UNIVERSITÀ DEGLI STUDI DI MACERATA

CORSO DI DOTTORATO DI RICERCA IN
QUANTITATIVE METHODS FOR POLICY EVALUATION

CICLO XXXVIII

The integration of Spatiotemporal Neural Networks in Portfolio Optimization

SUPERVISORI DI TESI

Chiar.mo Prof. Andrea Bucci
Chiar.mo Prof. Luca Riccetti

DOTTORANDO

Dott.ssa Diana Komis

COORDINATORE

Chiar.ma Prof.ssa Margherita
Scoppola

ANNO 2026

SDA
scuola di dottorato

Preface

I want to start this thesis by recognizing and appreciating how God works in my life, as he provided me with the road, the connections, the knowledge, the strength, the peace and all resources needed at the right time.

I then give all the appreciation to myself, for being disciplined, committed and not allow anything or anyone to stop me.

I am very grateful for my supervisors; Prof.Andrea Bucci, you have played the biggest role in the success of each detail of this thesis, you have not just been a supervisor, but the best leader and the role model I needed in this journey thank you. Prof.Luca Riccetti, I remember your presence in my master's defense, and my PhD interview since then you have given me support in both the professional level and the personal level, understanding my circumstances and standing up for me, thank you.

I dedicate my thesis to my mother, Nadejda and each of my brothers: Essam, Laith, Haytham and Michail. Even though we can spend years without meeting, I am proud of each one of you, and I will always be there for you all, and I know you will do the same for me when needed. Thank you.

Talking about family, I want to thank you Natasha for being there for my mother at the time that I could not do that due to distance you are the real example that family is not just blood.

I am extremely grateful for my only real friend Hadi, for believing in me at the time that I did not believe in myself, for being my support, and for just being there, thank you.

For people I met along the way, Walid and your family, Raneem and Sara, Thank you. Father Ayman, I hope you get well soon and thank you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Let us be clear on the fact that if asset movements at a specific timing were known, we would all be rich, but it is real life, where nothing is actually known. This is why we need portfolio optimization, a fundamental field in finance that requires constant decision-making of the best diverse set of assets to achieve the best possible trade-off between expected return and risk. Making the best decision is the root of modern investment management, and it enables the construction of portfolios that maximize returns for a given level of risk or minimize risk for a specific return target (Lucchetti et al., 2024). Portfolio optimization plays a crucial role in portfolio management by putting together risk-balancing frameworks; for example, the Risk Balancing Frontier identifies the optimal portfolio boundaries by combining tracking error volatility and Value-at-Risk constraints (Lucchetti et al., 2024). The need for quantitative methods designed specifically for financial data and markets led to the integration of financial econometrics in the early 20th century, including sophisticated forecasting methods for covariance matrices of asset returns that directly improved the accuracy and reliability of portfolio optimization by capturing the evolving dependence structures among assets (Bucci et al., 2022; Riccetti, 2013).

The main objective of portfolio management is to balance the trade-off of the twin pillars, risk and return; therefore, their accurate forecasting is a

must. Accurate estimates of expected risk and return guide investment decisions. Reliable risk forecasts—especially covariance matrices—play a central role in portfolio optimization because they capture the dynamic relationships among assets that determine diversification benefits (Bucci et al., 2022). The ability to model and predict the time-varying volatilities and correlations of asset returns directly influences portfolio performance, risk-adjusted returns, and overall investment strategies. Financial econometrics advanced techniques, including copula-GARCH models, have been developed to model complex dependence structures and tail dependencies that characterize financial returns, thereby improving both asset allocation and risk management (Riccetti, 2013). Furthermore, recent advances in geometric deep learning and graph-based neural networks tackle the high-dimensional challenges of covariance forecasting and improve predictive accuracy, which is vital for building large-scale portfolios (Bucci et al., 2022, 2024).

We cannot close our eyes to the profound influence of the arrival of deep learning and neural network methodologies on various fields, including financial econometrics and asset modeling. Traditional statistical models like GARCH, VAR and factor models cannot focus on temporal dependencies and cross-sectional correlations at the same time, which limits their ability to capture the complex, dynamic interactions that characterize financial data across time and space (Engle, 2002). Neural networks now play a key role in modeling high-dimensional and complex datasets. They offer greater flexibility and achieve stronger predictive performance (Zhang, 2018; Yu et al., 2019). While neural networks have been used in finance for years, their use has expanded rapidly as they can learn non-linear patterns without relying on explicit model assumptions, outperforming traditional methods in forecasting asset returns and volatility (Yang and Bristol, 1999). Also, with the further growth of deep learning, models like Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs) have more effectively captured temporal dependencies and multivariate relationships (Fischer and Krauss, 2018).

We can clearly say that the use of neural networks did not stop after capturing temporal dependencies; it has expanded to capture spatial dependencies like interactions among financial assets, markets, or regions (Kumar et al., 2024). For example, Yu et al. (2019) have proposed a graph-based neural network framework that integrates spatial information through graph convolutions with temporal sequence modeling for forecasting high-frequency asset prices. Similarly, Zhang (2021) shows how spatiotemporal graph neural networks (STGNNs) effectively model the interconnectedness of global equity markets by capturing the propagation of shocks across regions over time.

It is obvious that it is not easy to forecast the tools used to measure the riskiness of the assets even with the availability of all these traditional and advanced methodologies. Forecasting realized covariance matrices as a measurement of financial risk (Andersen et al., 2003) is still facing several challenges due to their dynamic nature and the need for positive semi-definiteness. The biggest one is the fact that as the number of assets rises, these matrices grow quadratically in size, making estimation computationally intensive, which increases noise and estimation error. This is known as the "curse of dimensionality" (Callot et al., 2017). Other challenges include the fact that capturing the dynamic, nonlinear, and long-memory characteristics of volatility and covariance is difficult (Bucci et al., 2024).

Forecasting returns is even more challenging; indeed, according to the core principles of the Efficient Market Hypothesis (EMH), the presence of any return predictability would indicate a degree of market inefficiency as prices follow a random walk (Bollerslev, 2019); Market microstructure effects, investor's behavior, and economic regimes create complex nonlinear dynamics and lead to biased return forecasts (Marconi, 2025). Moreover, as much as important accurate return predictions are crucial for portfolio management, stable performance is challenging due to noise, liquidity, and asset-specific dynamics (Laurent et al., 2010).

Starting from this point and from the belief that the spatiotemporal net-

work's ability to model complex interactions across multiple scales positions them as essential tools for analyzing systemic risk, asset return correlations, and volatility dynamics (Silva et al., 2020); while this might not be enough to address 100% of all the challenges, it is definitely a step forward into achieving more accurate forecasts compared to other models. Furthermore, the mixture of risk and return predictions from these models with concepts from graph theory can lead to an advantage in the world of portfolio analysis. This thesis aims to examine the ability of spatiotemporal neural networks combined with the graph theory to forecast volatility and returns, and then use the forecasts for portfolio optimization.

We begin our analysis in Chapter 2, where our focus is addressing challenges related to the forecasting of realized covariance matrices, a crucial component of portfolio optimization and risk management in financial markets, due to their dynamic nature and the need for positive semi-definiteness. This is done by building upon Adaptive Graph Convolutional Recurrent Network (AGCRN), a spatiotemporal model that was originally developed for traffic forecasting (Bai et al., 2020), to identify asset-specific patterns and determine inter-asset dependencies in a dynamic framework. AGCRN introduces a *Node Adaptive Parameter Learning* (NAPL) module to study node-specific patterns and a *Data Adaptive Graph Generation* (DAGG) module to generate the graph structure. This approach directly tackles the limitations identified earlier—specifically the inability of traditional models to handle high-dimensional matrices, capture nonlinear volatility spillovers, and adapt to evolving market structures.

The chapter demonstrates that AGCRN achieves superior predictive accuracy compared to traditional and machine learning models across multiple distance metrics, including Frobenius, Euclidean, and Procrustes distances, with the statistical Giacomini-White test and Model Confidence Set analysis after making sure that all matrices are positive definite by performing an eigen-decomposition of each predicted matrix.

To further assure the effectiveness of the volatility predictions generated in Chapter 2 using adaptive graph convolutional recurrent networks and to align further with the main purpose of our thesis, in Chapter 3 we benefit from one of the main concepts from the graph theory, Maximum Spanning Tree (Maxst) which uses Prim's algorithm (Prim, 1957), which filters the asset network to retain the connections that contribute most to risk reduction (West, 2001) after transferring each predicted realized covariance matrix into a graph.

This novel portfolio optimization framework successfully generates a portfolio profile with the highest Sharpe Ratio, Sortino Ratio and Information Ratio among all competing strategies, including traditional Markowitz and GMV portfolios, with a critical trade-off as they exhibit a higher tail risk (VaR and ES) than the most conservative models, indicating that AGCRN-MaxST's aggressive return-seeking nature exposes it to greater potential losses during extreme market events.

In Chapter 4, we shift the focus to the use of Spatiotemporal Covariance Neural Networks (STVNN), a model that was designed for multivariate time series forecasting (Cavallo et al., 2024). STVNN operates through a novel Spatio-Temporal Covariance Filter (STVF), which performs convolutions over both the evolving covariance matrices and temporal windows of observations by translating time series covariance matrices into graphs to capture how spatial dependencies evolve. Simultaneously, it studies the temporal dynamics within each variable, modeling these two dimensions jointly. Specifically, STVNN operates in an online learning framework, allowing it to adapt to non-stationarity by continuously updating its parameters as new data arrives (Cavallo et al., 2024).

This chapter shifts the focus from the predictions of volatility and focuses instead on the forecasting of returns. The STVNN approach helps overcome the EMH hurdle by exploiting transient inefficiencies that arise from market microstructure effects and short-lived structural breaks. Rather than promising unconditional arbitrage, STVNN focuses on conditional predictability, en-

hancing forecasting performance in real-world financial markets.

The chapter demonstrates that STVNN achieves superior predictive accuracy compared to traditional and machine learning models across multiple distance metrics, including Euclidean distances and Mean Squared Error (MSE), with the statistical Giacomini-White test and Model Confidence Set analysis.

In order to further understand the accuracy of the STVNN predictions, we have performed portfolio optimization using Markowitz with predicted returns as expected returns and the realized covariances as the covariance matrix. The STVNN portfolio outperforms other models in maximizing returns, demonstrating strong metrics like Sharpe, Sortino, Information, Treynor Ratios, and Jensen's Alpha. It generates impressive returns relative to total risk and systematic market exposure, indicating effective market timing. However, it has high volatility, significant drawdowns during market downturns, frequent rebalancing, and concentrated holdings. It also faces higher tail risk under extreme market conditions.

In summary, this thesis systematically demonstrates that by integrating graph theory with spatiotemporal neural networks, we can significantly overcome the high-dimensional challenges in forecasting both volatility and returns, thereby charting a new and more accurate path for modern portfolio optimization.

Chapter 2

A spatiotemporal RNN for predicting Realized Covariance

2.1 Introduction

Financial investments have been a source of income for specialized operators and private individuals for more than a century. Historically, a large percentage of investors, ranging between 40% and 60%, prefer low-risk investment products rather than seeking high returns (Lewis, 2024). The tool used in measuring the riskiness of securities is volatility, which, in the presence of multiple securities, becomes a matrix of volatility and co-volatilities. The major problem with this risk measure is that it is not observable.

Among the methods implemented through the years to estimate the unobservable volatility of multiple assets, in the last couple of decades the overall scene has been stolen by the realized volatility. In particular, a realized covariance matrix (RCOV) of asset returns is a statistical measure used to quantify the riskiness of several financial asset returns over a particular period using high-frequency price data (Andersen et al., 2003; Barndorff-Nielsen and Shephard, 2002).

As shown in Andersen et al. (2003), this nonparametric measure accurately estimates latent volatility, which is crucial for effective risk management and

asset allocation (Ledoit and Wolf, 2004). Furthermore, realized covariance matrices have improved the ability to predict volatility and covariances, serving as a critical input in many financial models. Better investment strategies are made by better forecasting abilities (Barndorff-Nielsen and Shephard, 2002). The dynamic character of these matrices, which capture the temporal evolution of asset correlations, has made effective asset allocation easier, particularly under volatile market conditions (Corsi, 2009). In addition, the complexity of the temporal dependencies that characterize financial data made it more interesting to utilize models to predict it.

The task of predicting realized covariance matrices has been complicated due to a variety of challenges. Hundreds of assets have frequently been included in financial portfolios, leading to high-dimensional covariance matrices, as noted by Yu et al. (2018). As the number of assets increases, the covariance matrix becomes large, making traditional multivariate volatility models intractable. Classic ARCH and GARCH frameworks (Engle, 2002) often fall under the "curse of dimensionality", yielding unreliable estimates when modeling all pairwise interactions (Callot et al., 2017). Even with high-frequency realized measures, managing high-dimensional covariance matrix is not straightforward (Hautsch et al., 2012). Researchers have explored dimensionality reduction via factor models or shrinkage (Asai and McAleer, 2015; Gribisch et al., 2020), yet preserving all important information while keeping models parsimonious is an open challenge.

Furthermore, the relationship between asset volatilities and correlations can be highly non-linear. Traditional econometric models (*e.g.*, GARCH and its multivariate extensions) capture some stylized facts like clustering, but they struggle with more complex nonlinear interdependencies between multiple assets (Kumar et al., 2024). For example, volatility spillovers and leverage effects may induce nonlinear patterns that linear models or Gaussian assumptions cannot fully capture. The challenge is to model such nonlinear dynamics so that, for instance, co-movements during extreme events or asymmetric re-

sponses are predicted correctly, since the complexity of the problem has not been efficiently captured by linear models, as pointed out by Kipf and Welling (2017) in their research on graph-based methods.

In addition to factor models and shrinkage techniques, copula-based approaches have been widely explored to address the challenge of modeling non-linear dependencies in financial returns. Copulas allow for the separation of marginal distributions from the dependence structure, enabling the modeling of complex, non-linear, and tail dependencies between assets, features that traditional correlation-based models often miss (Di Clemente and Romano, 2021). The practical application of copulas, especially in high-dimensional settings, is hindered by significant estimation and computational challenges. For example, the increase in the number of assets causes the standard copula models to become intractable; on the other side, advanced constructions like vine copulas require careful specification and can be sensitive to model selection and overfitting (Riccetti, 2012). Thus, the use of the theoretically appealing framework for modeling non-linear dependencies offered by copulas in large-scale covariance prediction remains limited by these practical constraints.

Another challenge is that assets tend to exhibit spatial dependencies between sectors or markets due to their economic linkages, geographical proximity, and network-based connections. In fact, assets can be represented as nodes in a network (or a "spatial" graph) with edges representing the correlations linkages between them. These linkages are dynamic with evolving fundamentals or contagion effects. A static correlation matrix or a prefixed network fails to reflect the shifts during structural breaks or crises. Moreover, figuring out which asset influences the other (the network structure) is itself challenging. There may be sectoral clusters, leading-lagging relationships, or other latent structures. Capturing the "interdependence in volatility (spillover effects) and correlation" across assets in a flexible yet interpretable way is difficult (Zhang et al., 2025). Although Wang et al. (2020) address this issue by emphasizing the significance of capturing spatial dependencies in financial networks, the

spatial interdependencies incorporation remains mostly unexplored.

Additionally, the time-evolving relationships between assets require models that can adjust to dynamic graph structures. The subject has been addressed by Bai et al. (2020), who have demonstrated the impact of developing graph structures on model performance. To obtain more accurate predictions of the realized covariance matrices, it is imperative to handle these challenges. Predicting realized covariance matrices has been approached through traditional and modern methodologies. Traditional approaches are mainly based on the use of a vector autoregressive (VAR) model in combination with a parameterization (Bucci et al., 2022). VAR models do capture the linear interdependencies within multiple time series and are straightforward to estimate (Sims, 1980; Zhang et al., 2025). However, they are constrained by their linear nature and tend to be unable to capture the non-linear relationships of financial data. Despite being commonly utilised, these approaches often fall short in dealing with complex temporal dependencies and high-dimensionality inherent in realized covariance matrices, leading to less precise predictions. Moreover, the use of a parametrisation may lead to results that are hardly interpretable and the choice of a proper parametrisation may be highly sample-related (Bucci et al., 2022).

In the last few years, machine learning approaches have received attention for predicting realized covariance matrices. Machine learning models, such as neural networks, have been used to analyse non-linear dynamics and temporal dependencies in financial time series (Chen et al., 2018). According to Bucci (2020), neural networks have been able to capture linear and nonlinear behaviors in volatility without the need to have a full understanding of the data-generating process. Recent studies have begun to apply graph neural networks to financial volatility modeling with promising results. For example, Zhang et al. (2025) augment a classic Heterogeneous Autoregressive (HAR, Corsi, 2009) volatility model with graph-based neighbor information (constructed from industry sectors or learned graphs) and report "statistically and eco-

nomically significant improvements" in forecasting large realized covariance matrices. Graph-based approaches, such as Graph Convolutional Networks (GCNs, Kipf and Welling, 2017), have leveraged the inherent structure of financial markets by treating assets as nodes in a graph, with edges representing correlations or other relationships (Kipf and Welling, 2017). The dependencies among assets have been identified with promising results using these methods also in the paper of Brini and Toscano (2024). However, the availability of spatial information is still an important challenge for these methods, often limitating their capacity to completely capture the inter-dependencies within assets.

This highlights a fundamental trade-off in existing deep learning models. Architectures that excel at capturing complex temporal dynamics, such as Long Short-Term Memory (LSTM) networks, often treat the asset series as a simple collection, overlooking the crucial network of interdependencies between them. Conversely, models designed explicitly for structured data, like standard Graph Convolutional Networks (GCNs), may not adequately capture long-range temporal patterns. The challenge is further magnified by the dynamic nature of financial markets, where these inter-asset relationships constantly evolve (Gao et al., 2020). Therefore, a significant gap remains for a unified framework that can simultaneously model complex temporal patterns, dynamic spatial interdependencies, and the non-linearities inherent in high-dimensional financial data, all while remaining computationally feasible.

In this context, PyTorch Geometric Temporal and models like the Adaptive Graph Convolutional Recurrent Network (AGCRN) give substantial benefits for predicting financial volatility and realized covariance matrices as they are specifically designed for processing a spatiotemporal signal, offering a robust framework that integrates both spatial and temporal aspects in neural machine learning models (Rozemberczki et al., 2021). The AGCRN excels at capturing deep spatial and temporal correlations via its adaptive modules, which automatically infer inter-dependencies among data points without needing pre-

defined graphs (Bai et al., 2020). This comprehensive strategy could not only enhance the forecasting accuracy of RCOV matrices, but could also increase the model's resilience and flexibility across varied market circumstances. By integrating the benefits of graph-based and recurrent neural network models, the AGCRN offers a substantial improvement in the field, overcoming some of the main drawbacks of standard deep learning techniques (Bai et al., 2020).

We propose the use of the AGCRN method, originally developed for traffic forecasting, for realized covariance forecasting. The AGCRN combines Graph Convolutional Networks (GCNs) with Recurrent Neural Networks (RNNs) to deal with the complex nature of spatiotemporal data. This integration allows a comprehensive understanding of the interconnections among financial assets (spatial correlations) and their changes over time (temporal correlations), which are important for accurate RCOV matrix predictions. Prior applications of AGCRN in traffic forecasting have proved its efficacy in tackling comparable spatiotemporal issues, showing high potential for success in financial settings as well (Bai et al., 2020; Kipf and Welling, 2017; Rozemberczki et al., 2021). In addition, Jonnalagadda and Hashemi (2023) use an AGCRN to forecast the El Niño-Southern Oscillation (ENSO) Indicator based on a graph of ocean grid cells. Their model substantially outperforms both classical statistical models and state-of-the-art dynamical climate models for long-lead (12-18 month) ENSO predictions. Similarly, in air quality prediction (a spatiotemporal problem with monitoring stations as graph nodes), adaptive graph RNN models have shown excellent performance, capturing how weather and pollution propagate through space over time (Chen et al., 2024).

AGCRN is explicitly designed to deal with a large number of related time series (nodes) without blowing up the parameter count. It introduces a *Node Adaptive Parameter Learning* (NAPL) module that factorizes the weight parameters for graph convolution, effectively reducing dimensionality (Bai et al., 2020). Instead of learning a separate set of parameters for each asset pair (which would be enormous), NAPL uses a low-dimensional embedding for each

node and a shared "weight pool". These are combined to generate node-specific weight matrices efficiently. This factorization maintains a unique model for each asset's interactions but with far fewer total parameters, alleviating overfitting even when the number of assets (nodes, n) is large. In essence, AGCRN provides a parsimonious representation that can scale to high-dimensional covariance matrices, unlike classical models that would require estimating $\mathcal{O}(n^2)$ parameters or imposing heavy restrictions. Another key feature of the AGCRN is the *Data Adaptive Graph Generation* (DAGG) module, which learns to connectivity among nodes (assets) directly from data, rather than relying on a fixed pre-defined adjacency matrix. Each asset is a learnable embedding, and during training the model infers the strength of link between every pair of assets using those embeddings (through a learned similarity function). This means the correlation network is not fixed, it is adaptive. AGCRN effectively discovers which assets have strongly connected volatilities or correlations, capturing the latent volatility spillover network (Zhang et al., 2025). Because the graph structure is optimized for the forecasting task, it can reflect evolving relationships: if certain connections become more important during a crisis, the learning process will adjust the edge weights accordingly. In AGCRN, the graph is a dynamic representation (learned in training) of the interdependence among assets. This directly addresses the challenge of dynamic spatial correlations, *i.e.*, the model does not assume any two assets' relation is constant. This has also the advantage that AGCRN avoids errors from a misspecified adjacency matrix (*e.g.*, missing an important linkage or including a spurious one). While standard financial GNNs construct a graph of assets, our approach formulates the problem on a graph of asset-pairs. Each node in our graph represents a unique (co)variance element. This allows the model to learn not just how assets influence each other, but how asset relationships influence other relationships. For instance, the model can learn that a rising correlation between Apple and Microsoft might predict a subsequent rise in correlation between Google and Amazon, a dynamic that a standard asset-level graph

cannot directly capture.

