might first extract spatial features at each timestamp, which are then fed into an LSTM to capture the underlying temporal dynamics, or vice versa [71]. Even though this method offers flexibility and simplicity for combining state-of-the-art methods for spatial and temporal relationships, it may fail to explicitly capture more complex spatiotemporal patterns, since the spatial and temporal components are not processed jointly [116].

One way to overcome the aforementioned challenge is to integrate the graph structure directly within a sequence-based model. These *graph recursive models* treat the temporal dimensions as a sequence of snapshots, where spatial embeddings are constructed at each time step through graph convolutions. For example, graph convolutions can be integrated in the autoregressive and moving average components of the VARIMA framework by replacing the parameter matrices with graph convolutional filters. Respectively, for the non-linear extension of VARIMA with RNN variants, a similar replacement of the parameter matrices induces graph-structure bias into the temporal mechanisms. This architectural paradigm processes spatial and temporal relationships in a joint manner, however the RNN variants lead to an undesirable model bias by putting more emphasis on the latter [116], and face difficulties with parallelization and vanishing gradients.



Kronecker: $\mathbf{S}_\otimes = \mathbf{S}_T \otimes \mathbf{S}_\otimes$; $|\mathcal{E}_\otimes| = 2|\mathcal{E}_T||\mathcal{E}|$

Cartesian: $\mathbf{S}_\times = \mathbf{S}_T \otimes \mathbf{I}_N + \mathbf{I}_T \otimes \mathbf{S}$; $|\mathcal{E}_\times| = T|\mathcal{E}| + N|\mathcal{E}_T|$

Strong: $\mathbf{S}_\boxtimes = \mathbf{S}_\otimes + \mathbf{S}_\times$ $|\mathcal{E}_\boxtimes| = |\mathcal{E}_\otimes| + |\mathcal{E}_\times|$

Figure 2.3: Different product graphs and the formation of multivariate temporal data over them, adapted from [116]

Another approach is to extend the graph representation by considering snapshots of the graph signal at every time step and connect them successively with temporal edges. These larger graph representations put equal weight in spatial and temporal dependencies, addressing the underlying temporal bias of the graph recursive models [116]. Nevertheless, the design problem of connecting temporally-adjacent graph snapshots with each other is not trivial. This is a crucial decision for creating powerful STGNN models, especially considering the computational overhead of performing graph convolutions, or other graph-based operations, on the larger graph. *Product graphs*, equipped with desirable theoretical properties, such as permutation equivariance, and a higher degree of interpretability, offer a range of choices for combining temporal graph replicas [116]. Three primary types of product graphs are commonly used to represent spatiotemporal relationships:

- ***Kronecker Product Graph***. The only types of relationships (edges) that exist with this graph is among spatially neighboring nodes for successive time steps. For a GSO $\mathbf{S} \in \mathbb{R}^{N\times N}$ of the static graph $\mathcal{G}$, the Kronecker product graph is defined by the following GSO:

$$\mathbf{S}_\otimes = \mathbf{S}_T \otimes \mathbf{S} \tag{2.25}$$

  where $\mathbf{S}_T$ is a subdiagonal identity matrix of size $T \times T$. It corresponds to the graph with the gray edges in Figure 2.3.

- ***Cartesian Product Graph***. This graph can be seen as the complement of the Kronecker product graph, since it involves spatial connections and temporal connections for the same node across successive time steps. The Cartesian product graph is defined by the following GSO:

$$\mathbf{S}_\times = \mathbf{S}_T \otimes \mathbf{I}_N + \mathbf{I}_T \otimes \mathbf{S} \tag{2.26}$$

  where $\mathbf{I}_N$ and $\mathbf{I}_T$ are identity matrices of size $N \times N$ and $T \times T$ respectively. It corresponds to the graph with the green and red edges in Figure 2.3.

- ***Strong Product Graph***. This graph is more general and involves all possible connections of the two previous product graphs. The strong product graph is defined by the following GSO:

$$\mathbf{S}_{\boxtimes} = \mathbf{S}_T \otimes \mathbf{I}_N + \mathbf{I}_T \otimes \mathbf{S} + \mathbf{S}_T \otimes \mathbf{S} \tag{2.27}$$

  It corresponds to the graph with the green, gray, and red edges in Figure 2.3.

Product graphs provide a robust foundation for integrating spatial and temporal dependencies, enabling joint spatiotemporal learning. By leveraging the unique properties of different product types, they offer a flexible and comprehensive approach to modeling complex interactions [116]. Optimizing their computational efficiency and exploring novel applications are key challenges for fully realizing their potential in STGNNs.

2.5.3. Learning with Dynamic Graphs

The aforementioned paradigms assume that the underlying graph structure remains unchanged across time. In practice, most real-world graphs have a dynamic structure that evolves across time, causing the static graph models to fail. To overcome these limitations, several approaches exist that account for a time-varying graph topology. Originating from the graph recursive model paradigm, the *evolving GNN models* [112] employ temporally-evolving weights through an RNN, instead of directly applying the RNN on the GNN-extracted node features to learn the temporal dynamics. Namely, a GNN operates at each time stamp on the current state of the graph signal, producing spatial node embeddings. Subsequently, the RNN model updates the GNN weights by combining their previous values and the information from the last computed node embeddings. Depending on the RNN variant and the problem parameters, the GNN weights could correspond to the hidden state or the output of the RNN model. This model paradigm aims to capture the temporal graph dynamics through its time-dependent parameters, and therefore is capable of handling non-static graphs.

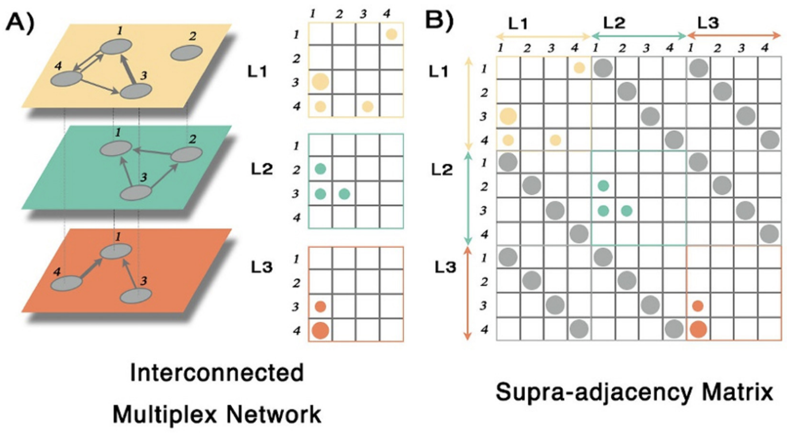


Figure 2.4: Schematic representation of 3 layers of a dynamic graph (A) and the corresponding supra max adjacency matrix (B), adapted from [84]

Another approach stems from the product graph models. Again, the dynamic graph can be represented by a larger static graph, consisting of multiple layers [61]. The connections among the nodes of this larger graph are expressed through a *supra adjacency matrix* of size $(NT) \times (NT)$, where N is the number of variables and T is the number of layers (time steps). Across the diagonal, the matrices of size $N \times N$ represent the spatial connections within the respective layers, at specific time steps. The off-diagonal matrices represent the temporal connections between nodes of different layers. The difference with product graphs is that, here, connections between non-successive time steps are permitted. Even though this allows a GNN to capture more complex spatiotemporal relationships, it also increases the computational overhead of processing the graph. To ease the computational burden, the cross-layer temporal connections may be allowed only between the same node of the original graph, similar to the Cartesian product. Moreover, a temporal window w indicates the maximum time lag between layers for which connections are allowed. Layers with longer time intervals would not be directly connected [61]. Figure 2.4 illustrates a graph with the above restrictions and a temporal window $w = 2$.

2.6. Conclusion

In this chapter, we discussed the background material necessary for the research carried out in this thesis. Beginning with static data analysis, PCA was introduced as a fundamental technique for dimensionality reduction and data decomposition. Building upon this, coVariance Neural Networks were presented, demonstrating how covariance matrices can be leveraged within the GNN framework. The discussion then transitioned to the temporal domain, covering the fundamentals of time series and various modeling techniques, including statistical methods and principal component-based approaches. Finally, STGNNs were explored as a framework for capturing both spatial and temporal dependencies in multi-variable sequential data. These topics establish the theoretical foundations for the subsequent chapters, where we will propose a novel covariance-based GNN approach for multivariate time series modeling and forecasting.

3

Literature Review

In this chapter, we will provide an overview of the relevant literature for the research carried out in this thesis. We begin by introducing the general landscape of time series modeling in Section 3.1, outlining key challenges and motivations behind different approaches. Section 3.2 explores statistical techniques, ranging from classical time series models, such as Exponential Smoothing and ARIMA, to principal component-based methods, like Temporal PCA and Online PCA algorithms. Then, Section 3.3 shifts to non-graph Deep Learning approaches, presenting key works utilizing RNNs and their variants, CNNs, and attention mechanisms, as well as hybrid architectures that integrate PCA-based techniques with these models. Section 3.4 reviews spatiotemporal GNNs, covering various architectures for processing spatial and temporal dependencies, before focusing on covariance-based STGNNs, which have a central role in this thesis. Finally, this chapter concludes with a discussion in Section 3.5.

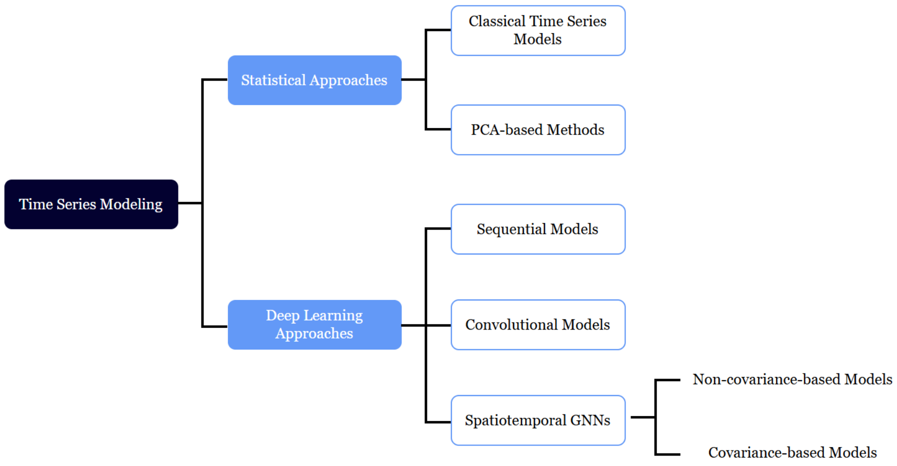


Figure 3.1: Classification of methods for time series modeling

3.1. Introduction

Time series modeling is a fundamental problem in data analysis, where the objective is to understand temporal dependencies and patterns within sequential data. These models can be applied to a variety of tasks, including forecasting [91], anomaly detection [12], data imputation [44], and classification [70]; however, the emphasis of this thesis has been mainly placed on models developed for forecasting. Traditional statistical methods have long been employed for this task, offering theoretically grounded, mathematical approaches to capturing underlying patterns [68]. More recently, deep learning techniques have emerged as powerful alternatives, leveraging neural networks to model complex, nonlinear relationships in time series data [58]. Beyond purely temporal models, graph approaches have gained traction, particularly for settings where spatial dependencies influence temporal dynamics [75]. In this chapter, we adhere to this taxonomy of methods for time series modeling, with a further distinction for methods utilizing a covariance matrix of the data, such as PCA [78] and covariance-based spatiotemporal GNNs [31]. Our framework for categorizing these methods is shown in Figure 3.1.

3.2. Statistical Approaches

Exponential Smoothing (ES) and ARIMA models are two well established statistical approaches in time series modeling, each offering distinct advantages. ES methods, such as the Holt-Winters model [67, 152], are effective in applications with strong seasonal patterns and trend indications, like rainfall forecasting [149]. Recently, they have been extended with the grey generating operator [94], addressing their limitations in settings with irregular and highly dynamic data, with applications in air quality index estimation [154] and electricity price prediction [117]. Then, ARIMA models constitute a cornerstone of time series analysis due to their ability to capture linear relationships, such as autocorrelation [15]. They have been widely applied in environmental modeling [83], including air quality [73, 97, 164], noise pollution [52, 60], and traffic pollution forecasting [151]. Additionally, ARIMA models have been extensively explored in financial time series applications, such as stock price prediction [85, 139] and market volatility estimation [92]. In multivariate settings, vector autoregressive models extend these capabilities, enabling applications in wind forecasting [167] and socio-economic trend analysis [142].

Despite their effectiveness with capturing trend-seasonality components and linear relationships respectively, ES and ARIMA models struggle with more complex time series characterized by non-linearity and long-term dependencies [18]. These limitations have been augmented with the emergence of neural networks and deep learning techniques, which can capture more intricate, non-linear patterns at the expense of interpretability [58]. In response, hybrid models have been proposed, leveraging statistical techniques with deep learning methods to leverage their complementary strengths. For instance, ES models have been combined with RNNs and their variants [131, 132, 21], as well as with Transformer models [153], in various applications. Similarly, ARIMA models have been integrated with RNNs for wind speed forecasting [95], and further extended with attention mechanisms [163] or CNNs [74] for financial forecasting. Nevertheless, these hybrid models have an increased complexity, because of the data-driven nature of the deep learning component and the theoretical assumptions of the statistical component [62].

PCA-based Models

PCA has long been a cornerstone for dimensionality reduction and feature extraction, particularly in time-domain analysis through its extended framework in TPCA. By decomposing time series into orthogonal components, principal component methods isolates dominant patterns that contribute most to the variance, offering insights into underlying temporal structures [78]. Focusing on the multivariate setting, MSSA constitutes an effective method for performing TPCA, with applications in modeling traffic flows [102] and galaxy dynamics [146]. The MSSA paradigm has led to numerous variants for spatiotemporal modeling. In [3], MSSA is integrated with the cumulative sum statistic from sequential hypothesis testing into a factor model for a financial system, aiming to detect abrupt changes in the underlying series dynamics in an online manner. Further works in the same direction suggest that variants of this model have comparable or even better performance than deep learning alternatives, such as a LSTM, on the same task [1]. Apart from the MSSA paradigm, in [87], a custom spatiotemporal PCA method is proposed for multivariate datasets. It employs an extended covariance matrix across space and time [81] and replaces the original spatiotemporal data with a multivariate functional dataset [59], aiming to increase the expressivity and address the "*curse of dimensionality*" often associated with conventional spatiotemporal PCA approaches.

Nevertheless, the application of PCA-based techniques becomes less effective with streaming or non-stationary data settings. As traditional PCA fails to account for evolving data distributions, online PCA methods address this limitation by updating the principal components incrementally as new data arrives [14]. A plethora of PCA variants have been proposed, incorporating various mechanisms to handle the challenges in the online setting. In [29], a comparative study evaluates different algorithms, such as batch

PCA, incremental PCA [5], and stochastic gradient approaches (Oja's rule [108, 107]), over computation time and estimation error with high dimensional data. Stemming from incremental PCA, various adaptive online methods have been suggested for dimensionality and principal component adjustments [103, 2, 104]. Moreover, with non-stationary data, almost all of the underlying data variation is often accredited to a few components, which may be spurious and lead to misinterpretations [109]. Motivated by this observation and an application of PCA on the first difference of bond returns [39], in [63] the focus shifts on applying PCA on the forecast errors rather than the time series themselves. Finally, it is crucial to develop algorithms addressing both the limitations of an online setting and the underlying non-stationarity, such as the probabilistic PCA method in [101], since many problems fall into both categories.

The PCA framework also has a number of limitations, similar to Exponential Smoothing and ARIMA models. PCA operates on a sample covariance matrix, which is a perturbed version of the true covariance matrix based on a finite number of samples. For eigenvalues that are close to each other, PCA proves to be unstable in constructing their principal components, since they could largely deviate with small changes in the available data [77], and thus lead to irrreproducible statistical results [43]. Then, considering the online setting, various PCA variants have been proposed to account for missing values, high time and space complexity, slow computation speed, and low statistical accuracy [16, 29]. Still, they either require careful hyperparameter optimization, since they have a limited number of observations in their disposal, or they are accompanied by significant tradeoffs with respect to accuracy, convergence, and computational speed [29]. Moreover, for multivariate time series data, temporal PCA approaches estimate the correlations between points across space and time, constructing a larger spatiotemporal covariance matrix [78]. However, the processing this matrix is associated with high computational costs, which limit scalability and the available temporal memory [31].

3.3. Sequential and Convolutional Models

Deep learning models have gained significant traction for time series modeling due to their ability to learn complex temporal dependencies directly from the data. Recurrent Neural Networks [38] are a class of deep learning architectures specifically designed to model sequential data by retaining past information through a hidden state, which is updated at every time step [144]. Because of their focus on temporal dependencies, numerous applications of RNN models have been suggested in environmental problems [33], healthcare analytics [6], and modeling traffic patterns [69] among others. To allow more intricate representations, bidirectional RNNs process sequential data also in reverse order, which facilitates the interpretation of earlier observations [27, 160]. However, traditional RNNs suffer from the vanishing and exploding gradient problems, impeding their learning of longer-term relationships [113]. To mitigate these issues, various RNN variants have been proposed, such as LSTM [66] and GRU [35]. These architectures have been adjusted to various problem settings, ranging from stock trend prediction [64, 51] and supply chain [111] optimization to natural language processing [105, 49], speech recognition [137, 121], and recommender systems [42, 8, 99]. Besides recurrence-based architectures, Convolutional Neural Networks have also been adapted to time series problems. Their ability to exploit spatial or temporal locality in the data and their fewer parameters in comparison to RNNs are the main motivating factors for exploring their application in time series modeling [166, 13].

More recently, transformer-based architectures have emerged as powerful tools for time series modeling, primarily due to their ability to capture long-range patterns using self-attention mechanisms [148]. Most variants, such as in [89, 169, 96, 170], introduce newly designed attention modules within the vanilla transformer architecture, aiming to incorporate time series inductive biases into it. Another approach is to design a new transformer architecture to account for sequential data, as in [36, 118]. Nevertheless, these models focus only on temporal relationships. In the multivariate time series setting, distinct attention blocks are constructed for temporal and spatiotemporal relationships and are interwined with graph convolution networks, such as the works in [22, 159] for traffic flow forecasting and [162] for modeling pedestrian trajectories. Moreover, hybrid models of statistical techniques, like PCA, and the aforementioned deep learning methods have gained attention. For instance, the works in [158] and [93] integrate PCA and SSA with the LSTM architecture for wind power and wind speed forecasting respectively. These principal component-based techniques act as a preprocessing step, reducing the dimensionality of the input and extracting main components of the series, in order to facilitate the work of the subsequent deep learning model. In a similar manner, in [51] PCA is integrated with LSTM and GRU models for stock trend prediction, while in [34, 135] MSSA is integrated with LSTM models for electric load forecasting.

Despite promising results, these deep learning models face limitations when applied to real-world time series. Most of these methods treat the time series as independent or flattened inputs, thus not explicitly modeling the spatial relations between different series. This limits the expressiveness of these models with respect to inter-variable patterns and leads to an implicit bias in favor of temporal relationships [75]. As a result, there is a growing interest in models that account for both spatial and temporal dynamics, capturing more intricate spatiotemporal relationships and improving generalization [75].

3.4. SPATIOTEMPORAL GNN METHODS

For multivariate time series modeling, one main advantage of STGNNs over non-graph deep learning approaches is how they process inter-variable dependencies. They achieve this by representing the series and their pairwise interactions with graph structures and processing them in a similar manner as GNNs with static graphs [75]. Most of the approaches operating in the spectral domain approximate the graph convolution operations with Chebyshev polynomials, heavily inspired by the seminal work in [40]. Some of these works focus on effectively entangling Chebyshev polynomial convolution layers and temporal convolution layers, in order to learn both spatial and temporal dependencies coherently, as in [161]. Others aim to jointly capture spatiotemporal relationships by moving the graph to the spectral domain through GFT and performing frequency-domain convolutions before moving back to the time domain, as in [26]. The work in [88] alternatively proposes the construction of multiple spatiotemporal graphs with different functionalities and processing them through convolutions and a custom spatiotemporal attention mechanism to extract meaningful patterns.

Then, inspired by the more intuitive approach of spatial-based GNNs [155], several methods focus on modeling inter-variable dependencies in the spatial domain. By viewing the spatial-domain graph convolutions as diffusion processes [54], one can achieve spatiotemporal modeling by incorporating graph diffusion layers into RNN-based architectures, as in [157]. Alternatively, by viewing the graph convolution as a message passing/information propagation process [55], the work in [98] proposes graph propagation with temporal polynomial coefficients, while the work in [45] suggests using a transformer-based model with multi-head spatiotemporal self-attention blocks to capture cross-spatial-temporal correlations. Apart from these two subcategories, other approaches extend the pre-existing spectral domain methods to the spatial domain, such as the work in [133] adapting the spectral graph convolutions [86] to synchronous graph convolutions on localized spatiotemporal graphs. Finally, some hybrid models combine both spectral-based and spatial-based GNN techniques, such as performing Chebyshev polynomial convolutions on a global level and message passing on a local level [165].

With respect to inter-temporal relationships, most spatiotemporal GNNs rely on well-established deep learning architectures, which have been proven to be effective in modeling temporal dynamics. As mentioned earlier, RNN-based architectures are often coupled with graph diffusion [157, 90] or with spectral graph convolutions [7]. Instead of RNN-based models, others focus on integrating graph convolution mechanisms with CNN models, both in the spatial [161] and the spectral domain [26]. Another option is attention-based architectures, such as transformers, with most related works either constructing distinct spatial and temporal attention mechanisms [168], or using transformer layers to produce representations of the individual temporal dependencies of each series and, afterwards, model the inter-variable dependencies through them [119]. Hybrid methods have also been proposed [143], however it is not clear how the aforementioned temporal models should be optimally combined for the task at hand [75].

The majority of these GNN-based methods for spatiotemporal modeling either process the spatial and temporal dependencies in a sequential manner, or the graph spatial-processing operations are integrated within the temporal-processing architecture. The former approach fails to jointly capture the dependencies on a data-level, while the latter often limits the expressivity and introduces a bias in favor of the temporal over the spatial patterns [116]. One response to this design limitation is to create larger product graphs to represent both spatial and temporal dependencies equally, facilitating the extraction of spatiotemporal patterns in a joint manner [116]. However, the rule for creating edges within the product graph, and therefore determining the information propagation, is not clear, while they are also associated with high computational complexity [116]. Moreover, modeling the inter-variable dependencies usually requires an underlying graph structure, which may not be available in various multivariate time series problems [126].

COVARIANCE-BASED MODELS

In a static data setting, coVariance Neural Networks [126] leverage the sample covariance matrix as the underlying graph structure for modeling inter-variable dependencies. Unlike traditional adjacency or Laplacian-based GSOs, the covariance matrix encapsulates second-order statistical dependencies between data dimensions, aligning naturally with PCA's ability to reveal dominant modes of variance. PCA-based approaches are often intertwined with deep learning techniques in order to remove redundant information from the features at every layers, as described in Section 3.3. VNNs are attempting to exploit this beneficial functionality of PCA, along with the low number of parameters of GNNs. Furthermore, VNNs are more stable to perturbations in the sample covariance matrix from the finite data, extending from the stability of coVariance Filters, as explained in Section 2.2.2. In the seminal work [126], a set of experiments for age prediction from cortical thickness brain data at different resolutions hinted that the performance of VNNs can be transferred across resolutions without re-training. This led to a series of additional works focusing on brain age prediction [129, 128, 130, 125], with a focus on this performance transferability property of VNNs. Moreover, another desirable property of VNNs would be explainability. The VNNs explanatory capabilities are also explored on this same task in [127, 124]. Overall, the stability of VNNs to perturbations in the sample covariance matrix, along with performance transferability and a significant degree of explainability, constitute a notable advancement in graph-based neural network frameworks.

In [31], the VNN framework is extended to multivariate time series data. This *Spatio-Temporal VNN* (STVNN) approach addresses the challenges of PCA with streaming and temporal data, as well as adapting to distributional shifts. First, STVNNs handle streaming data by parametrically updating the covariance matrix as new data becomes available. Then, over a time horizon T, STVNNs apply a spatiotemporal covariance filter on the corresponding series, with a weighted sum operator aggregating the outputs of the convolutions at different time steps. Experiments showcase STVNN's superior stability over PCA-based methods, faster adaptability to distribution shifts, and, in forecasting tasks, more consistent performance across datasets and single-step predictions than LSTM, VNN, hybrid TPCA-MLP, and hybrid VNN-LSTM models [31]. One of the key takeaways from this work is the importance of using the sample covariance matrix as inductive bias in this spatiotemporal setting, with STVNN offering a robust framework for utilizing it.

In high-dimensional static data settings, the VNN framework suffers from more difficult covariance estimation and a more dense sample covariance matrix regardless of the state of the true one. In response to this, in [30], hard and soft thresholding sparsification strategies have been suggested for sparse true covariance matrices and stochastic sparsification strategies if their state is unknown. These methods lower the computational time for a VNN forward pass, maintain the stability property, and, often, even lead to improved performance. The reason is that many of the estimated entries of the covariance matrix may be spurious, similar to what is suggested in [109]. These spurious correlations would negatively impact the modeling of the underlying data distributions, however many of them are eliminated through the covariance sparsification process. Nevertheless, these techniques have not yet been adapted to the STVNN in the multivariate time series setting, where the same limitations with respect to the covariance matrix persist. Another limitation of the work in [31] is that covariances among variables at time lags other than zero are implicitly captured through the temporal sums, providing a lower computational overhead. This relates to the motivation behind product graph models [116], as a mode to explicitly capture these spatiotemporal dependencies.

3.5. Discussion

This chapter reviewed different approaches for time series modeling. Statistical models, like Exponential Smoothing and ARIMA, have been widely applied in time series forecasting for their robust theoretical foundation and their ability to capture trends and linear dependencies. PCA-based methods, including temporal PCA and MSSA, have also been used for dimensionality reduction and to extract dominant temporal patterns in multivariate settings. While effective in structured or stationary environments, the former approaches struggle with non-linear dynamics and long-range dependencies, while the latter are unstable to perturbations in the sample covariance networks and are often associated with high computational costs. Deep learning models, such as RNNs, LSTMs, CNNs, and Transformers, have recently emerged as powerful solutions for learning complex temporal features directly from data. However, these models still face challenges with multivariate data, because they do not explicitly capture the spatial relations between different series.

Spatiotemporal GNNs extend deep learning models by explicitly incorporating spatial relationships among variables using graph structures, allowing for more expressive representations in multivariate time series modeling. These approaches typically operate in either the spectral or spatial domain, leveraging graph convolution mechanisms to model inter-variable dependencies alongside temporal dynamics. Temporal modeling is usually achieved through established architectures like RNNs, CNNs, or attention-based mechanisms, which are integrated with the graph-based components. Nevertheless, most STGNNs process spatial and temporal dependencies either sequentially or embed the spatial component within the temporal one, limiting the capacity to model joint spatiotemporal interactions. Attempts to overcome this through more unified formulations, such as product graphs, often lead to increased model complexity and computational costs. Furthermore, the assumption of a predefined graph structure poses a challenge, as such information is often unavailable or difficult to infer reliably in many time series applications.

STVNNs extend the VNN framework, which uses the sample covariance matrix as an inductive bias to model inter-variable dependencies, to the time series domain. Unlike traditional graph neural networks, this approach captures second-order statistical structure without requiring a predefined graph. STVNNs handle streaming data by updating the covariance matrix parametrically and apply a spatiotemporal filter to extract patterns across time. Compared to PCA-based and deep learning hybrid methods, they offer improved stability, adaptability to distribution shifts, and consistent forecasting performance. However, like their static counterparts, STVNNs face challenges with high-dimensional data and dense, potentially spurious covariance estimates. While sparsification strategies have been proposed to address this in static settings, their integration into the spatiotemporal framework remains unexplored. Additionally, covariances beyond lag-zero are only implicitly modeled via a temporal sum, thus unable to directly capture cross-dependencies between the time series.

Building on this limitation of the STVNN model, this thesis directly addresses the challenge of modeling lagged spatiotemporal covariances. Specifically, the proposed model extends the STVNN framework by constructing a larger sample covariance matrix that captures the cross-dependencies between time series variables across multiple time lags. This approach aims to enhance the expressiveness of the model, drawing a direct parallel to TPCA. To address the increased computational complexity associated with this larger covariance structure, sparsification strategies are also integrated into this model, ensuring more efficient computation while preserving the accuracy of the covariance estimation. By exploring these covariance sparsification techniques in the multivariate time series setting, this approach offers a more scalable and robust solution for explicitly modeling the underlying relationships of multivariate time series and capturing spatiotemporal patterns.

4 Methodology

In this chapter, we present our proposed approach for multivariate time series modeling with an extended covariance-based GNN model. We begin with the definition of the problem in Section 4.1, where we introduce the modeling setting. In Section 4.2, we present the proposed model architecture, based on an extended covariance representation that accounts for lagged dependencies, and its extension that incorporates thresholding sparsification strategies to enhance efficiency. Then, in Section 4.3, we provide a theoretical analysis of the model's stability with respect to perturbations in the sample covariance matrix. Finally, we conclude this chapter with a summarization of the main contributions in Section 4.4.

4.1. Problem Formulation

We aim to process multivariate time series $\mathbf{x}_t \in \mathbb{R}^N$, composed of N variables evolving over time in vectors $\ldots, \mathbf{x}_{-1}, \mathbf{x}_0, \mathbf{x}_1, \ldots$. To capture both temporal and cross-variable dependencies, we extend the classical covariance formulation to account for lagged interactions across time. Under the assumption of stationarity, the mean vector $\boldsymbol{\mu}$ and the lag-τ covariance matrix $\mathbf{C}_\tau$ are defined as:

$$\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}_t] \text{ and } \mathbf{C}_\tau = \mathbb{E}[(\mathbf{x}_t - \boldsymbol{\mu})(\mathbf{x}_{t-\tau} - \boldsymbol{\mu})^\top]. \tag{4.1}$$

To jointly model temporal and spatial correlations, we consider an extended representation of the multivariate time series $\tilde{\mathbf{x}}_t = [\mathbf{x}_{t-T+1}^\top, \ldots, \mathbf{x}_{t-1}^\top, \mathbf{x}_t^\top]^\top$, by concatenating T consecutive time steps. The corresponding extended covariance matrix is:

$$\tilde{\mathbf{C}} = \mathbb{E}[(\tilde{\mathbf{x}}_t - \tilde{\boldsymbol{\mu}})(\tilde{\mathbf{x}}_t - \tilde{\boldsymbol{\mu}})^\top], \tag{4.2}$$

where $\tilde{\boldsymbol{\mu}}$ denotes the mean of the extended time series vector. The matrix $\tilde{\mathbf{C}}$ captures joint spatial and temporal correlations through its block structure, where diagonal blocks correspond to lag-zero covariance matrices and off-diagonal blocks correspond to lagged covariance matrices, as depicted in Figure (4.2).