In this chapter, we apply the AGCRN method to forecast the daily realized covariance matrices of 50 assets composing the S&P 500 index. In a rolling window exercise, we show that the proposed methodology has higher predictive accuracy with respect to competing models, including also deep learning methods applied to the vectorization of the covariance matrices.

This chapter is organized as follows. In Section 2.2, we introduce the methodology, while Section 2.3 presents the results of the AGCRN's application. Section 2.4 concludes.

2.2 Methodology

2.2.1 Realized covariance matrix

The realized covariance matrix (RCOV) provides a non-parametric estimate of the integrated covariance matrix over a specified time period, for further details see Jin et al. (2021).

Consider an n-dimensional vector of log-prices $p(\tau)$, where $\tau \in \mathbb{R}^+$ represents continuous time. Assuming that $p(\tau)$ follows a semi-martingale process, it may be stated that:

$$dp(\tau) = \mu(\tau)d\tau + \mathbf{A}(\tau)dW(\tau).$$

Here, $\mu(\tau)$ represents the instantaneous drift, $W(\tau)$ is a normal n-variate Wiener process, and $\mathbf{A}(\tau)$ is a $n \times n$ matrix reflecting instantaneous volatility. The spot covariance matrix $\Theta(\tau) = \mathbf{A}(\tau)\mathbf{A}(\tau)^\top$ reflects the immediate covariance of asset returns (Protter, 2005; Barndorff-Nielsen and Shephard, 2002).

The integrated covariance matrix over a trading day t is given by:

$$\mathbf{IC}_t = \int_{t-1}^{t} \Theta(\tau)\, d\tau.$$

The realized covariance matrix $\mathbf{RCOV}_t$ serves as an ex-post estimate of $\mathbf{IC}_t$ obtained from high-frequency return data. The simplest variant of RCOV

is computed using $m+1$ evenly spaced intraday log-prices. The j-th intraday return vector on day t is defined as:

$$\mathbf{r}_{j,t} = p\left(t-1+\frac{j}{m}\right) - p\left(t-1+\frac{j-1}{m}\right), \quad j = 1, 2, \ldots, m.$$

It follows that a measure of the covariance matrix of asset returns in the period t can be obtained simply by aggregating the outer products of these intraday return vectors:

$$\mathbf{RCOV}_t = \sum_{j=1}^{m} \mathbf{r}_{j,t}\mathbf{r}_{j,t}^{\top},$$

which has dimension $n \times n$. In this chapter, we focus on the prediction of the $\tilde{n} = n(n+1)/2$ unique elements of $\mathbf{RCOV}_t$. We also assume access to node-level feature vectors $\mathbf{x}_{i,t}$ for each asset i at time t, capturing relevant financial indicators (*e.g.*, lagged returns, trading volumes, etc.)[1]. Our goal is to predict $\mathbf{RCOV}_t$ using a rolling window forecasting approach to produce one-step-ahead predictions.

To exploit the cross-asset relationships, we represent the system of assets as a graph $G = (V, E)$. Each asset corresponds to a node $v_i \in V$ (for $i = 1, \ldots, n$), and edges signify pairwise dependencies between assets. The realized covariance matrix $\mathbf{RCOV}_t$ at time t can be interpreted as a weighted complete graph among the n nodes, where the weight $(\mathbf{RCOV}_t)_{ij}$ on edge (i, j) reflects the strength of co-movement (covariance) between asset i and j during day t. Rather than fixing a graph a priori, we allow the model to learn the graph structure adaptively from data. This is motivated by prior findings that pre-specified graphs (*e.g.*, based on sector or correlation thresholds) may be suboptimal or introduce biases (Bai et al., 2020). Instead, our approach uses the time series data itself to infer an evolving dependency structure, which is especially crucial in finance where correlations can change over time.

[1] In traditional factor models, covariance matrices are decomposed into two components: $\mathbf{\Sigma}_t = \mathbf{B}_t\mathbf{\Sigma}_{f,t}\mathbf{B}_t^{\top} + \mathbf{\Sigma}_{\varepsilon,t}$, where $\mathbf{\Sigma}_{f,t}$ represents common factor covariance (first factor) and $\mathbf{\Sigma}_{\varepsilon,t}$ captures idiosyncratic residuals (second factor). While our node features $\mathbf{x}_{i,t}$ may implicitly encode aspects of both, our approach avoids explicit factor identification, instead letting the model learn latent dependencies directly from the data. For classical factor frameworks, see Fama and French (1993); Brito et al. (2023).

At each time step, the node feature matrix $\mathbf{X}_t$ (lagged returns, volumes, technical indicators, etc.) serves as the input node attributes. These features ensure that both idiosyncratic properties, like an asset's own volatility or trading activity, and some common market signals are presented to the model. The graph representation is then used to propagate information across assets. In essence, the graph provides a structural inductive bias: it enables the model to aggregate information from related assets when forecasting each asset's volatility and covariance. In our framework, the graph itself is not static but learned through an Adaptive Graph Convolutional Recurrent Network (AGCRN), which we detail next.

2.2.2 Adaptive Graph Convolutional Recurrent Network (AGCRN)

Our model builds upon the AGCRN architecture (Bai et al., 2020), originally proposed for traffic sensor networks, and adapts it to financial covariance forecasting. The architecture consists of three key components: (i) Node Adaptive Parameter Learning (NAPL) for node-specific filter generation, (ii) Data Adaptive Graph Generation (DAGG) for inferring the latent graph structure, and (iii) a Graph Convolutional Recurrent module (built with Gated Recurrent Units) for temporal modeling. We describe each in turn, including the design choices and mathematical formulation. An overview is that each asset (node) is endowed with a trainable embedding that both parameterizes its unique model weights and determines its connections to other nodes. These embeddings thus tie together NAPL and DAGG, providing a unified representation for each asset.

Financial assets often exhibit distinct temporal patterns. To capture such heterogeneity, we employ Node-Adaptive Parameter Learning, which gives each node its own set of learnable filter parameters while still sharing statistical strength across nodes. Rather than assigning completely independent parameters to each asset (which would be infeasible for large n), NAPL fac-

torizes the GCN weight matrices into a common basis plus node-specific coefficients. In this way, the model can learn a set of basis filters applicable to all assets, and the unique embedding of each asset will produce a customized linear combination of these basis filters.

Consider a single graph convolution layer in the model, which transforms input features of dimension F_{in} to output features of dimension F_{out} for each node. In a standard Graph Convolutional Network (GCN), this is done through a weight matrix $\mathbf{W} \in \mathbb{R}^{F_{\text{in}} \times F_{\text{out}}}$ that is shared across all nodes, applied after message-passing with the adjacency matrix. In our NAPL-enhanced GCN, we introduce: i) a trainable node embedding matrix $\mathbf{E} \in \mathbb{R}^{n \times d}$, where d is a chosen embedding dimension, with $d \ll n$ for parameter efficiency; ii) a weight pool (and similarly, a bias pool) that serves as the shared bank of basis parameters. Specifically, let $\Theta^S \in \mathbb{R}^{d \times (F_{\text{in}} \cdot F_{\text{out}})}$ denote the weight pool, whose rows can be viewed as a basis vector (when reshaped to $F_{\text{in}} \times F_{\text{out}}$). For a given node i, we generate its specific weight matrix $\mathbf{W}_i \in \mathbb{R}^{F_{\text{in}} \times F_{\text{out}}}$ by combining these basis filters according to the node's embedding $\mathbf{e}_i \in \mathbb{R}^{1 \times d}$ (the ith row of $\mathbf{E}$). Formally, if $\Theta^S = [\text{vec}(\mathbf{W}_1), \text{vec}(\mathbf{W}_2), \ldots, \text{vec}(\mathbf{W}_d)]$, where $\mathbf{W}_k$ is the k-th basis weight matrix, then the node-specific weight is:

$$\text{vec}(\mathbf{W}_i) = \mathbf{e}_i \Theta^S = \sum_{k=1}^{d} e_{ik} \text{vec}(\mathbf{W}_k^*)$$

where $\text{vec}(\cdot)$ denotes vectorization. Equivalently, we have that $\mathbf{W}_i = \sum_{k=1}^{d} e_{ik} \mathbf{W}_k^*$. Thus, $\mathbf{e}_i$ acts as a set of coefficients that extracts a customized filter from the shared weight pool. The same procedure is applied to generate a node-specific bias $b_i \in \mathbb{R}^{F\text{out}}$ using a bias pool $b^S \in \mathbb{R}^{d \times F_{\text{out}}}$. By this design, all nodes share the underlying basis filters, but each node emphasizes different aspects of them through $\mathbf{e}_i$, effectively learning node-specific patterns from a set of candidate patterns. This dramatically reduces the number of free parameters (especially when n is large) and provides a smooth parameter space for learning nodes with similar embeddings will have similar filters, which is a form of regularization.

In a forward pass of the GCN layer, the computation for node i uses its custom weights, $\mathbf{W}_i$. Let $\mathbf{h}_j^{(l)}$ be the input feature vector of node j in layer l. The output of the NAPL-GCN layer for node i is:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j=1}^{n} A_{ij}\mathbf{h}_j^{(l)}\mathbf{W}_i + b_i\right), \quad i = 1, \ldots, n$$

where $\sigma(\cdot)$ is an activation function (*e.g.*, ReLU[2]) and A_{ij} is the (i, j) element of the adaptive adjacency matrix (discussed below). Notice that $\mathbf{W}_i$ (and b_i) is fixed for the target node i while summing over its neighbors j, this means node i linearly combines incoming feature-information using its own filter. In matrix form, defining $\mathbf{H}^{(l)} \in \mathbb{R}^{n \times F_{\text{in}}}$ as the matrix of all node features at layer l (the i-th row is $\mathbf{h}_i^{(l)}$), and $\mathbf{W} = \mathbf{E}, \Theta^S$ as the assembled weight matrices for all nodes (an $n \times (F_{\text{in}}F_{\text{out}})$ matrix that can be viewed as stacking $\mathbf{W}_1, \ldots, \mathbf{W}_n$), the layer-wise computation can be written compactly as:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{A}}\mathbf{H}^{(l)}\overline{W} + \mathbf{b}\right),$$

where $\tilde{\mathbf{A}}$ is the normalized adjacency (see below), $\overline{\mathbf{W}}$ denotes the operation of applying each $\mathbf{W}_i$ to the i-th row of $\mathbf{H}^{(l)}$, and $\mathbf{b}$ is the matrix of biases (each row i equals b_i). In practice, we implement this NAPL layer efficiently by distributing the node-specific weight matrices across the batch dimension. NAPL thus endows the GCN with the flexibility to treat each asset differently, which is crucial in financial data modeling.

Instead of requiring a pre-defined adjacency matrix $\mathbf{A}$ to perform graph convolutions, we learn the graph structure directly from data in an end-to-end fashion. Financial asset connections (correlations) are not static; they vary with market conditions. DAGG aims to infer the latent inter-dependencies among assets automatically, rather than relying on heuristic graphs (which might be based on historical correlations, industry sectors, etc.). By learning the graph, we allow the model to discover and adjust relationships that are

[2]ReLU (Rectified Linear Unit) is an activation function defined as $f(x) = \max(0, x)$. It outputs the input if it is positive, otherwise zero, introducing non-linearity and helping neural networks learn complex patterns.

most relevant for the prediction task, potentially improving performance and adaptability.

We introduce a trainable graph embedding matrix $\mathbf{E}^{(g)} \in \mathbb{R}^{n \times d_g}$ (of dimension d_g) for the nodes, which will be used to generate the adjacency. In our implementation, we tie this graph embedding to the same $\mathbf{E}$ used in NAPL (*i.e.*, we use a common embedding for both weight generation and graph generation). This not only reduces parameters but also enforces that a single node representation encodes information for both its own dynamics and its connectivity, leading to consistent and interpretable embeddings. Given $\mathbf{E}^{(g)}$, we compute an adaptive adjacency matrix $\mathbf{A}$ by a similarity function of the embeddings. A simple and effective choice (also used in Bai et al., 2020) is to take the inner product between node embeddings:

$$\tilde{\mathbf{A}} = s\left(\mathbf{E}^{(g)}\mathbf{E}^{(g)\top}\right). \tag{2.1}$$

Here, $\mathbf{E}^{(g)}\mathbf{E}^{(g)\top} \in \mathbb{R}^{n \times n}$ produces a raw affinity score for each pair of nodes; $s(\cdot)$ is a normalization function. We apply a row-wise normalization such that each node's outgoing weights sum to 1 (effectively a softmax over each row, or a symmetric normalization as in GCNs), this ensures the graph convolution will be well-conditioned and prevents trivial scale issues. In practice, one can also include self-connections by setting the diagonal entries of $\mathbf{A}$ appropriately, or adding an identity matrix before normalization (Kipf and Welling, 2017). We denote the resulting normalized adjacency matrix as $\tilde{\mathbf{A}}$. This $\tilde{\mathbf{A}}$ will be used in all graph convolution operations in our model (replacing any fixed adjacency). Its entries are learnable since $\mathbf{E}^{(g)}$ is learned during training. The DAGG module thus produces a data-driven graph that reflects the learned correlations between assets. We emphasize that this approach is fully inductive: no external information about asset connections is needed, *i.e.* the graph is inferred from the time series data itself. This aligns with recent self-adaptive GCN approaches, but here with the added interpretability that the learned $\mathbf{E}^{(g)}$ positions assets in an embedding space where dot-products correspond to relatedness.

Although our base model uses the first-order graph convolution (immediate neighbors) as in Kipf and Welling (2017), one could extend the adjacency to incorporate multi-hop connections. For example, using a polynomial of $\tilde{\mathbf{A}}$ (such as $\tilde{\mathbf{A}}^2$ for second-order neighbors, or a Chebyshev polynomial expansion of the graph Laplacian) would allow information to propagate through multi-step relationships. In this work, we found that stacking multiple graph-convolutional recurrent layers is sufficient to capture higher-order structure, so we did not explicitly use higher-order polynomials in the filter. However, the normalization $s(\cdot)$ in Eq. (2.1) effectively rescales the eigenvalues of $\mathbf{A}$ (a form of spectral normalization) which keeps the graph filter localized and stable. This means that the learned adjacency $\tilde{\mathbf{A}}$ can be interpreted as defining a one-step propagation operator.

To capture temporal dependencies in the covariance time series, we integrate the above graph modules into a recurrent neural network architecture. Specifically, we employ a Gated Recurrent Unit (GRU) for each node, but replace the GRU's dense matrix operations with graph convolution operations. This produces a graph-convolutional GRU where information from neighboring nodes (as determined by $\tilde{\mathbf{A}}$) is used in the gating and state updating of each node at each time step. The motivation is that an asset's volatility at time t may be better predicted by looking not only at its own past, but also at the past of related assets (especially during market shocks where correlations spike).

Let $\mathbf{h}_{i,t}$ denote the hidden state for node i at time t, representing the latent dynamics of asset i that contribute to future covariance. We initialize $\mathbf{h}_{i,0}$ to zero or a learned initial vector. At each time step t, the GRU updates each $\mathbf{h}_{i,t}$ by combining the previous state $\mathbf{h}_{i,t-1}$ and the new inputs at time t. In our model, the input for node i at time t consists of its feature $\mathbf{x}_{i,t}$ (from $\mathbf{X}_t$) as well as potentially information from other nodes via the graph. We denote by $[\mathbf{H}_{t-1}|\mathbf{X}_t] \in \mathbb{R}^{n\times(d_h+F)}$ the concatenation of the previous hidden states and current features for all nodes (where d_h is the hidden state dimension per

node). The core update equations of the graph GRU are:

$$\mathbf{R}_t = \sigma\Big(\tilde{\mathbf{A}}, [\mathbf{H}_{t-1}|\mathbf{X}_t], \Theta^r\Big) \tag{2.2}$$

$$\mathbf{Z}_t = \sigma\Big(\tilde{\mathbf{A}}, [\mathbf{H}_{t-1}|\mathbf{X}_t], \Theta^z\Big) \tag{2.3}$$

$$\widetilde{\mathbf{H}}_t = \tanh\Big(\tilde{\mathbf{A}}, [\mathbf{R}_t \odot \mathbf{H}_{t-1}|\mathbf{X}_t], \Theta^h\Big) \tag{2.4}$$

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \widetilde{\mathbf{H}}_t, \tag{2.5}$$

which are respectively the reset gate, the update gate, the candidate gate and the state update. Here $\sigma(\cdot)$ is the sigmoid function and $\odot$ denotes elementwise (Hadamard) multiplication. All quantities are matrices of size $n \times d_h$ (one row per node). $\Theta^r, \Theta^z, \Theta^h$ are the learnable weight sets for reset gate, update gate, and candidate state computation, respectively (each implements a NAPL-generated linear transform with bias, as described earlier). In other words, for each gate we have a shared pool of parameters and node embeddings that produce node-specific weight matrices, just as in the static NAPL-GCN layer. The adjacency matrix $\tilde{\mathbf{A}}$, learned via DAGG, is used to perform the graph-convolution on the concatenated input $[\mathbf{H}_{t-1}|\mathbf{X}_t]$, so that each node i aggregates information from its neighbors before applying the gate weights. Intuitively, $\mathbf{R}_t[i]$ (the i-th row of $\mathbf{R}_t$) decides how much of asset i's previous state to reset based on both asset i's own memory and its neighbors' memories; $\mathbf{Z}_t[i]$ decides how much to update the new state versus keep the old state, thus controlling long-term memory. The candidate state $\widetilde{\mathbf{H}}_t[i]$ is a proposal for the new state, computed from the current inputs and the reset old state (again including neighbors). Finally, $\mathbf{H}_t[i]$ is the new hidden state for node i, interpolating between the old state and the candidate via the update gate. This mechanism, by design, can capture complex temporal patterns via gating while accounting for network effects via the graph convolution in each gate. All the parameters, including node embeddings E, weight pools Θ^S for each gate, are learned jointly by backpropagation through time, using the sequence of loss gradients.

Notably, AGCRN uses the same node embedding matrix $\mathbf{E}$ across all layers

and time steps, *i.e.* the embeddings that generate node-specific weights in NAPL are the very embeddings used in DAGG for graph construction. This shared embedding serves as a common thread linking the spatial (graph) and temporal aspects of the model. It means each asset i is represented by a fixed vector $\mathbf{e}_i$ throughout, which influences both how it interacts with others (graph connectivity) and how it processes information. This consistency is found to improve interpretability and acts as a regularizer: rather than learning disjoint representations for different parts of the model, the node embedding is constrained to satisfy multiple roles, discouraging it from overfitting any single facet.

2.2.3 Positive-definiteness

One important consideration is ensuring that the outputs $\widehat{\mathbf{RCOV}}_{T+1}$ remain valid covariance matrices. The aforementioned method does not explicitly guarantee positive-definiteness of the predicted $\widehat{\mathbf{RCOV}}$. To ensure positive definiteness, we apply a post-processing adjustment already used in the literature of covariance matrix forecasting (Higham, 1988; Bucci et al., 2022). Specifically, we perform an eigen-decomposition of each predicted matrix and truncate any negative eigenvalues to zero (or a small positive ϵ). That is, if $\widehat{\mathbf{RCOV}} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$, where $\mathbf{Q}$ is the matrix containing the eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues, $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$, we replace $\mathbf{\Lambda}$ with

$$\mathbf{\Lambda}^+ = \text{diag}(\lambda_1^+, \ldots, \lambda_n^+)$$

where

$$\lambda_i^+ = \max(\lambda_i, 0)$$

and reconstruct

$$\widehat{\mathbf{RCOV}}^+ = \mathbf{Q}\Lambda^+\mathbf{Q}^\top.$$

This adjusted $\widehat{\mathbf{RCOV}}^+$ is symmetric Positive Semi-Definite (PSD) and is used as the final forecast. In practice, we rarely observed significantly negative

eigenvalues given that when the model is trained well, $\widehat{\mathbf{RCOV}}$ is usually near-PD. Therefore, this step acts as a safety check to correct any mild indefiniteness. An alternative way to guarantee Symmetric Positive Definite (SPD) is to parameterize the output via a matrix factorization, *e.g.* predicting a lower-triangular Cholesky factor (Halbleib-Chiriac and Voev, 2011), or using a matrix logarithm transformation (Bauer and Vorkink, 2011). We decided not to embed such constraints into the network to keep the architecture simpler and outputs interpretable in the original space, and instead rely on the eigenvalue adjustment which is efficient and does not interfere with the learning of the model. This approach is consistent with practices in financial covariance modeling where a near-semidefinite matrix is projected to the positive-definite cone as a post-processing step if needed (Fan et al., 2012).