Our objective is to learn a function $\boldsymbol{\Phi}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{C}}; \mathbf{h})$ that takes as input an extended data sample $\tilde{\mathbf{x}}_t$ with temporal memory of size T and the extended covariance matrix $\tilde{\mathbf{C}}$ including lagged covariances up to lag $T-1$. This function maps the input to a target output $\mathbf{y}$, which may represent a future instance $\mathbf{x}_{t+\theta}$ at horizon $\theta > 0$, or, alternatively, a class label. Here, $\mathbf{h}$ constitutes the parameter set of function $\boldsymbol{\Phi}(\cdot)$. The extended covariance matrix $\tilde{\mathbf{C}}$ serves as an inductive bias, in order to reduce the parameters and computational complexity of the model in comparison to purely data-driven models, as well as develop a solution directly related to TPCA.

In practice, the extended covariance matrix must be estimated in a streaming setting, as neither the true extended covariance matrix is available nor a reliable batch of data to retrieve it from. As a result, we update the estimates of the extended mean $\hat{\tilde{\boldsymbol{\mu}}}_t$ and the extended covariance matrix $\hat{\tilde{\mathbf{C}}}_t$ in a recursive manner, as described in Section 4.2.1. Additionally, the model parameters are also updated in an online fashion, based on the current estimate $\mathbf{h}_t$ and the model loss $\mathcal{L}(\boldsymbol{\Phi}(\tilde{\mathbf{x}}_t, \hat{\tilde{\mathbf{C}}}_t; \mathbf{h}_t), \mathbf{y})$ at time t.

4.2. Proposed Model

The proposed model architecture is illustrated in Figure 4.1. At a time step t, the Lagged spatiotemporal coVariance Neural Network (LVNN) receives as input an extended data sample $\tilde{\mathbf{x}}_t$, consisting of the T most recent observation of the time series. This input is initially used to update the extended sample covariance matrix $\hat{\tilde{\mathbf{C}}}_t$ in an online fashion, in order to improve the estimate of the true extended covariance matrix $\tilde{\mathbf{C}}$ and reflect the most recent dynamics of the series. Once updated, $\hat{\tilde{\mathbf{C}}}_t$ serves as the GSO over which our model operates, comprising a set of Lagged spatiotemporal coVariance Filters (LVF). The filters apply localized polynomial transformations of $\hat{\tilde{\mathbf{C}}}_t$ to the extended input signal, allowing the model to aggregate and propagate information through the extended sample covariance matrix. The output contains a set of node embeddings that encode rich, localized spatiotemporal representations of each variable across the observation window. These embeddings can either be used directly for a downstream task or further processed by a readout layer. An in-depth description of the individual components of the model is provided below.

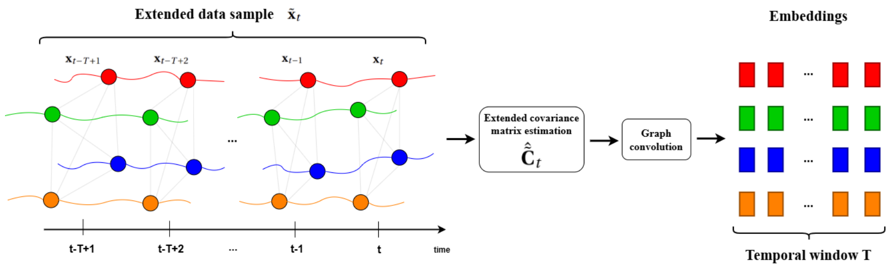


Figure 4.1: Lagged spatiotemporal covariance filter pipeline. We observe a new time series sample $\mathbf{x}_t$ and construct the extended data sample $\tilde{\mathbf{x}}_t$ to update the extended covariance estimate $\hat{\hat{\mathbf{C}}}_t$. The estimate of the extended covariance matrix is employed as a weighted graph adjacency matrix, leading to a fully-connected graph across all variables and time steps. The LVNN performs convolutions over this graph, producing a set of spatiotemporal embeddings.

4.2.1. EXTENDED COVARIANCE MATRIX

Figure 4.2 illustrates the structure of an extended covariance matrix used to capture both spatial and temporal dependencies in a multivariate time series. On the left, the large $T \times T$ block matrix represents the full extended covariance matrix across a temporal window of length T, where $\hat{\mathbf{C}}_\tau$ encodes the pairwise covariance terms between variables at time lag τ (for the lag-0 covariance blocks capturing spatial relationships, the subscript 0 is omitted). Under the assumption of stationarity, the sample covariance matrices $\hat{\mathbf{C}}_\tau$ with the same time lag τ are equivalent. The matrix is organized such that the entry at block column i and block row j corresponds to the covariance between the signal at time step $t-T+i$ and $t-T+j$, up to lag $T-1$. To the right, a magnified view of an individual $N \times N$ covariance block $\hat{\mathbf{C}}$ shows its internal structure for N time series variables, where each element $c_{k,l}$ denotes the covariance between variables k and l at the corresponding time lag. The full matrix forms the backbone of the LVFs, enabling joint learning of spatio-temporal relationships.

In this data streaming setting, the estimate of the extended covariance matrix is updated every T time steps. Because the extended data samples consist of T consecutive observations of the N time series variables, smaller intervals between updates would lead to overlaps among the extended data samples and, hence, they would not be independent. Therefore, moving from time step t to $t+T$, a new extended data sample $\tilde{\mathbf{x}}_{t+T}$ is constructed from the observations between time steps $t+1$ and $t+T$. Since our model operates in an online setting, $\tilde{\mathbf{x}}_{t+T}$ will be used to update the estimates of the extended data sample mean $\hat{\hat{\boldsymbol{\mu}}}_t$ and the extended covariance matrix $\hat{\hat{\mathbf{C}}}_t$ according to:

$$\hat{\hat{\boldsymbol{\mu}}}_{t+T} = \alpha_t \hat{\hat{\boldsymbol{\mu}}}_t + \beta_t \tilde{\mathbf{x}}_{t+T}, \tag{4.3}$$

$$\hat{\hat{\mathbf{C}}}_{t+T} = \xi_t \hat{\hat{\mathbf{C}}}_t + \zeta_t (\tilde{\mathbf{x}}_{t+T} - \hat{\hat{\boldsymbol{\mu}}})(\tilde{\mathbf{x}}_{t+T} - \hat{\hat{\boldsymbol{\mu}}})^\top \tag{4.4}$$

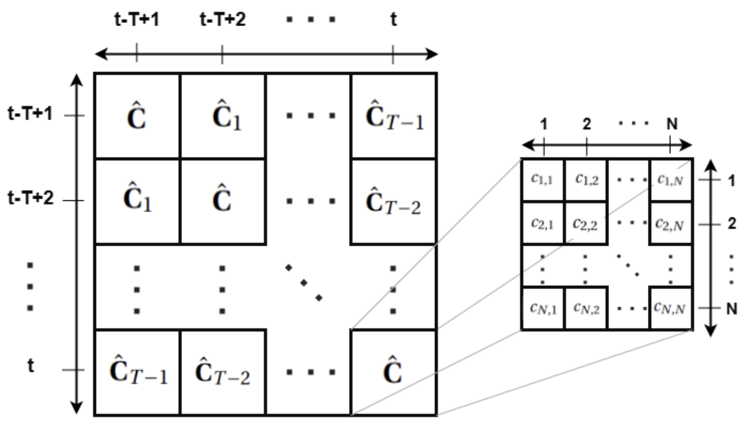
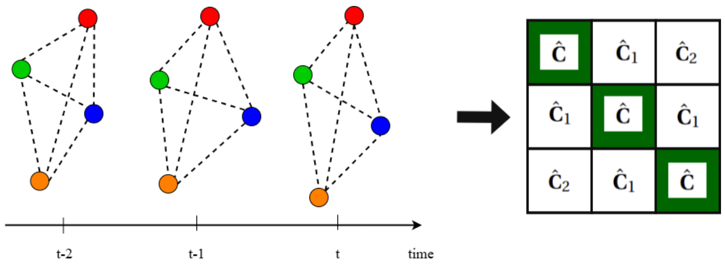


Figure 4.2: The structure of the extended sample covariance matrix for N variables and a temporal window size T, at time t. It consists of $T \times T$ blocks, capturing the covariance between the variables at two time steps t_1 and t_2, with a time lag $|t_2 - t_1|$. Therefore, each covariance block has a size $N \times N$, for each pair of variables between the two time steps.

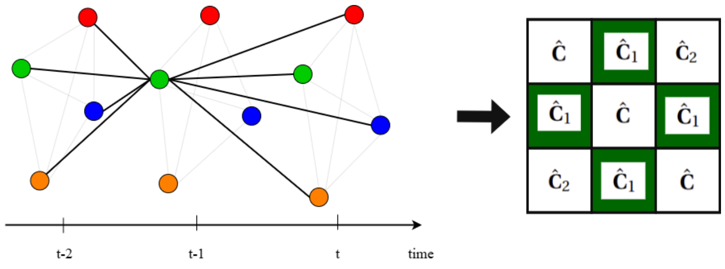
where $\alpha_t, \beta_t, \xi_t, \zeta_t \in [0,1]$ are scalar update coefficients. These coefficients account for both stationary ($\alpha_t = t/(t+T)$, $\beta_t = \zeta_t = T/(t+T)$, $\xi_t = (t-T)/t$) and non-stationary ($\alpha_t = \xi_t = 1-\gamma$, $\beta_t = \zeta_t = \gamma$) settings, where an additional parameter $\gamma \in [0,1]$ controls the contributions of $\tilde{\mathbf{x}}_{t+T}$ to the previous estimates.

To gain a better understanding of how the extended covariance matrix is updated, let us consider a setting with $N = 4$ time series variables and a temporal window size $T = 3$. As a new extended data sample becomes available at time t, Figures 4.3a-4.3c illustrate how different relationships contribute to the updates of different blocks of the extended covariance matrix. In Figure 4.3a, for the lag-0 case, only spatial relationships between variables within the same time step are present, updating the covariance blocks on the diagonal. In Figure 4.3b, the relationships between variables at consecutive time steps are used to update the lag-one covariance terms, reflected by the off-diagonal $\hat{\mathbf{C}}_1$ blocks in the extended covariance matrix. Finally, in Figure 4.3c, relationships between variables located two time steps apart update the $\hat{\mathbf{C}}_2$ off-diagonal blocks in the corners of the matrix.

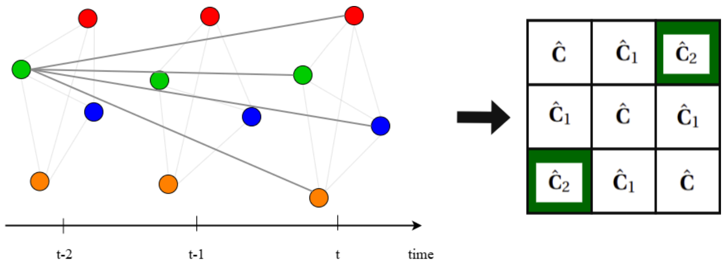
Depending on the target task, the extended sample covariance matrix can be processed differently: for tasks like anomaly detection or data imputation, bidirectional spatiotemporal dependencies may be desirable; however, for single-step forecasting, we opt to retain only the temporal relationships flowing backward in time. As shown in Figure 4.4, we zero-out the blocks below the diagonal, which, in our formulation of the problem, correspond to edges from previous to more recent time steps, therefore effectively disabling the forward-looking temporal connections. First, as we will discuss in Section 4.2.2, this structure creates nested temporal neighborhoods, allowing our model to focus on more recent time steps and incrementally enrich the embeddings with earlier information. Furthermore, this approach significantly improves the computational efficiency of processing the extended sample covariance matrix, since it zeroes out $T \times (T-1)/2$ of



(a) Update of Lag-Zero Covariance Blocks



(b) Update of Lag-1 Covariance Blocks



(c) Update of Lag-2 Covariance Blocks

Figure 4.3: A visualization of the updates of the extended sample covariance matrix in a setting with $N = 4$ variables and a temporal window size $T = 3$. The updated covariance blocks with each lagged connection are highlighted with green. For the lag-1 and lag-2 covariance block updates, we only draw the relevant relationships for the green node at $t-1$ and $t-2$ respectively.

its $T \times T$ blocks.

The presence of spurious covariance terms and the substantial computational overhead are two key challenges associated with the extended covariance matrix. As the number of variables N and the temporal window size T increase, the dimensionality of the extended covariance matrix grows quadratically with each of them, leading to scalability issues in both memory and computation. At the same time, the finite data sample estimate of the extended covariance matrix may contain spurious or noisy correlations, especially in high-dimensional settings with limited data. These entries introduce mis-

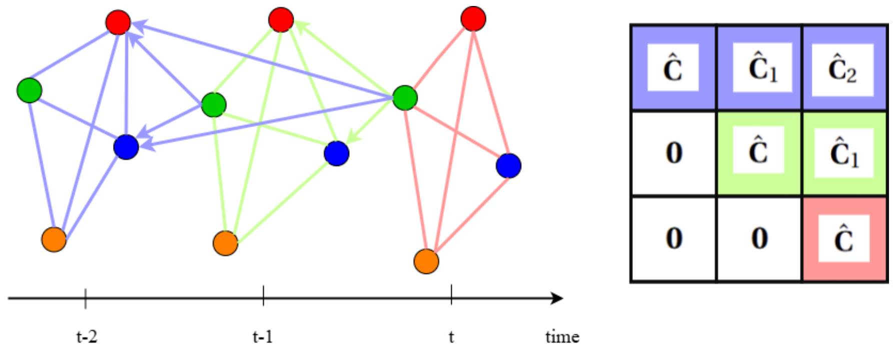


Figure 4.4: A visualization of the graph structure of the block upper triangular extended covariance matrix in a setting with $N = 4$ variables and a temporal window size $T = 3$. Spatial connections are bidirectional, while temporal connections are unidirectional towards the past. Each color represents a different temporal neighborhood, starting from time step t and incrementally incorporating each of the previous time steps. For the lag-1 and lag-2 temporal connections, we only draw the relevant edges originating from the green nodes and leading to the red and blue ones.

leading dependencies into the graph structure and negatively impact the quality of the produced embeddings. To address these issues, assuming that the true extended covariance matrix is sparse, we adapt two thresholding-based sparsification methods from [30] to our setting. This sparsification approach reduces the computational overhead and aims to prune spurious correlations, thus helping the model focus on meaningful spatiotemporal relationships.

The first approach is *hard thresholding*, where we remove the weaker entries of the extended sample covariance matrix. To achieve this in the stationary setting, we apply a threshold that decreases over time (since we process more data and get a more accurate estimate of the extended sample covariance matrix). All entries of the matrix with an absolute value below this threshold are set to zero, effectively pruning the weaker correlations. This technique has been shown to be especially effective in high-dimensional low-data settings [10].

Definition 1 (Hard Thresholding). *Given the extended sample covariance matrix* $\hat{\hat{\mathbf{C}}}$ *and a coefficient* $\tau > 0$, *the hard thresholding function is* $\eta(\hat{\hat{\mathbf{C}}})_{ij} = \hat{\hat{c}}_{ij}$ *if* $|\hat{\hat{c}}_{ij}| \geq \tau\sqrt{\frac{T}{t}}$, 0 *otherwise.*

Alternatively, while hard thresholding applies a strict threshold and keeps the remaining entries intact, *soft thresholding* reduces the magnitude of these remaining entries by subtracting a value equivalent to the current threshold from all of them. This technique often leads to more reliable covariance estimates and can be especially effective in settings where noise is expected to affect the whole matrix [41].

Definition 2 (Soft Thresholding). *Given the extended sample covariance matrix $\hat{\hat{\mathbf{C}}}$ and a coefficient $\tau > 0$, we define the soft thresholding function as $\eta(\hat{\hat{\mathbf{C}}})_{ij} = \hat{\hat{c}}_{ij} - sign(\hat{\hat{c}}_{ij})\tau\sqrt{\frac{T}{t}}$ if $|\hat{\hat{c}}_{ij}| \geq \tau\sqrt{\frac{T}{t}}$, 0 otherwise.*

In both these thresholding functions, we have incorporated the time window T inversely proportional to time variable t to account for the estimate updates every T time steps and, hence, assume statistical independence among the extended observations $\tilde{\mathbf{x}}_t$. We elaborate more on this assumption in Section 4.3, where we discuss the stability of L-STVFs and, subsequently, L-STVNNs under perturbations in the estimate of the extended covariance matrix.

4.2.2. LAGGED SPATIOTEMPORAL COVARIANCE NEURAL NETWORKS

The LVNN architecture is inspired by the STVNN architecture in [31] and the VNN architecure in [126]. The module we propose is depicted in Figure 4.1. It can jointly learn spatiotemporal patterns from multivariate time series data by leveraging $\hat{\hat{\mathbf{C}}}_t$ as an inductive bias, resulting in a set of representations for each variable across the time window T of the extended data input $\tilde{\mathbf{x}}_t$. A key notion for developing the LVNN architecture is the Lagged spatiotemporal coVariance Filter (LVF), a polynomial in the extended covariance matrix $\hat{\hat{\mathbf{C}}}_t$.

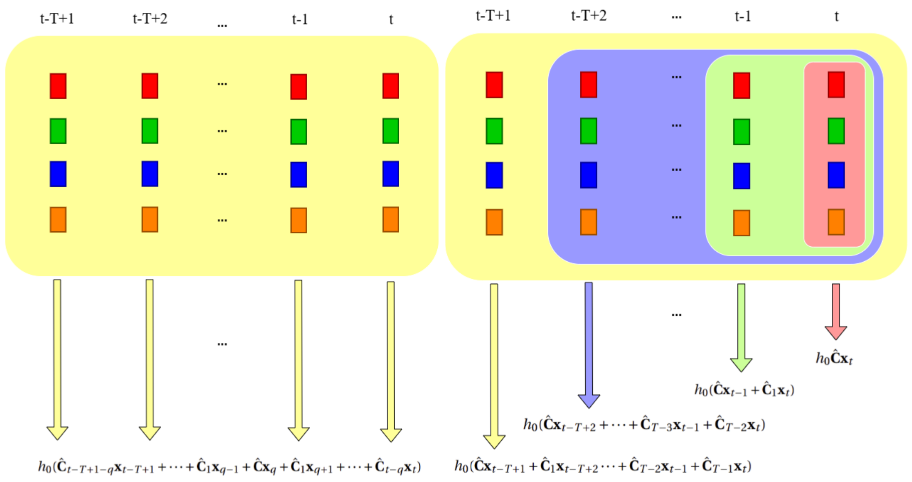
Definition 3 (Lagged spatiotemporal coVariance Filter (LVF)). *The input-output relation of an LVF of order K is defined as*

$$\mathbf{z}_t := \mathbf{H}(\hat{\hat{\mathbf{C}}}_t, \tilde{\mathbf{x}}_t) = \sum_{k=0}^{K} h_k \hat{\hat{\mathbf{C}}}_t^k \tilde{\mathbf{x}}_t \tag{4.5}$$

where h_k are the filter parameters.

The formulation of LVF is analogous to that of the standard graph convoloutional filters [72]. The spatiotemporal propagation depth of the filter is controlled by the order K, leading to a linear combination of signal values that lie at most K hops away in extended covariance graph $\hat{\hat{\mathbf{C}}}_t$. It should be noted that LVF is inherently agnostic to the choice of the time window T, allowing it to seamlessly operate on extended input structures of varying lengths without modification to its core architecture.

The LVF fundamentally differs from the spatiotemporal coVariance filter in [31] in how it models temporal interactions. The LVF leverages the extended sample covariance matrix $\hat{\hat{\mathbf{C}}}_t$, including cross-temporal covariances up to lag $T-1$, to jointly encode both spatial and temporal relationships. This formulation allows LVF to process spatiotemporal dependencies with a single graph filtering step, capturing direct interactions between variables across both space and time dimensions and leading to greater parameter efficiency. The cross-temporal interactions often contain important information about the


t-T+1
t-T+2
...
t-1
t
$h_0(\hat{\mathbf{C}}_{t-T+1-q}\mathbf{x}_{t-T+1} + \cdots + \hat{\mathbf{C}}_1\mathbf{x}_{q-1} + \hat{\mathbf{C}}\mathbf{x}_q + \hat{\mathbf{C}}_1\mathbf{x}_{q+1} + \cdots + \hat{\mathbf{C}}_{t-q}\mathbf{x}_t)$
$h_0\hat{\mathbf{C}}\mathbf{x}_t$
$h_0(\hat{\mathbf{C}}\mathbf{x}_{t-1} + \hat{\mathbf{C}}_1\mathbf{x}_t)$
$h_0(\hat{\mathbf{C}}\mathbf{x}_{t-T+2} + \cdots + \hat{\mathbf{C}}_{T-3}\mathbf{x}_{t-1} + \hat{\mathbf{C}}_{T-2}\mathbf{x}_t)$
$h_0(\hat{\mathbf{C}}\mathbf{x}_{t-T+1} + \hat{\mathbf{C}}_1\mathbf{x}_{t-T+2} \cdots + \hat{\mathbf{C}}_{T-2}\mathbf{x}_{t-1} + \hat{\mathbf{C}}_{T-1}\mathbf{x}_t)$

(a) Full Extended Covariance Matrix

(b) Block Upper Triangular Extended Covariance Matrix

Figure 4.5: The output embeddings of an LVF of order 1, where the input is an extended data input of temporal window T, with the full extended covariance matrix and the corresponding block upper triangular matrix.

underlying dynamics of the multivariate time series that can be used to improve their modeling. On the contrary, in the STVF setting, a bank of T independent spatial filters, each operating on a lagged input $\mathbf{x}_{t-t'}$ with $t' = 0, 1, ..., T-1$ using the same sample lag-zero covariance matrix $\hat{\mathbf{C}}_t$, is followed by a summation of the produced embeddings across time to capture temporal interactions. These independent filter banks for each lag offer greater flexibility in capturing lag-specific dynamics and combining them. Moreover, STVF processes a single covariance matrix of size $N \times N$, while the LVF handles an extended covariance matrix of size $NT \times NT$.

Figure 4.5a illustrates the output of an LVF of order 1 applied to the full extended sample covariance matrix, where the input consists of an extended data sample with temporal window T. Representing the underlying graph structure through the full extended covariance matrix induces a fully connected graph over both variables and time steps. In this configuration, the filter aggregates information across the entire temporal window T, with the contribution of each variable at each time step modulated solely by the corresponding covariance term. Consequently, the LVF is unable to isolate the influence of specific time lags and becomes highly sensitive to the accuracy of the estimated covariance structure. Figure 4.5b illustrates the output of the LVF of order 1 applied to the block upper triangular of the extended sample covariance matrix, as described in Section 4.2.1. This setup naturally induces nested temporal neighborhoods, enabling the model to distinguish and emphasize specific time windows. For example, the embeddings at time t are computed using lag-zero covariance terms and the signal values at time t (pink area), while the embeddings at time $t-1$ incorporate both lag-one and lag-zero covariance terms, as well as the signal values at times t and $t-1$, respectively (green area). Moreover, we observe that the first set of embeddings focuses on the most recent time step and the pertinent spatial covariance terms, while each subsequent set of embeddings increases the temporal window towards the past. This shifts the focus of the LVF to the more recent time steps and allows it to construct embeddings without the potentially misleading effect of past information. By modifying the structure of the covariance matrix over which the LVF operates, this approach enhances the expressiveness of the resulting representations, enabling the model to capture patterns over temporal windows of increasing length.

In the spectral domain, assuming the eigendecomposition $\hat{\tilde{\mathbf{C}}}_t = \hat{\mathbf{V}}\hat{\boldsymbol{\Lambda}}\hat{\mathbf{V}}^\top$ of the extended covariance matrix, we can compute the graph Fourier transform of the LVF output $\bar{\mathbf{z}}_t$ as:

$$\bar{\mathbf{z}}_t = \hat{\mathbf{V}}^\top \mathbf{z}_t = \sum_{k=0}^{K} h_k \hat{\boldsymbol{\Lambda}}^k \hat{\mathbf{V}}^\top \tilde{\mathbf{x}}_t \tag{4.6}$$

which holds from the orthonormality of the eigenvectors. In accordance with the spectral analysis of graph convolutional filters, we can further define the frequency response of LVF for each component i as:

$$[\bar{\mathbf{z}}_t]_i = h(\hat{\lambda}_i)[\tilde{\mathbf{x}}_t]_i. \tag{4.7}$$

This shows that the LVF processes the extended time series data by amplifying or attenuating their principal components. Taking into account the result from [126, Theorem 1], Equation (4.7) shows that there exist filterbanks of narrowband LVFs that enable the recovery of the TPCA transformation. In contrast to STVF in [31], by employing the extended covariance matrix $\hat{\tilde{\mathbf{C}}}_t$, the LVF naturally aligns with the formulation of TPCA and establishes a direct connection with it.

Given the LVF, we can now introduce the LVNN architecture.

Definition 4 (Lagged spatiotemporal coVariance Neural Network (LVNN)). *The LVNN is a layered architecture where each layer $l = 1, \ldots, L$ comprises a LVF nested into a pointwise nonlinearity $\sigma(\cdot)$, i.e.,*

$$\mathbf{z}_t^l = \sigma(\mathbf{H}^l(\hat{\tilde{\mathbf{C}}}_t, \mathbf{h}^l, \mathbf{z}_t^{l-1})) = \sigma(\sum_{k=0}^{K} h_k^l \hat{\tilde{\mathbf{C}}}_t^k \mathbf{z}_t^{l-1}), \qquad \text{for } l = 1, \ldots, L \tag{4.8}$$

with input $\mathbf{z}_t^0 = \tilde{\mathbf{x}}_t$.

In principle, the architecture of LVNNs is similar to the architecture of graph neural networks with the extended covariance matrix $\hat{\tilde{\mathbf{C}}}_t$ as GSO. The inter-layered activation functions increase the expressiveness of LVNNs beyond the linear mappings of LVFs, allowing them to learn joint, non-linear, spatiotemporal representations of the multivariate time series. The output of the last layer constitutes the output of the LVNN $\mathbf{z}_t^L := \boldsymbol{\Phi}(\tilde{\mathbf{x}}_t, \hat{\tilde{\mathbf{C}}}_t; \mathbf{h})$, and comprises the final learned set of embeddings. The learnable parameters $\mathbf{h} = \{h_k^l\}$ span all filter coefficients across all layers and are of order $\mathcal{O}((K+1)L)$. Further, we can enhance the representation capabilities of LVNNs by extending each layer with parallel filter banks [71], able to handle multiple parallel inputs and outputs.

Specifically, at a layer l, a filterbank of $F_{in} \times F_{out}$ filters processes F_{in} input features and transforms them into F_{out} output features as:

$$[\mathbf{z}_t^l]_f = \sigma(\sum_{g=1}^{F_{in}} \mathbf{H}^l(\hat{\tilde{\mathbf{C}}}_t, \mathbf{h}, \mathbf{z}_{t,g}^{l-1})), \qquad \text{for } f = 1, ..., F_{out}, \; l = 1, ..., L, \tag{4.9}$$

where $\mathbf{z}_{t,g}^{l-1}$ refers to the g-th input feature of the embeddings of the l-th layer.

Online learning. The LVNN is designed to operate with streaming data and adapt to distribution shifts by recursively updating the filter parameters in an online manner. Specifically, with a slight abuse of notation, let $\mathbf{h}_t$ denote the concatenation of all coefficients h_k^l for each layer l and order k at time t. Then, at time t, as a new data sample $\mathbf{x}_t$ becomes available, the extended data sample $\tilde{\mathbf{x}}_t$ is constructed and, after the update of the covariance matrix, the model parameters are updated as:

$$\mathbf{h}_{t+1} = \mathbf{h}_t - \eta \nabla_t \mathcal{L}(\mathbf{\Phi}(\tilde{\mathbf{x}}_t, \hat{\tilde{\mathbf{C}}}_t; \mathbf{h}_t)), \tag{4.10}$$

where $\eta > 0$ is the learning rate, and $\mathcal{L}(\cdot)$ is the task-specific loss function.

Computational complexity. According to Equation 4.9, each LVNN layer processes an input of size NT with an extended covariance matrix of size $NT \times NT$, resulting in a computational complexity of the order $\mathcal{O}(N^2T^2KF_{in}F_{out})$. The term N^2T^2, which originates from the size of the extended covariance matrix, degrades the model's performance with large datasets, in terms of both number of variables and long temporal memory. However, practical implementations rely on sparse estimates of large covariance matrices, via sparsification techniques like the ones we discussed in 4.2.1, reducing the computational complexity to $\mathcal{O}(|E|KF_{in}F_{out})$, where $|E|$ is the number of non-zero entries in the sparse extended covariance matrix.

4.3. STABILITY ANALYSIS

In a data streaming setting, the LVNN operates on an extended sample covariance matrix $\hat{\tilde{\mathbf{C}}}_t$, which is estimated from finite data up to time t. This perturbation in the extended covariance matrix results in a subsequent perturbation in the embeddings w.r.t. an LVNN trained with the true extended covariance matrix $\tilde{\mathbf{C}}$. Moreover, the model parameters $\mathbf{h}_t$ are updated online, which requires a sub-optimality regret-like analysis w.r.t. the optimal parameters $\mathbf{h}^*$ over the complete dataset. Here, we consider the robustness of LVNN to both these sources of perturbation, which we refer to as *covariance uncertainty error* and *parameter sub-optimality error* respectively.

The subsequent analysis is formulated under the assumption of stationary time series, where the true extended covariance matrix does not change over time and convergence is feasible. We also consider an extended data sample of time length T every T time steps. This ensures that there is no overlapping between consecutive extended data samples, and, therefore, we can assume they are statistically independent. Furthermore, we adjust the following assumptions from [31] to our setting, which we will use throughout our analysis:

Assumption 1. *The frequency response $h(\lambda)$ of a LVF is Lipschitz. That is, there exists a constant $P > 0$ such that:*

$$\left|h(\lambda_i) - h(\lambda_j)\right| \leq P\left|\lambda_i - \lambda_j\right|, \tag{4.11}$$

for each pair of distinct eigenvalues (λ_i, λ_j) of the extended covariance matrix $\tilde{\mathbf{C}}$.

Assumption 2. *[138, Theorem 5.6.1] For an extended multivariate time series $\tilde{\mathbf{x}}_t$ with extended covariance matrix $\tilde{\mathbf{C}}$ the following holds:*

$$\mathbb{P}\left(\|\tilde{\mathbf{x}}_t\| \leq G\sqrt{\mathbb{E}[\|\tilde{\mathbf{x}}_t\|^2]}\right) \geq 1 - \delta, \tag{4.12}$$

where $G \geq 1$ is a constant, $\delta \approx 0$, and $\|\cdot\|$ is the l2-norm.

Assumption 3. *[100, Theorem 4.1] Given the frequency response $h(\lambda)$ of a LVF, the eigenvalues $\{\lambda_i\}_{i=0}^{NT-1}$ and $\{\hat{\lambda}_i\}_{i=0}^{NT-1}$ of the true and extended sample covariance matrix, respectively, satisfy:*

$$\operatorname{sign}(\lambda_i - \lambda_j)2\hat{\lambda}_i > \operatorname{sign}(\lambda_i - \lambda_j)(\lambda_i + \lambda_j), \tag{4.13}$$

for each pair of distinct eigenvalues (λ_i, λ_j) of the extended covariance matrix $\tilde{\mathbf{C}}$.

The role of each of these assumptions is analogous to that for the stability analysis of STVNN in [31, Section 5.1]. As. 1 limits the discriminability of LVF by upper bounding the change rate of its frequency response with constant P. As. 2 pertains to the variance of the distribution of the extended multivariate data, with higher values of G corresponding to higher variance. As. 3 addresses the approximation error of the extended sample covariance eigenvalues w.r.t the ones of the true extended covariance matrix. It holds for each eigenvalue pair (λ_i, λ_j) with probability at least $1 - 2k_i^2/(N\left|\lambda_i - \lambda_j\right|)$, where $k_i = (\mathbb{E}\left[\left\|\tilde{\mathbf{x}}_t\tilde{\mathbf{x}}_t^\top\mathbf{v}_i\right\|^2\right] - \lambda_i^2)^{1/2}$ is related to the kurtosis of the extended data distribution for each eigenvalue-eigenvector pair $(\lambda_i, \mathbf{v}_i)$ of the extended covariance matrix.

4.3.1. STABILITY OF LAGGED SPATIOTEMPORAL COVARIANCE FILTER

We begin by establishing the stability of LVF, as defined in (4.5). This constitutes a paramount component of this work, as it is essential for the stability analysis of LVNN and allows the comparison with STVNN [31] and TPCA [78] on a theoretical basis.

Theorem 1. *Consider a multivariate time series $\mathbf{x}_t \in \mathbb{R}^N$, define $\tilde{\mathbf{x}}_t \in \mathbb{R}^{N\times T}$ as an extended representation of $\mathbf{x}_t$ from T consecutive time steps with an underlying extended covariance matrix $\tilde{\mathbf{C}}$ and let the extended sample covariance matrix estimated until time step t be $\hat{\tilde{\mathbf{C}}}_t$. Additionally, let the instances satisfy w.l.o.g. $\|\tilde{\mathbf{x}}_t\| \leq 1$, let assumptions 1-3 hold, and let the learning rate $\eta > 0$ be small enough to guarantee convergence. Denote also the filter parameters optimized over the complete dataset by $\mathbf{h}^*$ and those optimized online until time step t using the online update in (4.10) by $\mathbf{h}_t$. Then, the following holds with* *probability at least* $(1-e^{-\epsilon})(1-2e^{-u})$:

$$\left\|\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}_t,\tilde{\mathbf{x}}_t)-\mathbf{H}(\tilde{\mathbf{C}},\mathbf{h}^*,\tilde{\mathbf{x}}_t)\right\| \leq$$
$$\underbrace{\frac{1}{\sqrt{t}}PT^{3/2}N\left(k_{\max}e^{\epsilon/2}+QG\left\|\tilde{\mathbf{C}}\right\|\sqrt{\log(NT)+u}\right)}_{\text{covariance uncertainty}} \quad + \quad \underbrace{\frac{T\left\|\mathbf{h}^*\right\|^2}{2\eta t}}_{\text{parameter sub-optimality}} \quad + \quad \mathcal{O}\left(\frac{1}{t}\right), \tag{4.14}$$

where Q *is an absolute constant,* $k_{max} = max_j k_j$, *and* $k_j = (\mathbb{E}\left[\left\|\tilde{\mathbf{x}}_t\tilde{\mathbf{x}}_t^\top\mathbf{v}_j\right\|^2\right]-\lambda_j^2)^{1/2}$ *is related to the kurtosis of the data distribution for each eigenvalue-eigenvector pair* $(\lambda_j,\mathbf{v}_j)$ *of the extended covariance matrix. Here,* $\epsilon, u > 0$ *can be arbitrarily large and* $\|\cdot\|$ *denotes the* l_2*-norm for vectors and the spectral norm for matrices.*

Proof. See Appendix A.1.

The result highlights the effect of the online updates in the extended covariance matrix and the parameters on the stability of LVF. For comparison, we report the stability bound of STVF [31, Theorem 1]:

$$\left\|\mathbf{H}(\hat{\mathbf{C}}_t,\mathbf{h}_t,\mathbf{x}_{T:t})-\mathbf{H}(\mathbf{C},\mathbf{h}^*,\mathbf{x}_{T:t})\right\| \leq$$
$$\underbrace{\frac{1}{\sqrt{t}}PTN\left(k_{\max}e^{\epsilon/2}+QG\left\|\mathbf{C}\right\|\sqrt{\log N+u}\right)}_{\text{covariance uncertainty}} \quad + \quad \underbrace{\frac{\left\|\mathbf{h}^*\right\|^2}{2\eta t}}_{\text{parameter sub-optimality}} \quad + \quad \mathcal{O}\left(\frac{1}{t}\right), \tag{4.15}$$

With respect to this upper bound in (4.15) and our result in (4.14), we make the following observations:

Number of time samples. Focusing on the covariance uncertainty term in (4.14), we observe that it decreases with a rate $\mathcal{O}(1/\sqrt{t})$. Given that the parameter sub-optimality term decreases with a rate of $\mathcal{O}(1/t)$ (along with the final term of our upper bound), we deduce that the scale factor of the upper bound is mainly determined by the covariance uncertainty term. The upper bound of STVF in (4.15) showcases the same behavior.

Temporal window size. As expected, a larger time window T leads to a lower stability for the LVF. As the extended covariance matrix grows quadratically with T, there are more elements that need to be estimated and, hence, an increased uncertainty in the extended covariance estimation. With respect to the STVF upper bound in (4.15), we first focus on the covariance uncertainty term. Besides the spectral norm of the extended covariance matrix $\left\|\tilde{\mathbf{C}}\right\|$, which replaces the spectral norm of the lag-zero covariance matrix $\|\mathbf{C}\|$, the main differences between the two bounds are w.r.t to the temporal window T. Specifically, we observe the LVF covariance uncertainty bound depends on $T^{3/2}$, in contrast to T for STVF. This additional $T^{1/2}$ factor originates from our update rate every T time steps, to ensure statistical independence. Moreover, having NT nodes in our graph representation leads to T being incorporated into [138, Theorem 5.6.1] and, consequently, to the factor $\sqrt{\log(NT)+u}$ instead of $\sqrt{\log N+u}$. As a result, the stability of LVF is more severely affected as T increases w.r.t. the stability of STVF. Finally, regarding the parameter sub-optimality term, we observe an additional T factor which again originates from our sampling rate assumption, however it does not affect the filter bound as much as the covariance uncertainty term.

Block upper triangular. We aim to establish a similar upper stability bound for the block upper triangular extended sample covariance matrix (which we denote as $\hat{\tilde{\mathbf{U}}}$). The parameter sub-optimality error remains the same, however directly deriving a covariance uncertainty error bound involving the spectral norm is challenging. To address this, we perform a similar analysis of the covariance uncertainty error for the block upper triangular matrix, and derive the following upper bound:

$$\left\| \mathbf{H}(\hat{\tilde{\mathbf{U}}}_t, \mathbf{h}^*, \tilde{\mathbf{x}}_t) - \mathbf{H}(\tilde{\mathbf{U}}, \mathbf{h}^*, \tilde{\mathbf{x}}_t) \right\| \leq P\sqrt{NT}\, \|\mathbf{E_U}\|\, (1+\sqrt{NT}) \;+\; \mathcal{O}(\|\mathbf{E_U}\|^2). \tag{4.16}$$

The proof for this upper bound is included in Appendix A.2 Then, we leverage two key properties from norm equivalences. First, we use the fact that the spectral norm of any matrix is upper bounded by the corresponding Frobenius norm, i.e., $\|\mathbf{A}\| \leq \|\mathbf{A}\|_F$. Second, a matrix $\mathbf{A}'$, that is obtained by zeroing out entries of a matrix $\mathbf{A}$, cannot have a larger Frobenius norm, i.e., $\left\|\mathbf{A}'\right\|_F \leq \|\mathbf{A}\|_F$. Using the first property on the estimation error of the block upper triangular matrix and the second property on its relationship with the estimation error of the full extended covariance matrix, we derive:

$$\|\mathbf{E_U}\| \leq \|\mathbf{E}\|_F \tag{4.17}$$

Adjusting [23, Theorem 4] to our setting, and taking into account that the extended sample covariance matrix is dense and our sampling rate is T, we get:

$$\|\mathbf{E}\|_F \leq QT\sqrt{\frac{N}{t}}, \tag{4.18}$$

where Q is a constant. Using the inequalities 4.17 and 4.18 in 4.16, and ignoring the last error term $\mathcal{O}(\|\mathbf{E}\|^2)$ since we consider that it becomes too small after a few training steps ($t \gg 0$), we derive:

$$\left\| \mathbf{H}(\hat{\tilde{\mathbf{U}}}_t, \mathbf{h}^*, \tilde{\mathbf{x}}_t) - \mathbf{H}(\tilde{\mathbf{U}}, \mathbf{h}^*, \tilde{\mathbf{x}}_t) \right\| \leq \frac{1}{\sqrt{t}} QPT^{3/2} N(1+\sqrt{NT}). \tag{4.19}$$

We observe that this bound scales quadratically with T, meaning that using the block upper triangular matrix leads to lower stability than using the full extended covariance matrix. However, this bound also decreases with a rate $\mathcal{O}(1/\sqrt{t})$, as we process more extended data samples of the time series.

4.3.2. STABILITY OF LVF WITH THRESHOLDING SPARSIFICATION

Then, we extend the stability analysis of LVF by incorporating hard-thresholding sparsification, as defined in Section 4.2.1. Here, the optimal parameters $\mathbf{h}^*$ are optimized over the complete dataset using the hard-thresholded true extended covariance matrix. As a result, this technique will not affect the parameter sub-optimality error, but only the error caused by the extended covariance estimation.

Theorem 2. *Let the true extended covariance $\tilde{\mathbf{C}}$ belong to the sparse class $\mathcal{C} = \{\tilde{\mathbf{C}} : \tilde{c}_{ii} \leq M,\ \sum_{j=1}^{NT} \mathbb{1}[\tilde{c}_{ij} \neq 0] \leq c_0, \forall\}$ where $M > 0$ is a constant, $\mathbb{1}(\cdot)$ is the indicator function, and c_0 is the maximum number of non-zero elements in each row of $\tilde{\mathbf{C}}$. Consider a hard-thresholded extended sample covariance matrix $\hat{\tilde{\mathbf{C}}}_t$ following the definition in Def. 1 with $\tau = M'\sqrt{NT}$ and M' large enough. With a high probability, it holds that:*

$$\left\| \mathbf{H}(\hat{\tilde{\mathbf{C}}}_t, \mathbf{h}_t, \tilde{\mathbf{x}}_t) - \mathbf{H}(\tilde{\mathbf{C}}, \mathbf{h}^*, \tilde{\mathbf{x}}_t) \right\| \leq$$

$$\underbrace{\frac{1}{\sqrt{t}} P c_0 T \sqrt{N \log(NT)} \left(1 + \sqrt{NT}\right)}_{\text{covariance uncertainty}} + \underbrace{\frac{T \|\mathbf{h}^*\|^2}{2\eta t}}_{\text{parameter sub-optimality}} + \mathcal{O}\left(\frac{1}{t}\right), \tag{4.20}$$

Proof. See Appendix A.2.

Comparison with LVF bound. Comparing the result in (4.20) with the previous one for the standard LVF in (4.14), we notice that the covariance uncertainty term decreases again with a rate of $\mathcal{O}(1/\sqrt{t})$, dominating the stability bound for large t. Then, we observe that the role of time window T and number of variables N remains the same across the thresholded and non-thresholded LVF. Specifically, expanding the covariance uncertainty terms of both filters and focusing only on variables T and N, we see that the most intensive term would contain $T^{3/2} N \sqrt{\log(NT)}$ in both cases. However, the main advantage of the hard-thresholded LVF is the replacement of the spectral norm of the extended covariance matrix $\|\tilde{\mathbf{C}}\|$ with c_0, the maximum number of non-zero elements in each of its rows. Working with the assumption that the true extended covariance matrix is sparse, we have $c_0 \ll \|\tilde{\mathbf{C}}\|$, therefore hard thresholding improves the stability of LVF. This is analogous to the observation in [30], where the stability of a hard-thresholded VNN is compared to that of a dense VNN.

Then, in a similar manner, we proceed with the stability analysis of LVF with soft-thresholding sparsification, as defined in Section 4.2.1. Soft thresholding has been found to provide more reliable covariance estimates when the underlying data follows the spiked covariance estimates [41]. This means that extended data points follow:

$$\tilde{\mathbf{x}}_i = \sum_{q=1}^{r} \sqrt{\beta_q} u_{q,i} \mathbf{v}_q + \mathbf{z}_i, \tag{4.21}$$

where $\mathbf{v}_i, \ldots, \mathbf{v}_r \in \mathbb{R}^{NT}$ are orthonormal vectors with exactly c_0 non-zero entries of magnitudes with a lower bound θ / c_0 for some constant $\theta > 0$, $u_{g,i} \sim \mathcal{N}(0,1)$ and $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ are i.i.d., and $\beta_q \in \mathbb{R}_+$ is a measure of the signal-to-noise ratio. Again, we re-define the optimal parameters $\mathbf{h}^*$ as those optimized over the complete dataset with the soft-thresholded extended covariance matrix.

Theorem 3. *Consider a soft-thresholded estimate of the extended covariance matrix $\hat{\hat{\mathbf{C}}}_t$ as per Def. 2, with $\tau = M'\sqrt{\log(NT/c_0^2)}$ and M', C two large enough constants. Let the eigenvalues of the true extended covariance matrix $\{\lambda_i\}_{i=0}^{NT-1}$ be all distinct. Then, the following holds with high probability:*

$$\left\|\mathbf{H}(\hat{\hat{\mathbf{C}}}_t,\mathbf{h}_t,\tilde{\mathbf{x}}_t)-\mathbf{H}(\tilde{\mathbf{C}},\mathbf{h}^*,\tilde{\mathbf{x}}_t)\right\| \leq$$
$$\underbrace{\frac{1}{\sqrt{t}}PT\sqrt{N}Cc_0\max(1,\lambda_{\max})\sqrt{\max(\log\frac{NT}{c_0^2},1)}\left(1+\sqrt{NT}\right)}_{\text{covariance uncertainty}}+\underbrace{\frac{T\|\mathbf{h}^*\|^2}{2\eta t}}_{\text{parameter sub-optimality}}+\mathcal{O}\left(\frac{1}{t}\right), \tag{4.22}$$

Proof. See Appendix A.3.

Comparison with hard-thresholded LVF bound. The main takeaways of the LVF with soft thresholding, w.r.t. the stability of the standard LVF in (4.14), are the same as the ones for the hard thresholding. Consequently, we focus on the comparison of the two thresholded filters. Analogous to [30], we observe that soft thresholding provides better stability than hard thresholding in low-data settings under spiked covariance, since the hard-thresholded bound contains $\sqrt{\log(NT)}$, while the soft-thresholded bound contains the smaller term $\sqrt{\log(NT)/c_0^2}$.

4.3.3. STABILITY OF LVNN

Finally, we extend the stability analysis w.r.t to the covariance uncertainty to a LVNN, composed of L layers and F filterbanks per layer. We only account for the covariance uncertainty error and not for the sub-optimality of the parameters, because the LVNN function is highly non-convex and we cannot guarantee the convergence of the parameters.

Theorem 4. *Consider a true extended covariance matrix $\tilde{\mathbf{C}}$, a extended sample covariance matrix $\hat{\hat{\mathbf{C}}}_t$ and a bank of LVF with frequency response terms $|h(\lambda)| \leq 1$ and nonlinearity $\sigma(\cdot)$ such that $|\sigma(a)-\sigma(b)| \leq |a-b|$. Given a generic β_t, if the filters satisfy $\left\|\mathbf{H}(\hat{\hat{\mathbf{C}}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t) - \mathbf{H}(\tilde{\mathbf{C}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)\right\| \leq \beta_t$, then the LVNN satisfies:*

$$\left\|\mathbf{\Phi}(\hat{\hat{\mathbf{C}}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)-\mathbf{\Phi}(\tilde{\mathbf{C}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)\right\| \leq LF^{L-1}\beta_t. \tag{4.23}$$

The proof follows directly from [47, Theorem 4], for a generic β_t. In our case, β_t is the bound corresponding to the first term on the r.h.s. of (4.14), if we do not employ any sparsification techniques on the extended sample covariance matrix, or to the first term on the r.h.s. of (4.20) or (4.22), if we apply hard or soft thresholding sparsification respectively. In any case, this bound decreases with the number of extended data samples with rate $\mathcal{O}(1/\sqrt{t})$.

Comparison with STVNN bound. The comparison between the bound in (4.23) and that of STVNN in [31, Theorem 2] reduces to the comparison of their respective coVariance filters. Since the bound of STVF grows linearly with T and the bound of LVF grows super-linearly (specifically $T^{3/2}\sqrt{\log T}$), LVNN is less stable than STVNN w.r.t T. However, LVNN remains a more stable approach than a VNN model with the temporal values as node features, which would have a stability bound of $L(FT)^{L-1}$ [31]. Specifically, from [126, Theorem 3], the bound of the VNN model would grow quadratically with T, w.r.t to the number of layers, while the LVNN bound grows sub-quadratically.

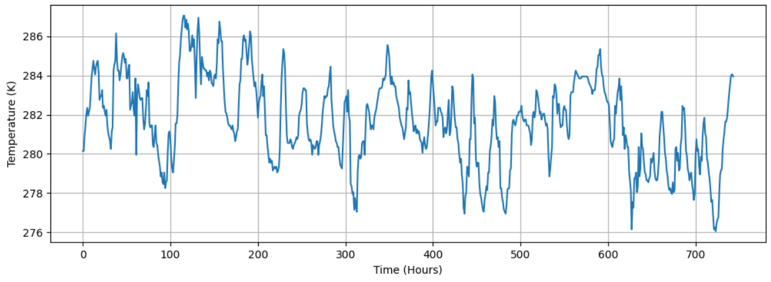
Comparison with TPCA bound. Further, we compare the stability of LVNN with the stability of online TPCA w.r.t the uncertainties in the extended covariance matrix. It is easy to show that the bound of online TPCA inversely depends on the smallest difference between eigenvalues of the covariance matrix, by working in a similar manner as in the stability proof [31, Proposition 1] for online PCA. This eigenvalue difference causes instabilities when these eigenvalues are close [80], by exploding the stability bound of online TPCA. However, via As. 1, we diminish this effect with the Lipschitz constant, which guarantees the output stability by limiting the filter's discriminability. This shows that LVNN shows improved stability compared to online TPCA, while the non-linear mappings of LVNN improve the discriminability of LVF and, hence, address this undesirable effect of achieving improved stability.

4.4. CONCLUSION

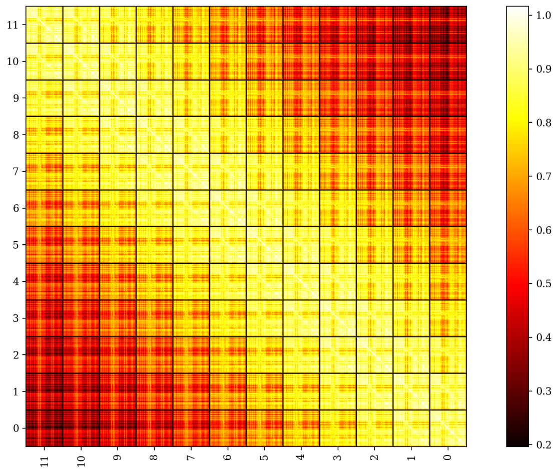
To summarize, we introduced a novel architecture for multivariate time series modeling, the Lagged spatiotemporal coVariance Neural Network (LVNN), which leverages spatiotemporal convolutions with an online estimation of the extended covariance matrix as inductive bias to represent these interactions. The extended covariance matrix contains spatial and cross-temporal relationships between the time series with a lag $\tau = 0, 1, \ldots, T-1$, allowing LVNN to directly capture dynamic spatiotemporal dependencies with a single graph filtering operation. We demonstrated that this architecture enables principled processing aligned with temporal PCA, while benefiting from the robustness and flexibility of graph-based convolutional methods. Furthermore, we theoretically established the stability of LVNN under perturbations in the extended covariance and the online update of the parameters, positioning it between online TPCA and STVNNs in terms of stability-performance trade-offs. In the next chapter, we empirically evaluate LVNN across forecasting tasks and compare it to relevant baselines.

5 NUMERICAL EXPERIMENTS

In this chapter, we will present the numerical experiments and results carried out for evaluating the performance of our proposed LVNN model. Initially, we detail the experimental settings in Section 5.1, elaborating on the selected datasets and the design of our experimental procedure. In Section 5.2, we present our experimental results, aiming to practically confirm the stability of LVNN, benchmark its forecasting performance against other baseline models, and examine the effects of thresholding sparsification on the extended sample covariance matrix. Finally, in Section 5.3, we offer a discussion of the results.



(a) Temperature recordings at a station in Brest, France



(b) Extended covariance matrix

Figure 5.1: The time series of the temperature recordings on a specific station and the extended covariance matrix over the complete Molene dataset up to a lag of 12 hours.

5.1. EXPERIMENTAL SETUP

In this section, we outline the main components of our experimental environment. First, in Subsection 5.1.1, we introduce the 3 real-world datasets we are using in our experiments and discuss their covariance dynamics and the data preprocessing and splits. Then, in Subsection 5.1.2, we discuss the baseline models that we will compare the LVNN against. In Subsection 5.1.3, we present the loss metrics we employ for evaluating our model's performance. Afterwards, in Subsection 5.1.4, we describe the details of the LVNN model evaluation pipeline and the hyperparameter search.

5.1.1. DATASETS

To evaluate our model, we consider three real-world datasets of different variable and temporal sizes, resolutions, and dynamics. A brief overview of each follows here.

MOLENE DATASET

The first dataset we consider in our work is the **Molene dataset** [56]. The original data were collected as part of the Archipel Molene operation from the French national meteorological service in January 2014, containing three types of measurements (wind, temperature, and air) for a number of weather stations in the area of Brest, France. Specifically, it contains 744 hourly measurements across $N = 37$ stations. Figure 5.1a shows the hourly temperature measurements at a specific station.

In this work, we only consider the temperature measurements of the weather stations. This is the smallest dataset among the ones included in this work, in terms of total observations per variable (station). As a result, we would expect that a strong prior, such as an adjacency matrix based on station coordinates, would significantly aid learning. However, in our streaming data setting, the size of the dataset would pose a challenge for capturing the underlying dynamics with the extended sample covariance matrix.

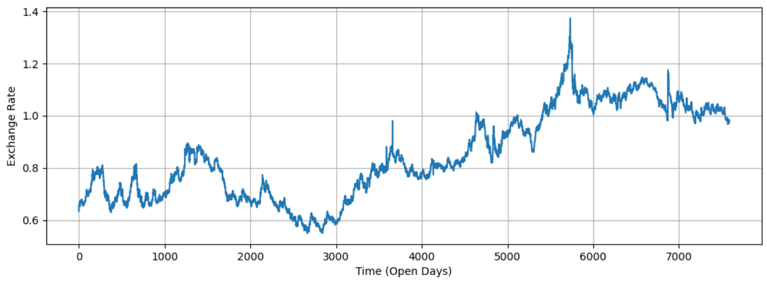
Figure 5.1b depicts the extended covariance matrix for the complete Molene dataset up to lags of 12 hours. First, we observe that the time series exhibit high positive covariances for lags up to 4 hours. This indicates that the temperatures are strongly correlated across nearby time steps in an hourly resolution, justified by the proximity of the stations (all located in the same geographical area) and the gradual changes in temperature over time. Then, as the lags between the series increase up to 10 hours, the temperature covariances across the stations tend to significantly decrease. This decay in covariance suggests that the influence of past temperature values weakens over time.

EXCHANGE RATE DATASET

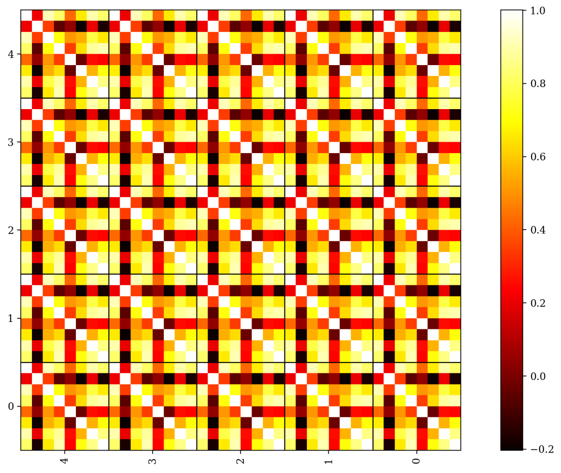
The second dataset we consider is the **Exchange-Rate** [156]. It contains the daily exchange rates of $N = 8$ countries' currencies (Australia, UK, Canada, Switzerland, China, Japan, New Zealand, and Singapore) against the US dollar. These exchange rates are recorded for 7588 open days, from 1990 to 2016. Because of the nature of exchange rates, the corresponding time series do not often exhibit abrupt changes, but periods of high volatility tend to cluster together. Similar to other financial time series, they are typically non-stationary. Figure 5.2a shows the daily fluctuations of the exchange rate of the Swiss franc within this period.

Among the selected datasets, this is the smallest one in terms of variables (countries' currencies) and significantly larger than Molene w.r.t the available observations per variable. Consequently, we expect that there are enough data for accurately estimating the extended covariance matrix in the online setting. Moreover, the small number of variables means that there are in general less cross-temporal covariance terms to be estimated and, hence, a smaller risk of incorporating spurious correlations.

Figure 5.2b depicts the extended covariance matrix for the complete exchange rate dataset up to lags of 7 days. Notably, the covariance blocks remain remarkably consistent across all lags, indicating that the statistical relationship between exchange rate values is largely invariant to temporal differences. This homogeneity reflects the smooth temporal dynamics of financial markets, where dependencies evolve slowly and are less likely to exhibit sudden changes across short time lags. Moreover, in comparison to the Molene extended covariance matrix in Figure 5.1b, the covariance values within the same temporal block are more distinct, ranging from highly positive (close to 1) to slightly negative (close to -0.2).



(a) Swiss franc exchange rate



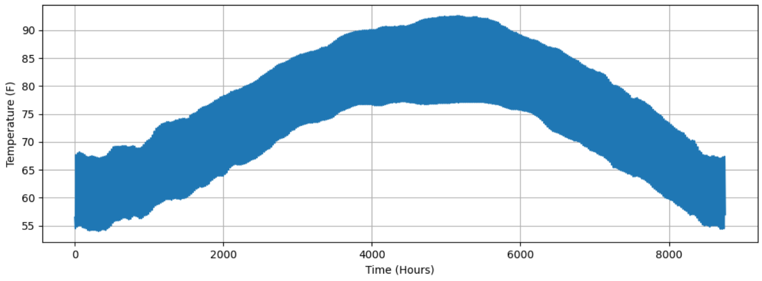
(b) Extended covariance matrix

Figure 5.2: The time series of the daily exchange rate of the Swiss franc against the US dollar from 1990 to 2016 and the extended covariance matrix over the complete exchange rate dataset up to a lag of 7 days.

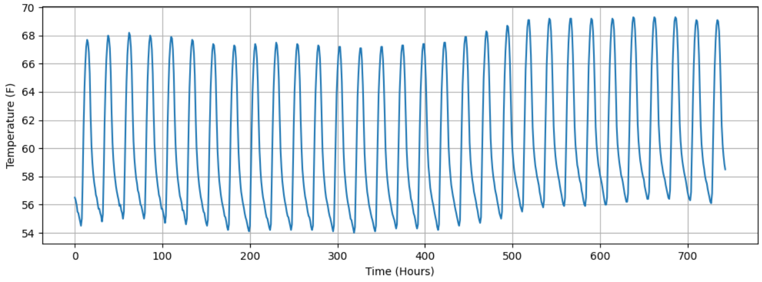
NOAA DATASET

The third dataset we consider is the **NOAA dataset** [4]. This dataset has been issued by the National Oceanic and Atmospheric Administration as part of an effort led by the World Meteorological Organization to compute climatological standard normals every 30 years. It contains 8579 hourly temperature measurements across $N = 109$ weather stations in the United States, recorded over the year 2010. We present two plots of the temperature measurements of a specific station, to gain a better insight into the dynamics of the time series. Figure 5.3a shows the hourly temperature recordings of the station across the year, where we can observe seasonal temperature patterns, with a rise during the summer months and a decline during the winter months. Then, Figure 5.3b shows the temperature measurements for the first month of 2010, where we can observe daily cyclical patterns of temperature fluctuations across the 31 days of January.

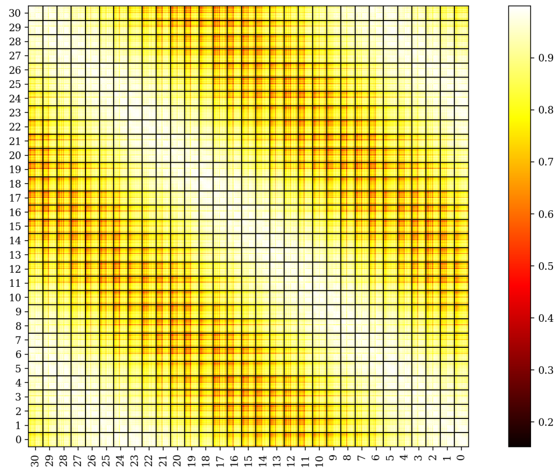
Among the selected datasets, the NOAA dataset is the largest both in terms of variables (stations) and total observations per variable. Similar to the Exchange-Rate dataset,



(a) Temperature recordings at a US station across a year



(b) Temperature recordings at a US station across a month



(c) Extended covariance matrix

Figure 5.3: The time series of the temperature recordings on a specific US station, across the year and the first month of 2010, and the extended covariance matrix over the complete NOAA dataset up to a lag of 31 days.

the number of observations seems sufficient for accurately estimating the extended covariance matrix in an streaming fashion. However, the large number of variables means that there is probably an increased presence of spurious cross-temporal covariance terms, which might negatively impact the model's forecasting performance.

Figure 5.3c depicts the extended covariance matrix for the complete NOAA dataset up to lags of 31 days. Initially, the time series exhibit very high positive covariance up to time lags of 5 days, which is subsequently weakened up to time lags of 21 days. This behavior agrees with that of the Molene dataset, despite the different resolutions between them. Afterwards, we observe another period of very high positive covariance among the time series for time lags between 22 and 26 days, which is indicative of long-term temperature patterns.

DATA PRE-PROCESSING AND SPLIT

In the data preprocessing stage, we begin by loading the raw time series data and then apply standard normalization to ensure consistent scaling across the series. Specifically, we fit a *Standard Scaler* on the complete datasets to compute the mean and standard deviation, and then use it to transform the data. This is crucial for training the LVNN model and our baselines, as it ensures that all variables contribute equally during optimization by preventing those with larger numerical ranges from dominating the learning process and improving convergence. After normalization, for the LVNN model, we reshape the data by concatenating successive time steps into extended data samples based on the pre-defined temporal window T. This converts the series into overlapping sequences that will serve as input to the LVNN, where each sample contains T consecutive time steps. Despite assuming a sampling rate T in our theoretical analysis in Chapter 4 for stationary settings, the non-stationary nature of our real-world datasets allows us to consider an extended data sample at each time step. Finally, we also locate and construct the target forecasting values, based on the forecasting horizon τ.