2.3 Empirical application

2.3.1 Dataset

The analysis is conducted on a high-dimensional dataset comprising daily returns and daily realized covariance matrices for 50 highly liquid and capitalized companies that are components of the S&P 500 index. A list with tickers and company names is reported in Table 2.1. The data, sourced from Bloomberg, spans from April 1st, 2021, to April 1st, 2025, covering a total of 1,060 trading days. A key input to our model, the daily realized covariance matrices, were computed using 5-minute intraday returns.

2.3.2 Data processing

When constructing the node features matrices, we leveraged the symmetry of RCOV matrices by focusing on the diagonal elements, which relate to each asset individually, and the upper off-diagonal elements, which capture the risk associated with each pair of assets. To enhance processing, these variables were standardized to facilitate the training and processing.

Table 2.1: **List of the 50 US stocks used in this study**

Ticker	**Stock**	**Ticker**	**Stock**	**Ticker**	**Stock**
AAPL	Apple Inc.	GE	General Electric Co.	NKE	Nike Inc.
ABT	Abbott Laboratories	GOOG	Alphabet Inc. Class A.	NVDA	NVIDIA Corp.
ACN	Accenture	GS	Goldman Sachs Group Inc.	ORCL	Oracle Corp.
ADBE	Adobe Inc.	HD	Home Depot Inc.	PEP	PepsiCo, Inc.
AMGN	Amgen Inc.	IBM	IBM Corp.	PFE	Pfizer Inc.
AMT	American Tower Corp.	INTU	Intuit Inc.	PG	Procter & Gamble Co.
AMZN	Amazon.com Inc.	ISRG	Intuitive Surgical Inc.	QCOM	Qualcomm Inc.
AXP	American Express Co.	JNJ	Johnson & Johnson	T	AT&T Inc.
BAC	Bank of America Corp.	JPM	JPMorgan Chase & Co.	TJX	TJX Companies Inc.
CAT	Caterpillar Inc.	KO	Coca-Cola Co.	TMO	Thermo Fisher Scientific Inc.
CMCSA	Comcast Corp.	LLY	Eli Lilly & Co.	TXN	Texas Instruments Inc.
COST	Costco Wholesale Corp.	LOW	Lowe's Companies Inc.	UNH	UnitedHealth Group Inc.
CRM	Salesforce Inc.	MA	Mastercard Inc.	UNP	Union Pacific Corp.
CSCO	Cisco Systems Inc.	MCD	McDonald's Corp.	VZ	Verizon Communications Inc.
CVX	Chevron Corp.	MRK	Merck & Co. Inc.	WFC	Wells Fargo & Co.
DHR	Danaher Corp.	MSFT	Microsoft Corp.	WMT	Walmart Inc.
DIS	The Walt Disney Co.	NFLX	Netflix Inc.		

When creating balance between the feature matrices (node-level attributes) with the embedding matrices (pairwise asset relationships), we designed embeddings specifically for asset pairs where for each unique combination of two assets (i, j), we computed a corresponding embedding vector. Afterwards, min-max scaling was applied on the embedding matrices aiming to ensure stability during training.

The data were then converted into graphical representations where the diagonal and off-diagonal elements of the RCOV matrices served as nodes, and lagged returns were used as node embeddings. The dataset was split into 70% training observations ($T_{train} = 742$) and 30% testing observations ($T_{test} = 318$). Since we applied a rolling-window prediction, we always used a window of 742 observations to make one-step-ahead predictions, and the model has been re-trained at each step.

2.3.3 Hyperparameter's setting

All analyses were implemented using Python 3.9.13. The deep-learning-based AGCRN model was developed using the following key libraries[3].

Five parameters are crucial for the implementation of the Node Adaptive Graph Convolution Layer. The `in_channels` parameter refers to the number of input features, which in our case is set to 1, as each node has a single feature. The `out_channels` parameter refers to the number of the predicted output features, which in our case is set to 1, again as each node has a single feature. The `hidden_dimensions` refers to the number of hidden layers which was determined through experimentation to balance model capacity and empirical performance. We chose a value of 1, which provides a rich representation for each node while ensuring compatibility with subsequent layers and the task at hand. This choice was based on metrics such as validation loss and model

[3]Additional libraries used for data preparation, results evaluation, and model simulations include `numpy` 1.26.4, `pandas` 2.2.3, `yfinance` 0.2.54, `networkx` 3.2.1, `matplotlib` 3.8.3, `scikit-learn` 1.4.0, `keras` 2.15.0, `statsmodels` 0.14.2, `pickleshare` 0.7.5, `tqdm` 4.66.2, and `scipy` 1.13.1.. Specifically, we utilized `torch` version 2.2.2, `torch-geometric` version 2.4.0, and `torch-geometric-temporal` version 0.54.0.

interpretability. The parameter K is set to 1 to capture a broader context around each node without over-smoothing, as determined by evaluating the model's performance on a validation set. Finally, the embedding dimension is set to 2.

The model is optimized using the Adam optimizer[4], which is run for up to 100 epochs. An early stopping strategy is employed with a patience of 10 epochs to prevent overfitting. The learning rate is set to 1×10^{-3}, chosen based on the model's convergence rate and stability during training.

2.3.4 Forecasting results

Learned Parameters

During the training process, the NAPL dynamically adjusts the weights based on the node embeddings, allowing the model to adapt to the underlying graph structure. The weights and biases within the NAPL layer are crucial for its operation. The `weights_pool` tensor has a shape of $[2, 1, 2, 2]$, and the `bias_pool` tensor has a shape of $[2, 2]$. The first dimension of these tensors corresponds to the number of node embedding dimensions (1), the second dimension represents the order of the filter ($K = 1$), the third dimension is the sum of input channels and output channels (1 input feature plus 1 hidden state features), and the fourth dimension reflects the requirements of the gates (twice the number of output channels, *i.e.*, $2 \cdot 1$).

The `gate` module is responsible for calculating gating signals, similar to those in GRUs or LSTMs. It uses a `weights_pool` tensor of shape $[2, 1, 2, 2]$ and a `bias_pool` tensor of shape $[2, 2]$. These shapes are chosen to ensure that the gates can handle the combined input and hidden state features properly.

The `update` module, which updates the hidden states based on the gated input, uses a `weights_pool` tensor of shape $[2, 1, 2, 1]$ and a `bias_pool` ten-

[4]The Adam optimizer is a stochastic gradient descent algorithm that adapts the learning rate for each parameter based on the magnitude of the gradient. It is widely used due to its stability and efficiency in deep learning applications. For more details, see Kingma and Ba (2014).

sor of shape $[2, 1]$. Here, the last dimension corresponds to the number of output channels (2), reflecting the need to maintain the hidden state features accurately.

The hidden state in the AGCRN model maintains the temporal dynamics of the nodes and has a shape of $[1, 1275, 2]$, where 1 represents the batch size, 1275 is the number of nodes in the graph, 50 of which are the diagonal elements of the realized covariance matrix and 1225 are the upper off-diagonal once, and 1 is the number of output channels, representing the hidden state features for each node. This structure ensures that each node's state is updated based on its features and the features of its neighbors, allowing the model to capture both local and broader graph structures.

2.3.5 Results

As shown in Figure 1, during training both the training and validation loss curves show a clear decreasing trend, which is a good indication of effective learning throughout the epochs. The final training loss averaged 0.1063 with a standard deviation of 0.1695 and the final validation loss averaged 0.0197 with a standard deviation of 0.0430. While there is some variance in the training loss, the substantially lower validation loss indicates that the model has achieved good generalization performance without overfitting.

Two additional metrics were evaluated during training: the Root Mean Square Error (RMSE) curves similarly displayed a consistent downward slope, highlighting improvements in the model's predictive accuracy over time. The final average RMSE for the training data was 0.1062 ($\pm$ 0.1249), while for the validation data it was notably lower at 0.0703 ($\pm$ 0.0779), indicating excellent generalization capabilities. The lower validation RMSE compared to training RMSE demonstrates that the model performs even better on unseen data, which is a positive indicator of robust learning. The MAE plots mirror the trends observed in the loss and RMSE curves, showing a steady reduction in error rates as training progresses. By the end of the training, the average MAE

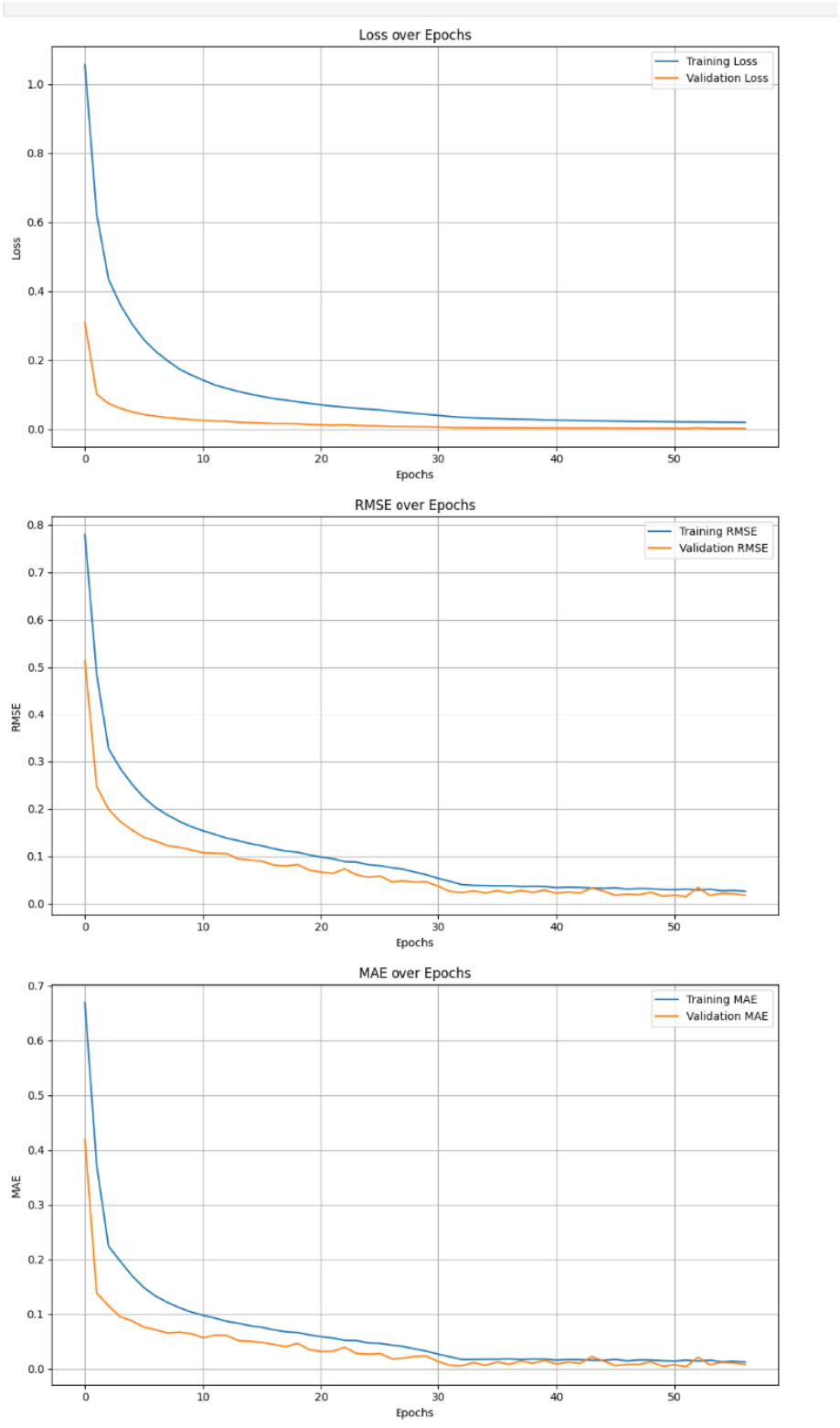


Figure 2.1: **Training and Validation Metrics**

for the training data was 0.0694 ($\pm$ 0.1026), while for the validation data, it reached 0.0397 ($\pm$ 0.0587). These results align with those seen in other metrics, confirming that the model's predictions are both reliable and robust, with the validation performance consistently outperforming the training metrics across all evaluation measures.

2.3.6 Predictions comparison and discussion

We then focused on the statistical evaluation of the predictive performance of our approach with respect to alternative methods. Specifically, we compared it with the predictions from a Long Short-Term Memory (LSTM) method with 1 as hidden dimension and 1 as number of layers , a Vanilla Elman Recurrent Neural Network (RNN) with 2 as hidden dimension and 1 as number of layers; for both machine learning models, we used an Adam optimizer running for 100 epochs. An early stopping strategy is employed with a patience of 10 epochs to prevent overfitting. The learning rate of these ML methods is set to 1×10^{-3}. The set of competing models also comprehends a random walk (RW) model (*i.e.*, $\widehat{\mathbf{RCOV}}_{t+1} = \widehat{\mathbf{RCOV}}_t$) and a single-lag Vector Autoregressive (VAR) model applied to the single values of realized covariances. The predictions for the AGCRN and VAR models went through a process to ensure comparability as each prediction, initially in the form of a vector, was reconstructed into a symmetric matrix. This reconstructed the matrices were de-normalized and eigenvalue clipping process was implemented to ensure positive definiteness (see Section 2.2.3).

Performance Metrics

We used different loss functions to evaluate the predictive accuracy of each method with respect to the observed values of the realized covariance matrices.

To be coherent with the related Patton and Sheppard (2009), we used the Frobenius and Euclidean distances. The former provides a way to quantify

similarities and differences of predicted and actual values using this formula:

$$\text{Frobenius Norm Error} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij} - b_{ij})^2}.$$

On the other hand, the Euclidean norm measures the difference between two vectors, representing the "distance" between the predicted and actual values using the following formula:

$$\text{Euclidean Norm Error} = \sqrt{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}.$$

Since the covariance matrices lie on a Riemannian manifold (Bucci et al., 2024), we also used a specific geometric-aware metric, *i.e.* the Procrustes distance (Dryden et al., 2009; Dryden and Mardia, 2016). This is an alternative measure used to compare the shapes of two or more configurations of points, involving superimposing one configuration onto another through translation, rotation, and scaling to minimize the sum of squared distances between corresponding points.

Finally, we used the R-squared value (R^2) calculated by fitting a linear regression model to the predicted values from each method against the true values. This metric assesses how well these predictions can be explained by a linear relationship with the true values (Mincer and Zarnowitz, 1969).

For each set of predictions $\hat{y}_{\text{pred}}$, a linear model was fitted:

$$y_{\text{true}} = \beta_0 + \beta_1 \cdot \hat{y}_{\text{pred}} + \upsilon_t$$

where y_{true} is the observed value of each RCOV element in the test sample, $\hat{y}_{\text{pred}}$ is one of the prediction sets (RW, AGCRN, VAR, LSTM & RNN), β_0 and β_1 are the coefficients determined by the regression. The R-squared of each regression was calculated for each element of RCOV, then we used the average of the R^2s to compute a metric.

Predictive Accuracy measures

In order to statistically evaluate the predictive accuracy of AGCRN model, we implemented the Giacomini-White test (Giacomini and White, 2006), whose

results are presented in Table 2.2. Ecco la traduzione in inglese:

The Giacomini-White test is the ideal candidate for evaluating forecasts derived from a rolling window scheme, whereas it does not permit the use of a recursive scheme. More specifically, they show that when forecasts are constructed using a rolling window scheme with a fixed window d, as in the present work, then the null hypothesis is given by:

$$H_0 : E\big[(Y_{t+1} - f_t(\hat{\beta}1t))^2 - (Yt + 1 - g_t(\hat{\beta}2t))^2 \mid It\big] = 0 \tag{2.6}$$

where $f_t(\hat{\beta}_{1t})$ and $g_t(\hat{\beta}_{2t})$ are two forecasting models for the conditional mean of the variable of interest Y_{t+1}, given a quadratic loss function.

The fact that the test is primarily based on forecasts from a rolling window scheme allows it to dispense with certain assumptions required by other tests, such as Diebold and Mariano (1995) and West (1996). In particular, the assumption of stationarity for the observations is no longer necessary, making the test applicable to a broader class of models, including all linear and non-linear, semi-parametric or non-parametric, nested and non-nested models.

As shown in Table 2.2, we observe that AGCRN consistently demonstrated superior predictive performance over the competing models across both error metrics. When using the Frobenius distance, AGCRN outperformed RNN and RW (with test statistics equal to -3.089 and -2.778, respectively), both of which are statistically significant at the 1% level, and it exhibited even greater superiority over the VAR model with a highly significant statistic of -9.417. In comparison to LSTM, AGCRN performed better, although the difference was not statistically significant, as indicated by a test statistic of -0.865. Regarding Euclidean distance, AGCRN again achieved stronger results: it outperformed LSTM (-2.474, significant at 5%), RNN (-4.523, significant at 1%), and achieved very large and highly significant test statistics compared to RW and VAR (-7.958 and -27.259, both significant at the 1% level). However, in terms of the Procrustes distance, although LSTM outperformed AGCRN (1.914, without statistical significance) it achieved stronger results in comparison to other models, as it outperformed RNN (-2.803, significant at 1%), and

achieved very large and highly significant test statistics compared to RW and VAR (-2.803 and -22.99, both significant at the 1% level). In general, these results confirm the robust predictive accuracy of AGCRN, with clear advantages over traditional models such as RW and VAR, and comparatively competitive results against other neural networks, although the margin of improvement over LSTM and RNN is less substantial and often statistically modest.

Model Confidence Set (MCS)

To simultaneously evaluate the predictive accuracy of the model proposed here, we also performed the Model Confidence Set (MCS) (Hansen et al., 2011). The MCS is a statistical procedure designed to identify a subset of models that are statistically indistinguishable in terms of performance, given a predefined significance level (α). The MCS iteratively eliminates poorly performing models based on loss functions until only the superior models remain. It is implemented using metrics such as Frobenius, Euclidean, or Procrustes distances and relies on equivalence tests and elimination rules.

For example, in the provided analysis referred to in Table 2.3, the MCS was applied with $\alpha = 0.1$ and 10,000 bootstrap replicates to evaluate the predictive accuracy of AGCRN against other models, including LSTM, RNN, RW, and VAR. The metric $T_{\max}$[5] was computed to rank the models and determine their inclusion in the Superior Set of Models (SSM).

The analysis began by computing loss functions (Frobenius, Euclidean, and Procrustes distances) for each model's predictions compared to observed data. These loss values were then used to perform the MCS procedure, which iteratively removed inferior models.

For instance, RW and VAR were consistently dropped early in the MCS procedure due to their poor predictive performance, as reflected by their sub-

[5] $T_{\max}$ is the maximum statistic (often called the "max-t" statistic) used in the MCS procedure. It is computed as the largest absolute value of the pairwise test statistics comparing the predictive performance of each model. $T_{\max}$ is used to iteratively eliminate the worst-performing models until the SSM is identified.

Table 2.2: **Realized covariance forecasting results with Giacomini-White test**

Panel A: Frobenius

	LSTM	RNN	RW	VAR
AGCRN	-0.865	-3.089 ***	-2.778 ***	-9.417 ***
LSTM		-2.542 **	-2.761 ***	-9.362 ***
RNN			2.750 ***	-9.356 ***
RW				-3.165 **

Panel B: Euclidean

	LSTM	RNN	RW	VAR
AGCRN	-2.474 **	-4.523 ***	-7.958 ***	-27.259 ***
LSTM		-1.865 *	-7.916 ***	-27.240 ***
RNN			-7.898 ***	-27.161 ***
RW				-7.1563 ***

Panel C: Procrustes

	LSTM	RNN	RW	VAR
AGCRN	1.914	-2.803 ***	-5.895 ***	-22.99 ***
LSTM		-5.924 ***	-5.909 ***	-23.01 ***
RNN			-5.861 ***	-22.89 ***
RW				-3.359 ***

Note: LSTM represents the predictions from Long Short-Term Memory model, RNN represents the predictions from Recurrent Neural Networks model, RW denotes a prediction from a random walk, and VAR represents the predictions through a VAR(1) on the elements of RCOV. The test statistics are computed using a loss difference $L_1 - L_2$, where L_1 is the loss of the model in row and L_2 of that in column. *, ** and *** denote respectively a rejection at a 10%, 5% and 1% significance level.

stantially higher average losses across all metrics (Frobenius: 7.107×10^{-6} and 9.332×10^{-6}; Euclidean: 0.001761 and 0.002043; Procrustes: 0.0002755 and 0.00027) along with negligible probabilities of inclusion in the Superior Set of Models ($p_{\text{MCS}} < 0.001$). The final SSM included AGCRN together with LSTM and RNN, which showed statistical equivalence at the 10% significance level ($\alpha = 0.1$). Notably, AGCRN achieved perfect inclusion probabilities in Frobenius and Euclidean MCS tests ($p_{\text{MCS}} = 1.000$), confirming its robustness in predictive accuracy; although its p_{MCS} dropped below 0.001 for the Procrustes metric, AGCRN still recorded the second lowest average loss (0.0002188) among all models. Across all metrics, AGCRN produced the smallest average losses (Frobenius: 5.351×10^{-6}, Euclidean: 0.001449, Procrustes: 0.0002188), with unwavering inclusion in the SSM over $10,000$ bootstrap replicates[6], underscoring its consistent and reliable forecasting performance. In contrast, the LSTM and RNN models, while sometimes retained in the SSM (*e.g.*, LSTM's $p_{\text{MCS}} = 0.4139$ in Frobenius and 1.000 in Procrustes), generally exhibited higher losses or lower robustness across some metrics. In general, these findings highlight the clear superiority of AGCRN in predicting realized covariance matrices, achieving results that are compatitive with the LSTM once and achieving substantial loss reductions relative to the other once. Which is also proved with Mincern- Zarnowitz R^2 where the AGCRN came in second place after LSTM with a small margin of difference but it did outperform RNN, VAR & RW models.