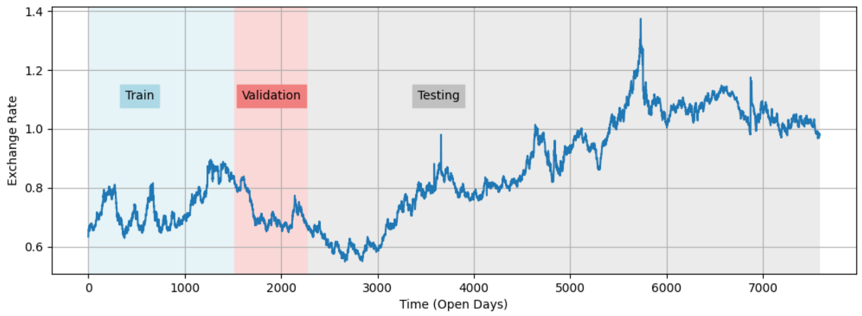


Figure 5.4: An example of the data split on the Swiss franc exchange rate. The first 20% of the time series is used for training, the subsequent 10% is used for validation, and the last 70% is used for testing.

For the data split, we allocate 20% of the dataset for training, 10% for validation, and the remaining 70% for testing, hence simulating a streaming data setting. Since we are working with time series data, it is essential to preserve the temporal order during the split to avoid data leakage and to ensure that the model is tested on actually unseen future observations. Maintaining the chronological structure is also essential because of the changing dynamics and distributional shifts of the selected datasets. It allows the model to learn how to adapt to evolving patterns and changing statistical properties over time, which is a key aspect of online learning in non-stationary environments. Figure 5.4 shows an example of this data split on the time series of the Swiss franc exchange rate.

5.1.2. BASELINE MODELS

To contextualize the performance of LVNN, we benchmark it against various baselines on single-step forecasting with different forecasting horizons, using the aforementioned datasets. Below, we group these models according to the classification provided in Chapter 3.

PCA-BASED METHODS

- **TPCA** [79]. This model initially processes the multivariate time series data with temporal PCA to extract spatiotemporal features. These features are then passed as input to an MLP for the forecasting task. Besides the comparison on single-step forecasting in 5.2.2, we also compare the learned embeddings of TPCA with the ones of LVNN in 5.2.1, in order to corroborate the stability of LVNN against TPCA.

SEQUENTIAL MODELS

- **LSTM**. This is a vector LSTM model that is trained online on the multivariate time series. This comparison allows to showcase the importance of covariance networks as inductive biases for spatiotemporal relational learning.

COVARIANCE-BASED GNN METHODS

- **VNN** [126]. This is the original VNN model with the previous T time snapshots of the multivariate time series being treated as node features. As a tabular covariance-based GNN model that ignores temporal relationships, the comparison with this model highlights the importance of explicitly learning these temporal dependencies.

- **VNNL (VNN-LSTM)**. This is a hybrid disjoint model, where a static VNN is followed by a shared LSTM across the different time series. We compare the joint spatiotemporal learning module of LVNN with the distinct learning modules of VNNL, separately handling the spatial and temporal dependencies.

- **STVNN** [31]. This is the original STVNN model, performing T independent spatial convolutions with the lag-zero sample covariance matrix at the previous T time snapshots and, then, summing the learned embeddings across time. The LVNN replaces the temporal sum of STVNN by employing the extended sample covariance matrix of the multivariate series, containing covariance terms at time lags $\tau = 0, 1, \ldots, T-1$. Therefore, we aim to compare these two approaches, highlighting the importance of including cross-temporal dependencies in the graph filtering operation.

5.1.3. Evaluation Metrics

To evaluate the forecasting performance of LVNN and the baseline models, we use three complementary metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and symmetric Mean Absolute Percentage Error (sMAPE) [120]. This combination allows us to capture both the scale of absolute errors and the model's behavior relative to the predicted and true values. These metrics provide a balanced perspective on accuracy, robustness, and interpretability, and are particularly suited for time series tasks.

- **Mean Squared Error** is a common metric that measures the average of the squared differences between predicted and actual values. Formally, for a set of predictions $\hat{\mathbf{y}}$ and corresponding true values $\mathbf{y}$, the MSE is defined as:

  $$\mathrm{MSE} = \frac{1}{n}\sum_{i=0}^{n}(\hat{\mathbf{y}}_i - \mathbf{y}_i)^2. \tag{5.1}$$

  By squaring the errors, MSE places greater weight on larger deviations, making it particularly sensitive to outliers. Therefore, this differentiable loss function is a more useful metric when large errors are especially undesirable.

- **Mean Absolute Error** calculates the average absolute difference between predicted and actual values, providing a straightforward measure of prediction accuracy. It is defined as:

  $$\mathrm{MAE} = \frac{1}{T}\sum_{i=0}^{n}\left|\hat{\mathbf{y}}_i - \mathbf{y}_i\right|. \tag{5.2}$$

  Unlike MSE, MAE treats all errors equally, regardless of their magnitude, making it more robust to outliers. Moreover, it is easier to interpret as it is expressed in the same units as the original data.

- **Symmetric Mean Absolute Percentage Error** is a scale-independent metric that expresses the prediction error as a percentage of the average magnitude of actual and predicted values. It is given by:

  $$\mathrm{sMAPE} = \frac{100}{n}\sum_{i=0}^{n}\frac{\left|\hat{\mathbf{y}}_i - \mathbf{y}_i\right|}{(\left|\mathbf{y}_i\right| + \left|\hat{\mathbf{y}}_i\right|)/2}. \tag{5.3}$$

  sMAPE addresses the limitations of traditional MAPE, particularly its instability when actual values are near zero. By symmetrically normalizing the error, sMAPE provides a more balanced and meaningful percentage error measure, especially for time series data with varying scales.

In Section 5.2, we focus primarily on sMAPE. However, results for all three metrics are provided in Appendix E for a more comprehensive evaluation of the model's performance.

5.1.4. Experimental Design

The model evaluation pipeline begins with loading, preprocessing, and splitting the dataset as described previously. All procedures of our framework (training, validation, and testing) are conducted with a batch size of 1 to allow continuous updates and mimic an online learning setting. At each step, the extended sample covariance matrix $\hat{\hat{\mathbf{C}}}_t$, operating as a GSO for the LVNN, is dynamically updated with the new available extended data sample. During training, the principal eigenvalue of $\hat{\hat{\mathbf{C}}}$ tends to become large for some of the datasets. To ensure numerical stability, we keep the scale of $\hat{\hat{\mathbf{C}}}_t$ consistent over time by applying trace-normalization (therefore using $\hat{\hat{\mathbf{C}}}_t/\mathrm{trace}(\hat{\hat{\mathbf{C}}}_t)$). We select MSE as the training loss for backpropagation. After each epoch, we transition to the validation phase of the model evaluation pipeline. The model with the best validation MAE across all epochs is selected and used for testing. Before proceeding to testing, the GSO of the selected model is re-initialized with the extended sample covariance matrix computed from the combined training and validation data. This ensures that the model begins testing with a comprehensive view of the prior dynamics of the time series.

Category	**Hyperparameter**	**Search Space**
Training	Learning rate	$\{0.001, 0.0001\}$
	Online coefficient γ	$\{0.01, 0.05, 0.1, 0.3\}$
	Optimizer	Adam, SGD
Architecture	Number of layers	$\{1, 2, 3\}$
	Feature size per-layer	$\{8, 16, 32, 64, 128, 256\}$
	Order of graph filters K	$\{1, 2, 3, 4\}$
	Temporal memory T	$\{2, 3, 5, 8, 12\}$

Table 5.1: A tabular description of the hyperparameter search for the numerical experiments.

To perform forecasting, we produce a weighted sum of the embeddings generated by LVNN and, then, apply a 2-layer MLP on it. This allows us to draw a more direct comparison with the STVNN model, which uses the same post-embeddings component for single-step forecasting. We perform hyperparameter optimization for the LVNN and the baseline models on each of the datasets in 5.1.1 and for three forecasting horizons $\{1, 3, 5\}$. To achieve this, we conduct a hyperparameter search over a predefined search space, allowing us to identify the most suitable configuration for our model. The full set of hyperparameter values considered during the search is summarized in Table 5.1. The LVNN and STVNN, among the baselines, are optimized over all the hyperparameters presented in Table 5.1. For VNN and VNN-LSTM, we use the best corresponding hyperparameter configuration of STVNN. For TPCA and LSTM, we use a fixed architecture and only optimize the temporal memory T. All models are trained for 80 epochs on the Molene dataset and for 40 epochs on the exchange rate and NOAA datasets. The selected non-linearity is LeakyReLU with a negative slope of 0.1. The average performance and standard deviation over 5 experiments are reported. For each dataset and forecasting horizon, the best hyperparameter configuration for the LVNN and the baselines is provided in Appendix C. All tests were performed with Pytorch v2.1.0, using a 13th Gen Intel(R) Core(TM) i9-13900H CPU (2600 Mhz, 14 Cores, 20 Logical Processors).

5.2. RESULTS

This section presents the empirical results from the experiments conducted. The results are naturally divided w.r.t the research questions they are aiming to answer. In Subsection 5.2.1, we corroborate the stability of the LVNN when trained online with finite samples against TPCA. Then, in Subsection 5.2.2, we compare the forecasting performance of LVNN against the baseline models. Finally, in Subsection 5.2.3, we evaluate the effect of hard and soft thresholding techniques on the performance of LVNN. The main discussion on the results is provided in Section 5.3.

5.2.1. Stability Analysis

Motivated by our theoretical analysis in Section 4.3, the aim of this experiment is to assess the stability of the proposed LVNN model under varying temporal memory window sizes T. Stability here refers to the robustness of LVNN to perturbations in the extended covariance matrix and, consequently, in the learned embeddings, as discussed in Section 4.3. To evaluate stability, we compute the difference between the embeddings learned with the online-estimated and the true extended covariance matrices.

Our first experiment is conducted in a stationary environment, consistent with the setting of our theoretical analysis. To achieve this, we construct several synthetic stationary datasets. Initially, we sample observations $\mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$, where $\mathbf{C}$ is a fixed covariance matrix. We then construct the time series by enforcing temporal causality over the sampled observations as $\mathbf{x}_t = \sum_{t'=0}^{\tau} h_{t'} \mathbf{z}_{t-t'}$, where $h_{t'} = h'_{t'} / \sqrt{\sum_t h'_t}$ and $h'_t = e^{-t}$ for $t = 0, \ldots, \tau$. We set $\tau = 9$ for our experiments. Three synthetic datasets are generated, each with covariance matrix $\mathbf{C}$ having different eigenvalue tail sizes $(0.1, 0.5, 0.9)$, reflecting differences in the kurtosis and eigenvalue closeness. A more detailed description of the synthetic dataset generation process is provided in Appendix B. Each dataset is split 20/10/70 into training, validation, and test sets, respectively, and the LVNN is run with different values of T ($\{2, 5, 10\}$). For our analysis, we consider the embedding difference on the test set. During testing, LVNN parameters are updated online, and the embeddings are recorded at each time step. We then retain the final parameter set and, using the true extended covariance matrix, compute the corresponding optimal embeddings. Each experiment is repeated on 10 independently generated time series using the same parameters, and the average results are reported.

Figure 5.5 shows the embedding differences of LVNN and TPCA across the test sets of the synthetic stationary datasets. We observe that LVNN consistently outperforms TPCA for all temporal memory window sizes T. This finding supports our theoretical stability results, indicating that LVNN is more robust to finite sample perturbations in the extended covariance matrix than TPCA. It is also aligns with the conclusions of [126] and [31], which highlight the enhanced stability of graph convolution methods over PCA-based approaches. Furthermore, the stability gap between LVNN and TPCA becomes more pronounced for datasets with heavier-tailed eigenvalue distributions, where eigenvalues are closer to each other and TPCA stability deteriorates. In contrast, LVNN maintains similar stability levels as the tail weight increases, illustrating its design trade-off - reduced discriminability between close eigenvalues for increased stability - as described in Assumption 1. Finally, examining LVNN's performance as T increases, we find that the model becomes increasingly unstable. This observation aligns with our theoretical upper bound on stability from Equation (4.14), which grows sub-quadratically with T.

We next evaluate the stability of LVNN in non-stationary settings using the three real-world datasets introduced in Section 5.1.1. Due to their non-stationarity, these datasets

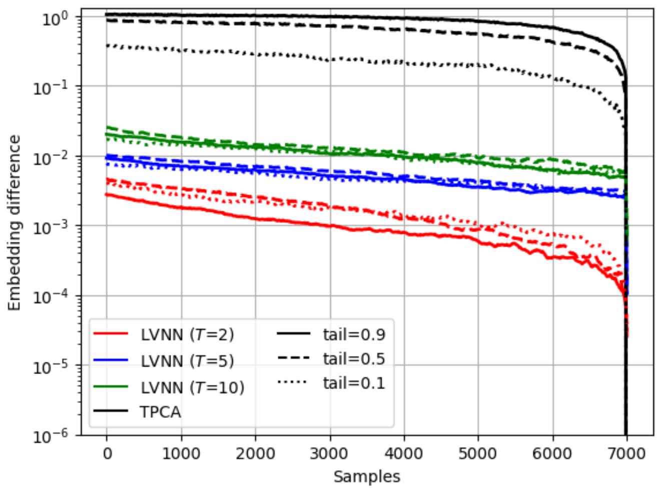
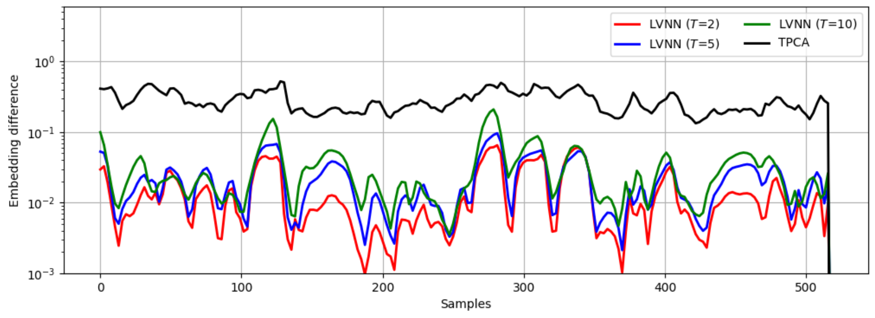


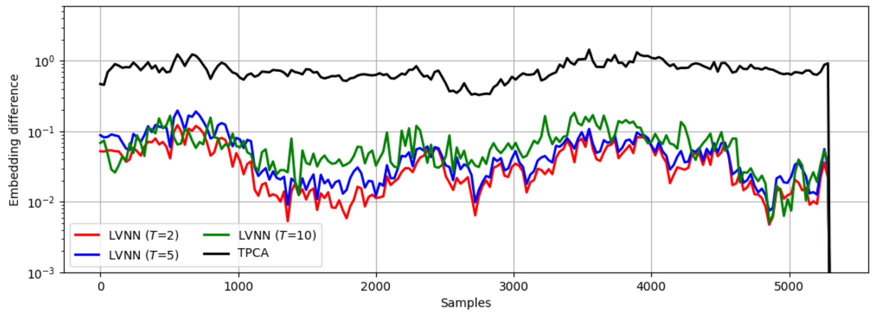
Figure 5.5: Embedding difference of the LVNN (i.e., $\left\|\boldsymbol{\Phi}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)-\boldsymbol{\Phi}(\tilde{\mathbf{C}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)\right\|$) and TPCA with estimated and optimal covariance parameters on synthetic datasets for distinct temporal memory window sizes T and covariance eigenvalues distribution tails.

exhibit time-varying covariance dynamics and lack a true extended covariance matrix. Therefore, we use the extended covariance matrix computed over the entire dataset as a proxy for the “true” structure. Again, we run LVNN using the same set of temporal memory window sizes T, and analyze the embeddings on each dataset’s test set.

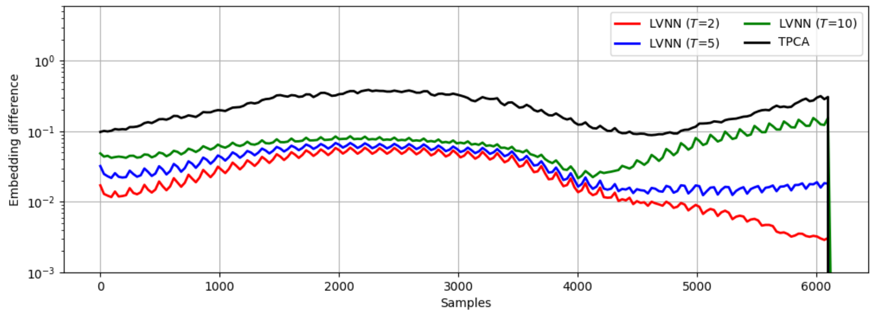
Figures 5.6a to 5.6c show the embedding differences for LVNN and TPCA across the test sets of the three real-world datasets. In all cases, LVNN outperforms TPCA for every temporal window size T, further corroborating the results from the stationary experiments. We also find that larger T values generally lead to greater divergence between the online-estimated and final covariance embeddings, and hence increased instability for LVNN. However, this trend is less consistent and less pronounced than in the stationary case. This is due to the assumption that the extended covariance matrix over the complete dataset acts as a benchmark: the non-stationarity of the time series implies that not all samples contribute towards learning the dynamics of this final matrix. This effect is evident in the oscillatory embedding differences of LVNN across the test sets, in contrast to the smoother behavior observed in Figure 5.5. For instance, in the final test samples of the NOAA dataset (Figure 5.6c), where the time series behavior more closely matches that of the final covariance matrix and there are less oscillations, we observe that $T = 2$ leads to a steady decrease in embeddings difference, $T = 5$ keeps it relatively constant, and $T = 10$ increases it towards the end of the test set. Nonetheless, because the final covariance matrix is only a proxy, these findings should be interpreted cautiously as indicative of LVNN’s stability in non-stationary conditions.



(a) Molene Dataset



(b) Exchange Rate Dataset



(c) NOAA Dataset

Figure 5.6: Embedding difference of LVNN (i.e., $\left\|\mathbf{\Phi}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)-\mathbf{\Phi}(\tilde{\mathbf{C}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)\right\|$) and TPCA with estimated and "optimal" covariance and parameters on the selected real-world datasets for distinct observation windows T.

5.2.2. SINGLE-STEP FORECASTING

The main goal of the following experiments is to assess the effectiveness of the learned spatiotemporal embeddings by LVNN when applied to a downstream task, specifically single-step forecasting. We conduct our experiments on the three real-world datasets presented in Section 5.1.1, following the preprocessing steps and dataset splits provided there. For each dataset, we report results over forecasting horizons of 1, 3, and 5 time steps using the baselines reported in Section 5.1.2. Performance is measured using the MSE, MAE, and sMAPE loss metrics (Section 5.1.3) averaged over 5 runs. However, because the relative performance of the models with each of these loss metrics is similar, we only discuss the sMAPE loss metric in this Section and provide the full results in Appendix E. We evaluate two versions of our proposed model. The first uses the full extended sample covariance matrix as a graph representation matrix and is denoted by LVNN. The second uses the block upper triangular extended sample covariance matrix and is denoted by T-LVNN.

Table 5.2 provides the forecasting results on the Molene dataset across three forecasting horizons. Among all the baselines, STVNN achieves the best results across all forecasting horizons, reflecting its ability to effectively capture meaningful spatiotemporal patterns. Both versions of our proposed model, using the full extended sample covariance matrix and the block upper triangular matrix, match or outperform the baselines across all horizons. In particular, LVNN consistently outperforms TPCA and achieves comparable or better results than LSTM, VNN, and VNNL, but its performance remains slightly weaker than than of STVNN across all settings. To interpret this performance gap, we consider the characteristics of the Molene dataset, which include 37 time series variables but a limited number of training observations. This makes accurate estimation of the covariance terms challenging, potentially introducing noise that LVNN is not sufficiently equipped to handle by design. As discussed in Section 4.2.2, LVNN lacks the flexibility to modulate the influence of these noisy covariance terms in the learned embeddings. In contrast, STVNN operates solely on the lag-zero sample covariance matrix and processes the graph signal at each time step independently, making it more robust to these challenges. Notably, T-LVNN performs comparably or slightly better than STVNN, demonstrating that the block upper triangular matrix improves LVNN's expressivity by constructing increasing temporal neighborhoods for each set of embeddings, moving backwards in time. The performance boost over STVNN may also be attributed to the better utilization of the cross-temporal covariance terms, which offer additional information for modeling the time series.

Table 5.3 presents the single-step forecasting results on the Exchange Rate dataset across forecasting horizons 1, 3, and 5. Among all baselines, STVNN, VNN, and VNNL perform similarly, outperforming LSTM and TPCA by a large margin. Our LVNN model, using the full extended sample covariance matrix, achieves substantially better performance than all baselines across all horizons, while T-LVNN with the block upper triangular matrix consistently yields the best results overall. This performance difference can be partially interpreted by the larger number of training observations and the smaller number of time series variables ($N = 8$) in the Exchanger Rate dataset, which allows for a more accurate estimation of the extended covariance matrix with less noise in the covariance terms. As a result, both LVNN and T-LVNN are able to effectively utilize the information

Model	Forecasting Horizon		
	1	3	5
TPCA	0.37 ± 0.00	0.55 ± 0.00	0.66 ± 0.00
LSTM	0.28 ± 0.00	0.46 ± 0.01	0.59 ± 0.01
VNN	0.20 ± 0.00	0.41 ± 0.01	0.63 ± 0.01
VNNL	0.20 ± 0.00	0.40 ± 0.00	0.60 ± 0.00
STVNN	$\underline{\mathit{0.19}} \pm 0.01$	$\underline{\mathit{0.39}} \pm 0.01$	$\mathbf{0.58} \pm 0.01$
LVNN (ours)	0.20 ± 0.00	0.41 ± 0.01	0.60 ± 0.01
T-LVNN (ours)	$\mathbf{0.18} \pm 0.00$	$\mathbf{0.38} \pm 0.00$	$\mathbf{0.58} \pm 0.01$

Table 5.2: Results for single-step forecasting on the test-split of the Molene dataset. All results are averaged over 5 runs.

provided by the lagged covariances. The slight edge of T-LVNN over LVNN can be attributed to the simpler selected hyperparameter configurations for the two models [cf. Appendix C]. Overall, these experiments demonstrate that effectively leveraging lagged covariance terms in the embedding learning process leads to improved performance in downstream tasks such as single-step forecasting.

Model	Forecasting Horizon		
	1	3	5
TPCA	1.95 ± 0.02	2.01 ± 0.02	2.08 ± 0.02
LSTM	1.22 ± 0.04	1.33 ± 0.05	1.39 ± 0.04
VNN	0.55 ± 0.01	0.88 ± 0.01	1.10 ± 0.01
VNNL	0.55 ± 0.00	0.87 ± 0.03	1.08 ± 0.01
STVNN	0.53 ± 0.01	0.84 ± 0.01	1.06 ± 0.01
LVNN (ours)	$\underline{\mathit{0.50}} \pm 0.00$	$\underline{\mathit{0.81}} \pm 0.01$	$\underline{\mathit{1.02}} \pm 0.01$
T-LVNN (ours)	$\mathbf{0.49} \pm 0.00$	$\mathbf{0.80} \pm 0.01$	$\mathbf{1.00} \pm 0.01$

Table 5.3: Results for single-step forecasting on the test-split of the exchange rate dataset. All results are averaged over 5 runs.

Table 5.4 presents the forecasting results on the NOAA dataset across three forecasting horizons. Among the baseline models, LSTM achieves the best performance at horizons 3 and 5, although it performs worse than most baseline models at horizon 1. The proposed LVNN model with the full extended sample covariance matrix shows comparable results to STVNN and outperforms all other baselines at horizon 1, but its performance degrades slightly at longer horizons. However, using the block upper triangular matrix, T-LVNN consistently improves upon LVNN and outperforms most baselines at horizon 1 and 5, and performs comparable to STVNN at horizon 3. Similar to the experiments with the Molene dataset, we observe a clear performance improvement when using T-LVNN, which could be partly explained by the large number of time series variables ($N = 107$), and therefore large number of introduced covariance terms, in the

NOAA dataset. This highlights the benefit of using the block upper triangular matrix in handling datasets with a relatively large number of variables, such as NOAA and Molene. Overall, both LVNN variants consistently exhibit strong forecasting performance across varying time horizons.

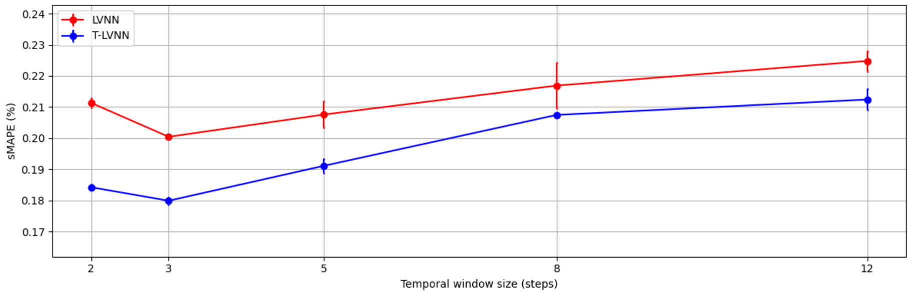
Model	Forecasting Horizon 1	3	5
TPCA	3.03 ± 0.04	4.78 ± 0.04	5.78 ± 0.03
LSTM	2.02 ± 0.04	$\mathbf{3.15} \pm 0.06$	$\mathbf{3.33} \pm 0.11$
VNN	1.42 ± 0.01	$\underline{\mathit{3.16}} \pm 0.13$	4.06 ± 0.26
VNNL	1.75 ± 0.02	3.30 ± 0.11	4.21 ± 0.22
STVNN	1.36 ± 0.02	3.22 ± 0.10	3.82 ± 0.17
LVNN (ours)	$\underline{\mathit{1.35}} \pm 0.01$	3.26 ± 0.11	3.89 ± 0.16
T-LVNN (ours)	$\mathbf{1.25} \pm 0.03$	3.20 ± 0.05	$\underline{\mathit{3.66}} \pm 0.09$

Table 5.4: Results for single-step forecasting on the test-split of the NOAA dataset. All results are averaged over 5 runs.

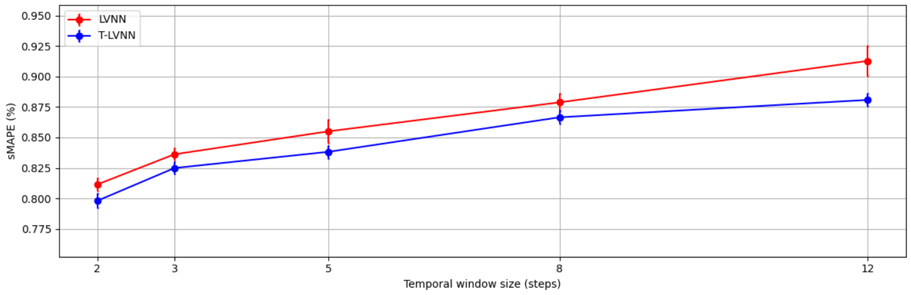
TEMPORAL WINDOW SIZE ANALYSIS

Figure 5.7 presents the forecasting performance of the best-performing LVNN configurations across varying temporal window sizes for the three real-world datasets discussed earlier. Since the behaviors across different settings are similar, we report results for a single representative horizon per dataset in this section, while the full set of results across all horizons and datasets is provided in Appendix D. Overall, we observe that T-LVNN, which leverages the block upper triangular part of the extended sample covariance matrix, consistently performs comparably to or better than the standard LVNN across all window sizes, forecasting horizons, and datasets. This supports our hypothesis that retaining only spatial and backward temporal connections would enhance the flexibility of LVNN and allow it to generate more expressive embeddings that emphasize more recent time windows. Additionally, we find that the optimal temporal window size varies across datasets and horizons, suggesting that it is closely tied to the inherent characteristics of each dataset.

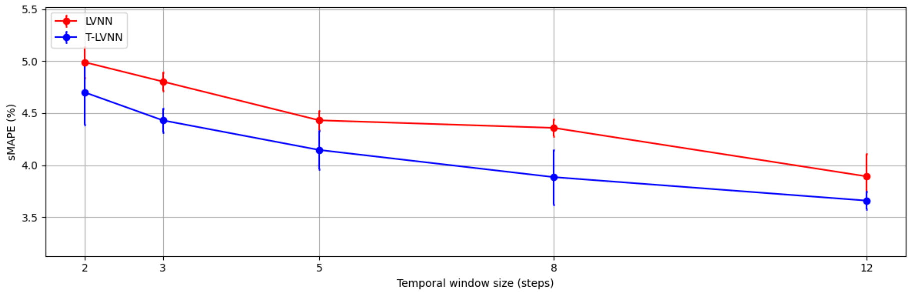
For the Molene dataset at horizon 1 (Figure 5.7a), both LVNN and T-LVNN achieve their best forecasting performance at a temporal window size of $T = 3$. This marks an improvement over their performance at $T = 2$, indicating that incorporating past signal values and lag-2 covariance terms contributes positively to the learned representations. However, as the temporal window continues to grow, forecasting error increases, suggesting that larger windows introduce additional noisy or spurious correlations, particularly given the limited number of observations in this dataset. For the Exchange Rates dataset at horizon 3 (Figure 5.7b), performance consistently degrades as the temporal window size increases beyond $T = 2$. This pattern suggests that, due to the highly non-stationary nature of currency exchange rates, LVNN benefits more from focusing on recent information rather than extending the temporal window to incorporate past



(a) Molene Dataset at forecasting horizon 1



(b) Exchange Rate Dataset at forecasting horizon 3



(c) NOAA Dataset at forecasting horizon 5

Figure 5.7: Forecasting performance of the best LVNN configurations across different temporal horizons, with three different datasets.

observations. In contrast, the NOAA dataset at horizon 5 (Figure 5.7c) displays the opposite behavior. Namely, performance steadily improves with larger temporal windows, reaching its optimum at $T = 12$. Despite the high dimensionality of this dataset, which increases the number of estimated covariance terms as T grows, the longer temporal context appears to enhance the quality of the learned representations. This can likely be attributed to the nature of temperature data, which exhibits more robust spatiotemporal trends, and the abundance of available observations, which supports accurate covariance estimation even with longer time windows.