[6]A p_{MCS} of 1.000 indicates that AGCRN was never eliminated in any of the 10,000 bootstrap replicates, demonstrating unequivocal superiority.

Table 2.3: **Average loss functions and MCS p-values in realized covariance forecasting**

	Frobenius		**Euclidean**		**Procrustes**		
Model	**Avg. loss**	p_{MCS}	**Avg. loss**	p_{MCS}	**Avg. loss**	p_{MCS}	R^2
AGCRN	5.351e-06	**1.000**	0.001449	**1.000**	0.0002188	<0.001	0.7141
LSTM	5.357e-06	**0.4139**	0.001450	<0.001	0.0002187	**1.000**	**0.7162**
RNN	5.366e-06	<0.001	0.001451	<0.001	0.0002190	<0.001	0.7061
VAR	9.332e-06	<0.001	0.002043	<0.001	0.0002755	<0.001	0.1074
RW	7.107e-06	<0.001	0.001761	<0.001	0.0002545	<0.001	0.1470

Note: Underlined values denote the lowest loss functions for each setting. The AGCRN, LSTM, and RNN models use a single lag of historical data for input. The models are trained with standard MSE loss functions. The p-value refers to the probability of being included in the SSM over $10,000$ block bootstrap replicates (in bold values with a p_{MCS} greater than 0.75).

2.4 Conclusion

This chapter studies the possibility of predicting realized covariance matrices using Adaptive Graph Convolutional Recurrent Network (AGCRN), which leverages both spatial and temporal dependencies and offers a robust framework that outperforms traditional models as well as deep learning models in several metrics. The AGCRN's ability to dynamically adapt to evolving relationships between assets, facilitated by its Node Adaptive Parameter Learning (NAPL) and Data Adaptive Graph Generation (DAGG) modules, makes it particularly well-suited for capturing the complex dynamics of financial markets.

The empirical result from this study confirms AGCRN's superior predictive accuracy, as evidenced by the lower Frobenius norm, Euclidean norm, and Procrustes distance compared to other models. Additionally, statistical tests such as the Giacomini-White test confirm that AGCRN outperforms traditional models like RW and VAR while maintaining competitiveness with advanced neural network models. This is further supported by the Model Confidence Set (MCS) analysis underscoring AGCRN's reliability across different evaluation metrics.

For our future research, we want to check how AGCRN performs under extreme market conditions and outliers, which is critical to improving its robustness in real-world applications; we also want to explore its performance in other financial contexts, such as portfolio optimization and risk analysis, to further emphasize its possible benefits. This research contributes to the growing literature on machine learning applications in finance by demonstrating the efficacy of AGCRN in overcoming the obstacles linked with predicting realized covariance matrices.

Chapter 3

Portfolio optimization using Spanning Tree combined with Adaptive Graph Convolutional Recurrent Networks

3.1 Introduction

Portfolio optimization became a keystone of asset management with the introduction of Modern Portfolio Theory (MPT) by Markowitz (1952), given that it provides an objective way to balance risk and return within asset's combinations achieving different investment objectives (Palomar, 2025). The MPT formalizes the trade-off between risk and return, encapsulated in the concept of the efficient frontier which is composed by portfolios that offer the highest expected return for a given risk level or the lowest risk for a given expected return. Mathematically, it is derived as the set of solutions to the mean-variance optimization problem, where portfolio variance is minimized subject to constraints on expected return and the sum of portfolio weights. Portfolios lying on the efficient frontier are considered "efficient" because no other feasible portfolio exists with higher expected return for the same or lower risk, or with

lower risk for the same or higher expected return (Best, 2010; Gunjan and Bhattacharyya, 2022).

Following Markowitz's 1952 publication, different techniques have been defined aiming to obtain optimal portfolios of assets (Zhang, 2023; Salo et al., 2023). For instance, after Merton (1980) proved that the estimation of average returns is highly error-prone, one of the most used variants of MPT became the Global Minimum Variance (GMV) optimization, which focuses exclusively on minimizing portfolio risk by optimizing asset weights to achieve the lowest possible variance, without requiring estimates of expected returns. Another extension of MPT is the tangency portfolio, a model that was introduced as a cornerstone of the Capital Asset Pricing Model (CAPM) in the seminal work of Sharpe (1964) and operationalized via the Sharpe ratio in 1966. This model represents the point where the Capital Market Line (CML) tangents the efficient frontier, offering maximal risk-adjusted returns and allowing to identify the optimal portfolio on the efficient frontier that maximizes the Sharpe ratio by combining risky assets with a risk-free asset.

Although there are plenty of models used for portfolio optimization, many challenges are still not fully addressed. For instance, the traditional mean-variance optimization remains affected by the portfolios phenomenon known as "Markowitz optimization enigma" stating that the model's performance is influenced by the input parameters' sensitivity, *i.e.* a minor error in the estimation of the expected returns or covariance matrices leads to unstable portfolio performance (Michaud, 1989). Furthermore, the MPT's reliance on variance as a risk measure has been widely criticized for penalizing upside and downside risk equally, ignoring tail risks critical to investors (Belabbes et al., 2024). Moreover, despite the efforts of Bayesian and subjective models (Black and Litterman, 1991) to lower estimation errors by integrating market equilibrium and subjective views, they remain constrained by their dependence on equilibrium returns and the ad hoc specification of investor views, which may conflict with empirical market dynamics (Belabbes et al., 2024). In ad-

dition, the risk parity allocation model faces criticism for equating volatility with risk and failing to account for asymmetric tail dependencies, particularly in leveraged environments. Robust optimization frameworks, as well, are not exempt from errors since they often incur in prohibitive computational costs when scaling to large portfolios, while heuristic approaches like equal weighting or buy-and-hold strategies lack of adaptability to shifting market regimes and suffer from implicit sector or liquidity biases. These challenges highlight the need for models that harmonize theoretical rigor with empirical robustness.

In the first chapter, we have built on Adaptive Graph Convolutional Recurrent Network (AGCRN) model that introduces two innovations: Node Adaptive Parameter Learning and Data Adaptive Graph Generation (Bai et al., 2020), to address some critical limitations in the realized covariance matrix forecasting like high-dimensionality, nonlinear dependencies, and time-varying correlations. AGCRN also mitigates input sensitivity by learning covariance structures directly from high-frequency returns, bypassing error-prone return assumptions, while its adaptive graph convolutions address nonlinear tail dependencies. By integrating graph-based spatial relationships with temporal RNN modules, we proved the competitiveness of AGCRN with other machine learning techniques like LSTM and its superior forecasting accuracy over VAR and heuristic benchmarks, as validated by the Model Confidence Set test.

Realized covariance matrices are a cornerstone input for portfolio optimization as they provide accurate, high-frequency measures of asset return co-movements that capture the dynamic dependencies essential for risk assessment and allocation decisions (Bucci et al., 2024). Therefore, we can say that the advancement of forecasting them using AGCRN provides a robust foundation for portfolio optimization.

In this chapter, we leverage the use of the realized covariances as ex-post estimates of integrated covariance of asset returns over a specific period (Barndorff-Nielsen and Shephard, 2002) as inputs to compute distance matrices that are essential for the application of Spanning Tree optimization

(Mantegna, 1999). A distance matrix is a symmetric matrix that quantifies the pairwise dissimilarities between assets by transforming the information from the correlation driven from the covariance matrix into geometric distances, thereby mapping financial relationships into a metric space suitable for network analysis and hierarchical clustering. In financial applications, each element of the distance matrix represents the effective "distance" between two assets, typically derived from their correlation coefficient, such that highly correlated assets are mapped closer together and weakly or negatively correlated assets are farther apart; this transformation is fundamental for filtering noise, uncovering the hierarchical organization of markets, and enabling robust portfolio optimization and risk management strategies by highlighting diversification opportunities and systemic risk channels that are not evident from raw correlations alone (Mantegna, 1999; Tumminello et al., 2010; Pozzi et al., 2013).

The concept of spanning trees[1] is fundamental in graph theory and plays a central role in numerous optimization problems, where the goal is to find a tree that optimally balances connectivity and cost or weight (Cormen et al., 2009). Most of the optimization techniques based on a graph rely on the Minimum Spanning Tree (MST) (Danko et al., 2022), a network-based method that gained importance in finance since Mantegna's seminal application to stock correlation analysis (Mantegna, 1999). This approach is capable of reveling the hierarchical market structures by distilling complex relationships into sparse, interpretable networks (Tumminello et al., 2010) as it is used to filter and represent the most significant relationships among a set of nodes by keeping the minimal total edge weights, where the edge weights derived from the constructed distance matrices (Millington and Niranjan, 2021).

However, the standard Minimum Spanning Tree (MST) is fundamentally misaligned with the goal of portfolio diversification when using a correlation-

[1] A *spanning tree* of a connected, undirected graph $G = (V, E)$ is a subgraph that is a tree containing all vertices V of the original graph. In other words, a spanning tree connects every vertex in the graph without forming any cycles and contains exactly $|V| - 1$ edges (Cormen et al., 2009).

based distance metric. The core objective of our optimization is to minimize portfolio risk, which is achieved by selecting assets that are as uncorrelated as possible. Our distance matrix is constructed such that a low distance signifies a high correlation (*e.g.*, using the common transformation $d_{ij} = 2(1 - \rho_{ij})$, where ρ_{ij} is the correlation between assets i and j).

A direct application of the MST would seek to minimize the sum of edge weights (distances). This would lead to preferentially connecting the assets with the lowest distances, in other words, the most highly correlated assets. The resulting tree would represent a "minimum diversification" portfolio, chaining together assets that move in tandem, and thus amplifying systematic risk. This is exactly the opposite of what a prudent portfolio construction strategy aims to achieve.

To align the graph optimization with our financial objective, we must instead prioritize the asset pairs with the highest distances, as these represent the lowest correlations and therefore offer the greatest diversification benefits. The appropriate tool for this task is the Maximum Spanning Tree (MaxST) (Pemmaraju and Skiena, 2022). By maximizing the sum of the edge weights (distances), the MaxST identifies the backbone structure of a well-diversified portfolio. It filters the asset network to retain the connections that contribute the most to risk reduction (West, 2001).

We use the forecasts of the realized covariance matrices of the 50 assets from the AGCRN model introduced in the first chapter as our input data to generate the distance matrices, and the edge weights are then used in the implementation of the MaxST to obtain the optimized portfolio. This helps in creating portfolios that are characterized by enhanced diversification, high performance in risk-adjusted return terms, balanced risk levels, and effective active management qualities compared to those obtained from other models.

To make this chapter more interesting, we inserted additional analysis where we were interested to understand whether a change in the methodological process to use AGCRN to predict distances and then implemented

maximum spanning trees to get the the portfolios from these predictions and compared them to the once from the realized covariance predictions.

This chapter is organized as follows. In Section 3.2, we introduce the methodology, while Section 3.3 presents the results of the optimized portfolios, in section 3.4 we have the results of the distance predictions. Section 3.5 concludes.

3.2 Methodology

3.2.1 Construction of Distance Matrices

Distance matrices are a very fundamental tool in financial network analysis and portfolio optimization, as they transform the information captured by correlation matrices into a metric space suitable for hierarchical clustering and graph-based modeling. When distance matrices are driven from correlations, they encode these dependencies as geometric distances, which lead to enabling the construction of structures that reveal most of the significant relationships in a financial system (Mantegna, 1999). This transformation is crucial for filtering out noise and spurious correlations, especially in high-dimensional portfolios, thereby improving the robustness of risk management and asset allocation strategies (Tumminello et al., 2010). Moreover, distance matrices facilitate the identification of diversification opportunities and systemic risk channels by highlighting the hierarchical organization of assets, which is not readily apparent from raw correlations alone (Pozzi et al., 2013). Their use is well established in the literature for both empirical studies and practical applications in constructing diversified, stable portfolios and understanding market structure.

Let us consider the availability of high-frequency data for a set of n assets. Given a daily covariance matrix, $\mathbf{\Sigma}_t$, computed as follows:

$$\mathbf{\Sigma}_t = \sum_{\tau=1}^{N_t} \mathbf{r}_{\tau,t} \mathbf{r}'_{\tau,t}$$

where $\mathbf{r}_{\tau,t}$ is the n-dimensional vector of asset returns in the τ-th intra-period of day t and N_t is the number of high-frequency observations in the t-th day, we can derive the correlation matrix, $\mathbf{P}_t$, as

$$\mathbf{P}_t = \mathbf{R}_t^{-1} \mathbf{\Sigma}_t \mathbf{R}^{-1}$$

where $\mathbf{R}_t$ is a diagonal matrix with asset-specific standard deviations on the diagonal. Then, we can compute the distance matrix, $\mathbf{D}_t$, whose ij-th element is given by

$$d_{ij,t} = \sqrt{2 \cdot \max(1 - \rho_{ij,t}, 0)}$$

where $\rho_{ij,t}$ is the correlation between asset i and j at time t.

This formula ensures that the distances are not negative and that they satisfy the properties of a Euclidean metric, which is essential for graph-based analyses that will be done later, such as MaxST construction (Mantegna, 1999). The use of $\max(1 - \rho_{ij,t},\ 0)$ provides numerical stability by preventing the square root of negative values, which can arise from estimation or rounding errors in the correlation matrix. By converting correlations to distances in this manner, the methodology bridges the gap between traditional covariance-based risk modeling and modern network-theoretic approaches, enabling more robust and interpretable portfolio optimization (Tumminello et al., 2010; Pozzi et al., 2013).

3.2.2 Maximum Spanning Tree (MaxST) Construction

A Maximum Spanning Tree (MaxST) of a connected, undirected, weighted graph $G = (V, E)$ is a spanning tree whose total edge weight is maximized. In other words, it is a subgraph that connects all vertices V without forming cycles, such that the cumulative weight of its edges is maximized among all possible spanning trees for that graph (West, 2001).

Our portfolio construction method leverages the MaxST to identify the most significant diversification relationships within a set of assets. The process is executed through the following sequential steps:

1. Use distances to calculate the strength of the node to identify the root node.

2. Transform distances in a weight matrix for MaxST construction.

3. Compute the Maximum Spanning Tree (MaxST) from the transformed matrix through Prim's algorithm, initiated from the identified root node.

4. Recalculate the strength of each node using only the edges present in the resulting MaxST and their original distance values.

5. Normalize the node strenghts to produce the portfolio weights.

Node strength and root selection

In a weighted graph, node strength is a measure of a node's total connection intensity, calculated as the sum of the weights of all edges incident to it. This metric is fundamental for identifying central or influential nodes within a network (Barrat et al., 2004; Newman, 2010).

For the initial fully connected graph based on the distance matrix $\mathbf{D} = [d_{ij}]$, where each element d_{ij} represents the distance or dissimilarity between nodes i and j, the strength of node i is:

$$s_i = \sum_{j \in V, j \neq i} d_{ij}.$$

In our methodology, the node with the highest initial strength is selected as the root node. This choice ensures that the tree construction process begins from a central asset, *i.e.*, one that is, on average, most dissimilar to all other assets in the portfolio.

Distance transformation for MaxST construction

Standard algorithms for finding spanning trees are mainly designed to find a Minimum Spanning Tree (MST). To adapt these algorithms for our purpose of finding a Maximum Spanning Tree, we must first transform the edge weights.

Given the original distance matrix, $\mathbf{D}$, we create a new weight matrix, $\mathbf{W} = [w_{ij}]$, using the following transformation:

$$w_{ij} = d_{\max} - d_{ij},$$

where $d_{\max} = \max_{i,j} d_{ij}$ is the single largest distance value in the entire matrix.

This transformation inverts the weight hierarchy: the largest original distances (d_{ij}) become the smallest new weights (w_{ij}), and vice versa. Consequently, applying an MST algorithm to the transformed matrix $\mathbf{W}$ is mathematically equivalent to finding the MaxST of the original matrix $\mathbf{D}$. An algorithm minimizing the sum of the new weights will effectively maximize the sum of the original distances:

$$\arg\min_{T} \sum_{(i,j)\in T} w_{ij} = \arg\max_{T} \sum_{(i,j)\in T} d_{ij}$$

where T represents any spanning tree of the graph. This is a computationally efficient and standard technique in combinatorial optimization (Pemmaraju and Skiena, 2022).

MaxST construction via Prim's algorithm

With the transformed weights, we construct the tree using the Prim's algorithm. This greedy algorithm builds a spanning tree by iteratively adding the "cheapest" edge that connects a vertex in the growing tree to a vertex outside of it (Prim, 1957).

The algorithm proceeds as follows:

1. Start with a tree containing only the selected root node. All other nodes are considered unvisited.

2. Identify the edge with the minimum weight (w_{ij}) that connects a node inside the tree to a node outside the tree.

3. Add this minimum-weight edge and the corresponding unvisited node to the tree.

4. Repeat the iterative step until all nodes from the original graph have been included in the tree.

The final result is a Minimum Spanning Tree of the transformed weights, which corresponds to the Maximum Spanning Tree of the original distances.

Final Portfolio Weighting

Once the MaxST is identified, we revert to the original distance values (d_{ij}) for the edges that form this tree. The final portfolio weights are derived from the node strengths calculated within this sparse MaxST structure.

The strength of node i in the MaxST, denoted $s_i^{\max ST}$, is the sum of the original distances of the edges connected to it in the tree:

$$s_i^{\max ST} = \sum_{j \in \text{neighbors in MaxST}} d_{ij}.$$

A higher strength value indicates that the asset is a key node in the diversified structure, connected to other assets via paths of high dissimilarity (low correlation).

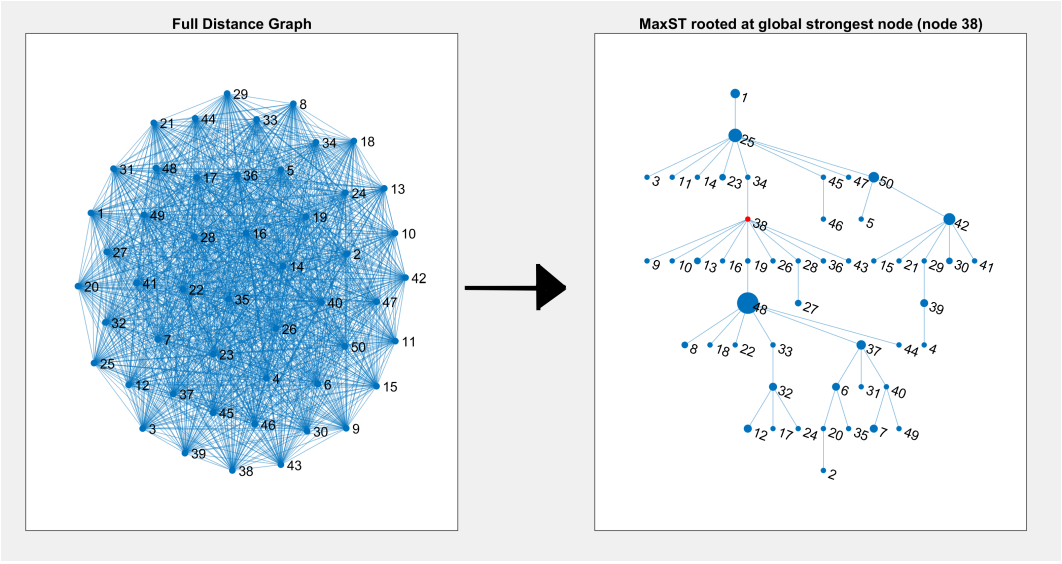
These strength values are then normalized to create the final portfolio weights (w_i), ensuring they sum to one:

$$w_i = \frac{s_i^{\text{MaxST}}}{\sum_{k \in V} s_k^{\text{MaxST}}}.$$

This allocation scheme assigns a larger capital share to assets that are more central to the portfolio's diversification structured as captured by the Maximum Spanning Tree.