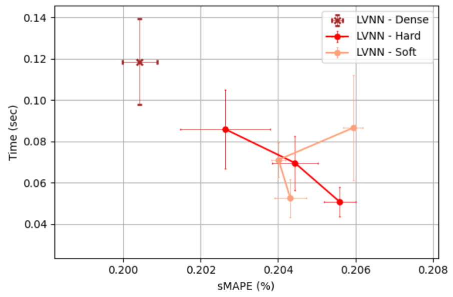
5.2.3. THRESHOLDING SPARSIFICATION

The goal of this final set of experiments is to evaluate how hard and soft thresholding sparsification techniques affect the forecasting performance and computational efficiency of the LVNN model. To this end, we measure the runtime of the forward pass for the best-performing LVNN and T-LVNN configurations from the previous section across the entire test set for each dataset. Forecasting accuracy (measured by % sMAPE loss) and computational time are reported for all datasets and forecasting horizons in Figures 5.8, 5.9, and 5.10, while the full results are reported in Appendix E. We investigate the effect of progressively increasing sparsification rates (25%, 50%, and 75%) to better understand the model's behavior under a more sparse covariance structure and to examine how this interacts with each dataset. Since we adapt these sparsification techniques from the static to the online setting, we dynamically compute the appropriate threshold at each forward pass to maintain the desired sparsity level and apply it to the updated extended covariance matrix in real time. Overall, we find no significant performance difference between hard and soft thresholding strategies. In high-dimensional scenarios, sparsification leads to substantial improvements in computational efficiency, while performance degradation remains minimal. The sparsified models continue to perform at levels comparable to their dense counterparts and remain competitive with the baseline models discussed in the previous section.

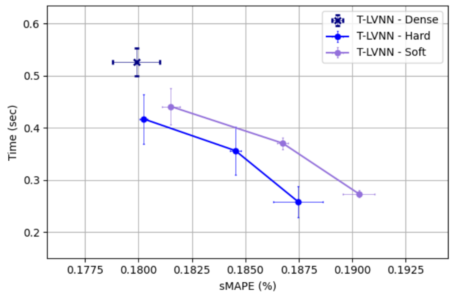
Figure 5.8 compares forecasting performance and computation time for a forward pass over the entire test set using the LVNN and T-LVNN models under different sparsification strategies across forecasting horizons 1, 3, and 5 on the Molene dataset. Since the two models have substantially different optimal hyperparameter configurations, we present their results in separate plots for clarity. Across all horizons and for both models, we observe that both hard and soft thresholding sparsification significantly reduce computational time, with no consistently clear distinction between the two strategies. A 25% sparsification rate yields a moderate reduction in runtime of approximately 15%–20%, while a 75% sparsification rate leads to a more substantial decrease, up to 65% in computational overhead (Figure 5.8d). Given that the Molene dataset includes $N = 37$ variables and the best-performing configurations utilize a temporal window size of $T = 3$, the graph convolutions over the extended covariance graph contribute significantly to the overall runtime. Consequently, the sparsification techniques have a pronounced impact on computational efficiency. In terms of forecasting performance, both models show some degradation across all sparsification levels. However, the T-LVNN model maintains a consistent advantage, since its worst-reported performance in each setting still outperforms the best results of the standard LVNN. This further supports our earlier hypothesis regarding the enhanced flexibility and expressivity of T-LVNN when using the block upper triangular matrix. Moreover, when comparing these results to the baseline models presented in Section 5.2.2, we find that even with sparsification, both LVNN variants achieve comparable or superior scores, further confirming the effectiveness of incorporating cross-temporal covariance terms in a computationally efficient manner.

Figure 5.9 presents the forecasting performance and computation time of the LVNN and T-LVNN models on the Exchange Rates dataset. Unlike in the experiments with the Molene dataset, we report the results of both models within the same plots, as they share similar hyperparameter configurations. We first observe that the impact of thresholding-based sparsification on computational overhead is notably smaller, with reductions ranging from approximately 10% to 25%. This can be attributed to the characteristics of the Exchange Rates dataset and the corresponding model configurations. Specifically, the extended covariance matrix is relatively small due to the limited number of variables ($N = 8$) and the use of a short temporal window ($T = 2$), resulting in lower overall computational overhead during a forward pass and, thus, diminishing the relative impact of sparsification. This also explains why we cannot see a clear improvement in computational efficiency when using the T-LVNN model, which operates on the block upper triangular matrix. Nonetheless, both hard and soft thresholding still yield moderate improvements in efficiency, especially at higher sparsification rates. In terms of forecasting performance, the models' performance remains very close to that without any sparsification across all settings, while still outperforming the baseline models from Section 5.2.2 regardless of the sparsification level. Notably, in Figure 5.9b, we even observe an improvements in performance under sparsification. This suggests that pruning weak or noisy covariance terms may act as a regularization mechanism, encouraging the model to focus on more robust and informative dependencies in the data.

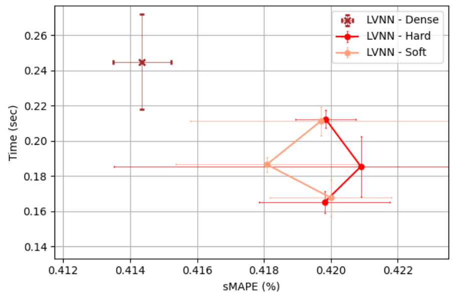
Figure 5.10 displays the forecasting performance and computation time of LVNN and T-LVNN on the NOAA dataset. Due to the large number of variables in this dataset ($N = 109$) and the longer temporal windows used in the selected configurations ($T = 2$ in Figure 5.10a, $T = 8$ in Figure 5.10b, $T = 12$ in Figure 5.10c), the resulting extended covariance matrices are significantly larger than in the previous datasets. As a result, sparsification has a much more substantial effect on computational efficiency, with clearly more pronounced improvements, ranging from approximately 15% with a sparsification rate of 25% to approximately 70% with a sparsification rate of 75% (Figure 5.10c). Additionally, in Figures 5.10a and 5.10c, where LVNN and T-LVNN have comparable hyperparameter configurations, it becomes evident that the block upper triangular structure employed by T-LVNN not only enhances expressivity but also contributes to improved computational efficiency. In terms of forecasting performance, both models remain highly competitive across all sparsification levels and forecasting horizons. T-LVNN consistently retains its performance from Section 5.2.2, demonstrating its robustness even under heavier sparsification. LVNN, while slightly less accurate, maintains solid performance and remains competitive with most baselines. These results further confirm that sparsification is more beneficial in high-dimensional settings, where the extended covariance matrix monopolizes the model's computational requirements.



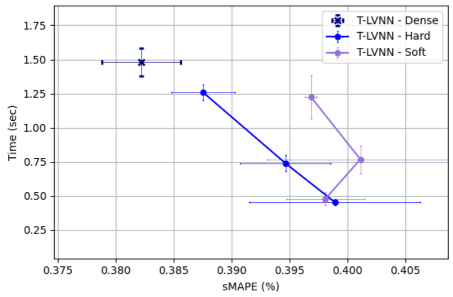
(a) LVNN / Molene dataset at forecasting horizon 1



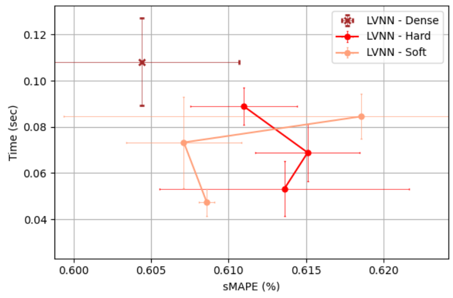
(b) T-LVNN / Molene dataset at forecasting horizon 1



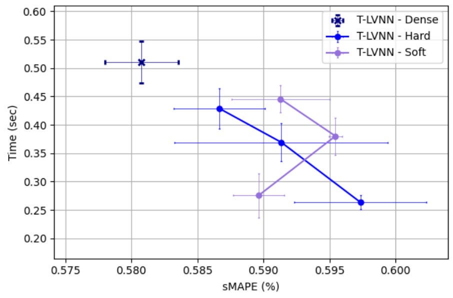
(c) LVNN / Molene dataset at forecasting horizon 3



(d) T-LVNN / Molene dataset at forecasting horizon 3

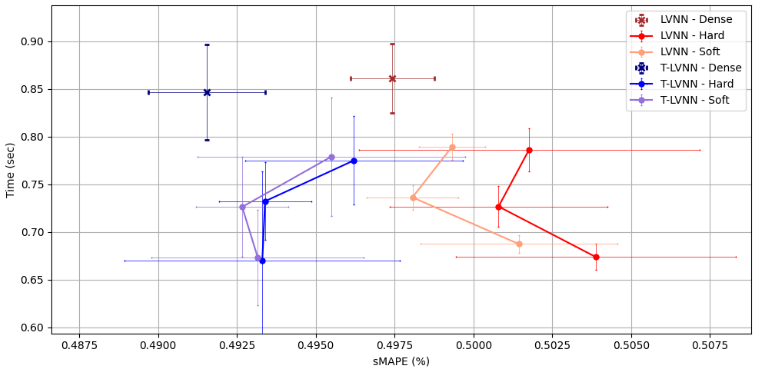


(e) LVNN / Molene dataset at forecasting horizon 5

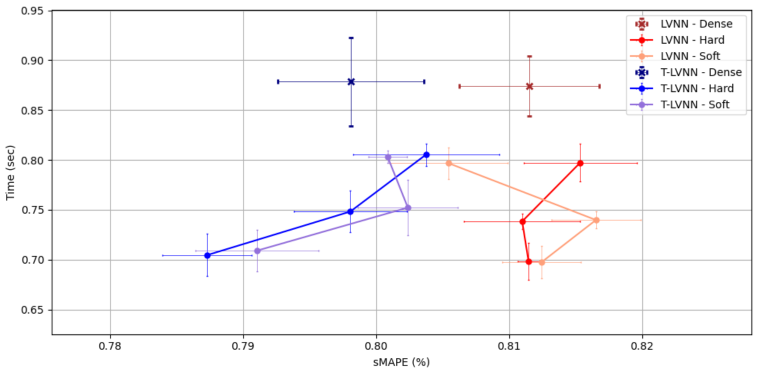


(f) T-LVNN / Molene dataset at forecasting horizon 5

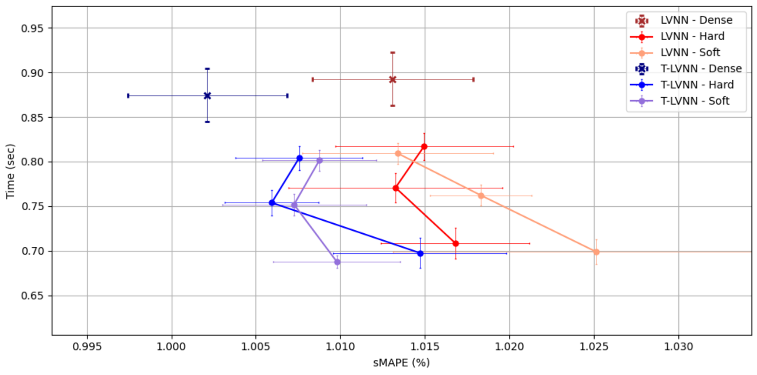
Figure 5.8: The sMAPE loss and computation time for a forward pass over the entire test for LVNN and T-LVNN on the Molene dataset. From top to bottom, we report results for adjusted thresholds to achieve sparsification rates of 25%, 50%, and 75%, for hard and soft thresholding. Horizontal axis reports the sMAPE loss, with the performance improving towards the left.



(a) Exchange Rates dataset at forecasting horizon 1

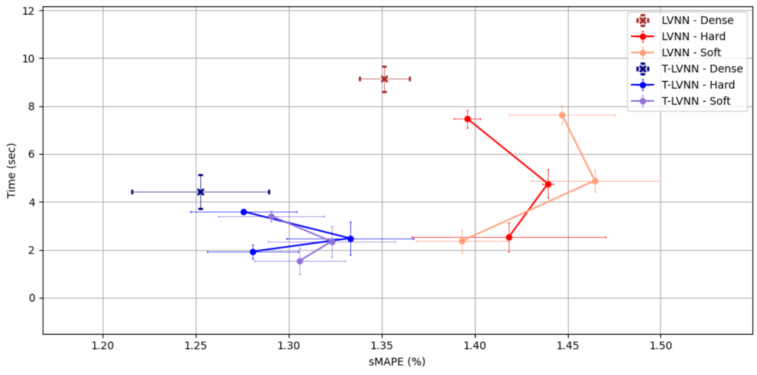


(b) Exchange Rates dataset at forecasting horizon 3

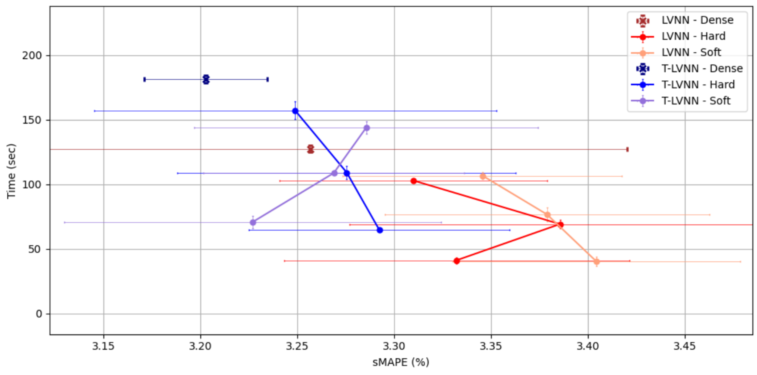


(c) Exchange Rates dataset at forecasting horizon 5

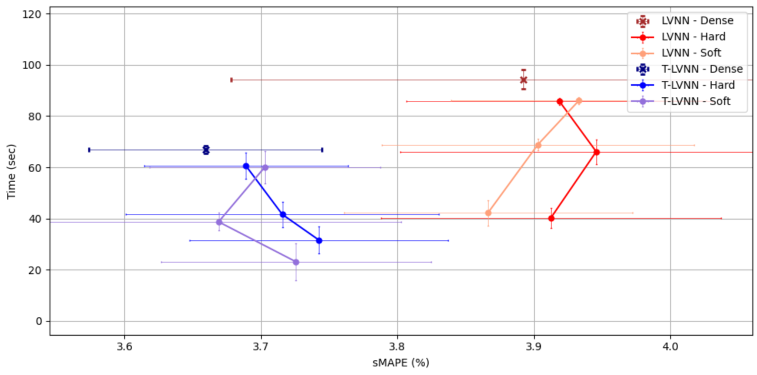
Figure 5.9: The sMAPE loss and computation time for a forward pass over the entire test for LVNN and T-LVNN on the Exchange Rates dataset. From top to bottom, we report results for adjusted thresholds to achieve sparsification rates of 25%, 50%, and 75%, for hard and soft thresholding. Horizontal axis reports the sMAPE loss, with the performance improving towards the left.



(a) NOAA dataset at forecasting horizon 1



(b) NOAA dataset at forecasting horizon 3



(c) NOAA dataset at forecasting horizon 5

Figure 5.10: The sMAPE loss and computation time for a forward pass over the entire test for LVNN and T-LVNN on the NOAA dataset. From top to bottom, we report results for adjusted thresholds to achieve sparsification rates of 25%, 50%, and 75%, for hard and soft thresholding. Horizontal axis reports the sMAPE loss, with the performance improving towards the left.

5.3. DISCUSSION

Effectively learning from multivariate time series requires models that can capture both spatial correlations between variables and temporal dependencies across time steps. This challenge is particularly pronounced in online and non-stationary settings, where data arrives sequentially and the underlying relationships may evolve over time. To address this, we introduced the Lagged spatiotemporal coVariance Neural Network (LVNN), a graph-based learning architecture that leverages lagged correlations to represent and learn from multivariate time series. By constructing spatiotemporal graphs from the extended sample covariance matrix, LVNN generalizes prior covariance-based methods and provides a more expressive framework for modeling intricate spatiotemporal interactions, moving beyond the conventional focus on lag-zero correlations.

We began by investigating the robustness of LVNN under uncertainty in the finite-sample estimation of the extended covariance matrix. Through theoretical analysis, we showed that LVNN remains stable under such perturbations, with an upper stability bound that improves as more data becomes available and scales sub-quadratically with the temporal window size. These theoretical guarantees were validated empirically in both stationary and non-stationary environments, where LVNN demonstrated consistently stable behavior and improved upon the limitations of techniques like Temporal PCA. These findings offer foundational support for incorporating lagged covariance structure into online learning without compromising robustness.

Next, we assessed the effectiveness of the learned embeddings in a downstream single-step forecasting task across three real-world datasets and multiple forecasting horizons. While the standard LVNN performed competitively with strong baselines, the variant that selectively retains only spatial and backward temporal connections, by using the block upper triangular portion of the covariance matrix, achieved the most consistent performance. This result highlights the value of enforcing temporal structure in the graph, which increases the model's flexibility by allowing it to focus on recent, more relevant information. By capturing how variable interactions evolve over time, rather than treating them as static or temporally independent, LVNN is able to construct richer and more informative representations. These findings confirm that modeling cross-temporal covariance terms strengthens the underlying graph structure and enhances the model's ability to exploit spatiotemporal dependencies, ultimately improving forecasting performance in dynamic environments. We began by investigating the robustness of LVNN under uncertainties in the finite sample estimation of the extended covariance matrix. Through theoretical analysis, we demonstrated that LVNN remains stable under these perturbations, with an upper stability bound that improves as more data becomes available and grows sub-quadratically with the temporal window size.

Finally, we explored thresholding-based sparsification as a method to improve computational efficiency. Both hard and soft thresholding techniques yielded significant reductions in runtime, particularly in high-dimensional scenarios, while preserving predictive performance. This demonstrates that lagged covariance-based models can be made more scalable without sacrificing performance, and that removing weak or noisy correlations may even serve as a useful regularization mechanism. In specific cases, sparsification even led to slight improvements in forecasting performance, suggesting it not only reduces complexity but can also enhance generalization by focusing the model on more robust spatiotemporal patterns.

In conclusion, this thesis demonstrates the practical and theoretical benefits of incorporating lagged covariance information into graph-based models for multivariate time series. By moving beyond static or purely spatial representations, LVNN captures richer temporal dynamics and adapts effectively to streaming and evolving data. The combination of principled modeling, robust learning, and efficient implementation opens promising directions for future research at the intersection of graph learning and time series modeling.

6 Conclusion

In this chapter, we conclude this thesis and provide possible future work directions. Section 6.1 gives a summary of this research work, outlining the main takeaways. Section 6.2 answers the research questions posed in Chapter 1. Finally, Section 6.3 discusses promising future work directions that originate from the research work conducted in this thesis.

6.1. Thesis Summary

Chapter 1 motivated the integration of lagged covariance terms into multivariate time series modeling with covariance-based GNNs, introduced the research questions, and set the context for the following chapters. In Chapter 2, we presented the necessary background material for our research. We introduced the notion of coVariance Neural Networks (VNN), followed by statistical and PCA-based methods for modeling multivariate time series and spatiotemporal GNN architectures. Chapter 3 provided an overview of the relevant literature, covering different methods for time series modeling and, finally, focusing on covariance-based spatiotemporal GNN approaches, which have a central role in the research carried out in this thesis.

In Chapter 4, we discussed the main contribution of this thesis, the Lagged spatiotemporal coVariance Neural Network (LVNN) architecture for modeling multivariate time series. The LVNN leverages the extended covariance matrix as a graph representation matrix [cf. Section 4.2.1], and performs graph convolutions over this larger graph to produce spatiotemporal embeddings [cf. Section 4.2.2]. We proposed the use of the block upper triangular matrix and thresholding sparsification strategies to enhance the computational efficiency of LVNN. Additionally, we conducted a theoretical analysis of LVNN's stability under perturbations in the finite sample estimation of the extended covariance matrix, comparing its robustness against that of TPCA. In Chapter 5, we ini

In Chapter 5, we presented our empirical evaluation of the proposed LVNN model. First, we validated the stability of LVNN under both stationary and non-stationary conditions, confirming our theoretical findings from Chapter 4. Next, we evaluated the model's capabilities on single-step forecasting using three real-world datasets, where the LVNN with the block upper triangular matrix achieved the most consistent performance across the baselines. Finally, we explored the effects of thresholding-based sparsification, demonstrating a substantial reduction in computational cost with only minimal impact on forecasting performance.

6.2. Answers to Research Questions

In this section, we address the research questions posed in Chapter 1 and provide answers based on our research findings from Chapters 4 and 5.

(RQ1) *"How robust is the proposed model to perturbations introduced by the online estimation of the extended covariance matrix and model parameters?"*

To answer this question, we first theoretically proved that LVNN is stable to uncertainties caused by these online estimations in a stationary setting, in Section 4.3. Specifically, we observed that the upper stability bound decreases with a rate $\mathcal{O}(1/\sqrt{t})$ as more samples become available, while it grows sub-quadratically with the temporal window size T. Therefore, LVNN provides improved stability compared to Temporal PCA and VNN, but remains less stable than STVNN. Moreover, we derived the corresponding upper stability bounds for the LVNN variants with the block upper triangular extended sample covariance matrix and with the application of hard and soft thresholding sparsification. Then, in Section 5.2.1, we empirically validate our theoretical findings against TPCA in a stationary setting with synthetic data, as well as in non-stationary settings for three real-world datasets.

(RQ2) *"How effective are the learned embeddings from the proposed model when applied to a downstream task, such as single-step forecasting?"*

We answered this question in Section 5.2.2. We evaluated the performance of the proposed LVNN model on single-step forecasting across multiple forecasting horizons. We conducted our experiments on three real-world datasets with different characteristics: Molene and NOAA, containing temperature recordings, and Exchange Rates, containing time series of currency exchange rates. Overall, our results indicate that LVNN showcases the most consistent performance across all experiments. When using the full sample extended covariance matrix, LVNN achieves comparable or better results than the TPCA, LSTM, VNN, and VNN-LSTM benchmark models, while, compared to STVNN, it performs similarly or slightly worse on Molene and NOAA datasets, but consistently better on Exchange Rates. This is indicative of the inability of LVNN to isolate the contributions of specific time steps with the full sample extended covariance matrix. Then, using the block upper triangular of the matrix improves LVNN's performance across all experiments, leading to better results than STVNN in most settings. This supports our analysis in Sections 4.2.1 and 4.2.2, indicating that the block upper triangular matrix leads to nested temporal neighborhoods, allowing LVNN to modulate contributions from different time windows and enhance its expressiveness.

(RQ3) *"How can we make the model more computationally efficient and potentially robust to spurious correlations via thresholding sparsification?"*

To address this research question, we extended LVNN with both hard and soft thresholding sparsification techniques and evaluated its performance on the real-world datasets of the previous experiments. First, in Section 4.3, we theoretically proved that these sparsification strategies do not affect the upper stability bound of LVNN, w.r.t. the number of variables N or the temporal window size T. Then, in our empirical evaluation in Section 5.2.3, we tested varying sparsification rates (25%, 50%, and 75%) and observed a substantial improvement in computational time as the rate increased. In terms of predictive performance, the results show that the LVNN loss metrics slightly degrade as the sparsity levels increase, however remaining comparable to the original LVNN performance without sparsification. Notably, in specific cases, performance actually improves, suggesting that thresholding may effectively prune spurious covariance terms. Overall, there were not any strong patterns in the results to distinguish the application of hard and soft thresholding. These findings highlight that sparsification can efficiently address the computational overhead of the extended sample covariance matrix without significant sacrifices in performance.

Based on the answers to these research questions, we can now provide an answer to the main research question of this thesis.

(RQ) *"How can we learn online meaningful embeddings from multivariate time series by leveraging lagged covariance information as part of the GNN architecture?"*

The proposed LVNN model effectively captures and leverages lagged covariance information from multivariate time series by performing graph convolutions over spatiotemporal graphs. The underlying graph structure is defined by the finite sample extended covariance matrix, which encodes correlations across both variables and time. LVNN demonstrates stability under perturbations introduced by the finite sample estimation of the extended covariance matrix, and its stability is only moderately affected by the temporal window size, in contrast to Temporal PCA. Furthermore, both the standard LVNN using the full extended covariance matrix and its block upper triangular variant prove effective in single-step forecasting tasks across diverse datasets. Lastly, applying thresholding-based sparsification significantly reduces computational overhead without compromising the model stability or substantially affecting its forecasting performance.

6.3. FUTURE WORK

To conclude this thesis, we discuss possible future research directions. Taking into consideration the focal points of our research work, we identify the following directions: *(i)* improving the LVNN performance by enhancing its expressiveness and, subsequently, the learned embeddings; *(ii)* improving the computational efficiency of LVNN and mitigating the effect of spurious correlations by alternative sparsification techniques.

PROCESSING LAGGED COVARIANCE TERMS WITH DIFFERENT PARAMETERS

One limitation of the LVNN architecture lies in the use of shared parameters across all connections in the graph representation, corresponding to lagged covariance terms. As discussed in Section 4.2.2, the extended sample covariance matrix induces a fully connected graph where each signal value across the temporal window contributes to all learned embeddings. However, treating all connections with the same set of parameters significantly limits the model's flexibility. For instance, an LVNN cannot selectively amplify the contributions from lag-1 connections and attenuate the contributions of lag-2 connections. While we partially addressed this issue by altering the underlying graph structure through the block upper triangular extended sample covariance matrix, the expressivity of the model could be further enhanced by retaining the full graph structure and assigning distinct parameters to connections based on their time lag. Figure 6.1 demonstrates the benefits of using distinct parameters with an application of a filter of order 1 on a setting with $N = 4$ variables and a temporal window size $T = 3$. The greater flexibility of this filter is evident, especially if we compare the embeddings expressions with those of the proposed LVF in Figure 4.5a.

ALTERNATIVE STRUCTURAL SPARSIFICATION STRATEGIES

Another limitation of LVNN is the computational overhead and spurious correlations associated with the extended sample covariance matrix. In this thesis, we experimented with the block upper triangular extended sample covariance matrix, which improved the computational efficiency of LVNN. However, it still retains all lagged covariance terms within the selected temporal window, thus not addressing the issue of spurious covariance terms. Inspired by the supra-max adjacency framework [84], another approach would be to decouple the temporal window size from the time lags considered in the

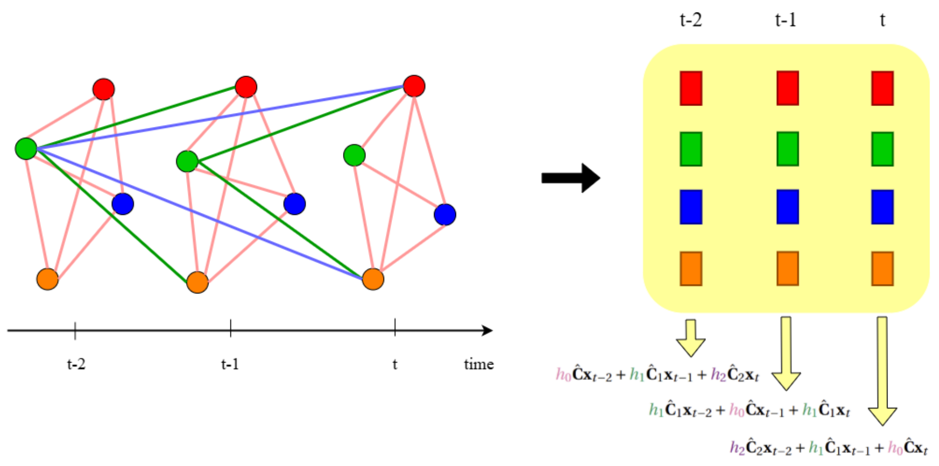
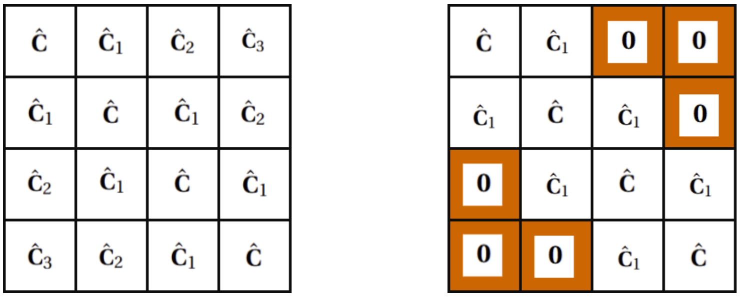

t-2
t-1
t
time
t-2
t-1
t
$h_0\hat{\mathbf{C}}\mathbf{x}_{t-2} + h_1\hat{\mathbf{C}}_1\mathbf{x}_{t-1} + h_2\hat{\mathbf{C}}_2\mathbf{x}_t$
$h_1\hat{\mathbf{C}}_1\mathbf{x}_{t-2} + h_0\hat{\mathbf{C}}\mathbf{x}_{t-1} + h_1\hat{\mathbf{C}}_1\mathbf{x}_t$
$h_2\hat{\mathbf{C}}_2\mathbf{x}_{t-2} + h_1\hat{\mathbf{C}}_1\mathbf{x}_{t-1} + h_0\hat{\mathbf{C}}\mathbf{x}_t$

Figure 6.1: A visualization of the graph structure of the extended covariance matrix in a setting with $N = 4$ variables and a temporal window size $T = 3$. A filter of order 1 performs graph convolutions over the induced graph. Covariances of different lags are depicted in a different color, as they correspond to different filter parameters (which are visible in the embeddings expressions). For the lag-1 (green) and lag-2 (purple) temporal connections, we only draw the relevant edges originating from the green nodes and leading to the subsequent red and orange ones.

graph structure. Figure 6.2b shows an example of a setting with temporal window size $T = 4$, where retaining only lag-zero and lag-1 covariance terms results in a sparsification rate of 37,5%. By evaluating the relevance of each time lag to the task at hand, we can construct more computationally efficient spatiotemporal graphs, tailored to the temporal dynamics of the data. Therefore, this structural sparsification approach would allow the model to focus on the most informative time lags within the temporal window, while reducing the computational overhead.



$\hat{\mathbf{C}}$	$\hat{\mathbf{C}}_1$	$\hat{\mathbf{C}}_2$	$\hat{\mathbf{C}}_3$
$\hat{\mathbf{C}}_1$	$\hat{\mathbf{C}}$	$\hat{\mathbf{C}}_1$	$\hat{\mathbf{C}}_2$
$\hat{\mathbf{C}}_2$	$\hat{\mathbf{C}}_1$	$\hat{\mathbf{C}}$	$\hat{\mathbf{C}}_1$
$\hat{\mathbf{C}}_3$	$\hat{\mathbf{C}}_2$	$\hat{\mathbf{C}}_1$	$\hat{\mathbf{C}}$

(a) Full Extended Covariance Matrix

$\hat{\mathbf{C}}$	$\hat{\mathbf{C}}_1$	$\mathbf{0}$	$\mathbf{0}$
$\hat{\mathbf{C}}_1$	$\hat{\mathbf{C}}$	$\hat{\mathbf{C}}_1$	$\mathbf{0}$
$\mathbf{0}$	$\hat{\mathbf{C}}_1$	$\hat{\mathbf{C}}$	$\hat{\mathbf{C}}_1$
$\mathbf{0}$	$\mathbf{0}$	$\hat{\mathbf{C}}_1$	$\hat{\mathbf{C}}$

(b) Matrix with Lag-Zero and Lag-1 Covariance Blocks

Figure 6.2: The extended sample covariance matrix in a setting with $T = 4$. By retaining the temporal window size T, but only considering the lag-zero and lag-1 covariance blocks (and zeroing out the orange blocks), the matrix is sparsified by 37,5%.

References

[1] Anish Agarwal, Abdullah Alomar, and Devavrat Shah. "On multivariate singular spectrum analysis and its variants". In: *ACM SIGMETRICS Performance Evaluation Review* 50.1 (2022), pp. 79–80.

[2] Anish Agarwal et al. "Adaptive principal component regression with applications to panel data". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 77104–77118.

[3] Arwa Alanqary, Abdullah Alomar, and Devavrat Shah. "Change Point Detection via Multivariate Singular Spectrum Analysis". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 23218–23230. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/c348616cd8a86ee661c7c98800678fad-Paper.pdf.

[4] Anthony Arguez et al. "NOAA's 1981–2010 US climate normals: an overview". In: *Bulletin of the American Meteorological Society* 93.11 (2012), pp. 1687–1697.

[5] Raman Arora et al. "Stochastic optimization for PCA and PLS". In: *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 2012, pp. 861–868. DOI: 10.1109/Allerton.2012.6483308.

[6] Mohammed Badawy, Nagy Ramadan, and Hesham Ahmed Hefny. "Healthcare predictive analytics using machine learning and deep learning techniques: a survey". In: *Journal of Electrical Systems and Information Technology* 10.1 (2023), p. 40.

[7] Lei Bai et al. "Adaptive graph convolutional recurrent network for traffic forecasting". In: *Advances in neural information processing systems* 33 (2020), pp. 17804–17815.

[8] Trapit Bansal, David Belanger, and Andrew McCallum. "Ask the gru: Multi-task learning for deep text recommendations". In: *proceedings of the 10th ACM Conference on Recommender Systems*. 2016, pp. 107–114.

[9] Alaa Bessadok, Mohamed Ali Mahjoub, and Islem Rekik. "Graph neural networks in network neuroscience". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.5 (2022), pp. 5833–5848.

[10] Peter J. Bickel and Elizaveta Levina. "Covariance regularization by thresholding". In: *The Annals of Statistics* 36.6 (Dec. 2008). ISSN: 0090-5364. DOI: 10.1214/08-aos600. URL: http://dx.doi.org/10.1214/08-AOS600.

[11] Jacob Bien and Robert J. Tibshirani. "Sparse estimation of a covariance matrix". In: *Biometrika* 98.4 (2011), pp. 807–820. ISSN: 00063444, 14643510. URL: http://www.jstor.org/stable/23076173 (visited on 03/31/2025).

[12] Ane Blázquez-García et al. *A review on outlier/anomaly detection in time series data*. 2020. arXiv: 2002.04236 [cs.LG]. URL: https://arxiv.org/abs/2002.04236.

[13] Anastasia Borovykh, Sander Bohte, and Cornelis W. Oosterlee. *Conditional Time Series Forecasting with Convolutional Neural Networks*. 2018. arXiv: 1703.04691 [stat.ML]. URL: https://arxiv.org/abs/1703.04691.

[14] Christos Boutsidis et al. "Online principal components analysis". In: *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '15. San Diego, California: Society for Industrial and Applied Mathematics, 2015, pp. 887–901.

[15] George.E.P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[16] Matthew Brand. "Incremental singular value decomposition of uncertain data with missing values". In: *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I 7*. Springer. 2002, pp. 707–720.

[17] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer Texts in Statistics. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-29854-2. URL: https://doi.org/10.1007/978-3-319-29854-2.

[18] Chris Brooks. "Linear and Non-linear (Non-)Forecastability of High-frequency Exchange Rates". In: *Journal of Forecasting* 16.2 (1997), pp. 125–145. DOI: https://doi.org/10.1002/(SICI)1099-131X(199703)16:2<125::AID-FOR648>3.0.CO;2-T.

[19] Daniel Brooks. "Deep learning and information geometry for time-series classification". PhD thesis. Sorbonne Universite, 2020.

[20] Robert Goodell. Brown. *Statistical forecasting for inventory control*. eng. New York: McGraw-Hill, 1959.

[21] Umair Muneer Butt et al. "Hybrid of deep learning and exponential smoothing for enhancing crime forecasting accuracy". In: *PLOS ONE* 17.9 (Sept. 2022), pp. 1–22. DOI: 10.1371/journal.pone.0274172. URL: https://doi.org/10.1371/journal.pone.0274172.

[22] Ling Cai et al. "Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting". In: *Transactions in GIS* 24.3 (2020), pp. 736–755.

[23] T. Tony Cai, Cun-Hui Zhang, and Harrison H. Zhou. "Optimal rates of convergence for covariance matrix estimation". In: *The Annals of Statistics* 38.4 (2010), pp. 2118–2144. DOI: 10.1214/09-AOS752. URL: https://doi.org/10.1214/09-AOS752.

[24] Laurent AF Callot, Anders B Kock, and Marcelo C Medeiros. "Modeling and forecasting large realized covariance matrices and portfolio choice". In: *Journal of Applied Econometrics* 32.1 (2017), pp. 140–158.

[25] Sean D Campbell and Francis X Diebold. “Weather forecasting for weather derivatives”. In: *Journal of the American Statistical Association* 100.469 (2005), pp. 6–16.

[26] Defu Cao et al. “Spectral temporal graph neural network for multivariate time-series forecasting”. In: *Advances in neural information processing systems* 33 (2020), pp. 17766–17778.

[27] Wei Cao et al. “Brits: Bidirectional recurrent imputation for time series”. In: *Advances in neural information processing systems* 31 (2018).

[28] José Vinícius de Miranda Cardoso, Jiaxi Ying, and Daniel Perez Palomar. “Algorithms for learning graphs in financial markets”. In: *arXiv preprint arXiv:2012.15410* (2020).

[29] Hervé Cardot and David Degras. “Online Principal Component Analysis in High Dimension: Which Algorithm to Choose?” In: *International Statistical Review* 86 (2015), pp. 29–50. URL: https://api.semanticscholar.org/CorpusID:17287648.

[30] Andrea Cavallo, Zhan Gao, and Elvin Isufi. *Sparse Covariance Neural Networks*. 2024. arXiv: 2410.01669 [cs.LG]. URL: https://arxiv.org/abs/2410.01669.

[31] Andrea Cavallo, Mohammad Sabbaqi, and Elvin Isufi. “Spatiotemporal Covariance Neural Networks”. In: *Machine Learning and Knowledge Discovery in Databases. Research Track*. Springer Nature Switzerland, 2024, pp. 18–34. ISBN: 9783031703447. DOI: 10.1007/978-3-031-70344-7_2. URL: http://dx.doi.org/10.1007/978-3-031-70344-7_2.

[32] C. Chatfield. *The Analysis of Time Series: An Introduction*. Monographs on applied probability and statistics. Chapman and Hall, 1980. ISBN: 9780412224607. URL: https://books.google.nl/books?id=JR_vAAAAMAAJ.

[33] Yingyi Chen et al. “Applications of recurrent neural networks in environmental factor forecasting: a review”. In: *Neural computation* 30.11 (2018), pp. 2855–2881.

[34] Neeraj Choudhary et al. “Long short-term memory-singular spectrum analysis-based model for electric load forecasting”. In: *Electrical Engineering* 103 (Apr. 2021), pp. 1–16. DOI: 10.1007/s00202-020-01135-y.

[35] Junyoung Chung et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).

[36] Razvan-Gabriel Cirstea et al. “Triformer: Triangular, Variable-Specific Attentions for Long Sequence Multivariate Time Series Forecasting–Full Version”. In: *arXiv preprint arXiv:2204.13767* (2022).

[37] PJ Coen, ED Gomme, and Maurice G Kendall. “Lagged relationships in economic forecasting”. In: *Journal of the Royal Statistical Society. Series A (General)* 132.2 (1969), pp. 133–163.

[38] J.T. Connor, R.D. Martin, and L.E. Atlas. “Recurrent neural networks and robust time series prediction”. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 240–254. DOI: 10.1109/72.279188.

[39] Richard K. Crump and Nikolay Gospodinov. “On the Factor Structure of Bond Returns”. In: *Econometrica* 90.1 (Jan. 2022), pp. 295–314. DOI: 10.3982/ECTA17943. URL: https://ideas.repec.org/a/wly/emetrp/v90y2022i1p295-314.html.

[40] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. “Convolutional neural networks on graphs with fast localized spectral filtering”. In: *Advances in neural information processing systems* 29 (2016).

[41] Yash Deshpande and Andrea Montanari. *Sparse PCA via Covariance Thresholding*. 2016. arXiv: 1311.5179 [math.ST]. URL: https://arxiv.org/abs/1311.5179.

[42] Jiasheng Duan et al. “Long short-term enhanced memory for sequential recommendation”. In: *World Wide Web* 26.2 (2023), pp. 561–583.

[43] Eran Elhaik. “Principal Component Analyses (PCA)-based findings in population genetic studies are highly biased and must be reevaluated”. In: *Scientific reports* 12.1 (2022), p. 14683.

[44] Chenguang Fang and Chen Wang. *Time Series Data Imputation: A Survey on Deep Learning Approaches*. 2020. arXiv: 2011.11347 [cs.LG]. URL: https://arxiv.org/abs/2011.11347.

[45] Aosong Feng and Leandros Tassiulas. “Adaptive graph spatial-temporal transformer network for traffic flow forecasting”. In: *arXiv preprint arXiv:2207.05064* (2022).

[46] Z. Ferdousi and A. Maeda. “Unsupervised Outlier Detection in Time Series Data”. In: *22nd International Conference on Data Engineering Workshops (ICDEW’06)*. 2006, pp. x121–x121. DOI: 10.1109/ICDEW.2006.157.

[47] Fernando Gama, Joan Bruna, and Alejandro Ribeiro. “Stability properties of graph neural networks”. In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 5680–5695.

[48] Fernando Gama et al. “Graphs, Convolutions, and Neural Networks: From Graph Filters to Graph Neural Networks”. In: *IEEE Signal Processing Magazine* 37.6 (Nov. 2020), pp. 128–138. ISSN: 1558-0792. DOI: 10.1109/msp.2020.3016143. URL: http://dx.doi.org/10.1109/MSP.2020.3016143.

[49] Mahak Gambhir and Vishal Gupta. “Deep learning-based extractive text summarization with word-level attention mechanism”. In: *Multimedia Tools and Applications* 81.15 (2022), pp. 20829–20852.

[50] John Cristian Borges Gamboa. *Deep Learning for Time-Series Analysis*. 2017. arXiv: 1701.01887 [cs.LG]. URL: https://arxiv.org/abs/1701.01887.

[51] Ya Gao, Rong Wang, and Enmin Zhou. “Stock prediction based on optimized LSTM and GRU models”. In: *Scientific Programming* 2021.1 (2021), p. 4055281.

[52] N Garg et al. “Applications of Autoregressive integrated moving average (ARIMA) approach in time-series prediction of traffic noise pollution”. In: *Noise Control Engineering Journal* 63.2 (2015), pp. 182–194.

[53] Guillaume Garrigos and Robert M. Gower. *Handbook of Convergence Theorems for (Stochastic) Gradient Methods*. 2024. arXiv: 2301.11235 [math.OC]. URL: https://arxiv.org/abs/2301.11235.

[54] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. *Diffusion Improves Graph Learning*. 2022. arXiv: 1911.05485 [cs.SI]. URL: https://arxiv.org/abs/1911.05485.

[55] Justin Gilmer et al. *Neural Message Passing for Quantum Chemistry*. 2017. arXiv: 1704.01212 [cs.LG]. URL: https://arxiv.org/abs/1704.01212.

[56] Benjamin Girault. "Stationary graph signals using an isometric graph translation". In: *2015 23rd European Signal Processing Conference (EUSIPCO)*. IEEE. 2015, pp. 1516–1520.

[57] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[58] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Book in preparation for MIT Press. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[59] Tomasz Górecki et al. "Selected statistical methods of data analysis for multivariate functional data". In: *Statistical Papers* 59 (Nov. 2018), pp. 153–182. DOI: 10.1007/s00362-016-0757-8.

[60] Claudio Guarnaccia, Joseph Quartieri, and Carmine Tepedino. "Deterministic decomposition and seasonal ARIMA time series models applied to airport noise forecasting". In: *AIP Conference Proceedings*. Vol. 1836. 1. AIP Publishing. 2017.

[61] Mounir Haddad et al. "Temporalizing static graph autoencoders to handle temporal networks". In: *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ASONAM '21. Virtual Event, Netherlands: Association for Computing Machinery, 2022, pp. 201–208. DOI: 10.1145/3487351.3488333. URL: https://doi.org/10.1145/3487351.3488333.

[62] Zahra Hajirahimi and Mehdi Khashei. "Hybridization of hybrid structures for time series forecasting: A review". In: *Artificial Intelligence Review* 56.2 (2023), pp. 1201–1261.

[63] James D Hamilton and Jin Xi. *Principal Component Analysis for Nonstationary Series*. Working Paper 32068. National Bureau of Economic Research, Jan. 2024. DOI: 10.3386/w32068. URL: http://www.nber.org/papers/w32068.

[64] Chenyu Han and Xiaoyu Fu. "Challenge and opportunity: deep learning-based stock price prediction by using bi-directional LSTM model". In: *Frontiers in Business, Economics and Management* 8.2 (2023), pp. 51–54.

[65] Pradeep Hewage et al. "Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station". In: *Soft Computing* 24 (2020), pp. 16453–16482.

[66] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[67] C.C. Holt and Carnegie Inst of Tech Pittsburgh PA Graduate School of Industrial Administration. *Forecasting Seasonals and Trends by Exponentially Weighted Moving Averages*. O.N.R. research memorandum. Defense Technical Information Center, 1957. URL: https://books.google.gr/books?id=SAFFNwAACAAJ.

[68] Robin John Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. English. 2nd. Australia: OTexts, 2018.

[69] Ayad Ghany Ismaeel et al. "Traffic pattern classification in smart cities using deep recurrent neural network". In: *Sustainability* 15.19 (2023), p. 14522.

[70] Hassan Ismail Fawaz et al. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4 (Mar. 2019), pp. 917–963. ISSN: 1573-756X. DOI: 10.1007/s10618-019-00619-1. URL: http://dx.doi.org/10.1007/s10618-019-00619-1.

[71] Elvin Isufi and Gabriele Mazzola. *Graph-Time Convolutional Neural Networks*. 2021. arXiv: 2103.01730 [cs.LG]. URL: https://arxiv.org/abs/2103.01730.

[72] Elvin Isufi et al. *Graph Filters for Signal Processing and Machine Learning on Graphs*. 2024. arXiv: 2211.08854 [eess.SP]. URL: https://arxiv.org/abs/2211.08854.

[73] Anand Jaiswal, Cherian Samuel, and VM Kadabgaon. "Statistical trend analysis and forecast modeling of air pollutants". In: *Global Journal of Environmental Science and Management* 4.4 (2018), pp. 427–438.

[74] Lei Ji et al. "Carbon futures price forecasting based with ARIMA-CNN-LSTM model". In: *Procedia Computer Science* 162 (2019), pp. 33–38.

[75] Ming Jin et al. *A Survey on Graph Neural Networks for Time Series: Forecasting, Classification, Imputation, and Anomaly Detection*. 2024. arXiv: 2307.03759 [cs.LG]. URL: https://arxiv.org/abs/2307.03759.

[76] Ming Jin et al. *Towards Expressive Spectral-Temporal Graph Neural Networks for Time Series Forecasting*. 2025. arXiv: 2305.06587 [cs.LG]. URL: https://arxiv.org/abs/2305.06587.

[77] Ian T Joliffe and BJT Morgan. "Principal component analysis and exploratory factor analysis". In: *Statistical methods in medical research* 1.1 (1992), pp. 69–95.

[78] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[79] Ian Jolliffe and Jorge Cadima. "Principal component analysis: A review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374 (Apr. 2016), p. 20150202. DOI: 10.1098/rsta.2015.0202.

[80] Ian T Jolliffe. "Rotation of III-defined principal components". In: *Journal of the Royal Statistical Society Series C: Applied Statistics* 38.1 (1989), pp. 139–147.

[81] T.S. Jombart et al. "Revealing cryptic spatial patterns in genetic variability by a new multivariate method". In: *Heredity* 101 (Aug. 2008), pp. 92–103. DOI: 10.1038/hdy.2008.34.

[82] Zahra Karevan and Johan AK Suykens. “Transductive LSTM for time-series prediction: An application to weather forecasting”. In: *Neural Networks* 125 (2020), pp. 1–9.

[83] Jatinder Kaur, Kulwinder Singh Parmar, and Sarbjit Singh. “Autoregressive models in environmental forecasting time series: a theoretical and application review”. In: *Environmental Science and Pollution Research* 30.8 (2023), pp. 19617–19641.

[84] Muhammad Kazim et al. “Multilayer analysis of energy networks”. In: *Sustainable Energy, Grids and Networks* 39 (2024), p. 101407. ISSN: 2352-4677. DOI: https://doi.org/10.1016/j.segan.2024.101407. URL: https://www.sciencedirect.com/science/article/pii/S235246772400136X.

[85] Shakir Khan and Hela Alghulaiakh. “ARIMA model for accurate time series stocks forecasting”. In: *International Journal of Advanced Computer Science and Applications* 11.7 (2020).

[86] Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

[87] M Krzysko et al. “Spatio-temporal principal component analysis”. English. In: *Spatial Economic Analysis* 19.1 (2024), pp. 8–29. ISSN: 1742-1772. DOI: 10.1080/17421772.2023.2237532.

[88] Shiyong Lan et al. “Dstagnn: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting”. In: *International conference on machine learning*. PMLR. 2022, pp. 11906–11917.

[89] Shiyang Li et al. “Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting”. In: *Advances in neural information processing systems* 32 (2019).

[90] Yaguang Li et al. “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting”. In: *arXiv preprint arXiv:1707.01926* (2017).

[91] Bryan Lim and Stefan Zohren. “Time-series forecasting with deep learning: a survey”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 379.2194 (Feb. 2021), p. 20200209. ISSN: 1471-2962. DOI: 10.1098/rsta.2020.0209. URL: http://dx.doi.org/10.1098/rsta.2020.0209.

[92] Zhe Lin. “Modelling and forecasting the stock market volatility of SSE Composite Index using GARCH models”. In: *Future Generation Computer Systems* 79 (2018), pp. 960–972.

[93] Hui Liu, Xiwei Mi, and Yanfei Li. “Smart multi-step deep learning model for wind speed forecasting based on variational mode decomposition, singular spectrum analysis, LSTM network and ELM”. In: *Energy Conversion and Management* 159 (2018), pp. 54–64.

[94] Lianyi Liu and Lifeng Wu. “Holt–Winters model with grey generating operator and its application”. In: *Communications in Statistics - Theory and Methods* 51 (July 2020), pp. 1–14. DOI: 10.1080/03610926.2020.1797804.

[95] Ming-De Liu, Lin Ding, and Yu-Long Bai. “Application of hybrid model based on empirical mode decomposition, novel recurrent neural networks and the ARIMA to wind speed prediction”. In: *Energy Conversion and Management* 233 (2021), p. 113917.

[96] Shizhan Liu et al. “Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting”. In: *ICLR*. 2022.

[97] Tong Liu et al. “Time series forecasting of air quality based on regional numerical modeling in Hong Kong”. In: *Journal of Geophysical Research: Atmospheres* 123.8 (2018), pp. 4175–4196.

[98] Yijing Liu et al. “Multivariate time-series forecasting with temporal polynomial graph neural networks”. In: *Advances in neural information processing systems* 35 (2022), pp. 19414–19426.

[99] Yuwen Liu et al. “An attention-based category-aware GRU model for the next POI recommendation”. In: *International Journal of Intelligent Systems* 36.7 (2021), pp. 3174–3189.

[100] Andreas Loukas. “How Close Are the Eigenvectors of the Sample and Actual Covariance Matrices?” 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, Aug. 2017, pp. 2228–2237. URL: https://proceedings.mlr.press/v70/loukas17a.html.

[101] Cheng Lu et al. “Streaming variational probabilistic principal component analysis for monitoring of nonstationary process”. In: *Journal of Process Control* 133 (2024), p. 103134. ISSN: 0959-1524. DOI: https://doi.org/10.1016/j.jprocont.2023.103134. URL: https://www.sciencedirect.com/science/article/pii/S0959152423002214.

[102] Xuegeng Mao and Pengjian Shang. “Multivariate singular spectrum analysis for traffic time series”. In: *Physica A: Statistical Mechanics and its Applications* 526 (2019), p. 121063. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2019.121063. URL: https://www.sciencedirect.com/science/article/pii/S037843711930648X.

[103] Nico Migenda, Ralf Möller, and Wolfram Schenck. “Adaptive dimensionality reduction for neural network-based online principal component analysis”. In: *PLOS ONE* 16.3 (Mar. 2021), pp. 1–32. DOI: 10.1371/journal.pone.0248896. URL: https://doi.org/10.1371/journal.pone.0248896.

[104] Nico Migenda, Ralf Möller, and Wolfram Schenck. “Adaptive local Principal Component Analysis improves the clustering of high-dimensional data”. In: *Pattern Recognition* 146 (2024), p. 110030. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2023.110030. URL: https://www.sciencedirect.com/science/article/pii/S0031320323007276.

[105] M Nafees Muneera and P Sriramya. "An enhanced optimized abstractive text summarization traditional approach employing multi-layered attentional stacked LSTM with the attention RNN". In: *Computer Vision and Machine Intelligence Paradigms for SDGs: Select Proceedings of ICRTAC-CVMIP 2021*. Springer, 2023, pp. 303–318.

[106] Hussain Nizam et al. "Real-Time Deep Anomaly Detection Framework for Multivariate Time-Series Data in Industrial IoT". In: *IEEE Sensors Journal* 22.23 (2022), pp. 22836–22849. DOI: 10.1109/JSEN.2022.3211874.

[107] Erkki Oja. "Neural Networks, Principal Components, and Subspaces". In: *International Journal of Neural Systems* 01.01 (1989), pp. 61–68. URL: https://doi.org/10.1142/S0129065789000475.

[108] Erkki Oja. "Simplified neuron model as a principal component analyzer". In: *Journal of Mathematical Biology* 15.3 (Nov. 1982), pp. 267–273. ISSN: 0303-6812. DOI: 10.1007/BF00275687. URL: http://dx.doi.org/10.1007/BF00275687.

[109] Alexei Onatski and Chen Wang. "Spurious Factor Analysis". In: *Econometrica* 89.2 (Mar. 2021), pp. 591–614. DOI: 10.3982/ECTA16703. URL: https://ideas.repec.org/a/wly/emetrp/v89y2021i2p591-614.html.

[110] Antonio Ortega et al. "Graph Signal Processing: Overview, Challenges, and Applications". In: *Proceedings of the IEEE* 106.5 (2018), pp. 808–828. DOI: 10.1109/JPROC.2018.2820126.

[111] Massimo Pacella and Gabriele Papadia. "Evaluation of deep learning with long short-term memory networks for time series forecasting in supply chain management". In: *Procedia CIRP* 99 (2021), pp. 604–609.

[112] Aldo Pareja et al. *EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs*. 2019. arXiv: 1902.10191 [cs.LG]. URL: https://arxiv.org/abs/1902.10191.

[113] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. *On the difficulty of training Recurrent Neural Networks*. 2013. arXiv: 1211.5063 [cs.LG]. URL: https://arxiv.org/abs/1211.5063.

[114] Tuan Pham. "Time–frequency time–space LSTM for robust classification of physiological signals". In: *Scientific Reports* 11 (Mar. 2021), p. 6936. DOI: 10.1038/s41598-021-86432-7.

[115] Luana Ruiz, Luiz F. O. Chamon, and Alejandro Ribeiro. *Transferability Properties of Graph Neural Networks*. 2023. arXiv: 2112.04629 [cs.LG]. URL: https://arxiv.org/abs/2112.04629.

[116] Mohammad Sabbaqi and Elvin Isufi. *Graph-Time Convolutional Neural Networks: Architecture and Theoretical Analysis*. 2022. arXiv: 2206.15174 [cs.LG]. URL: https://arxiv.org/abs/2206.15174.

[117] Flavian Emmanuel Sapnken et al. "A whale optimization algorithm-based multivariate exponential smoothing grey-holt model for electricity price forecasting". In: *Expert Systems with Applications* 255 (2024), p. 124663. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2024.124663. URL: https://www.sciencedirect.com/science/article/pii/S0957417424015306.

[118] Amin Shabani et al. "Scaleformer: Iterative multi-scale refining transformers for time series forecasting". In: *arXiv preprint arXiv:2206.04038* (2022).

[119] Zezhi Shao et al. "Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting". In: *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 2022, pp. 1567–1577.

[120] Maxim Vladimirovich Shcherbakov et al. "A survey of forecast error measures". In: *World applied sciences journal* 24.24 (2013), pp. 171–176.

[121] Apeksha Shewalkar. "Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU". In: *Journal of Artificial Intelligence and Soft Computing Research* 9 (2019).

[122] Jonathon Shlens. *A Tutorial on Principal Component Analysis*. 2014. arXiv: 1404.1100 [cs.LG]. URL: https://arxiv.org/abs/1404.1100.

[123] D. I. Shuman et al. "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 (May 2013), pp. 83–98. ISSN: 1053-5888. DOI: 10.1109/msp.2012.2235192. URL: http://dx.doi.org/10.1109/MSP.2012.2235192.

[124] Saurabh Sihag, Gonzalo Mateos, and Alejandro Ribeiro. *Explainable Brain Age Gap Prediction in Neurodegenerative Conditions using coVariance Neural Networks*. 2025. arXiv: 2501.01510 [cs.LG]. URL: https://arxiv.org/abs/2501.01510.

[125] Saurabh Sihag, Gonzalo Mateos, and Alejandro Ribeiro. *Towards a Foundation Model for Brain Age Prediction using coVariance Neural Networks*. 2024. arXiv: 2402.07684 [q-bio.QM]. URL: https://arxiv.org/abs/2402.07684.

[126] Saurabh Sihag et al. *coVariance Neural Networks*. 2023. arXiv: 2205.15856 [cs.LG]. URL: https://arxiv.org/abs/2205.15856.

[127] Saurabh Sihag et al. *Explainable Brain Age Prediction using coVariance Neural Networks*. 2023. arXiv: 2305.18370 [q-bio.QM]. URL: https://arxiv.org/abs/2305.18370.

[128] Saurabh Sihag et al. "Novel Framework for Brain Age Prediction using Graph Neural Networks". In: *Alzheimer's & Dementia* 19.S17 (2023), e079038. DOI: https://doi.org/10.1002/alz.079038. eprint: https://alz-journals.onlinelibrary.wiley.com/doi/pdf/10.1002/alz.079038. URL: https://alz-journals.onlinelibrary.wiley.com/doi/abs/10.1002/alz.079038.

[129] Saurabh Sihag et al. *Predicting Brain Age using Transferable coVariance Neural Networks*. 2022. arXiv: 2210.16363 [cs.LG]. URL: https://arxiv.org/abs/2210.16363.

[130] Saurabh Sihag et al. *Transferability of coVariance Neural Networks and Application to Interpretable Brain Age Prediction using Anatomical Features*. 2023. arXiv: 2305.01807 [cs.LG]. URL: https://arxiv.org/abs/2305.01807.

[131] Slawek Smyl. "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting". In: *International Journal of Forecasting* 36.1 (2020). M4 Competition, pp. 75–85. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2019.03.017. URL: https://www.sciencedirect.com/science/article/pii/S0169207019301153.

[132] Slawek Smyl, Grzegorz Dudek, and Paweł Pełka. *ES-dRNN: A Hybrid Exponential Smoothing and Dilated Recurrent Neural Network Model for Short-Term Load Forecasting*. 2021. arXiv: 2112.02663 [cs.LG]. URL: https://arxiv.org/abs/2112.02663.

[133] Chao Song et al. "Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 01. 2020, pp. 914–921.

[134] G. W. Stewart and Ji guang Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

[135] Akylas Stratigakos et al. "Short-Term Net Load Forecasting with Singular Spectrum Analysis and LSTM Neural Networks". In: *Energies* 14.14 (2021). ISSN: 1996-1073. DOI: 10.3390/en14144107. URL: https://www.mdpi.com/1996-1073/14/14/4107.

[136] Wenyue Sun, Chuan Tian, and Guang Yang. "Network analysis of the stock market". In: *CS224W Project Report* (2015).

[137] Mohammad Amaz Uddin et al. "The Efficacy of Deep Learning-Based Mixed Model for Speech Emotion Recognition." In: *Computers, Materials & Continua* 74.1 (2023).

[138] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*. Vol. 47. Cambridge university press, 2018.

[139] SAL Wadi et al. "Predicting closed price time series data using ARIMA Model". In: *Modern Applied Science* 12.11 (2018), pp. 181–185.

[140] Renzhuo Wan et al. "Multivariate Temporal Convolutional Network: A Deep Neural Networks Approach for Multivariate Time Series Forecasting". In: *Electronics* 8.8 (2019). ISSN: 2079-9292. DOI: 10.3390/electronics8080876. URL: https://www.mdpi.com/2079-9292/8/8/876.

[141] Chia-Hung Wang et al. "A deep learning integrated framework for predicting stock index price and fluctuation via singular spectrum analysis and particle swarm optimization". In: *Applied Intelligence* 54.2 (Jan. 2024), pp. 1770–1797. ISSN: 0924-669X. DOI: 10.1007/s10489-024-05271-x. URL: https://doi-org.tudelft.idm.oclc.org/10.1007/s10489-024-05271-x.

[142] Jing Wang, Dan S Rickman, and Yihua Yu. "Dynamics between global value chain participation, CO2 emissions, and economic growth: Evidence from a panel vector autoregression model". In: *Energy Economics* 109 (2022), p. 105965.

[143] Xiaoyang Wang et al. “Traffic flow prediction via spatial temporal graph neural network”. In: *Proceedings of the web conference 2020*. 2020, pp. 1082–1092.

[144] Yuxuan Wang et al. *Deep Time Series Models: A Comprehensive Survey and Benchmark*. 2024. arXiv: 2407.13278 [cs.LG]. URL: https://arxiv.org/abs/2407.13278.

[145] Simon Wein et al. “Forecasting brain activity based on models of spatiotemporal brain dynamics: A comparison of graph neural network architectures”. In: *Network Neuroscience* 6.3 (2022), pp. 665–701.

[146] Martin D Weinberg and Michael S Petersen. “Using multichannel singular spectrum analysis to study galaxy dynamics”. In: *Monthly Notices of the Royal Astronomical Society* 501.4 (Dec. 2020), pp. 5408–5423. ISSN: 1365-2966. DOI: 10.1093/mnras/staa3997. URL: http://dx.doi.org/10.1093/mnras/staa3997.

[147] Qingsong Wen et al. *Transformers in Time Series: A Survey*. 2023. arXiv: 2202.07125 [cs.LG]. URL: https://arxiv.org/abs/2202.07125.

[148] Qingsong Wen et al. “Transformers in time series: A survey”. In: *arXiv preprint arXiv:2202.07125* (2022).

[149] I Komang Arya Ganda Wiguna et al. “Rainfall Forecasting Using the Holt-Winters Exponential Smoothing Method”. In: *Jurnal Info Sains : Informatika dan Sains* 13.01 (Mar. 2023), pp. 15–23. URL: https://ejournal.seaninstitute.or.id/index.php/InfoSains/article/view/2656.

[150] James Hardy Wilkinson. *The algebraic eigenvalue problem*. Oxford University Press, Inc., 1988.

[151] Billy M Williams and Lester A Hoel. “Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results”. In: *Journal of transportation engineering* 129.6 (2003), pp. 664–672.