In Figure 3.1, the resulting network structure is visualized. This graph illustrates the final Maximum Spanning Tree, where each node represents an asset. The size of a node is directly proportional to its calculated portfolio weight, and the red node highlights the root from which the tree construction was initiated.

Figure 3.1: **MaxST Networking**



Note: This figure illustrates the structure of the maximum spanning tree (MaxST) network, where the size of the node goes in hand with the weight in the portfolio

3.3 Empirical Application

3.3.1 Dataset and processing

The analyses were conducted on daily realized covariances obtained using the historical data of assets of the following 50 companies for a period ranging from April 1st, 2021 to April 1st, 2025, including Apple Inc., Abbott Laboratories, Accenture, Adobe Inc., Amgen, American Tower, Amazon, American Express, Bank of America, Caterpillar Inc., Comcast, Costco, Salesforce, Cisco, Chevron Corporation, Danaher Corporation, Walt Disney Company (The), GE Aerospace, Alphabet Inc. (Class C), Goldman Sachs, Home Depot (The), IBM, Intuit, Intuitive Surgical, Johnson & Johnson, JPMorgan Chase, Coca-Cola Company (The), Lilly (Eli), Lowe's, Mastercard, McDonald's, Merck & Co., Microsoft, Netflix, Nike, Inc., Nvidia, Oracle Corporation, PepsiCo, Pfizer, Procter & Gamble, Qualcomm, AT&T, TJX Companies, Thermo Fisher Scientific, Texas Instruments, UnitedHealth Group, Union Pacific Corporation, Verizon, Wells Fargo, and Walmart. The predictions of the realized covariances

in the testing sample ($T_{test} = 318$) obtained from AGCRN are used to calculate the correlation matrices from which we compute the distance matrices and obtain the weights from the MaxST, as detailed before.

3.3.2 A survey of competing portfolio optimization methods

To evaluate the performance of our proposed MaxST-AGCRN methodology, we benchmark it against a comprehensive set of established portfolio construction techniques. These competing models span the literature from the foundational principles of modern portfolio theory to recent innovations in machine learning and risk management. This section provides a brief overview of the theoretical underpinnings of each benchmark model used in our empirical analysis.

First, we apply two approaches which are direct descendants of the seminal work of Markowitz (1952), focusing on the trade-off between expected return and variance. The classic mean-variance (MV) framework seeks to find the optimal portfolio weights $\mathbf{w}$ that maximize an investor's utility, which rewards expected return and penalizes risk. The objective is formally stated as:

$$\max_{\mathbf{w}} \quad \mathbf{w}^\top \boldsymbol{\mu} - \frac{\gamma}{2}\mathbf{w}^\top \boldsymbol{\Sigma}\mathbf{w}$$

where $\boldsymbol{\mu}$ is the vector of expected asset returns, $\boldsymbol{\Sigma}$ is the covariance matrix, and γ represents the investor's risk aversion coefficient. While theoretically elegant, its practical application is mined by its high sensitivity to estimation errors in $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

As a direct response to the unreliability of return forecasts, the Global Minimum Variance (GMV) approach disregards expected returns and aims to find the portfolio with the lowest possible risk. The optimization is simplified to:

$$\min_{\mathbf{w}} \quad \mathbf{w}^\top \boldsymbol{\Sigma}\mathbf{w} \quad \text{subject to} \quad \mathbf{w}^\top \mathbf{1} = 1.$$

Its primary advantage is robustness, as it only requires a covariance matrix estimate. However, by ignoring returns, it may lead to portfolios with suboptimal growth potential.

The tangency (or max Sharpe Ratio) approach, instead, identifies the optimal portfolio of risky assets to be mixed with a risk-free asset, thereby maximizing the return per unit of total risk. It is the solution to maximizing the Sharpe Ratio:

$$\max_{\mathbf{w}} \quad \frac{\mathbf{w}^\top \boldsymbol{\mu} - r_f}{\sqrt{\mathbf{w}^\top \boldsymbol{\Sigma} \mathbf{w}}}$$

where r_f is the risk-free rate. While a cornerstone of the Capital Asset Pricing Model (CAPM), it shares the same sensitivity to input estimation errors as the standard MV portfolio.

Moving beyond variance, the Extreme Risk Index (ERI) approach (Chauhan, 2011; Mainik et al., 2015) focuses on mitigating tail risk. It employs principles from Extreme Value Theory (EVT) to model the behavior of returns during market stress, optimizing the portfolio to minimize the probability of extreme losses.

The Black-Litterman model (Black and Litterman, 1991) addresses the issue of unstable return forecasts by starting with a more robust prior: the market-implied equilibrium returns. It then allows an investor to incorporate their own subjective views on asset performance, blending them with the market prior in a Bayesian fashion to produce stable and intuitive portfolio weights.

Within the approaches which uses machine learning to optimize portfolio weights, we find the Hierarchical Risk Parity (HRP) model proposed by López de Prado (2018). This offers a novel approach that does not require matrix inversion and is thus more stable. The process involves three steps: (1) using hierarchical clustering to group assets based on their correlations, (2) reordering the covariance matrix based on this tree structure (quasi-diagonalization), and (3) recursively allocating weights between and within clusters using an inverse-variance approach.

Finally, we compare the MaxST portfolio performance with those from a naive model where capital is equally allocated across all N available assets ($w_i = 1/N$).

3.3.3 Portfolio Metrics

We use the following metrics to compare the portfolio performance:

- The annualized average return provides the expected annual return for portfolio optimization objective functions, serving as the primary return component in mean-variance optimization frameworks. Provided that the portfolio return is:

$$r_{p,t} = \sum_{i=1}^{n} w_{i,t} r_{i,t},$$

  we compute the annualized mean return as follows

$$\bar{\mu}_p = \bar{r}_p \times 252$$

  where $\bar{r}_p$ is the average portfolio returns over the out-of-sample observations;

- The annualized portfolio volatility (σ_p) quantifies the total risk of the portfolio. It is crucial for constructing the efficient frontier and is the denominator in many risk-adjusted performance measures and is computed as:

$$\sigma_p = \sqrt{252 \cdot \frac{1}{T} \sum_{t=1}^{T} (r_{t,p} - \bar{r}_p)^2};$$

- The Sharpe Ratio (SR) is a widely used risk-adjusted performance metric that quantifies the excess return earned per unit of total risk. It is calculated as:

$$\text{SR} = \frac{\overline{\mu}_p - r_f}{\sigma_p}$$

  where $\overline{\mu}_p$ is the annualized mean return, r_f is the risk-free rate, and σ_p is the annualized standard deviation of returns (Sharpe, 1966). The

Sharpe ratio is important as it allows investors to compare portfolios with different risk levels on a standardized basis;

- The Information Ratio (IR) measures the consistency of a portfolio's excess returns over a benchmark, adjusted for the volatility of those excess returns. A higher IR indicates a better ability to generate excess returns consistently;

- The Tracking Error (TE) measures the standard deviation of the difference between the portfolio's returns and the benchmark's returns. A lower TE indicates that the portfolio's performance closely follows the benchmark;

- The β_p measures the portfolio's volatility in relation to the overall market. A beta greater than 1 indicates the portfolio is more volatile than the market, while a beta less than 1 means it is less volatile;

- The Treynor Ratio (TR) is similar to the Sharpe Ratio but measures the excess return earned per unit of systematic risk, as measured by the portfolio's beta. It is calculated as

$$\text{TR} = \frac{\bar{\mu}_p - r_f}{\beta_p},$$

where β_p is the portfolio's beta;

- Jensen's Alpha (α) represents the portfolio's excess return relative to its expected return as predicted by the Capital Asset Pricing Model (CAPM). A positive alpha suggests the portfolio has outperformed the market on a risk-adjusted basis;

- The Maximum Drawdown (MDD) quantifies the largest observed loss from a portfolio's peak to its subsequent trough, providing a direct measure of its downside risk. It is computed as:

$$\text{MDD} = \max_{i \leq t} \left(\max_{j \leq i} W_j - W_i \right)$$

where W_i is the cumulative wealth at time i (Chekhlov et al., 2005). This metric is crucial for assessing resilience during adverse periods;

- The Diversification Ratio (DR) measures the diversification benefit achieved through portfolio construction. It is defined as the ratio of the portfolio's weighted average volatility to its overall volatility:

$$DR = \frac{\sum w_i \sigma_i}{\sigma_p}.$$

A higher ratio indicates greater diversification benefits;

- The average portfolio turnover measures the frequency and magnitude of changes in portfolio allocations, calculated as:

$$\tau_p = \frac{1}{T} \sum_{t=1}^{T} TO_t$$

where

$$TO_t = \sum_{i=1}^{N} |w_{i,t} - w_{i,t-1}|$$

where $w_{i,t}$ is the weight of asset i at time t (Grinold and Kahn, 2000). A high turnover can indicate active adaptation to new information but also leads to increased transaction costs. An increase in turnover suggests more frequent or larger re-balancing, possibly reflecting market instability or model sensitivity;

- The Value at Risk (VaR) estimates the maximum expected portfolio loss over a specified time horizon at a given confidence level (*e.g.*, 95% or 99%). It is computed from the empirical distribution of returns;

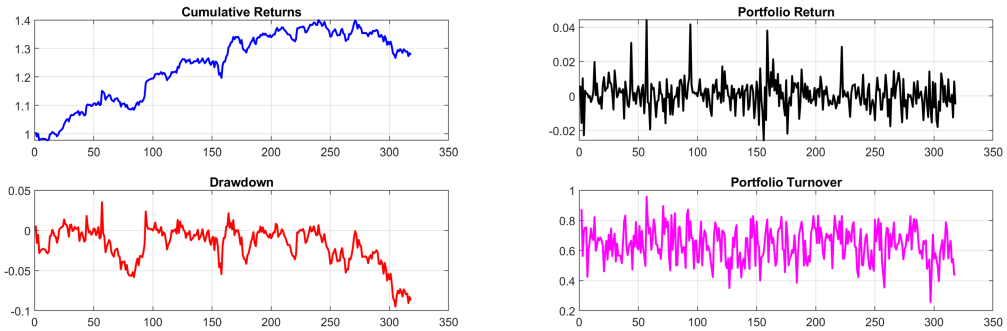
- The Expected Shortfall (ES), also known as Conditional VaR, measures the average loss in the worst-case scenarios that fall beyond the VaR threshold. It is considered a more comprehensive measure of tail risk than VaR.

3.3.4 Results

MaxST performance

In Figure 3.2, we can find a graph of some performance metrics of the MaxST-AGCRN optimized portfolios through the overall time frames. Portfolios from MaxST-AGCRN model showed a strong overall performance with cumulative returns reaching approximately 140% over the evaluation period, which indicates a robust capital appreciation. The portfolio exhibits well-controlled risk characteristics, with maximum drawdowns remaining mostly below -0.1 (10%), though there is a notable decline towards the end of the sample period where drawdowns get down to around -0.1. The return volatility appears well-managed, as shown in the portfolio returns panel by the fact that daily fluctuations are contained within a reasonable range of approximately $\pm 4\%$, without persistent clusters of extreme negative returns. These results somehow suggest that the MaxST-AGCRN approach effectively captures and responds to the evolving correlation structure. Portfolio turnover patterns are generally ranging between 0.4 and 0.8, with some periods of higher re-balancing activity reaching nearly 1.0. Showing the dynamic adaptability of MaxST-AGCRN to changing market relationships while avoiding excessive trading costs. Overall, MaxST-AGCRN model delivers good risk-adjusted performance through effective diversification, stable risk control, and adaptive portfolio re-balancing that responds to evolving asset inter-dependencies.

Figure 3.2: **MaxST portfolio Metrics**



Predictions comparison and discussion

Table 3.1 presents the performance metrics of the compared methods. In this table, MaxST refers to Maximum Spanning Tree model, MV denotes a mean-variance portfolio, LDP refers to the López de Prado (2018) portfolio, BL identifies a Black-Litterman portfolio. The results indicate a clear trade-off in the AGCRN-MaxST strategy between generating high returns and managing risk.

The MaxST-AGCRN portfolios exhibited the second highest annualized mean return ($\bar{\mu}_p = 0.000809$) after the tangency model, but outperforming traditional models like Mean-Variance. However, this high return is accompanied with the second highest volatility. A key strength of the MaxST-AGCRN framework is the fact that it has achieved the highest Sharpe Ratio (SR= 0.091757) indicating that it provides the best return per unit of total risk among all models tested; MaxST-AGCRN have also achieved the highest Sortino Ratio (equal to 0.147984), which is a critical metric for risk-averse investors as it demonstrates superior performance when considering only downside volatility. Another key strength is the fact that it achieved the highest Information Ratio (IR= 0.071559) indicating MaxST-AGCRN's ability to generate the best excess returns relative to tracking error.

In addition, MaxST-AGCRN ranks second in both Treynor Ratio (TR= 0.000804) and Jensen's Alpha ($\alpha = 0.000376$), showing consistent risk-adjusted out-performance. Furthermore, it exhibits a low Maximum drawdown (MDD= -0.095023), second only to the tangency portfolio, underscoring its resilience during market stress and its value for capital preservation. However, this resilience comes at the cost of high trading activity. The portfolio's turnover (τ_p) is strongly the highest, suggesting frequent rebalancing. This highlights a critical trade-off between adaptability to changing market conditions and the potential for significant transaction costs.

Its market sensitivity is also elevated, with ($\beta_p = 1.006745$) indicating closer-to-market exposure and less systematic risk dampening than some com-

petitors. In addition, the model's tail risk metrics (VaR and ES) are higher than those of the most conservative strategies. This indicates that the model's pursuit of high returns exposes it to greater potential losses in extreme market scenarios compared to risk-focused models like GMV or tangency. Finally, the Diversification Ratio (DR= 1.682143) is the second lowest, suggesting less dispersion of idiosyncratic risk than the competing models. In summary, the MaxST-AGCRN portfolio delivers strong risk-adjusted returns, controlled drawdowns, and stable tail-risk profiles while maintaining a reasonable diversification.

Table 3.1: **Performance metrics comparison across portfolio models**

Metric	MaxST	MV	GMV	Tangency	ERI	BL	HRP	Naive
$\overline{\mu}_p$	**0.000809**	0.000344	0.000347	**0.000940**	0.000329	0.000441	0.000424	0.000430
σ_p	0.008816	**0.006188**	**0.006310**	0.012415	0.006591	0.006920	0.006737	0.006999
SR	**0.091757**	0.055527	0.054911	**0.075680**	0.049983	0.063720	0.062975	0.061407
Sortino	**0.147984**	0.079911	0.079022	**0.108556**	0.071850	0.090097	0.088973	0.086394
IR	**0.071559**	-0.01489	-0.01408	**0.051950**	-0.01630	0.017051	-0.00529	0.000000
TE	0.084105	0.091910	0.093919	0.155760	0.097761	**0.010376**	0.016569	**0.000000**
TR	**0.000804**	0.000626	0.000631	**0.000862**	0.000592	0.000448	0.000446	0.000430
β_p	1.006745	0.548708	**0.549177**	1.090421	**0.556269**	0.984390	0.952142	1.000000
α	**0.000376**	0.000108	0.000110	**0.000471**	0.000090	0.000018	0.000015	0.000000
MDD	-0.095023	-0.08871	-0.09034	-0.15516	**-0.07694**	**-0.08631**	-0.08888	-0.09262
DR	1.682143	1.768485	1.765274	1.664695	1.932990	1.708189	**2.366031**	**2.319720**
τ_p	0.640256	0.035634	0.016552	0.059863	0.025790	**0.002091**	0.207843	**0.000000**
VaR_{95}	0.013339	**0.010095**	**0.010575**	0.022364	0.011475	0.012201	0.011801	0.012244
VaR_{99}	0.020304	**0.015734**	**0.016030**	0.031541	0.016838	0.018063	0.018398	0.019036
ES_{95}	0.017188	**0.013570**	**0.013839**	0.027997	0.014510	0.015892	0.015526	0.016138
ES_{99}	0.023561	**0.019516**	**0.019809**	0.034933	0.019911	0.024108	0.023870	0.024954

The best two results for each metric are in bold; the best is also underlined. MaxST denotes the portfolios optimized using Maximum spanning tree, MV denotes a mean-variance portfolio, HRP refers to the López de Prado (2018) portfolio, BL identifies a Black-Litterman portfolio. In the rows, $\overline{\mu}$ and σ represent the annualized portfolio return and portfolio volatility; SR is the Sharpe ratio, IR is the information ratio, TE is the tracking error, TR is the Treynor ratio, β is the CAPM beta, α is the Jensen's alpha, MDD is the maximum drawdown, DR is the diversification ratio, while VaR and ES are the Value-at-risk and the expected shortfall.

3.4 Additional Analysis

In this section, we were interested in answering this question: what will change if we decide to change the methodological order and use the AGCRN model to predict distance matrices themselves instead of RCOV matrices?

This comes from the idea that in order to avoid noise amplification from non-linear transformations of realized covariance (RCOV) forecasts (Bollerslev et al., 2016; Mantegna, 1999), we directly predict the distance matrices derived from realized correlation matrices in the forecasting process instead of forecasting the RCOV matrices themselves. The benefit lies in preserving metric relationships through explicit modelling of geometric constraints like ultrametric inequalities (Tumminello et al., 2005), which moderates structural distortions caused by logarithmic mappings of error-prone covariance estimates (Bollerslev et al., 2016). By operating directly on distance representations, we circumvent the dimensional complexity of full covariance matrices (Pozzi et al., 2013), enabling more stable learning of asset network dynamics while maintaining interpretability for clustering and portfolio optimization tasks (Hartkopf and Lillo, 2020). This approach also avoids time-varying reliability issues in RCOV estimates (Bollerslev et al., 2016), as distance matrices inherently filter noise through their metric properties (Tumminello et al., 2005). This leads to enhanced forecast consistency for downstream applications, *i.e.* models tailored to distance manifolds capture essential co-movement structures more parsimoniously (Pozzi et al., 2013), reducing error propagation in maximum-variance portfolio construction (Hartkopf and Lillo, 2020). Several empirical applications show that this strategy improves risk-adjusted returns by aligning forecasts with optimization geometry (Hartkopf and Lillo, 2020), rather than deferring non-linear transformations to post-prediction stages where residual errors compound (Mantegna, 1999).

In order to do this, we start the analysis by using the same dataset as in section 2.3, a sample which includes the daily realized covariances obtained using the hourly historical data for a period ranging from April 1st, 2021, to

April 1st, 2025, of assets of the following 50 companies.

The realized covariances are then used to calculate the correlation matrices from which we compute the distance matrices. As node embeddings for the AGCRN, we include lagged returns for each asset.

To construct the node feature matrices, we leverage the symmetry of distance matrices. However, as the diagonal elements of the distance matrices are zeros, we use only the triangular non-diagonal elements, which capture the distances between each pair of assets. To enhance processing, these variables have been standardized to facilitate the training and processing.

When creating balance between the feature matrices (node-level attributes) with the embedding matrices (pairwise asset relationships), we design the embeddings specifically for asset pairs, where for each unique combination of two assets (i, j), we compute a corresponding embedding vector. Afterwards, min-max scaling has been applied on the embedding matrices to ensure stability during the training process.

The data is then converted into graphical representations where upper off-diagonal elements of the distance matrices serve as nodes, and lagged volumes and returns are used as node embeddings. The dataset was split into 70% training observations ($T_{train} = 742$) and 30% testing observations ($T_{test} = 318$). Since we applied a rolling-window prediction, we always used a window of 742 observations to make one-step-ahead predictions. The distance matrices predicted using AGCRN models following the methodology 2.2.2 are then used for the implementation of the MaxST model to get the optimized portfolio for each time step.

As mentioned in section 2.3.3, the AGCRN implementation uses five parameters. For the prediction of distance matrices, the input features are set to 1, the output features are set to 1, and the hidden dimensions are determined through experimentation. We also use 1. The parameter K is set to 1 for distance matrices to prioritize immediate neighbor interactions over higher-order correlations. The embedding dimension is set to 2 for distance matrices to en-

sure sharper discrimination between proximity levels. The model is optimized using the Adam optimizer, which is run for up to 100 epochs and has an early stopping strategy to prevent overfitting and the learning rate is set to 1×10^{-3}.

In this section, the overall portfolio performance of the predicted distance matrices will first be compared with the other models, like a mean-variance (MV) portfolio, a portfolio built using the approach proposed by López de Prado (2018) (LDP), a naive portfolio (*i.e.*, with equal weights), an Extreme Risk Index (ERI) portfolio (Chauhan, 2011; Mainik et al., 2015), a GMV portfolio, a tangency portfolio and a portfolio constructed à la Black and Litterman (1991), as provided in Table 3.2. Then, we will compare the portfolio performance of the predicted distance matrices with the performance of the portfolio of the predicted RCOV matrices to understand if there is a real difference 3.3.