[152] Peter R. Winters. “Forecasting Sales by Exponentially Weighted Moving Averages”. In: *Manage. Sci.* 6.3 (Apr. 1960), pp. 324–342. ISSN: 0025-1909. DOI: 10.1287/mnsc.6.3.324. URL: https://doi-org.tudelft.idm.oclc.org/10.1287/mnsc.6.3.324.

[153] Gerald Woo et al. *ETSformer: Exponential Smoothing Transformers for Time-series Forecasting*. 2022. arXiv: 2202.01381 [cs.LG]. URL: https://arxiv.org/abs/2202.01381.

[154] Lifeng Wu et al. “Using grey Holt–Winters model to predict the air quality index for cities in China”. In: *Natural Hazards* 88 (Sept. 2017), pp. 1–10. DOI: 10.1007/s11069-017-2901-8.

[155] Zonghan Wu et al. “A comprehensive survey on graph neural networks”. In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[156] Zonghan Wu et al. “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*. 2020, pp. 753–763.

[157] Zonghan Wu et al. “Graph wavenet for deep spatial-temporal graph modeling”. In: *arXiv preprint arXiv:1906.00121* (2019).

[158] Qu Xiaoyun et al. “Short-term prediction of wind power based on deep long short-term memory”. In: *2016 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*. IEEE. 2016, pp. 1148–1152.

[159] Mingxing Xu et al. “Spatial-temporal transformer networks for traffic flow forecasting”. In: *arXiv preprint arXiv:2001.02908* (2020).

[160] Jinsung Yoon, William R Zame, and Mihaela Van Der Schaar. “Estimating missing data in temporal data streams using multi-directional recurrent neural networks”. In: *IEEE Transactions on Biomedical Engineering* 66.5 (2018), pp. 1477–1490.

[161] Bing Yu, Haoteng Yin, and Zhanxing Zhu. “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting”. In: *arXiv preprint arXiv:1709.04875* (2017).

[162] Cunjun Yu et al. “Spatio-temporal graph transformer networks for pedestrian trajectory prediction”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*. Springer. 2020, pp. 507–523.

[163] Zhiwen Zeng and Matloob Khushi. “Wavelet denoising and attention-based RNN-ARIMA model to predict forex price”. In: *2020 International joint conference on neural networks (IJCNN)*. IEEE. 2020, pp. 1–7.

[164] Lanyi Zhang et al. “Trend analysis and forecast of PM2. 5 in Fuzhou, China using the ARIMA model”. In: *Ecological indicators* 95 (2018), pp. 702–710.

[165] Qi Zhang et al. “Spatio-temporal graph structure learning for traffic forecasting”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 01. 2020, pp. 1177–1185.

[166] Bendong Zhao et al. “Convolutional neural networks for time series classification”. In: *Journal of systems engineering and electronics* 28.1 (2017), pp. 162–169.

[167] Yongning Zhao et al. “Correlation-constrained and sparsity-controlled vector autoregressive model for spatio-temporal wind power forecasting”. English. In: *IEEE Transactions on Power Systems* 33.5 (2018), pp. 5029–5040. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2018.2794450.

[168] Chuanpan Zheng et al. “Gman: A graph multi-attention network for traffic prediction”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 01. 2020, pp. 1234–1241.

[169] Haoyi Zhou et al. “Informer: Beyond efficient transformer for long sequence time-series forecasting”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 12. 2021, pp. 11106–11115.

[170] Tian Zhou et al. “Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting”. In: *International conference on machine learning*. PMLR. 2022, pp. 27268–27286.

A
Stability Proofs

A.1. Proof of Theorem 1

The following provides a proof of Theorem 1. To bound the filter output difference $\left\|\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}_t,\tilde{\mathbf{x}}_t)-\mathbf{H}(\tilde{\mathbf{C}},\mathbf{h}^*,\tilde{\mathbf{x}}_t)\right\|$, we follow a similar approach to the proof of the stability of STVF in [31, Appendix B]. First, we add and subtract the term $\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)$ within the norm and, using the triangle inequality, we have the following upper bound:

$$\begin{aligned}
&\left\|\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}_t,\tilde{\mathbf{x}}_t)-\mathbf{H}(\tilde{\mathbf{C}},\mathbf{h}^*,\tilde{\mathbf{x}}_t)\right\| \\
\leq&\left\|\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}_t,\tilde{\mathbf{x}}_t)-\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)\right\|+\left\|\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)-\mathbf{H}(\tilde{\mathbf{C}},\mathbf{h}^*,\tilde{\mathbf{x}}_t)\right\| \\
\leq&\quad \alpha_t + \beta_t
\end{aligned} \tag{A.1}$$

where $\left\|\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}_t,\tilde{\mathbf{x}}_t)-\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)\right\|\leq\alpha_t$ corresponds to the parameter sub-optimality error, while $\left\|\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)-\mathbf{H}(\tilde{\mathbf{C}},\mathbf{h}^*,\tilde{\mathbf{x}}_t)\right\|\leq\beta_t$ corresponds to the covariance uncertainty error. Next, we will upper bound each of these terms and, therefore, establish an upper bound for the lagged spatiotemporal covariance filter.

Parameter Sub-Optimality Error

Given $\mathbf{h}^*$ as the optimal set of coefficients for the multivariate time series forecasting problem, we formulate the problem of optimizing $\mathbf{h}_t$ as:

$$\min_{\mathbf{h}_t}\left\|\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}_t,\tilde{\mathbf{x}}_t)-\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)\right\|^2 \tag{A.2}$$

This constitutes an unconstrained convex optimization problem. Assuming that $\mathbf{h}_t$ is updated via online gradient descent with a learning rate $\eta>0$ small enough to guarantee convergence, we exploit the upper bound of the distance between the online update and the optimal solution in [53, Theorem 3.4] to obtain:

$$\alpha_t\leq\frac{T\,\|\mathbf{h}^*\|^2}{2\eta t} \tag{A.3}$$

Covariance Uncertainty Error

We begin by expressing the estimated extended covariance matrix as $\hat{\tilde{\mathbf{C}}}_t=\tilde{\mathbf{C}}+\mathbf{E}$, where $\mathbf{E}$ is an error matrix, and analyzing the Taylor expansion of $\hat{\tilde{\mathbf{C}}}_t^k$ as:

$$\hat{\tilde{\mathbf{C}}}_t^k=(\tilde{\mathbf{C}}+\mathbf{E})^k=\tilde{\mathbf{C}}^k+\sum_{r=0}^{k-1}\tilde{\mathbf{C}}^r\mathbf{E}\tilde{\mathbf{C}}^{k-r-1}+\bar{\mathbf{E}} \tag{A.4}$$

where $\bar{\mathbf{E}}$ is such that $\left\|\bar{\mathbf{E}}\right\|=\mathcal{O}(\|\mathbf{E}\|^2)$. For this proof of the standard L-STVF, we ignore this last term $\bar{\mathbf{E}}$, as we consider the error to be very small ($\|\mathbf{E}\|\ll 1$) after a few training steps ($t\gg 0$). Therefore, we analyze the filter output difference for the covariance uncertainty error by using the eigendecomposition of the true extended and sample extended covariance matrices, $\tilde{\mathbf{C}}=\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$ and $\hat{\tilde{\mathbf{C}}}_t=\hat{\mathbf{V}}\hat{\boldsymbol{\Lambda}}\hat{\mathbf{V}}^\top$ respectively, and the first two terms as shown in (A.4):

$$\begin{aligned}\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t,\mathbf{h}^*,\tilde{\mathbf{x}}_t)-\mathbf{H}(\tilde{\mathbf{C}},\mathbf{h}^*,\tilde{\mathbf{x}}_t) &= \sum_{k=0}^{K} h_k \sum_{r=0}^{k-1} \tilde{\mathbf{C}}^r\mathbf{E}\tilde{\mathbf{C}}^{k-r-1}\tilde{\mathbf{x}}_t\\ &= \sum_{i=0}^{NT-1} \bar{x}_{t,i}\sum_{k=0}^{K} h_k \sum_{r=0}^{k-1} \tilde{\mathbf{C}}^r\mathbf{E}\tilde{\mathbf{C}}^{k-r-1}\mathbf{v}_i\\ &= \sum_{i=0}^{NT-1} \bar{x}_{t,i}\sum_{k=0}^{K} h_k \sum_{r=0}^{k-1} \tilde{\mathbf{C}}^r\lambda_i^{k-r-1}\mathbf{E}\mathbf{v}_i \qquad \text{(A.5)}\end{aligned}$$

where $\bar{x}_{t,i}$ is the i-th entry of the graph Fourier transform of signal $\tilde{\mathbf{x}}_t$, which is $\bar{\mathbf{x}}_t = \mathbf{V}^\top\tilde{\mathbf{x}}_t$. We will now expand (A.5) into three terms, which we will bound separately and, therefore, compute an upper bound for the covariance uncertainty error:

$$\sum_{i=0}^{NT-1} \bar{x}_{t,i}\sum_{k=0}^{K} h_k \sum_{r=0}^{k-1} \tilde{\mathbf{C}}^r\lambda_i^{k-r-1}(\lambda_i\mathbf{I}_{NT}-\tilde{\mathbf{C}})(\hat{\mathbf{v}}_i-\mathbf{v}_i) \qquad \text{(A.6)}$$

$$+\sum_{i=0}^{NT-1} \bar{x}_{t,i}\sum_{k=0}^{K} h_k \sum_{r=0}^{k-1} \tilde{\mathbf{C}}^r\lambda_i^{k-r-1}(\hat{\lambda}_i-\lambda_i)\mathbf{v}_i \qquad \text{(A.7)}$$

$$+\sum_{i=0}^{NT-1} \bar{x}_{t,i}\sum_{k=0}^{K} h_k \sum_{r=0}^{k-1} \tilde{\mathbf{C}}^r\lambda_i^{k-r-1}((\hat{\lambda}_i-\lambda_i)\mathbf{I}_{NT}-\mathbf{E})(\hat{\mathbf{v}}_i-\mathbf{v}_i) \qquad \text{(A.8)}$$

First term (A.6). We start by plugging the eigendecompositions of the covariance matrices into (A.6) to get:

$$\begin{aligned}&\sum_{i=0}^{NT-1} \bar{x}_{t,i}\sum_{k=0}^{K} h_k \sum_{r=0}^{k-1} \tilde{\mathbf{C}}^r\lambda_i^{k-r-1}\mathbf{V}(\lambda_i\mathbf{I}_{NT}-\mathbf{\Lambda})\mathbf{V}^\top(\hat{\mathbf{v}}_i-\mathbf{v}_i)\\ =&\sum_{i=0}^{NT-1} \bar{x}_{t,i}\sum_{k=0}^{K} h_k \sum_{r=0}^{k-1} \lambda_i^{k-r-1}\mathbf{V}\mathbf{\Lambda}^r(\lambda_i\mathbf{I}_{NT}-\mathbf{\Lambda})\mathbf{V}^\top(\hat{\mathbf{v}}_i-\mathbf{v}_i)\\ =&\sum_{i=0}^{NT-1} \bar{x}_{t,i}\mathbf{V}\mathbf{L}_i\mathbf{V}^\top(\hat{\mathbf{v}}_i-\mathbf{v}_i) \qquad \text{(A.9)}\end{aligned}$$

where $\mathbf{L}_i$ is a diagonal matrix, whose j-th diagonal element is:

$$\mathbf{L}_{i,j}=\begin{cases}0, & \text{if}\quad i=j\\ h(\lambda_i)-h(\lambda_j), & \text{if}\quad i\neq j,\end{cases} \qquad \text{(A.10)}$$

where $h(\lambda_i)$ is the frequency response of the lagged spatiotemporal covariance filter defined in (4.5). Expanding this with the last terms in (A.9), we get:

$$[\mathbf{L}_i\mathbf{V}^\top(\hat{\mathbf{v}}_i-\mathbf{v}_i)]_j=\begin{cases}0, & \text{if}\quad i=j\\ (h(\lambda_i)-h(\lambda_j))\mathbf{v}_j^\top\hat{\mathbf{v}}_i, & \text{if}\quad i\neq j\end{cases} \qquad \text{(A.11)}$$

Then, we consider the norm of (A.9). By plugging in (A.11) and successively applying the triangle and Cauchy-Schwarz inequalities, the orthonormality of the eigenvectors, and the bound $\|\mathbf{y}\| \leq \sqrt{M} \max_i y_i$ for any arbitrary vector $\mathbf{y} \in \mathbb{R}^M$, we get:

$$\left\| \sum_{i=0}^{NT-1} \bar{x}_{t,i} \mathbf{V} \mathbf{L}_i \mathbf{V}^\top (\hat{\mathbf{v}}_i - \mathbf{v}_i) \right\| \leq \sum_{i=0}^{NT-1} \left| \bar{x}_{t,i} \right| \|\mathbf{V}\| \left\| \mathbf{L}_i \mathbf{V}^\top (\hat{\mathbf{v}}_i - \mathbf{v}_i) \right\|$$
$$\leq \sqrt{NT} \sum_{i=0}^{NT-1} \left| \bar{x}_{t,i} \right| \max_j \left| h(\lambda_i) - h(\lambda_j) \right| \left| \mathbf{v}_j^\top \hat{\mathbf{v}}_i \right| \tag{A.12}$$

Under As. 3, we leverage the result of [100, Theorem 4.1] to characterize the dot product of the eigenvectors of the true and sample extended covariance matrices:

$$\mathbb{P}(\left| \mathbf{v}_j^\top \mathbf{v}_i \right| \geq B) \leq \frac{T}{t} \left(\frac{2k_j}{B \left| \lambda_i - \lambda_j \right|} \right)^2 \tag{A.13}$$

where the term k_j is related to the kurtosis of the data distribution [100, 126]. Then, we set:

$$B = \sqrt{\frac{T}{t} \frac{2k_j e^{\epsilon/2}}{\left| \lambda_i - \lambda_j \right|}} \tag{A.14}$$

and, with respect to the last terms in (A.12), we get:

$$\max_j \left| h(\lambda_i) - h(\lambda_j) \right| \left| \mathbf{v}_j^\top \hat{\mathbf{v}}_i \right| \leq \max_j \frac{\left| h(\lambda_i) - h(\lambda_j) \right|}{\left| \lambda_i - \lambda_j \right|} \frac{2\sqrt{T} k_j e^{\epsilon/2}}{\sqrt{t}} \tag{A.15}$$

with probability at least $1 - e^{-\epsilon}$. Finally, by using the fact that the frequency responses $h(\lambda)$ of the filters are Lipschitz with a constant P by As. 1, and leveraging $\|\tilde{\mathbf{x}}_t\| \leq 1$ for all t, with $\sum_{i=0}^{NT-1} \left| \bar{x}_{t,i} \right| \leq \sqrt{NT} \|\tilde{\mathbf{x}}_t\|$ and $k_{\max} = \max_j k_j$, we bound the term in (A.6) as:

$$\left\| \sum_{i=0}^{NT-1} \bar{x}_{t,i} \sum_{k=0}^{K} h_k \sum_{r=0}^{k-1} \tilde{\mathbf{C}}^r \lambda_i^{k-r-1} (\lambda_i \mathbf{I}_{NT} - \tilde{\mathbf{C}})(\hat{\mathbf{v}}_i - \mathbf{v}_i) \right\| \leq \sum_{i=0}^{NT-1} \left| \bar{x}_{t,i} \right| \frac{2PT\sqrt{N} k_{\max} e^{\epsilon/2}}{\sqrt{t}}$$
$$\leq \frac{2}{\sqrt{t}} P k_{\max} e^{\epsilon/2} T^{3/2} N \tag{A.16}$$

with probability at least $1 - e^{-\epsilon}$.

Second term (A.7). We rewrite (A.7) as:

$$\sum_{i=0}^{NT-1} \bar{x}_{t,i} \sum_{k=0}^{K} h_k \sum_{r=0}^{k-1} \tilde{\mathbf{C}}^r \lambda_i^{k-r-1} (\hat{\lambda}_i - \lambda_i) \mathbf{v}_i = \sum_{i=0}^{NT-1} \bar{x}_{t,i} \sum_{k=0}^{K} h_k \sum_{r=0}^{k-1} \lambda_i^r \lambda_i^{k-r-1} (\hat{\lambda}_i - \lambda_i) \mathbf{v}_i$$
$$= \sum_{i=0}^{NT-1} \bar{x}_{t,i} \sum_{k=0}^{K} k h_k \lambda_i^{k-1} (\hat{\lambda}_i - \lambda_i) \mathbf{v}_i$$
$$= \sum_{i=0}^{NT-1} \bar{x}_{t,i} h'(\lambda_i) (\hat{\lambda}_i - \lambda_i) \mathbf{v}_i \tag{A.17}$$

where $h'(\lambda_i)$ is the derivative of $h(\lambda_i)$ w.r.t. λ. By taking the norm of (A.17), similarly to the first term, we get:

$$\left\| \sum_{i=0}^{NT-1} \bar{x}_{t,i} h'(\lambda_i)(\hat{\lambda}_i - \lambda_i)\mathbf{v}_i \right\| \le \sum_{i=0}^{NT-1} \left|\bar{x}_{t,i}\right| \left|h'(\lambda_i)\right| \left|(\hat{\lambda}_i - \lambda_i)\right| \|\mathbf{v}_i\| \tag{A.18}$$

Among the terms in the right-hand side of (A.18), we have that $\|\mathbf{v}_i\| = 1$, the filter derivative $h'(\lambda_i)$ is bounded by P from As. 1, and $\|\tilde{\mathbf{x}}_t\| \le 1$ for all t, with $\sum_{i=0}^{NT-1} \left|\bar{x}_{t,i}\right| \le \sqrt{NT}\,\|\tilde{\mathbf{x}}_t\|$. Then, to bound the eigenvalue difference, we use Weyl's Theorem [57, Theorem 8.1.6], where $\left\|\bar{\mathbf{E}}\right\| \le \alpha$ implies that $\left|\hat{\lambda}_i - \lambda_i\right| \le \alpha$ for any $\alpha > 0$, and the result from [138, Theorem 5.6.1]:

$$\mathbb{P}\left(\|\mathbf{E}\| \le \underbrace{Q\left(\sqrt{\frac{G^2T^2N(\log(NT)+u)}{t}} + \frac{G^2T^2N(\log(NT)+u)}{t} \right) \left\|\tilde{\mathbf{C}}\right\|}_{\alpha} \right) \ge 1 - 2e^{-u} \tag{A.19}$$

where Q is an absolute value and $G \ge 1$ derives from As. 2. Putting all these together, we derive the following upper bound for the term in A.7:

$$\begin{aligned} &\left\| \sum_{i=0}^{NT-1} \bar{x}_{t,i} \sum_{k=0}^{K} h_k \sum_{r=0}^{k-1} \tilde{\mathbf{C}}^r \lambda_i^{k-r-1} (\hat{\lambda}_i - \lambda_i)\mathbf{v}_i \right\| \\ \le &P\sqrt{NT}Q\left(\sqrt{\frac{G^2T^2N(\log(NT)+u)}{t}} + \frac{G^2T^2N(\log(NT)+u)}{t} \right) \left\|\tilde{\mathbf{C}}\right\| \end{aligned} \tag{A.20}$$

with probability at least $1 - 2e^{-u}$.

Third term (A.8). As in the proof of STVF [31], we can leverage the equations (65)-(68) in [126] with minimal changes and show that this term scales as $\mathcal{O}(1/t)$.

By bringing together the upper bounds in (A.16) and (A.20), along with the observation that (A.8) scales as $\mathcal{O}(1/t)$, we derive an upper bound for the covariance uncertainty error. Further combining this bound with the parameter sub-optimality bound in (A.3),we derive the upper bound of the lagged spatiotemporal covariance filter, as shown in (4.14).

A.2. PROOF OF THEOREM 2

The following provides a proof of Theorem 2. To bound the filter output difference when hard thresholding is applied, we will rely on our previous proof of the stability of L-STVF in Section A.1 and the corresponding proof of the stability of sparse VNN with hard thresholding in [30, Appendix C]. Again, we would break down the computation of the upper bound to bounding the parameter sub-optimality error and the covariance uncertainty error, as in (A.1). The parameter sub-optimality error remains the same as in (A.3) for the standard L-STVF. In order to compute the bound for the covariance uncertainty error, we perform the same analysis of the filter difference into the 3 terms in (A.6), (A.7), and (A.8), and address each one of them individually.

First term (A.6). From [150, Chapter 2, (10.2)], assuming the true extended covariance matrix $\tilde{\mathbf{C}}$ is positive definite with distinct eigenvalues, the first-order approximation of the i-th estimated covariance eigenvector $\hat{\mathbf{v}}_i$ is:

$$\hat{\mathbf{v}}_i \approx \mathbf{v}_i + \sum_{k=1,k\neq i}^{NT} \frac{\mathbf{v}_k^\top \mathbf{E}\mathbf{v}_i}{\lambda_k - \lambda_i} + \mathbf{D}, \tag{A.21}$$

with $\|\mathbf{D}\| = \mathcal{O}(\|\mathbf{E}\|^2)$. By projecting on $\mathbf{v}_j$ and taking the absolute value, we get:

$$\begin{aligned}\left|\mathbf{v}_j^\top \hat{\mathbf{v}}_i\right| \leq & \frac{\left|\mathbf{v}_j^\top \mathbf{E}\mathbf{v}_i\right|}{\left|\lambda_i - \lambda_j\right|} + \mathcal{O}(\|\mathbf{E}\|^2) \\ \leq & \frac{\|\mathbf{E}\|}{\left|\lambda_i - \lambda_j\right|} + \mathcal{O}(\|\mathbf{E}\|^2),\end{aligned} \tag{A.22}$$

where the last step follows from the spectral norm definition $\|\mathbf{E}\| = \max_{\mathbf{v},\|\mathbf{v}\|_2=1} \left|\mathbf{v}^\top \mathbf{E}\mathbf{v}\right|$.

Then, we consider the norm of (A.6), deriving the inequality in (A.12). By plugging (A.22) into (A.12), and using the Lipschitz property of the filter by As. 1 and the inequality $\sum_{i=0}^{NT-1} \left|\bar{x}_{t,i}\right| \leq \sqrt{NT}\,\|\bar{\mathbf{x}}_t\|$, we have:

$$\begin{aligned}\sqrt{NT} \sum_{i=0}^{NT-1} \left|\bar{x}_{t,i}\right| \max_{j\neq i} \left|h(\lambda_i) - h(\lambda_j)\right| \left|\mathbf{v}_j^\top \hat{\mathbf{v}}_i\right| \leq & \sqrt{NT}\,\|\mathbf{E}\| \sum_{i=0}^{NT-1} \left|\bar{x}_{t,i}\right| \max_{j\neq i} \frac{\left|h(\lambda_i) - h(\lambda_j)\right|}{\left|\lambda_i - \lambda_j\right|} \\ \leq & \sqrt{NT}\,\|\mathbf{E}\| \sum_{i=0}^{NT-1} \left|\bar{x}_{t,i}\right| P \\ \leq & PNT\,\|\mathbf{E}\|\end{aligned} \tag{A.23}$$

Second term (A.7). We leverage the expression (A.17) and Weyl's Theorem [57, Theorem 8.1.6] to derive:

$$\begin{aligned}\left\|\sum_{i=0}^{NT-1} \bar{x}_{t,i} h'(\lambda_i)(\hat{\lambda}_i - \lambda_i)\mathbf{v}_i\right\| \leq & \sum_{i=0}^{NT-1} \left|\bar{x}_{t,i}\right| \left|h'(\lambda_i)\right| \|\mathbf{E}\|\,\|\mathbf{v}_i\| \\ \leq & \sum_{i=0}^{NT-1} \left|\bar{x}_{t,i}\right| P\,\|\mathbf{E}\| \\ \leq & P\sqrt{NT}\,\|\mathbf{E}\|\end{aligned} \tag{A.24}$$

Third term (A.8). Analogous to [30], from Weyl's Theorem [57, Theorem 8.1.6] and the sin-theta theorem [134, Theorem V.3.6], we derive that this term scales as $\mathcal{O}(\|\mathbf{E}\|^2)$. Under small perturbation assumption, this term is dominated by the terms in (A.6) and (A.7).

By bringing together the bounds in (A.23) and (A.24), along with the previous observation that (A.8) scales as $\mathcal{O}(\|\mathbf{E}\|^2)$, we derive the following upper bound for the covariance uncertainty error w.r.t. $\|\mathbf{E}\|$:

$$\left\|\mathbf{H}(\hat{\tilde{\mathbf{C}}}_t, \mathbf{h}^*, \tilde{\mathbf{x}}_t) - \mathbf{H}(\tilde{\mathbf{C}}, \mathbf{h}^*, \tilde{\mathbf{x}}_t)\right\| \le P\sqrt{NT}\,\|\mathbf{E}\|\,(1+\sqrt{NT}) \;+\; \mathcal{O}(\|\mathbf{E}\|^2) \tag{A.25}$$

Adjusting [10, Theorem 1] to the extended covariance setting, given a true extended covariance matrix $\tilde{\mathbf{C}}$ and a hard-thresholded sample extended covariance matrix $\hat{\tilde{\mathbf{C}}}_t$, it holds with high probability that:

$$\left\|\hat{\tilde{\mathbf{C}}}_t - \tilde{\mathbf{C}}\right\| = \|\mathbf{E}\| \le c_0\sqrt{\frac{T\log(NT)}{t}} \tag{A.26}$$

By replacing (A.26) into (A.25), we compute the bound of the covariance uncertainty error. Combining this with the parameter sub-optimality error in (A.3), we derive the upper bound of L-STVF with hard thresholding, as shown in (4.20).

A.3. PROOF OF THEOREM 3

The following provides a proof of Theorem 3. Adjusting [41, Theorem 1] to the extended covariance setting, given a true extended covariance matrix $\tilde{\mathbf{C}}$ and a soft-thresholded sample extended covariance matrix $\hat{\tilde{\mathbf{C}}}_t$, it holds with high probability that:

$$\left\|\hat{\tilde{\mathbf{C}}}_t - \tilde{\mathbf{C}}\right\| = \|\mathbf{E}\| \le \sqrt{\frac{T}{t}} C c_0 \max(1,\lambda_0)\sqrt{\max\left(\log\frac{NT}{c_0^2}, 1\right)} \tag{A.27}$$

for a generic constant C. By replacing $\|\mathbf{E}\|$ from (A.27) into (A.25), we obtain the upper bound of the covariance uncertainty error with soft thresholding. Combining this bound with the upper bound in (A.3) for the parameter sub-optimality error, we derive the complete bound for the filter output difference, as shown in (4.22).

B

Synthetic Dataset Generation

To generate stationary synthetic datasets, we begin with a covariance matrix $\mathbf{C}$ and sample observations $\mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$. We then introduce temporal causality by constructing the multivariate time series $\mathbf{x}_t$ as a weighted sum of past observations: $\mathbf{x}_t = \sum_{t'=0}^{\tau} h_{t'} \mathbf{z}_{t-t'}$, where the weights $h_{t'}$ are normalized using $h_{t'} = h'_{t'} / \sqrt{\sum_t h'_{t'}}$ and $h_{t'} = e^{-t}$ for $t = 0, \ldots, \tau$. In our experiments, we set $\tau = 9$. To shape the eigenvalue distribution of the covariance matrices used in the synthetic datasets, we utilize the *sklearn.make_regression* function, which allows us to adjust the size of the distribution tail and thereby control the spacing between eigenvalues. Figure B.1 displays the eigenvalue spectra for the three covariance matrices used, where larger tails correspond to higher kurtosis and more closely spaced eigenvalues.

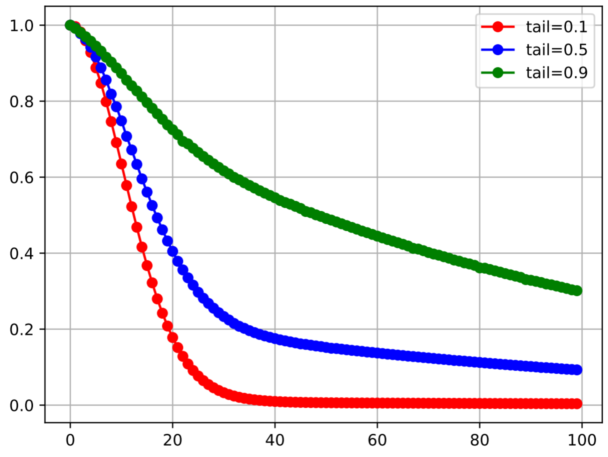


Figure B.1: Eigenvalue distribution for different tail sizes.

C
Hyper-parameter Configurations

TPCA Configuration

Dataset	Horizon	Parameters
		Temporal Window T
Molene	1	3
	3	3
	5	5
Exchange Rates	1	2
	3	2
	5	2
NOAA	1	3
	3	5
	5	8

LSTM Configuration

Dataset	Horizon	Parameters
		Temporal Window T
Molene	1	3
	3	3
	5	3
Exchange Rates	1	2
	3	2
	5	2
NOAA	1	3
	3	3
	5	5

VNN-VNNL-STVNN Configuration

Dataset	Horizon	Parameters					
		Learning Rate	Coefficient γ	Optimizer	Layers	Temporal Window T	Filter Order K
Molene	1	0.001	0.1	Adam	[32]	3	3
	3	0.001	0.1	Adam	[32,16]	3	3
	5	0.001	0.1	Adam	[32,16]	3	3
Exchange Rates	1	0.001	0.1	Adam	[32]	2	2
	3	0.001	0.1	Adam	[32]	2	2
	5	0.001	0.1	Adam	[64]	2	3
NOAA	1	0.001	0.05	Adam	[32]	2	2
	3	0.001	0.05	Adam	[32]	3	2
	5	0.001	0.05	Adam	[64,32]	5	2

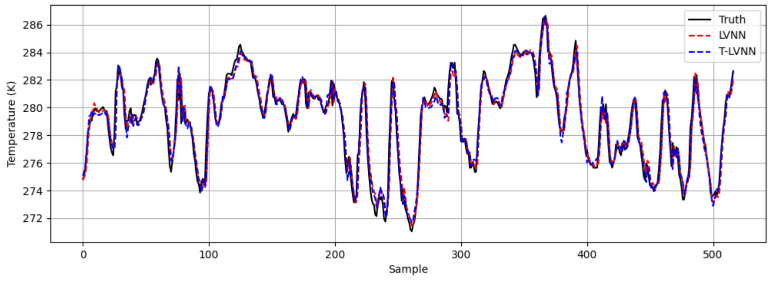
LVNN Configuration - Full Extended Covariance Matrix

		Parameters					
Dataset	Horizon	Learning Rate	Coefficient γ	Optimizer	Layers	Temporal Window T	Filter Order K
Molene	1	0.001	0.3	Adam	[32]	3	2
	3	0.001	0.05	Adam	[256]	3	2
	5	0.001	0.05	Adam	[32]	3	2
Exchange Rates	1	0.001	0.1	Adam	[16]	2	2
	3	0.001	0.1	Adam	[32]	2	2
	5	0.001	0.1	Adam	[16]	2	3
NOAA	1	0.001	0.05	Adam	[32]	2	3
	3	0.001	0.05	Adam	[64]	8	3
	5	0.001	0.05	Adam	[32]	12	2

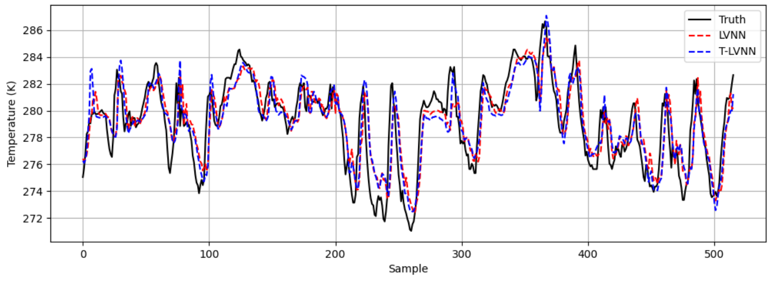
LVNN Configuration - Block Upper Triangular Matrix

		Parameters					
Dataset	Horizon	Learning Rate	Coefficient γ	Optimizer	Layers	Temporal Window T	Filter Order K
Molene	1	0.001	0.05	Adam	[32,16]	3	3
	3	0.001	0.05	Adam	[64,32]	3	4
	5	0.001	0.05	Adam	[64,32]	3	2
Exchange Rates	1	0.001	0.1	Adam	[16]	2	2
	3	0.001	0.1	Adam	[16]	2	3
	5	0.001	0.1	Adam	[16]	2	3
NOAA	1	0.001	0.05	Adam	[32]	2	3
	3	0.001	0.05	Adam	[64,32]	8	2
	5	0.001	0.05	Adam	[32]	12	2

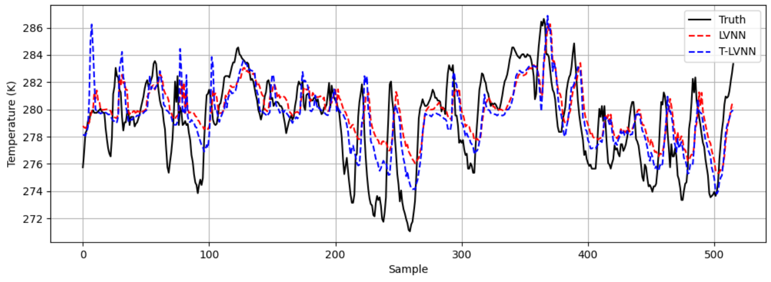
D
Single-Step Forecasting Supplementary Material



(a) Forecasting horizon 1

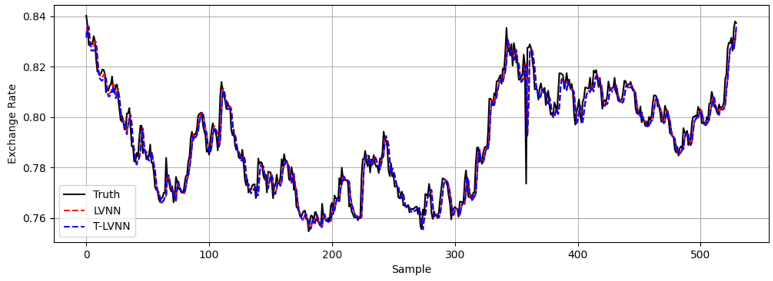


(b) Forecasting horizon 3

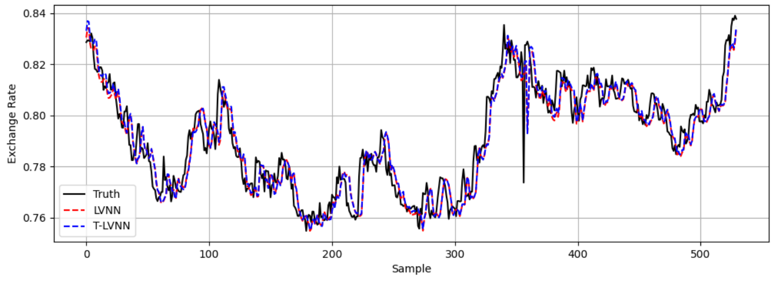


(c) Forecasting horizon 5

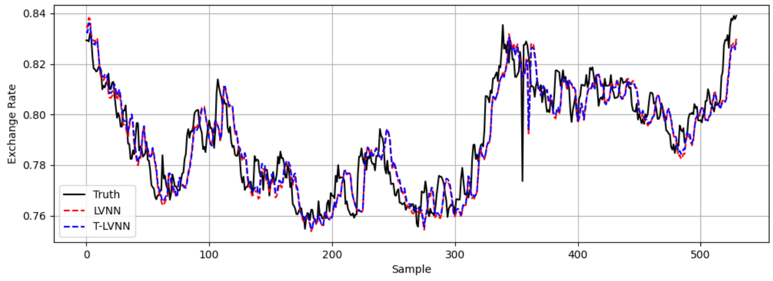
Figure D.1: True temperatures in degrees Kelvin and single-step predictions from LVNN and T-LVNN for a selection station from the Molene test set.



(a) Forecasting horizon 1

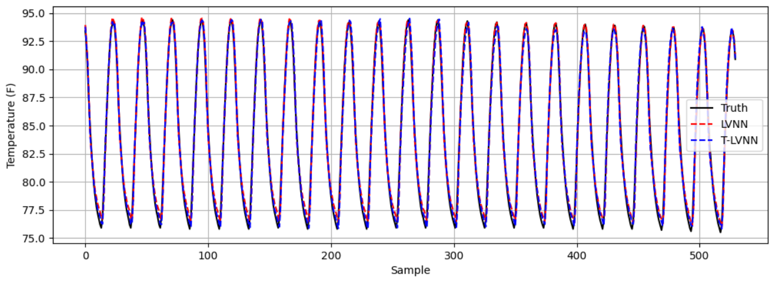


(b) Forecasting horizon 3

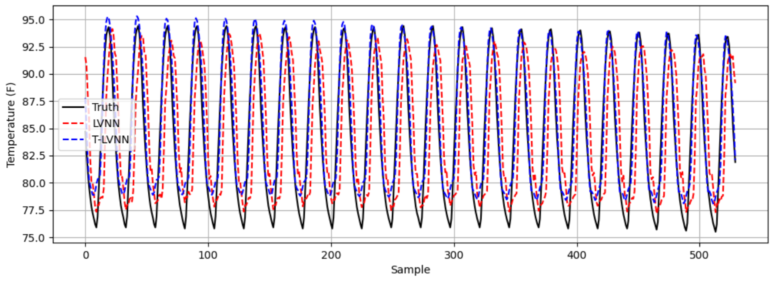


(c) Forecasting horizon 5

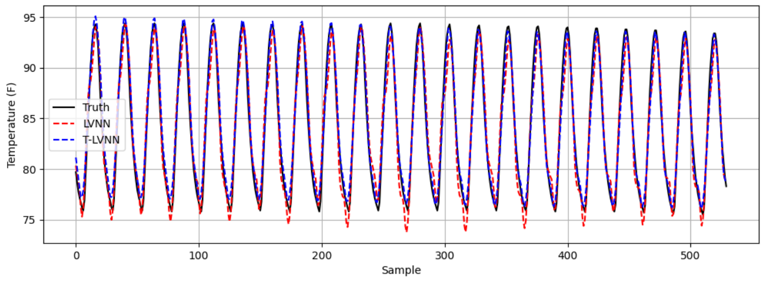
Figure D.2: True exchange rate and single-step predictions from LVNN and T-LVNN for the Swiss franc from the Exchange Rates test set, for 500 test samples.



(a) Forecasting horizon 1

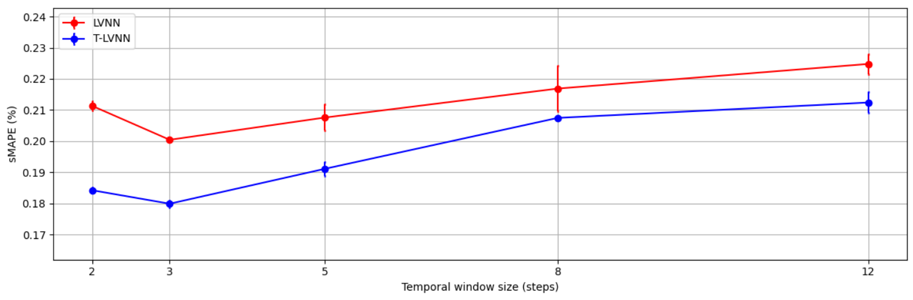


(b) Forecasting horizon 3

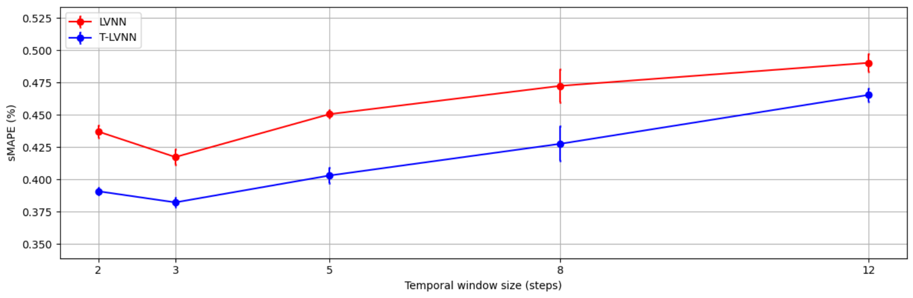


(c) Forecasting horizon 5

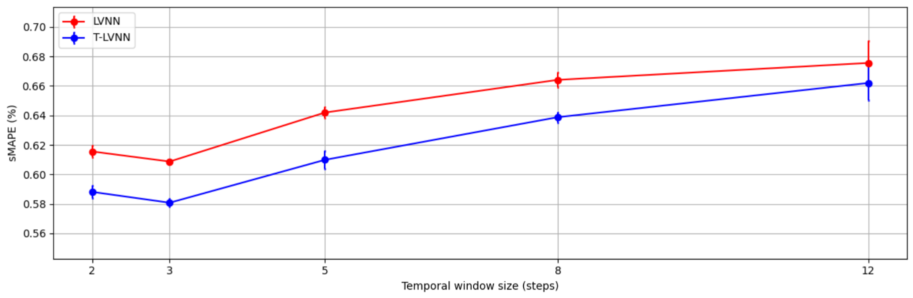
Figure D.3: True temperature in degrees Fahreneit and single-step predictions from LVNN and T-LVNN for a selected station from the NOAA test set, for 500 test samples.



(a) Forecasting horizon 1

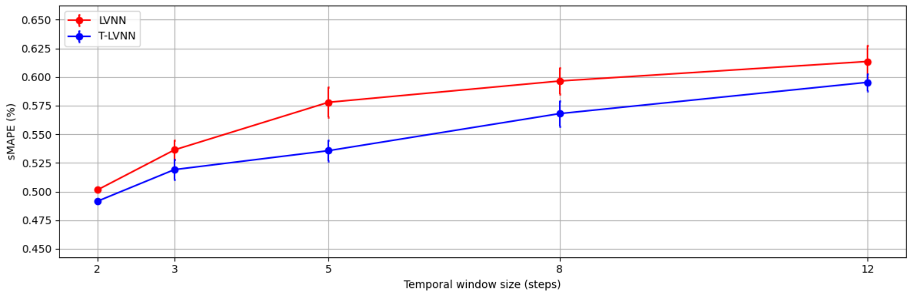


(b) Forecasting horizon 3

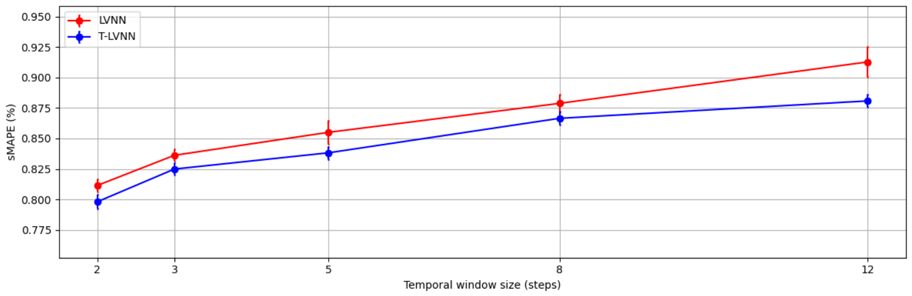


(c) Forecasting horizon 5

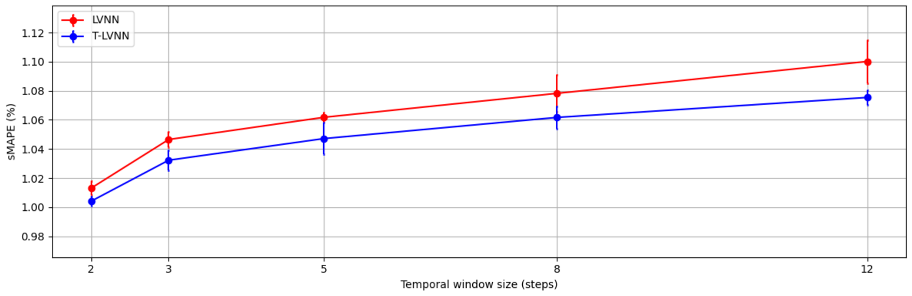
Figure D.4: Performance of LVNN and T-LVNN on the Molene dataset across three forecasting horizons, with different temporal window sizes T.



(a) Forecasting horizon 1

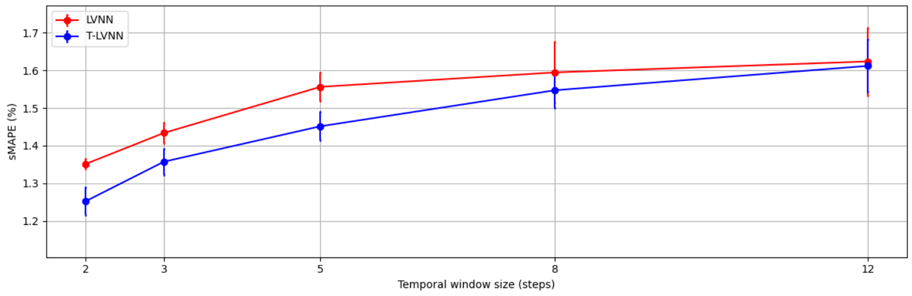


(b) Forecasting horizon 3

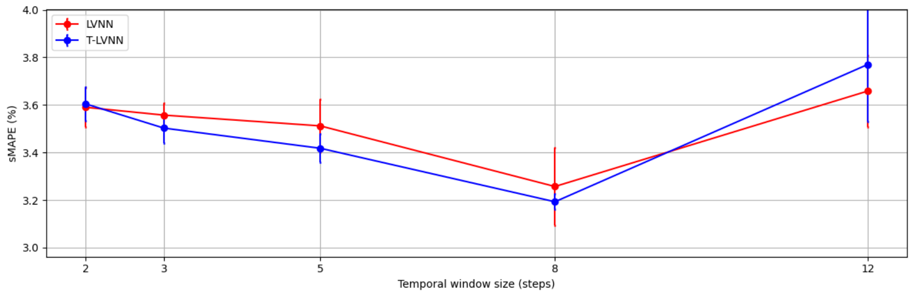


(c) Forecasting horizon 5

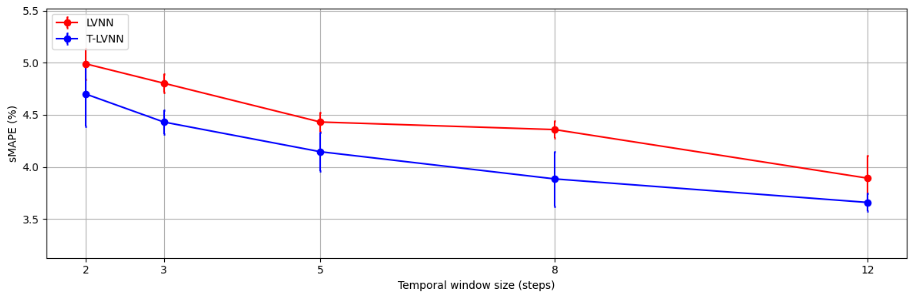
Figure D.5: Performance of LVNN and T-LVNN on the Exchange Rates dataset across three forecasting horizons, with different temporal window sizes T.



(a) Forecasting horizon 1



(b) Forecasting horizon 3



(c) Forecasting horizon 5

Figure D.6: Performance of LVNN and T-LVNN on the NOAA dataset across three forecasting horizons, with different temporal window sizes T.

E

Full Experimental Results

Molene - Single-Step Forecasting

	Forecasting Horizon								
	1			3			5		
Model	MSE	MAE	sMAPE (%)	MSE	MAE	sMAPE (%)	MSE	MAE	sMAPE (%)
TPCA	1.95 ± 0.01	1.04 ± 0.00	0.37 ± 0.00	4.21 ± 0.04	1.55 ± 0.00	0.55 ± 0.00	5.83 ± 0.05	1.86 ± 0.01	0.66 ± 0.00
LSTM	1.23 ± 0.04	0.79 ± 0.01	0.28 ± 0.00	3.30 ± 0.08	1.29 ± 0.02	0.46 ± 0.01	5.04 ± 0.14	1.67 ± 0.02	0.59 ± 0.01
VNN	0.57 ± 0.02	0.55 ± 0.01	0.20 ± 0.00	2.32 ± 0.07	1.15 ± 0.02	0.41 ± 0.01	5.25 ± 0.29	1.78 ± 0.04	0.63 ± 0.01
VNNL	0.62 ± 0.01	0.57 ± 0.01	0.20 ± 0.00	2.26 ± 0.01	1.13 ± 0.00	0.40 ± 0.00	4.56 ± 0.02	1.67 ± 0.01	0.60 ± 0.00
STVNN	*0.54* ± 0.01	*0.54* ± 0.01	*0.19* ± 0.01	*2.13* ± 0.07	*1.08* ± 0.02	*0.39* ± 0.01	**4.29** ± 0.06	**1.61** ± 0.01	**0.58** ± 0.01
LVNN (ours)	0.58 ± 0.01	0.55 ± 0.01	0.20 ± 0.00	2.27 ± 0.02	1.15 ± 0.02	0.41 ± 0.01	4.72 ± 0.1	1.65 ± 0.03	0.60 ± 0.01
T-LVNN (ours)	**0.48** ± 0.00	**0.50** ± 0.00	**0.18** ± 0.00	**1.99** ± 0.03	**1.07** ± 0.01	**0.38** ± 0.00	*4.33* ± 0.04	*1.63* ± 0.01	**0.58** ± 0.01

Exchange Rates - Single-Step Forecasting

	Forecasting Horizon								
	1			3			5		
Model	MSE	MAE	sMAPE (%)	MSE	MAE	sMAPE (%)	MSE	MAE	sMAPE (%)
TPCA	5.01 ± 0.07	1.36 ± 0.01	1.95 ± 0.02	5.23 ± 0.10	1.40 ± 0.01	2.01 ± 0.02	5.65 ± 0.05	1.45 ± 0.01	2.08 ± 0.02
LSTM	1.89 ± 0.16	0.85 ± 0.04	1.22 ± 0.04	2.47 ± 0.08	0.95 ± 0.02	1.33 ± 0.05	2.77 ± 0.20	0.99 ± 0.02	1.39 ± 0.04
VNN	0.54 ± 0.01	0.42 ± 0.00	0.55 ± 0.01	1.20 ± 0.02	0.66 ± 0.01	0.88 ± 0.01	1.71 ± 0.02	0.80 ± 0.01	1.10 ± 0.01
VNNL	0.51 ± 0.00	0.42 ± 0.00	0.55 ± 0.00	1.15 ± 0.04	0.64 ± 0.02	0.87 ± 0.03	1.64 ± 0.01	0.78 ± 0.00	1.08 ± 0.01
STVNN	0.49 ± 0.00	0.39 ± 0.00	0.53 ± 0.01	1.14 ± 0.01	0.64 ± 0.00	0.84 ± 0.01	1.63 ± 0.01	0.77 ± 0.00	1.06 ± 0.01
LVNN (ours)	*0.44* ± 0.00	**0.36** ± 0.00	*0.50* ± 0.00	*1.02* ± 0.00	*0.60* ± 0.00	*0.81* ± 0.01	*1.56* ± 0.01	*0.75* ± 0.00	*1.02* ± 0.01
T-LVNN (ours)	**0.43** ± 0.00	**0.36** ± 0.00	**0.49** ± 0.00	**1.00** ± 0.01	**0.59** ± 0.00	**0.80** ± 0.01	**1.53** ± 0.02	**0.74** ± 0.00	**1.00** ± 0.01

NOAA - Single-Step Forecasting

	Forecasting Horizon								
	1			3			5		
Model	MSE	MAE	sMAPE (%)	MSE	MAE	sMAPE (%)	MSE	MAE	sMAPE (%)
TPCA	5.50 ± 0.13	1.74 ± 0.02	3.03 ± 0.04	13.37 ± 0.21	2.75 ± 0.02	4.78 ± 0.04	19.19 ± 0.22	3.29 ± 0.01	5.78 ± 0.03
LSTM	2.89 ± 0.01	1.19 ± 0.01	2.02 ± 0.04	7.02 ± 0.25	1.86 ± 0.04	**3.15** ± 0.06	8.53 ± 0.55	**1.95** ± 0.06	**3.33** ± 0.11
VNN	1.30 ± 0.10	0.82 ± 0.04	1.42 ± 0.01	**5.60** ± 0.42	**1.75** ± 0.07	*3.16* ± 0.13	8.77 ± 0.48	2.25 ± 0.17	4.06 ± 0.26
VNNL	1.74 ± 0.01	0.99 ± 0.02	1.75 ± 0.02	6.07 ± 0.40	1.90 ± 0.09	3.30 ± 0.11	9.04 ± 0.58	2.45 ± 0.31	4.21 ± 0.22
STVNN	*1.15* ± 0.02	*0.77* ± 0.02	1.36 ± 0.02	6.10 ± 0.44	1.79 ± 0.06	3.22 ± 0.10	8.41 ± 0.24	2.08 ± 0.10	3.82 ± 0.17
LVNN (ours)	1.24 ± 0.04	*0.77* ± 0.01	*1.35* ± 0.01	5.94 ± 0.36	1.79 ± 0.07	3.26 ± 0.11	*8.14* ± 0.43	2.11 ± 0.13	3.89 ± 0.16
T-LVNN (ours)	**1.03** ± 0.07	**0.70** ± 0.02	**1.25** ± 0.03	*5.81* ± 0.32	*1.76* ± 0.03	3.20 ± 0.05	**7.72** ± 0.29	*2.01* ± 0.04	*3.66* ± 0.09

Molene - Sparsification

	Forecasting Horizon					
	1		3		5	
Model	sMAPE (%)	Time (sec)	sMAPE (%)	Time (sec)	sMAPE (%)	Time (sec)
LVNN - Dense	0.20 ± 0.00	0.12 ± 0.02	0.41 ± 0.00	0.24 ± 0.03	0.60 ± 0.01	0.11 ± 0.02
LVNN - Hard (25%)	0.20 ± 0.00	0.09 ± 0.02	0.42 ± 0.00	0.21 ± 0.01	0.61 ± 0.00	0.09 ± 0.01
LVNN - Hard (50%)	0.20 ± 0.00	0.07 ± 0.01	0.42 ± 0.01	0.19 ± 0.02	0.62 ± 0.00	0.07 ± 0.01
LVNN - Hard (75%)	0.21 ± 0.00	0.05 ± 0.01	0.42 ± 0.00	0.17 ± 0.01	0.61 ± 0.01	0.05 ± 0.01
LVNN - Soft (25%)	0.21 ± 0.00	0.09 ± 0.03	0.42 ± 0.00	0.21 ± 0.01	0.62 ± 0.02	0.08 ± 0.01
LVNN - Soft (50%)	0.20 ± 0.00	0.07 ± 0.01	0.42 ± 0.00	0.19 ± 0.00	0.61 ± 0.00	0.07 ± 0.02
LVNN - Soft (75%)	0.20 ± 0.00	0.05 ± 0.01	0.42 ± 0.00	0.17 ± 0.01	0.61 ± 0.01	0.05 ± 0.01
T-LVNN - Dense	0.18 ± 0.00	0.53 ± 0.03	0.38 ± 0.00	1.48 ± 0.1	0.58 ± 0.00	0.51 ± 0.04
T-LVNN - Hard (25%)	0.18 ± 0.00	0.42 ± 0.05	0.39 ± 0.00	1.26 ± 0.06	0.59 ± 0.00	0.43 ± 0.04
T-LVNN - Hard (50%)	0.18 ± 0.00	0.36 ± 0.05	0.39 ± 0.00	0.74 ± 0.06	0.59 ± 0.01	0.37 ± 0.03
T-LVNN - Hard (75%)	0.19 ± 0.00	0.26 ± 0.03	0.40 ± 0.01	0.45 ± 0.02	0.60 ± 0.01	0.26 ± 0.01
T-LVNN - Soft (25%)	0.18 ± 0.00	0.44 ± 0.03	0.40 ± 0.00	1.22 ± 0.16	0.59 ± 0.00	0.45 ± 0.02
T-LVNN - Soft (50%)	0.19 ± 0.00	0.37 ± 0.01	0.40 ± 0.01	0.55 ± 0.00	0.60 ± 0.00	0.38 ± 0.03
T-LVNN - Soft (75%)	0.19 ± 0.00	0.27 ± 0.01	0.40 ± 0.00	0.48 ± 0.05	0.59 ± 0.00	0.28 ± 0.04

Exchange Rates - Sparsification

	Forecasting Horizon					
	1		3		5	
Model	sMAPE (%)	Time (sec)	sMAPE (%)	Time (sec)	sMAPE (%)	Time (sec)
LVNN - Dense	0.50 ± 0.00	0.86 ± 0.04	0.81 ± 0.01	0.87 ± 0.03	1.01 ± 0.01	0.89 ± 0.03
LVNN - Hard (25%)	0.50 ± 0.01	0.79 ± 0.02	0.82 ± 0.00	0.80 ± 0.02	1.01 ± 0.01	0.82 ± 0.02
LVNN - Hard (50%)	0.50 ± 0.00	0.73 ± 0.02	0.81 ± 0.00	0.70 ± 0.02	1.01 ± 0.01	0.77 ± 0.02
LVNN - Hard (75%)	0.50 ± 0.00	0.67 ± 0.01	0.81 ± 0.00	0.69 ± 0.01	1.02 ± 0.01	0.71 ± 0.02
LVNN - Soft (25%)	0.50 ± 0.00	0.79 ± 0.01	0.80 ± 0.01	0.81 ± 0.01	1.01 ± 0.01	0.81 ± 0.01
LVNN - Soft (50%)	0.50 ± 0.00	0.74 ± 0.01	0.82 ± 0.00	0.74 ± 0.01	1.02 ± 0.00	0.76 ± 0.01
LVNN - Soft (75%)	0.50 ± 0.00	0.69 ± 0.01	0.81 ± 0.00	0.70 ± 0.02	1.02 ± 0.01	0.70 ± 0.01
T-LVNN - Dense	0.49 ± 0.00	0.85 ± 0.05	0.80 ± 0.00	1.48 ± 0.1	1.00 ± 0.00	0.87 ± 0.03
T-LVNN - Hard (25%)	0.50 ± 0.00	0.77 ± 0.05	0.80 ± 0.01	0.81 ± 0.01	1.01 ± 0.00	0.80 ± 0.01
T-LVNN - Hard (50%)	0.49 ± 0.00	0.73 ± 0.04	0.80 ± 0.00	0.75 ± 0.02	1.01 ± 0.00	0.76 ± 0.01
T-LVNN - Hard (75%)	0.49 ± 0.00	0.67 ± 0.09	0.79 ± 0.00	0.70 ± 0.02	1.01 ± 0.01	0.70 ± 0.02
T-LVNN - Soft (25%)	0.50 ± 0.00	0.78 ± 0.06	0.80 ± 0.00	0.80 ± 0.01	1.01 ± 0.00	0.80 ± 0.01
T-LVNN - Soft (50%)	0.49 ± 0.00	0.73 ± 0.05	0.80 ± 0.01	0.75 ± 0.02	1.01 ± 0.00	0.75 ± 0.01
T-LVNN - Soft (75%)	0.49 ± 0.00	0.67 ± 0.05	0.79 ± 0.00	0.71 ± 0.02	1.01 ± 0.00	0.69 ± 0.01

NOAA - Sparsification

	Forecasting Horizon					
	1		3		5	
Model	sMAPE (%)	Time (sec)	sMAPE (%)	Time (sec)	sMAPE (%)	Time (sec)
LVNN - Dense	1.35 ± 0.01	9.13 ± 0.54	3.26 ± 0.16	127.47 ± 2.88	3.89 ± 0.21	94.30 ± 3.77
LVNN - Hard (25%)	1.40 ± 0.01	7.47 ± 0.37	3.31 ± 0.07	102.96 ± 0.26	3.92 ± 0.11	85.81 ± 1.33
LVNN - Hard (50%)	1.44 ± 0.00	4.76 ± 0.68	3.39 ± 0.11	69.14 ± 3.36	3.95 ± 0.14	66.11 ± 4.84
LVNN - Hard (75%)	1.42 ± 0.01	2.53 ± 0.60	3.33 ± 0.09	41.13 ± 2.43	3.91 ± 0.12	40.18 ± 3.94
LVNN - Soft (25%)	1.45 ± 0.03	7.63 ± 0.38	3.35 ± 0.07	106.67 ± 2.49	3.93 ± 0.09	86.00 ± 1.37
LVNN - Soft (50%)	1.46 ± 0.03	4.89 ± 0.45	3.38 ± 0.07	76.62 ± 5.42	3.90 ± 0.11	68.85 ± 2.35
LVNN - Soft (75%)	1.39 ± 0.02	2.37 ± 0.46	3.87 ± 0.10	42.23 ± 4.98	3.40 ± 0.08	40.35 ± 3.48
T-LVNN - Dense	1.25 ± 0.03	4.41 ± 0.71	3.20 ± 0.03	181.55 ± 3.02	3.66 ± 0.09	66.88 ± 1.59
T-LVNN - Hard (25%)	1.28 ± 0.03	3.60 ± 0.09	3.25 ± 0.10	157.28 ± 6.65	3.69 ± 0.07	60.59 ± 5.22
T-LVNN - Hard (50%)	1.33 ± 0.03	2.48 ± 0.70	3.28 ± 0.10	109.06 ± 5.23	3.72 ± 0.11	41.60 ± 5.02
T-LVNN - Hard (75%)	1.28 ± 0.02	1.93 ± 0.29	3.29 ± 0.09	68.73 ± 0.18	3.74 ± 0.08	31.63 ± 5.26
T-LVNN - Soft (25%)	1.29 ± 0.03	3.41 ± 0.23	3.29 ± 0.09	143.77 ± 4.74	3.70 ± 0.09	60.10 ± 6.24
T-LVNN - Soft (50%)	1.32 ± 0.03	2.35 ± 0.64	3.27 ± 0.07	108.71 ± 0.81	3.67 ± 0.13	38.72 ± 3.47
T-LVNN - Soft (75%)	1.31 ± 0.02	1.54 ± 0.53	3.23 ± 0.07	70.79 ± 4.63	3.73 ± 0.09	23.11 ± 7.24