In Table 3.2, we can see that the DisMaxST portfolios exhibited the second highest annualized mean return ($\bar{\mu}_p = 0.000664$) after the tangency model, but outperformed classical models like mean-variance. However, this high return is accompanied by the second-highest volatility. A key strength of the DisMaxST framework is the fact that it has achieved the highest Sharpe Ratio (SR= 0.079799), indicating that it provides the best return per unit of total risk among all models tested; DisMaxST has also achieved the highest Sortino Ratio (equal to 0.123750), which is a critical metric for risk-averse investors as it demonstrates superior performance when considering only downside volatility. Another key strength is the fact that it achieved the highest Information Ratio (IR= 0.052345), indicating DisMaxST's ability to generate the best excess returns relative to tracking error.

In addition, DisMaxST ranks second in both the Treynor Ratio (TR= 0.000662) and Jensen's Alpha ($\alpha = 0.000233$), showing consistent risk-adjusted outperformance. Furthermore, it exhibits a low maximum drawdown (MDD= -0.113930), second only to the tangency portfolio, underscoring its resilience during market stress and its value for capital preservation. However, this resilience comes at the cost of high trading activity. The portfolio's turnover

(τ_p) is strongly the highest, suggesting frequent rebalancing. This highlights a critical trade-off between adaptability to changing market conditions and the potential for significant transaction costs.

Its market sensitivity is also elevated, with ($\beta_p = 1.002261$) indicating closer-to-market exposure and less systematic risk dampening than some competitors. In addition, the model's tail risk metrics (VaR and ES) are higher than those of the most conservative strategies. This indicates that the model's pursuit of high returns exposes it to greater potential losses in extreme market scenarios compared to risk-focused models like GMV or tangency. Finally, the Diversification Ratio (DR= 1.873733) comes in the middle of all the other models, suggesting less dispersion of idiosyncratic risk than the competing models. In summary, the DisMaxST portfolio delivers strong risk-adjusted returns, controlled drawdowns, and stable tail-risk profiles while maintaining a reasonable diversification.

Table 3.2: **Distance Performance metrics comparison across portfolio models**

Metric	DisMaxST	MV	GMV	Tangency	ERI	BL	HRP	Naive
$\overline{\mu}_p$	**0.000664**	0.000344	0.000347	**0.000940**	0.000329	0.000441	0.000424	0.000430
σ_p	0.008318	**0.006188**	**0.006310**	0.012415	0.006591	0.006920	0.006737	0.006999
SR	**0.079799**	0.055527	0.054911	**0.075680**	0.049983	0.063720	0.062975	0.061407
Sortino	**0.123750**	0.079911	0.079022	**0.108556**	0.071850	0.090097	0.088973	0.086394
IR	**0.052345**	-0.01489	-0.01408	**0.051950**	-0.01630	0.017051	-0.00529	0.000000
TE	0.070956	0.091910	0.093919	0.155760	0.097761	**0.010376**	0.016569	**0.000000**
TR	**0.000662**	0.000626	0.000631	**0.000862**	0.000592	0.000448	0.000446	0.000430
β_p	1.002261	0.548708	**0.549177**	1.090421	**0.556269**	0.984390	0.952142	1.000000
α	**0.000233**	0.000108	0.000110	**0.000471**	0.000090	0.000018	0.000015	0.000000
MDD	-0.113930	-0.08871	-0.09034	-0.15516	**-0.07694**	**-0.08631**	-0.08888	-0.09262
DR	1.873733	1.768485	1.765274	1.664695	1.932990	1.708189	**2.366031**	**2.319720**
τ_p	0.730431	0.035634	0.016552	0.059863	0.025790	**0.002091**	0.207843	**0.000000**
VaR_{95}	0.013316	**0.010095**	**0.010575**	0.022364	0.011475	0.012201	0.011801	0.012244
VaR_{99}	0.018247	**0.015734**	**0.016030**	0.031541	0.016838	0.018063	0.018398	0.019036
ES_{95}	0.016391	**0.013570**	**0.013839**	0.027997	0.014510	0.015892	0.015526	0.016138
ES_{99}	0.021665	**0.019516**	**0.019809**	0.034933	0.019911	0.024108	0.023870	0.024954

The best two results for each metric are in bold; the best is also underlined. DisMaxST denotes the portfolios optimized using Maximum spanning tree of the predicted distance matrices, MV denotes a mean-variance portfolio, HRP refers to the López de Prado (2018) portfolio, BL identifies a Black-Litterman portfolio. In the rows, $\overline{\mu}$ and σ represent the annualized portfolio return and portfolio volatility; SR is the Sharpe ratio, IR is the information ratio, TE is the tracking error, TR is the Treynor ratio, β is the CAPM beta, α is the Jensen's alpha, MDD is the maximum drawdown, DR is the diversification ratio, while VaR and ES are the Value-at-risk and the expected shortfall.

If we concentrate on the separate position among other models of both the performance of portfolio performance of the predicted RCOV matrices 3.1 and the portfolio performance of the predicted distance matrices 3.2, we can see that they have almost the same position; however, when considering them together as in 3.3, we can notice that while RcovMaxST slightly outperforms DisMaxST on classical risk-adjusted return metrics, DisMaxST offers advantages in diversification, achieving (DR= 1.873733) in comparison with (DR= 1.682143), tracking error, and certain tail risk metrics.

The trade-off between the models falls into the hands of the investor and his priorities: while RCOV-based matrices may be better for better average portfolio returns, the distance-based portfolios provide better risk mitigation and operational efficiency.

Table 3.3: **Performance metrics comparison across Distance verses RCOV-based portfolios**

Metric	DisMaxST	RcovMaxST
$\overline{\mu}_p$	0.000664	**0.000809**
σ_p	**0.008318**	0.008816
SR	0.079799	**0.091757**
Sortino	0.123750	**0.147984**
IR	0.052345	**0.071559**
TE	**0.070956**	0.084105
TR	0.000662	**0.000804**
β_p	**1.002261**	1.006745
α	0.000233	**0.000376**
MDD	**-0.113930**	-0.095023
DR	**1.873733**	1.682143
τ_p	0.730431	**0.640256**
VaR_{95}	**0.013316**	0.013339
VaR_{99}	**0.018247**	0.020304
ES_{95}	**0.016391**	0.017188
ES_{99}	**0.021665**	0.023561

The best two results for each metric are in bold; the best is also underlined. DisMaxST denotes the portfolios optimized using Maximum spanning tree of the predicted distance matrices and RcovMaxST denotes the portfolios optimized using Maximum spanning tree of the predicted RCOV matrices. In the rows, $\overline{\mu}$ and σ represent the annualized portfolio return and portfolio volatility; SR is the Sharpe ratio, IR is the information ratio, TE is the tracking error, TR is the Treynor ratio, β is the CAPM beta, α is the Jensen's alpha, MDD is the maximum drawdown, DR is the diversification ratio, while VaR and ES are the Value-at-risk and the expected shortfall.

3.5 Conclusion

In this chapter, we introduced and tested a novel portfolio optimization framework that integrates a predictive deep learning model, the Adaptive Graph Convolutional Recurrent Network (AGCRN), with a network-based filtering technique, the Maximum Spanning Tree (MaxST). Using the predicted realized covariances to compute the distance matrices to construct MaxST-based portfolios, our approach aims to balance adaptability with the structural parsimony needed to overcome the limitations of traditional optimization methods.

Our empirical application on a portfolio of 50 assets from the S&P 500 index demonstrates that the MaxST-AGCRN strategy offers a distinct and compelling risk-return profile. The model successfully generates the highest Sharpe Ratio, Sortino Ratio and Information Ratio among all competing strategies, including traditional Markowitz and GMV portfolios. Furthermore, it was among the highest in regards to mean returns, Treynor Ratio, and Jensen's alpha and one of the lowest in maximum drawdown.

However, our analysis also highlights critical trade-offs. The high returns and adaptability of the model are accompanied by a turnover rate that is substantially higher than all other strategies, which could lead to significant transaction costs in a real-world implementation. Moreover, the portfolio exhibited higher tail risk (VaR and ES) than the most conservative models, indicating that its aggressive return-seeking nature exposes it to greater potential losses during extreme market events.

In the other side of the coin, the trade-off between the risk and return have been addressed further in the additional analysis, where even though we got high level of risk-return adjustment as even if the mean return, Sharpe Ratio, Sortino Ratio and Information Ratio have slightly decreased; the levels of volatility tail risk (VaR and ES) have also decreased. This did not stop here; this was also accompanied with a good rate of increase in the diversification ratio.

In conclusion, the MaxST-AGCRN framework represents a significant step

toward integrating advanced neural networks with network theory for financial applications. It successfully addresses the "Markowitz enigma" by reducing input sensitivity and enhancing portfolio resilience without sacrificing the return potential. The primary limitations, *i.e.* high turnover and elevated tail risk, present clear avenues for future research. Potential enhancements could include the incorporation of transaction cost constraints directly into the optimization process or the integration of tail risk-aware objective functions to create a more balanced and practical investment strategy.

Keeping in mind the fact that the additional analysis has proven that there is indeed a space for improvement, for future research we will be focused on further improvements and better fitting of parameters to get the best ways for portfolio optimization.

Chapter 4

Spatiotemporal Covariance Neural Networks for Forecasting Returns

4.1 Introduction

Financial investments are important for allocating capital to productive opportunities, enhancing long-run economic growth, and improving household welfare through intertemporal consumption smoothing and wealth accumulation (Fedorowicz and Łopatka, 2022). They provide vehicles for investors to seek returns that outperform market benchmarks through active management, the value of which can be quantitatively assessed to ensure management fees are justified by performance (Riccetti, 2012). In the preceding chapters, we have addressed some challenges related to the forecasting of the tool used in measuring the riskiness of securities, volatility, which plays a big role in the investment related decisions. In this chapter, we will be focusing on the twin pillars of risk, *i.e.* return forecasting.

The first models used for forecasting returns date back to the late 1920s and were based on linear regressions and simple univariate autoregressive (AR) models, which were first formalised by Yule (1927). Their application to finan-

cial returns emerged in the 1960s with empirical tests of return autocorrelation in Fama's paper (Fama, 1965), representing one of the first empirical applications to asset returns. Since then, this topic has been addressed using many different models. In 1988, multivariate GARCH models were introduced to capture time-varying covariances and volatilities across multiple assets (Bollerslev et al., 1988). A few years later, Fama and French (1992) introduced the three-factor model, which enhanced the capital asset pricing model (CAPM) by including market, size, and value factors to better forecast returns. More recently, the Dynamic Conditional Correlation (DCC) GARCH model (Engle, 2002) was introduced to allow correlations between asset returns to vary over time in a computationally tractable way. In the last decade, machine learning models, like random forest, support vector machines, and neural networks have been applied to capture nonlinear patterns and dependencies that classical models may miss (Gu et al., 2020; Fischer and Krauss, 2018; Elnabarawy et al., 2020).

In the context of machine learning, deep learning architectures like Dense networks, Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Convolutional Neural Networks (CNN), and hybrid CNN-RNN models have pushed the boundaries in unraveling complex nonlinear interactions in multivariate time series (Hortúa et al., 2025), which discusses advanced forecasting techniques that extend beyond single-variable models to create joint models incorporating a diverse array of predictors, pushing financial return prediction beyond traditional econometric models. Techniques like correlation screening and walk-forward simulations that bring together many variables, including price levels, trading volumes, volatility indices, macroeconomic indicators, and sector-specific Exchange Traded Funds (ETFs), achieve much higher forecasting accuracy than relying on a single variable alone (Hull and Qiao, 2017) significantly outperforming univariate benchmarks in both out-of-sample accuracy and directional accuracy. The implementation of a dynamic, percentile-based thresholding mechanism for binary classification proves ad-

vantageous, particularly in adapting to various market conditions. Among the different models tested, deep learning architectures utilizing multivariate inputs, especially recurrent networks and hybrid convolutional-recurrent approaches, exhibit consistent generalization across different time horizons, including daily, weekly, and monthly forecasts. Conversely, while ensemble tree methods demonstrate a strong fit for in-sample data, they often fit well to in-sample data but struggle with out-of-sample data, particularly at lower frequencies. This comprehensive methodology exemplifies the advantages of multivariate forecasting techniques, underscoring their applicability in complex market environments (Hortúa et al., 2025).

Despite this progress, predicting returns remains complicated by several deep challenges. The most foundational is the Efficient Market Hypothesis (EMH), which posits that asset prices fully reflect all available information. Consequently, it should be impossible to consistently achieve returns above a "fair" rate without assuming additional risk. In its strictest form, this implies that prices follow a random walk, making returns unpredictable and serially uncorrelated. The presence of any return predictability would indicate a degree of market inefficiency or, alternatively, could suggest time-varying risk premia, which challenges the simple random walk assumption while still aligning with a broader concept of time-varying fair returns. Therefore, strong return predictability is fundamentally at odds with the core principles of the EMH (Bollerslev, 2019).

Another challenge is that market microstructure effects, investor behavior, and changing economic regimes give rise to complex nonlinear dynamics and temporal variability. Ignoring these nonlinear and time-varying features can lead to biased and inefficient forecasts, especially during periods of financial stress or market regime changes (Marconi, 2025; Guerrón-Quintana et al., 2023).

Furthermore, in portfolio management, it is essential to make accurate and stable predictions across a set of assets. Achieving consistent accuracy is chal-

lenging due to fluctuating levels of noise, liquidity, and asset-specific dynamics. Stability issues arise when models produce highly volatile forecasts for some assets, which can undermine portfolio management decisions. Ensuring stable forecast performance across the entire asset universe is a critical challenge, particularly in high-dimensional settings with limited training data (Laurent et al., 2010). Capturing complex, evolving correlations among assets is also essential for understanding risk contagion and co-movement in market dynamics. Correctly modeling these joint spatiotemporal structures is complicated by high dimensionality, non-stationarity, and noise in financial data. Traditional forecasting models often treat time series independently or fail to jointly model spatial correlations and temporal dynamics, resulting in suboptimal forecasts (Gundersen et al., 2024; Cao et al., 2020).

In the recent literature, these problems have been addressed by spatiotemporal graph neural networks (GNNs) and hybrid deep learning architectures that combine graph convolutional networks with recurrent units (Kumar et al., 2024). These models exploit the inherent graph structure of financial markets to learn both cross-asset (spatial) and temporal dependencies, showing improved performance over classical approaches by capturing inter-asset dynamics and time-varying relationships (Cao et al., 2020; Yin et al., 2023; Kumar et al., 2024). However, challenges remain in handling non-stationarity, managing noisy covariance estimates for graph construction, and developing efficient representations in high-dimensional settings (Wang, 2024).

To address these remaining gaps, this chapter introduces the SpatioTemporal coVariance Neural Network (STVNN), a model designed for multivariate time series forecasting (Cavallo et al., 2024). The STVNN model operates by translating time series covariance matrices into graphs to capture how spatial dependencies evolve. Simultaneously, it studies the temporal dynamics within each variable, modeling these two dimensions jointly. This is achieved through a novel SpatioTemporal Covariance Filter (STVF), which performs convolutions over both the evolving covariance matrices and temporal windows of

observations. Specifically, STVNN operates in an online learning framework, allowing it to adapt to non-stationarity by continuously updating its parameters as new data arrives.

It is important to clarify that our primary objective is to forecast asset returns. We use realized covariance matrices not as the prediction target, but as a dynamic, time-varying graph structure to better model the evolving inter-asset relationships. Our hypothesis is that by explicitly modeling these cross-asset linkages, we can generate more accurate return forecasts.

We are motivated to use the STVNN model for three primary reasons. First, it is capable of explicitly learning complex, nonlinear, and time-varying dependencies that traditional models often fail to capture. Second, its online framework is designed to handle noisy data and adapt to new information continuously. Third, by modeling covariance matrices as evolving graphs, STVNN captures both the spatial correlations between assets and their temporal behaviors, overcoming the limitations of models that cannot jointly represent these patterns (Cavallo et al., 2024; Kanungo, 2025). This integrated approach helps overcome the EMH hurdle by exploiting transient inefficiencies that arise from market microstructure effects and short-lived structural breaks. Rather than promising unconditional arbitrage, STVNN focuses on conditional predictability, enhancing forecasting performance in real-world financial markets.

In this chapter, we apply the STVNN model to forecast the daily returns of the same 50 assets composing the S&P 500 index used in the previous chapters. Here, we compare the performance of the STVNN with the alternative methods proven to be effective in the prediction of multivariate financial returns. In an expanding window exercise, we show that the proposed methodology has good predictive accuracy with respect to competing models, including also deep learning methods.

This chapter is organized as follows. In Section 4.2, we introduce the methodology, while Section 4.3 presents the results of the STVNN's application. Section 4.4 concludes.

4.2 Methodology

4.2.1 Spatiotemporal Covariance Neural Network (STVNN)

This chapter builds upon the Spatiotemporal Covariance Neural Network (STVNN) architecture, a model designed for multivariate time series forecasting (Cavallo et al., 2024). The model consists of three core components: (i) a dynamic graph construction mechanism; (ii) SpatioTemporal coVariance Filters (STVFs) that perform joint convolutions over the graph and time; iii) a multi-layer architecture for hierarchical feature extraction.

The objective of the network is to learn a function $\Phi(\mathbf{x}_{T:t}, \mathbf{C}; \mathbf{h})$ that maps a temporal window of inputs, $\mathbf{x}_{T:t} = \{\mathbf{x}_{t-T+1}, \ldots, \mathbf{x}_t\}$, and a time-varying covariance extimate, $\hat{\mathbf{C}}_t$, to a target output $\mathbf{y} = \mathbf{x}_{t+\tau}$. A central element of the STVNN is its use of a time-varying covariance matrix, $\hat{\mathbf{C}}_t$, as the graph shift operator. Using $\mathbf{C}$ helps reducing the number of parameters and computational complexity, since the cross-asset dependency structure is imposed and not learned. The key methodological choice is how to estimate this unobservable time-varying covariance matrix.

In their original work, Cavallo et al. (2024) propose a recursive online update rule for the covariance matrix, allowing it to adapt using low-frequency (*e.g.*, daily) data. Their approach is motivated by the limitations of classical methods, like Principal Component Analysis (PCA) for streaming time series data. In PCA, the features are computed by projecting the data onto the eigenvectors of the covariance matrix, $\hat{\mathbf{C}}_t = \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}\hat{\mathbf{V}}^\top$, where $\hat{\mathbf{V}}$ contains the eigenvectors and $\hat{\mathbf{\Lambda}}$ the diagonal matrix of eigenvalues. However, this approach can be unstable when data is limited or when eigenvalues are close in magnitude. To overcome this, STVNN builds upon graph signal processing principles, treating the multivariate time series as a signal on a graph. The original paper proposes estimating the covariance matrix via a recursive online update rule

$$\hat{\mathbf{C}}_{t+1} = \xi_t \hat{\mathbf{C}}_t + \zeta_t (\mathbf{x}_{t+1} - \hat{\boldsymbol{\mu}}_t)(\mathbf{x}_{t+1} - \hat{\boldsymbol{\mu}}_t)^\top \tag{4.1}$$

where the mean is also updated recursively, $\hat{\boldsymbol{\mu}}_{t+1} = \alpha\hat{\boldsymbol{\mu}}_t + \beta_t \mathbf{x}_{t+1}$. The scalar coefficients ($\alpha_t, \beta_t, \xi_t, \zeta_t \in [0, 1]$) can be set for stationary,

$$\alpha_t = \frac{t}{t+1}, \quad \beta_t = \zeta_t = \frac{1}{t+1}, \quad \xi_t = \frac{t-1}{t},$$

and non-stationary assumptions

$$\alpha_t = \xi_t = 1 - \gamma, \quad \beta_t = \zeta_t = \gamma,$$

with scalar $\gamma \in [0, 1]$ regulating the contributions of recent and past data. This method is essentially an exponentially weighted moving average (EWMA) of the covariance.

While effective for general streaming data, the original update rule may be too slow to capture the rapid shift common in financial markets. Our implementation deviates significantly by replacing this estimate with a more robust measure, *i.e.* the realized covariance (RCOV) matrix (Andersen et al., 2001, 2003). The RCOV leverages high-frequency intra-day data to compute a more accurate and responsive daily estimate of the covariance structure

$$\mathbf{RCOV}_t = \sum_{j=1}^{m} \mathbf{r}_{j,t} \mathbf{r}_{j,t}^{\top},$$

where m is the number of intra-daily observations and $\mathbf{r}_{j,t}$ is the vector of returns at day t and period j (see the second chapter for further details). By grounding the STVNN's graph structure in this high-frequency measure, we provide the model with a more accurate estimate of the market's dependency structure, which we hypothesize will lead to more accurate return forecasts.

The core processing unit of the STVNN is the SpatioTemporal coVariance Filter (STVF), which performs a joint convolution over space (the asset graph) and time (the lookback window). As defined by Cavallo et al. (2024), the input-output relation of an STVF with temporal memory T and spatial memory K is

$$\mathbf{z}_t = \mathcal{H}(\hat{\mathbf{C}}_t, \mathbf{h}_t, \mathbf{x}_{T:t}) = \sum_{t'=0}^{T-1} \sum_{k=0}^{K} h_{kt'} \hat{\mathbf{C}}_t^k \mathbf{x}_{t-t'},$$

where $h_{kt'}$ are learnable filter parameters and $\hat{\mathbf{C}}_t$ is the time-varying covariance which we estimate through the realized covariance matrix. This operation

aggregates information across space, since powers of the covariance matrix diffuse information across the asset graph up to K hops away, and time, since the outer sum aggregates these spatially-filtered features over the last T time steps. An illustration of the STVF is given in Figure 4.1.

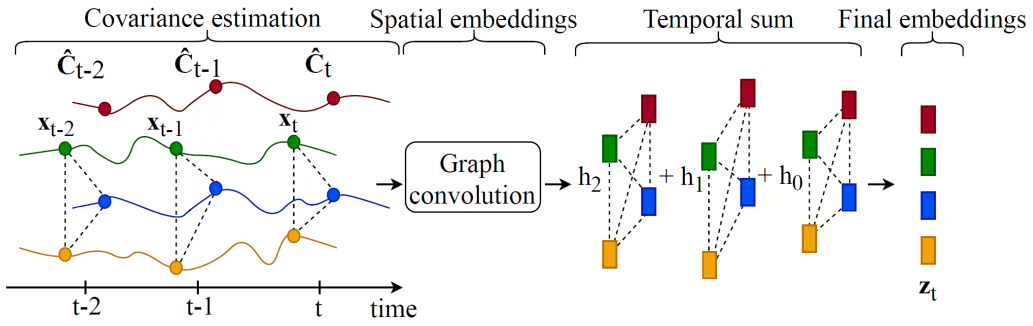


Figure 4.1: **Spatiotemporal covariance filter. Source: Cavallo et al. (2024)**

The full STVNN model is a layered architecture where each layer l nests an STVF within a pointwise non-linear acivation function $\sigma(\cdot)$. For layers $l = 1, \ldots, L$, the update is given by:

$$\mathbf{z}_t^l = \sigma\left(\mathcal{H}^l(\hat{\mathbf{C}}_t, \mathbf{h}_t, \mathbf{z}_{T:t}^{l-1})\right) = \sigma\left(\sum_{t'=0}^{T-1}\sum_{k=0}^{K} h_{kt'}^l \hat{\mathbf{C}}_t^k \mathbf{z}_{t-t'}^{l-1}\right), \quad l = 1, \ldots, L,$$

with the input $\mathbf{z}_{t-t'}^0 = \mathbf{x}_{t-t'}$ to the first layer being the raw return data, $\mathbf{z}_{t-t'}^0 = \mathbf{x}_{t-t'}$. The output of the final layer, $\mathbf{z}_t^L$, contains the hierarchical spatiotemporal representations used for forecasting

$$\mathbf{z}_t^L := \Phi(\mathbf{x}_{T:t}, \hat{\mathbf{C}}_t; \mathbf{h}).$$

The parameters comprise those of all filters in all layers,

$$\mathbf{h} = \{h_{kt'}^l\}_{k,t',l},$$

and are of order $\mathcal{O}((K+1)TL)$.

To further increase representational capacity, each layer can employ parallel filter banks. With F_{in} input features and F_{out} output features, the operation

becomes:

$$\mathbf{z}_t^{l,f} = \sigma\left(\sum_{g=1}^{F_{\text{in}}}\sum_{t'=0}^{T-1}\sum_{k=0}^{K} h_{kt'}^{l,f,g}\hat{\mathbf{C}}_t^k \mathbf{z}_{t-t'}^{l-1,g}\right), f = 1,\ldots,F_{\text{out}}. \tag{4.2}$$

This structure is analogous to using multiple channels in a traditional convolutional neural network (CNN).

To adapt to non-stationarity, the model's parameters $\mathbf{h}_t$, are updated continuously as new data arrives. Following each forecast, the parameters are adjusted via stochastic gradient descent:

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

where $\eta > 0$ is the learning rate, ∇_t is the gradient at time t, and $\mathcal{L}$ is the loss function.

For a single layer, the computational complexity is $\mathcal{O}(N^2TKF_{\text{in}}F_{\text{out}})$, where N is the number of nodes (variables), T is the temporal memory size, K the spatial filter order, and F_{in}, F_{out} are the number of input and output features per layer. If the realized covariance matrix is sparsified, this can be reduced to $\mathcal{O}(|E|\,T\,K\,F_{\text{in}}F_{\text{out}})$, where $|E|$ is the number of non-zero correlations. This linear scaling with temporal size T makes the STVNN significantly more efficient than methods like temporal PCA, whose projection costs scale quadratically with T as $\mathcal{O}(N^2T^2)$.

4.3 Empirical application

This section details the empirical analysis conducted to evaluate the forecasting performance of the STVNN for asset returns. We begin by describing the dataset and the experimental design, followed by a presentation of the benchmark models used for comparison. Finally, we present and discuss the results through a multi-faceted evaluation framework, encompassing statistical accuracy, tests for superior predictive ability, and economic significance via a portfolio optimization exercise.

4.3.1 Dataset

The analysis is conducted on the same dataset used in the previous chapters with daily returns and daily realized covariance matrices for 50 companies that are components of the S&P 500 index, as reported in Table 2.1. The data, sourced from Bloomberg, spans from April 1st, 2021, to April 1st, 2025, covering a total of 1,060 trading days. A key input to our model, the daily realized covariance matrices, were computed using 5-minute intraday returns.

The core of our experiment is to forecast one-day-ahead asset returns. In the STVNN framework, the sequence of past daily returns serves as the time-varying node features, while the daily realized covariance matrix functions as the time-varying graph adjacency matrix, $\hat{\mathbf{C}}_t$. To ensure numerical stability and facilitate the model training process, the daily return series for each asset were standardized using a 60-day rolling window.

4.3.2 Forecasting setting

The dataset is split into a training set and a testing set, with the first 70% of observations (742 trading days) used for initial model training and the remaining 30% (318 trading days) used for out-of-sample evaluation. We employ an expanding-window forecasting approach to simulate a realistic trading environment. After the initial training, the estimation window is expanded by one day at a time, and the models are re-trained to generate a sequence of 318 one-day-ahead forecasts. This ensures that all predictions are made using only information that would have been available at that time. The STVNN architecture was configured with two STVF layers, a temporal memory of five days, and a spatial memory of two hops, trained using the Adam optimizer over 100 epochs with an early stopping mechanism.

To evaluate the STVNN's performance, we compare it against a comprehensive set of established benchmark models. This includes deep learning architectures as a Long Short-Term Memory (LSTM) network and a vanilla Elman Recurrent Neural Network (RNN), both trained using 5 lags of the re-

turns as features. These models serve as benchmarks for time-series forecasting but operate on each asset's return series independently, thereby ignoring the cross-sectional dependency structure that STVNN explicitly models. We also include traditional econometric models, namely a spatial GARCH model to account for spatial dependencies and conditional heteroskedasticity, and a 5-lag Vector Autoregressive (VAR) model as a robust linear multivariate baseline. In fact, even if it does not directly model the covariances, our approach takes some interesting characteristics from the GARCH literature. We here exploits the cross-asset dependencies with the purpose to predict the returns, which is mainly what is done in multivariate GARCHs. If one would also model the covariances, a further research question may involve using the methodology presented in the first chapter of this thesis. Finally, a random walk (RW) model provides a naive benchmark representing the "no predictability" baseline implied by the Efficient Market Hypothesis.

The performance of all models is assessed through a multi-type framework. Statistical accuracy is measured using Mean Squared Error (MSE) and the Euclidean norm, while forecast calibration is evaluated using the Mincer-Zarnowitz R^2 (Mincer and Zarnowitz, 1969). Both the MSE and the R^2 are computed as averages among the N assets. To ensure our comparisons are statistically meaningful, we employ two state-of-the-art tests for superior predictive accuracy, *i.e.* the pairwise Giacomini-White (GW) test (Giacomini and White, 2006) and the Model Confidence Set (MCS) procedure (Hansen et al., 2011), which identifies a "Set of Superior Models" (SSM) from the candidate pool. Finally, to determine if statistical superiority translates into tangible economic value, we conduct a portfolio optimization exercise where return forecasts are used within a Markowitz mean-variance strategy.

4.3.3 Results

The out-of-sample forecasting statistical accuracy is first evaluated using the Giacomini-White test (Giacomini and White, 2006), whose results are pre-

sented in Table 4.1.

The pairwise Giacomini-White (GW) test provides strong statistical evidence for the superior conditional predictive ability of the STVNN model. The results, consistent across both the Mean Squared Error (Panel A) and Euclidean distance (Panel B) loss functions, reveal that STVNN significantly outperforms every competing model. This superiority is particularly notable when compared to its deep learning counterparts, LSTM and RNN. For instance, the test statistic against LSTM is a highly significant -2.287 under Euclidean. This suggests that STVNN's explicit modeling of cross-asset dependencies through the realized covariance graph provides a distinct forecasting advantage over sophisticated models that only capture temporal patterns.

Furthermore, the most striking results are the extremely large and highly significant negative test statistics against the Random Walk (RW) and VAR models. The test statistic of -14.93 against the RW under MSE provides compelling evidence against the weak-form efficient market hypothesis for our sample and highlights the substantial economic value embedded in the STVNN's forecasts. The consistent outperformance against all benchmarks underscores the robustness and effectiveness of the proposed spatiotemporal architecture.

Model Confidence Set (MCS)

To simultaneously evaluate the predictive accuracy of the model proposed here, we also performed the Model Confidence Set (MCS) (Hansen et al., 2011). The MCS is a statistical procedure designed to identify a subset of models that are statistically indistinguishable in terms of performance, given a predefined significance level (α). The MCS iteratively eliminates poorly performing models based on loss functions until only the superior models remain.

In the provided analysis referred to in Table 4.2, the MCS was applied with $\alpha = 0.1$ and 10,000 bootstrap replicates to evaluate the predictive accuracy of STVNN against other models, including LSTM, RNN, GARCH, RW, and

Table 4.1: **Returns forecasting results with Giacomini-White test**

Panel A: MSE

	LSTM	RNN	GARCH	RW	VAR
STVNN	-1.469	-4.795 ***	-2.612 ***	-14.93 ***	-2.190 **
LSTM		-4.639 ***	-0.7057	-14.93 ***	-1.728 *
RNN			4.568 ***	-14.33 ***	4.189 ***
GARCH				-14.88 ***	-0.968
RW					14.87 ***

Panel B: Euclidean

	LSTM	RNN	GARCH	RW	VAR
STVNN	-2.287 **	-5.194 ***	-2.179 **	-21.81 ***	-2.813 ***
LSTM		-4.990 ***	0.074	-21.79 ***	-2.266 **
RNN			4.972 ***	-20.56 ***	4.028 ***
GARCH				-21.70 ***	-2.079 **
RW					21.59 ***

Note: LSTM represents the predictions from Long Short-Term Memory model, RNN represents the predictions from Recurrent Neural Networks model, GARCH denotes for predictions from a Spatial GARCH, RW denotes a prediction from a random walk, and VAR represents the predictions through a VAR(5) on the elements of financial returns. The test statistics are computed using a loss difference $L_1 - L_2$, where L_1 is the loss of the model in row and L_2 of that in column. *, ** and *** denote respectively a rejection at a 10%, 5% and 1% significance level.

VAR. Across both the MSE and Euclidean loss functions, the STVNN is the only model retained in the Superior Set of Models, achieving a p_{MCS} of 1.000. This indicates that at a 10% significance level, the forecast performance of the STVNN is statistically superior to all alternatives, which are iteratively eliminated from the set ($p_{\text{MCS}} < 0.001$). This result provides powerful evidence that the STVNN is not just a competitive model, but is statistically distinguishable as the best-performing forecaster within our considered set.

Finally, the Mincer-Zarnowitz regression provides a nuanced perspective on the models' forecast calibration. The results, shown in the last column of Table 4.2, indicate that the GARCH model achieves the highest goodness-of-fit with an R^2 of 0.008601, followed by the RW model at 0.0034737 and then RNN at 0.0030491.

It is crucial to interpret these R^2 values within the context of financial return forecasting. The inherent randomness and efficiency of markets make returns notoriously difficult to predict, and consequently, explanatory power is expected to be very low. In this high-noise environment, a model that can consistently explain even a small fraction of the variance—such as the 0.86% captured by the GARCH model—can be of significant statistical and economic value.

While the deep learning models, including STVNN, did not lead on this specific metric, they remain highly competitive. This suggests that while the STVNN is statistically superior at minimizing overall prediction error (as shown by the GW and MCS tests), traditional econometric models like GARCH and VAR may have a slight edge in capturing a simple linear relationship between their forecasts and the true outcomes. This highlights a common trade-off between minimizing error and achieving linear calibration. Finally, the overwhelming evidence from the primary loss metrics and superiority tests solidifies STVNN's position as the leading candidate for forecasting, particularly when model robustness and overall accuracy are prioritized.

Table 4.2: **Average loss functions and MCS p-values**

	MSE		**Euclidean**		
Model	**Avg. loss**	p_{MCS}	**Avg. loss**	p_{MCS}	R^2
STVNN	0.0002724	**1.000**	0.10947	**1.000**	0.0018494
LSTM	0.0002727	**1.000**	0.10957	**1.000**	0.0012027
RNN	0.0002847	<0.001	0.11198	<0.001	0.0030491
GARCH	0.0002730	**0.9972**	0.10957	**1.000**	**0.008601**
VAR	0.0002737	**0.7256**	0.10993	**0.3853**	0.0019842
RW	0.0005569	<0.001	0.15837	<0.001	0.0034737

Note: Underlined values denote the lowest loss functions for each setting. The STVNN, LSTM, and RNN models use a single lag of historical data for input. The models are trained with standard MSE loss functions. The p-value refers to the probability of being included in the SSM over $10,000$ block bootstrap replicates (in bold values with a p_{MCS} greater than 0.75).

Economic significance and portfolio performance

In order to further understand the accuracy of the STVNN predictions, we have performed portfolio optimization using Markowitz with predicted returns as expected returns and the realized covariances as covariance matrix, we then have compared the results with those from a naive portfolio and the methods presented in Chapter 3. Specifically, we compared the STVNN with the Maximum Spanning Tree (MaxST) approach, a mean-variance (MV) portfolio, the portfolio using the approach of López de Prado (2018) (LDP), a naive portfolio, an Extreme Risk Index (ERI) portfolio (Chauhan, 2011; Mainik et al., 2015), a GMV portfolio, a tangency portfolio and a portfolio constructed following the approach of Black and Litterman (1991) (BL).

As in the previous chapter, to evaluate the performance of the portfolios, we used the following metrics:

- The annualized average return provides the expected annual return for portfolio optimization objective functions, serving as the primary return

component in mean-variance optimization frameworks. Provided that the portfolio return is:

$$r_{p,t} = \sum_{i=1}^{n} w_{i,t} r_{i,t},$$

we compute the annualized mean return as follows

$$\bar{\mu}_p = \bar{r}_p \times 252$$

where $\bar{r}_p$ is the average portfolio returns over the out-of-sample observations;

- The annualized portfolio volatility (σ_p) quantifies the total risk of the portfolio. It is crucial for constructing the efficient frontier and is the denominator in many risk-adjusted performance measures and is computed as:

$$\sigma_p = \sqrt{252 \cdot \frac{1}{T} \sum_{t=1}^{T} (r_{t,p} - \bar{r}_p)^2};$$

- The Sharpe Ratio (SR) is a widely used risk-adjusted performance metric that quantifies the excess return earned per unit of total risk. It is calculated as:

$$\text{SR} = \frac{\overline{\mu}_p - r_f}{\sigma_p}$$

where $\overline{\mu}_p$ is the annualized mean return, r_f is the risk-free rate, and σ_p is the annualized standard deviation of returns (Sharpe, 1966). The Sharpe ratio is important as it allows investors to compare portfolios with different risk levels on a standardized basis;

- The Information Ratio (IR) measures the consistency of a portfolio's excess returns over a benchmark, adjusted for the volatility of those excess returns. A higher IR indicates a better ability to generate excess returns consistently;

- The Tracking Error (TE) measures the standard deviation of the difference between the portfolio's returns and the benchmark's returns. A

lower TE indicates that the portfolio's performance closely follows the benchmark;

- The β_p measures the portfolio's volatility in relation to the overall market. A beta greater than 1 indicates the portfolio is more volatile than the market, while a beta less than 1 means it is less volatile;

- The Treynor Ratio (TR) is similar to the Sharpe Ratio but measures the excess return earned per unit of systematic risk, as measured by the portfolio's beta. It is calculated as

$$\text{TR} = \frac{\bar{\mu}_p - r_f}{\beta_p},$$

where β_p is the portfolio's beta;

- Jensen's Alpha (α) represents the portfolio's excess return relative to its expected return as predicted by the Capital Asset Pricing Model (CAPM). A positive alpha suggests the portfolio has outperformed the market on a risk-adjusted basis;

- The Maximum Drawdown (MDD) quantifies the largest observed loss from a portfolio's peak to its subsequent trough, providing a direct measure of its downside risk. It is computed as:

$$\text{MDD} = \max_{i \leq t} \left(\max_{j \leq i} W_j - W_i \right)$$

where W_i is the cumulative wealth at time i (Chekhlov et al., 2005). This metric is crucial for assessing resilience during adverse periods;

- The Diversification Ratio (DR) measures the diversification benefit achieved through portfolio construction. It is defined as the ratio of the portfolio's weighted average volatility to its overall volatility:

$$DR = \frac{\sum w_i \sigma_i}{\sigma_p}.$$

A higher ratio indicates greater diversification benefits;

- The average portfolio turnover measures the frequency and magnitude of changes in portfolio allocations, calculated as:

$$\tau_p = \frac{1}{T} \sum_{t=1}^{T} TO_t$$

where

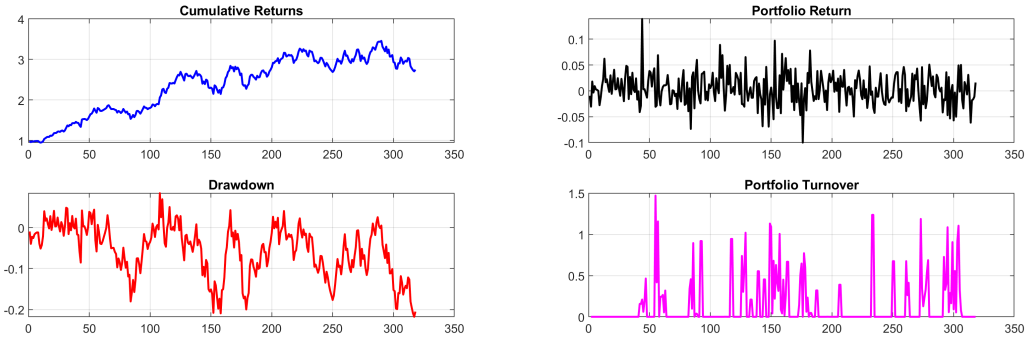
$$TO_t = \sum_{i=1}^{N} |w_{i,t} - w_{i,t-1}|$$

where $w_{i,t}$ is the weight of asset i at time t (Grinold and Kahn, 2000). A high turnover can indicate active adaptation to new information but also leads to increased transaction costs. An increase in turnover suggests more frequent or larger re-balancing, possibly reflecting market instability or model sensitivity;

- The Value at Risk (VaR) estimates the maximum expected portfolio loss over a specified time horizon at a given confidence level (*e.g.*, 95% or 99%). It is computed from the empirical distribution of returns;

- The Expected Shortfall (ES), also known as Conditional VaR, measures the average loss in the worst-case scenarios that fall beyond the VaR threshold. It is considered a more comprehensive measure of tail risk than VaR.

Figure 4.2: **STVNN portfolio Metrics**



In Figure 4.2, we can find a graph of some performance metrics of the STVNN optimized portfolios through the overall time frames. Portfolios from

STVNN model showed a strong overall performance with cumulative returns reaching approximately 350% over the evaluation period, which indicates a robust capital appreciation. The portfolio demonstrates moderate risk control, with most drawdowns remaining below 0.15 (15%), although there is a significant decline toward in the second half of the period where drawdowns approach 0.2 (20%). The return volatility appears well-managed, as shown in the portfolio returns panel by the fact that daily fluctuations daily changes are contained within roughly ±10%, and there are no sustained periods of extreme negative returns. These results somehow suggest that the STVNN approach effectively captures and responds to the evolving correlation structure. Portfolio turnover patterns are generally ranging between 0.0 and 1.0, with some periods of higher re-balancing activity reaching nearly 1.5. Showing the dynamic adaptability of STVNN to changing market relationships while avoiding excessive trading costs. Overall, STVNN model delivers stronger return generation paired with materially higher downside and market exposure.

Table 4.3 presents the performance metrics of the compared methods. The STVNN framework clearly emphasizes return maximization but at the expense of higher risk and trading activity. In fact, the STVNN portfolio achieved the highest annualized mean return ($\overline{\mu}_p = 0.003632$), more than triple that of the tangency model and substantially above traditional approaches. However, this elevated return is accompanied by the highest volatility ($\sigma_p = 0.030038$). A key strength of the STVNN framework is that it secures the highest Sharpe Ratio (SR= 0.120907), indicating the best return per unit of total risk among all models; it also attains the highest Sortino Ratio (Sortino= 0.194590), which is crucial for risk-averse investors as it measures performance considering only downside volatility. Furthermore, STVNN achieves the highest Information Ratio (IR= 0.116250), demonstrating superior excess returns relative to its tracking error.

In addition, STVNN leads in market-timing and systematic exposure metrics, posting the highest Treynor Ratio (TR= 0.001846) and Jensen's Alpha

($\alpha = 0.002786$), reflecting exceptional returns per unit of systematic risk. Its CAPM beta ($\beta_p = 1.966906$) reveals strong market sensitivity and close alignment with broad-market movements. Nevertheless, these strengths are tempered by significant drawbacks: STVNN exhibits the worst Maximum Drawdown (MDD= -0.217656), indicating the largest peak-to-trough decline and greatest vulnerability during market downturns. The portfolio's turnover ($\tau_p = 0.138647$) is highest as well, suggesting frequent rebalancing and potential for elevated transaction costs.

Keeping this in mind, STVNN exhibits the highest tail-risk metrics—$\text{VaR}_{95} = 0.042330$, $\text{VaR}_{99} = 0.068771$, $\text{ES}_{95} = 0.057469$, and $\text{ES}_{99} = 0.081031$, highlighting greater potential losses under extreme market scenarios. Its diversification ratio (DR= 1.029555) is the lowest among all models, suggesting concentrated exposures and limited dispersion of idiosyncratic risk. In summary, while STVNN excels in maximizing returns and risk-adjusted metrics, its elevated risk profile, pronounced drawdowns, frequent trading, and concentrated positions may deter highly risk-averse investors.

Table 4.3: **Performance metrics comparison of STVNN with competing models**

Metric	STVNN	MV	HRP	Naive	ERI	GMV	Tangency	BL
$\overline{\mu}_p$	**0.003632**	0.000344	0.000424	0.000430	0.000329	0.000347	**0.000940**	0.000441
σ_p	0.030038	**0.006188**	0.006737	0.006999	0.006591	**0.006310**	0.012415	0.006920
SR	**0.120907**	0.055527	0.062975	0.061407	0.049983	0.054911	**0.075680**	0.063720
Sortino	**0.194590**	0.079911	0.088973	0.086394	0.071850	0.079022	**0.108556**	0.090097
IR	**0.116259**	-0.01489	-0.00529	0.000000	-0.01630	-0.01408	**0.051950**	0.017051
TE	0.437209	0.091910	0.016569	**0.000000**	0.097761	0.093919	0.155760	**0.010376**
TR	**0.001846**	0.000626	0.000446	0.000430	0.000592	0.000631	**0.000862**	0.000448
β_p	1.966906	0.548708	0.952142	1.000000	**0.556269**	**0.549177**	1.090421	0.984390
α	**0.002786**	0.000108	0.000015	0.000000	0.000090	0.000110	**0.000471**	0.000018
MDD	**-0.217656**	-0.08871	-0.08888	-0.09262	-0.07694	-0.09034	**-0.15516**	-0.08631
DR	1.029555	1.768485	**2.366031**	**2.319720**	1.932990	1.765274	1.664695	1.708189
τ_p	0.138647	0.035634	0.207843	**0.000000**	0.025790	0.016552	0.059863	**0.002091**
VaR_{95}	0.042330	**0.010095**	0.011801	0.012244	0.011475	**0.010575**	0.022364	0.012201
VaR_{99}	0.068771	**0.015734**	0.018398	0.019036	0.016838	**0.016030**	0.031541	0.018063
ES_{95}	0.057469	**0.013570**	0.015526	0.016138	0.014510	**0.013839**	0.027997	0.015892
ES_{99}	0.081031	**0.019516**	0.023870	0.024954	0.019911	**0.019809**	0.034933	0.024108

The best two results for each metric are in bold; the best is also underlined. STVNN denotes Spatiotemporal Covariance Neural Networks, MV denotes a mean-variance portfolio, HRP refers to the López de Prado (2018) portfolio, BL identifies a Black-Litterman portfolio. In the rows, $\overline{\mu}$ and σ represent the annualized portfolio return and portfolio volatility; SR is the Sharpe ratio, IR is the information ratio, TE is the tracking error, TR is the Treynor ratio, β is the CAPM beta, α is the Jensen's alpha, MDD is the maximum drawdown, DR is the diversification ratio, while VaR and ES are the Value-at-risk and the expected shortfall.

As is clear from 4.4, STVNN beats both DisMaxST and RcovMaxST in annualized mean return ($\overline{\mu}_p$), sharpe ratio (SR), Sortino ratio, Information ratio (IR), and alpha (α). You get the highest risk-adjusted returns and superior excess performance with STVNN.

However, the magnificent return performance has a price, and it is even more clear that this price is paid by having the highest volatility (σ_p) and deeper maximum drawdown (MDD). This is translated by saying that you take more risk and face larger losses when you choose STVNN.

DisMaxST and RcovMaxST both offer lower volatility and more stable drawdown, making them better options if you want to limit risk. DisMaxST gives you the largest Diversification Ratio (DR), so if your aim is to have the best portfolio diversification, do so by directly predicting distances. also helps you cut your tracking error (TE).

STVNN is the lowest in the case of portfolio turnover (τ_p), which means you trade more aggressively and get higher market exposure.

However, the last choice is taken to the investor's choice; choose STVNN if you want maximum risk-adjusted returns and don't mind higher risk. Pick DisMaxST for better diversification and risk control. RcovMaxST gives you a balanced mix between return and risk.

4.4 Conclusion

The third chapter of this thesis studies the possibility of predicting financial returns using Spatiotemporal Covariance Neural Networks (STVNN), which leverages both spatial and temporal dependencies directly by acting on the principal components in each layer using a graph convolutional principle inspired by PCA and offers a robust framework that outperforms traditional models as well as deep learning models when comparing average distance between predicted and real data points. The STVNN's ability to process the covariance matrices as dynamic graph structures and temporal dependencies

Table 4.4: **Performance metrics STVNN VS Distance VS Rcov based portfolios**

Metric	STVNN	DisMaxST	RcovMaxST
$\overline{\mu}_p$	**0.003632**	0.000664	0.000809
σ_p	0.030038	**0.008318**	0.008816
SR	**0.120907**	0.079799	0.091757
Sortino	**0.194590**	0.123750	0.147984
IR	**0.116259**	0.052345	0.071559
TE	0.437209	**0.070956**	0.084105
TR	**0.001846**	0.000662	0.000804
β_p	1.966906	**1.002261**	1.006745
α	**0.002786**	0.000233	0.000376
MDD	**-0.217656**	-0.113930	-0.095023
DR	1.029555	**1.873733**	1.682143
τ_p	**0.138647**	0.730431	0.640256
VaR_{95}	0.042330	**0.013316**	0.013339
VaR_{99}	0.068771	**0.018247**	0.020304
ES_{95}	0.057469	**0.016391**	0.017188
ES_{99}	0.081031	**0.021665**	0.023561

The best two results for each metric are in bold; the best is also underlined.STVNN denotes Spatiotemporal Covariance Neural Networks, DisMaxST denotes the portfolios optimized using Maximum spanning tree of the predicted distance matrices and RcovMaxST denotes the portfolios optimized using Maximum spanning tree of the predicted RCOV matrices. In the rows, $\overline{\mu}$ and σ represent the annualized portfolio return and portfolio volatility; SR is the Sharpe ratio, IR is the information ratio, TE is the tracking error, TR is the Treynor ratio, β is the CAPM beta, α is the Jensen's alpha, MDD is the maximum drawdown, DR is the diversification ratio, while VaR and ES are the Value-at-risk and the expected shortfall.

within financial return series through the use of the Spatiotemporal Covariance Filters (STVF) increases how quickly it adapts to distribution shifts, makes it particularly well-suited for addressing nonlinear and time-varying dependencies, handling noisy data and capturing the complex dynamics of financial markets.

With this structure STVNN partially overcomes the limits imposed by the Efficient Market Hypothesis (EMH) by exploiting transient inefficiencies that arise due to market microstructure effects, short-lived momentum, and structural breaks. The empirical results from this study confirm its superior point-by-point predictive accuracy, as evidenced by the lower mean squared errors and Euclidean norm compared to other models. Additionally, statistical tests such as the Giacomini-White test confirm that STVNN outperforms both traditional models like GARCH, RW and VAR, and advanced neural network models like RNN and LSTM. This is further supported by the Model Confidence Set (MCS) analysis underscoring STVNN's reliability across different evaluation metrics.

In summary, STVNN's novelty lies in its graph convolution approach on covariance matrices with temporal memory, stable online learning mechanism, and superior performance on streaming data compared to all existing PCA-based and spatiotemporal models globally, making it particularly well-suited for financial return forecasting because it can robustly capture and adapt to the complex dependencies and dynamics in financial multivariate time series.

STVNN has further proved its ability to predict returns by delivering the strongest return profile and top risk-adjusted performance. When compared to other models, both models that are designed and studied in this thesis, or classical Markowitz, Global Minimum Variance (GMV), and other machine learning-driven portfolio methods. STVNN posts the highest scores in annualized return, Sharpe Ratio, Sortino Ratio, Information Ratio, Treynor Ratio, and Jensen's alpha, confirming superior excess-return generation and timing relative to systematic risk. However, it achieves this by taking more risk and

trading more aggressively than other methods; however it is somehow understandable, as this model was leveraged for forecasting returns, not risk. This edge comes with the highest volatility, the deepest maximum drawdown, elevated tail risk (VaR and ES), and a beta near two, leading to the greatest turnover. This raises transaction costs and increases sensitivity to market drawdowns.

For our future research, STVNN is still a new model giving us a lot of space to explore changes like deeper architectural variations and multiscale temporal dependencies that will allow us to get more accurate predictions. A further direction would foresee checking how STVNN performs under extreme market conditions and outliers, which is critical to improving its robustness in real-world applications. Also, we will further study how we can use STVNN in a way that can adjust its risk profile when considering portfolio optimization while keeping this amazing return performance.

Chapter 5

Conclusions

Portfolio analysis is the ground on which asset management is constructed. This thesis explores new ways to strengthen this foundation by focusing on its most critical concepts: risk and return.

This thesis was written with the primary objective of addressing the fundamental challenges in forecasting both realized covariance matrices (as a measure of volatility) and asset returns. We employ spatiotemporal neural network models to tackle these challenges and demonstrate their practical value for portfolio optimization and risk management. The core premise is that more accurate forecasts of risk and return lead to more informed investment decisions: reliable risk forecasts capture the dynamic asset relationships that shape diversification, while precise return predictions guide capital allocation.

Traditional statistical methods such as GARCH, VAR, and factor models have long been the backbone of financial econometrics. However, these models often struggle to capture temporal and cross-sectional correlations simultaneously, especially in high-dimensional settings, which limits their ability to capture the full picture and therefore give the most precise results in real financial applications. For example, GARCH models tend to struggle with high-dimensional multivariate settings due to their computational limits. VAR models capture linear interdependencies within time but fail to gain control over nonlinear relationships. Factor models simplify dimensionality but rely

on restrictive assumptions that may fail to capture the dynamic interactions among space and time. These limitations make it harder to deal with financial datasets that are characterized by high-frequency data, hundreds of assets, and complex and changing correlations due to market stress.

Deep learning and neural networks have reshaped financial econometrics by adding flexibility and improving predictive accuracy without the need for explicit model specification. They have pushed financial econometrics a big step forward due to their ability to understand nonlinear patterns directly from data. Advances in architectures such as Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNNs) have also provided the ability to study temporal and multivariate relationships at the same time; however, the effect of the spatial dependencies, like interactions among assets, sectors, and markets, is not fully acknowledged.

This thesis introduces a framework that combines advanced graph neural networks with recurrent architectures, focusing on both temporal and spatial dependencies to provide more accurate predictions. This approach reshapes financial data into a time series of graphical representations, which makes it easier to understand, therefore reducing the effect of the challenges like high dimensionality, non-linear volatility spillovers and the changes in asset correlations, especially with the changes in market regimes.

We build on the idea that the combination of spatiotemporal neural networks with graph theory and network analysis can improve the forecasting accuracy, giving us more information that will help in the decision-making process and increase the portfolio performance.

In summary, our main goal is to demonstrate that spatiotemporal neural networks can significantly improve the prediction accuracy of risk and return, thereby enhancing portfolio optimization and risk management in real-world financial markets by addressing the limitations of traditional econometric models and extending the flexibility of modern deep learning architectures. This research will contribute to quantitative finance and establish a foundation for

future innovations in financial econometrics.

In Chapter 2, we focused on forecasting the realised covariance matrices... (text unchanged)... these challenges include the curse of dimensionality, non-linear dynamics, the spatial dependencies, and the time-evolving nature of asset relationships.

We address this by building upon a spatiotemporal neural network developed for traffic forecasting called Adaptive Graph Convolutional Recurrent Network (AGCRN) that combines Graph Convolutional Networks (GCNs) with Recurrent Neural Networks (RNNs) to deal with the complex nature of spatiotemporal data to identify asset-specific patterns and determine inter-asset dependencies in a dynamic framework via its adaptive modules: Node Adaptive Parameter Learning (NAPL) and Data Adaptive Graph Generation (DAGG), which automatically infer inter-dependencies among data points without needing pre-defined graphs. As NAPL enables the network to capture node-specific temporal patterns by learning unique filters per asset node and DAGG dynamically infers the evolving inter-dependencies among assets by constructing a data-driven adjacency graph rather than relying on a pre-specified static network.

Moreover, to ensure positive definiteness of the forecasted realized covariance matrices, a post-processing eigenvalue adjustment is applied as it guarantees symmetric positive semi-definiteness without embedding complex constraints into the neural network architecture, maintaining simplicity and interpretability.

The predicted realized covariance matrices using AGCRN demonstrated superior forecasting performance compared to models such as Long Short-Term Memory (LSTM), vanilla Recurrent Neural Networks (RNN), random walk (RW), and Vector Autoregressive (VAR). Giacomini-White tests for predictive accuracy revealed that AGCRN consistently outperformed competing models across multiple error metrics, confirming its robust predictive accuracy with clear advantages over traditional models and competitive performance against

other neural networks.

Model Confidence Set (MCS) analysis further validated AGCRN's superiority, with AGCRN achieving perfect inclusion probabilities in Frobenius and Euclidean MCS tests and producing the smallest average losses across all metrics. The final Superior Set of Models included AGCRN, LSTM, and RNN, which were statistically equivalent at the 10% significance level, underscoring AGCRN's consistent and reliable forecasting performance.

In Chapter 3, we have focused on the use of the predictions of the AGCRN model to address the challenges related to portfolio optimization like estimation errors and input sensitivity, high-dimensionality and noisy estimates, inadequate risk measures, and static and simplistic correlation assumptions.

We do that by leveraging the forecasts of the AGCRN model mixed by integrating graph theory and using the Maximum spanning tree model (MaxST). By maximizing the sum of the edge weights (distances), the MaxST identifies the backbone structure of a well-diversified portfolio. It filters the asset network to retain the connections that contribute the most to risk reduction. By transforming predicted covariance matrices into distance matrices and applying the MaxST algorithm, the chapter aims to produce portfolios that deliver superior risk-adjusted returns while mitigating estimation noise and portfolio turnover issues typical of traditional mean-variance and minimum variance portfolio frameworks.

We employ Prim's greedy algorithm initiated from a root node (asset with highest node strength) to compute the MaxST structure efficiently. This produces a sparse, interpretable network encoding the strongest diversification relationships among assets. The resulting MaxST-AGCRN portfolio optimization framework generated the highest Sharpe Ratio, Sortino ratio, and Information Ratio among all competing strategies. However, these portfolios also faced trade-offs, namely elevated tail risk measures, indicating vulnerability during extreme market downturns. The sparse network structure derived from MaxST did, however, significantly reduce portfolio turnover compared to un-

filtered covariance-based strategies, limiting transaction costs.

We have also implemented an additional analysis comparing portfolios constructed from MaxST applied to AGCRN forecasted distances versus realized covariance distances. This confirms the robustness of this integrated approach, with AGCRN forecasts adding predictive power that translates into improved portfolio performance.

In the last Chapter, we shift the focus from forecasting volatility to the direct prediction of asset returns, which is among the most complex problems in financial econometrics, as, according to the Efficient Market Hypothesis (EMH), prices already reflect information available, leaving little systematic predictability; as a result, it should be impossible to achieve returns above the fair rate without assuming risk.

We develop and implement the Spatiotemporal Covariance Neural Network (STVNN), which is a model designed for multivariate time series forecasting as it operates by translating time series covariance matrices into graphs to capture its spatial tendencies at the same time as it studies the temporal dynamics within each asset returns; this is achieved by that performing convolutions over both covariance matrices and the temporal window of returns. STVNN also works in an online framework that continuously updates its parameters to adapt to non-stationarity.

We then connect with the main goal of the thesis, portfolio optimization, by using Markowitz with predicted returns as expected returns and the realized covariances as the covariance matrix. We compare the results to not only the results from classical Markowitz, Global Minimum Variance (GMV), and other machine learning-driven portfolio methods, but also the portfolios built in Chapter 3 using the mixture of AGCRN and MaxST models.

Statistically, the STVNN model delivered stronger return forecasts when compared to other models, including Long Short-Term Memory (LSTM) networks, vanilla Recurrent Neural Networks (RNN), random walk (RW), and Vector Autoregressive (VAR) models. Giacomini–White predictive accuracy

tests and Model Confidence Set (MST) results reinforced STVNN's dominance, showing it consistently outperformed or matched competing models across both Euclidean distances and MSE. These results confirm the value of modeling spatiotemporal dependencies jointly in asset return forecasting.

Economically, STVNN has further proved its ability to predict returns by delivering the strongest risk-adjusted performance. It strongly outperforms all other models in annualized return, Sharpe Ratio, Sortino Ratio, Information Ratio, Treynor Ratio, and Jensen's alpha, confirming superior excess-return generation and timing relative to systematic risk. This comes with a big cost, as STVNN takes more risk and trades more aggressively than other methods, resulting in the highest volatility, deepest maximum drawdown, elevated tail risk, and a beta near two raising transaction costs, and increasing sensitivity to market drawdowns.

The integration of spatiotemporal neural networks with graph theory represents a significant step forward in financial econometrics. In this thesis, we show that mixing models combining deep learning architectures with network science helps capture complex and multidimensional financial patterns that other methods might overlook, shaping how we understand financial markets as dynamic, interconnected systems as well as providing practical tools to improve prediction accuracy, which consequently supports decision-making needed to get closer to achieving investors' goals of investment.

The spatiotemporal frameworks developed in this thesis showed a clear ability to overcome barriers that have long challenged traditional models. Unlike models that struggle with high dimensionality or fail to capture nonlinear dynamics, our approaches - the Adaptive Graph Convolutional Recurrent Network (AGCRN) and the Spatiotemporal Covariance Neural Network (STVNN) - are designed to learn from these complexities. AGCRN learns both node-level temporal patterns and data-driven adjacency structures without predefined correlation matrices, while STVNN integrates evolving covariance structures directly into its training. The resulting predictive accuracy for both risk and re-

turn translated directly into more effective portfolio strategies, demonstrating how these advanced models can address core challenges like estimation error and static correlation assumptions. Despite these promising results, several important challenges and limitations remain to be acknowledged.

First, computational complexity remains a significant hurdle of modeling financial data. Even though the AGCRN model uses the NAPL parameter, which helps in reducing dimensionality, training and running spatiotemporal neural networks still need significant computational power and time, especially with the increase in the number of assets or in the time frame.

Second, the models showed sensitivity to market stress and tail risk; even though both AGCRN and STVNN have a strong predictive power, when the predictions were used in portfolio optimization they both show high value-at-risk (VaR) and expected shortfall (ES), revealing that the models remain vulnerable under market downturns. The STVNN has shown aggressive turnover value, and double the amount of beta exposure increases its reaction to market swings and exposes the portfolios to deeper drawdowns during any type of crisis.

Third, we recognize the practical cost of frequent portfolio rebalancing. STVNN has a data-adaptive nature, which makes portfolios highly responsive to new information; this double-edged sword comes with an increase in transaction cost and execution risk.

Fourth, interpretability remains an issue since graph-based designs aim to make the understanding of asset interactions and relationships easier through the use of adjacency matrices and network topologies, the overall structure of these models is hard to understand, which makes them undesirable, especially in a world where asset managers demand explanations they can trust.

Fifth, the models are sensitive to hyperparameter specification. In fact, spatiotemporal networks include several modeling choices that require careful handling according to the structures of the data, which could be very time-consuming, especially for financial institutions that often need real-time

forecasts.

Finally, we acknowledge the ongoing debate surrounding the Efficient Market Hypothesis (EMH) that reminds us that persistent excess returns usually come with increased risk.

The results of this thesis confirm the potential of combining spatiotemporal neural networks and graph theory in financial econometrics. Building on these promising results, several avenues for future research are evident.

For our future research, we first want to further experience how AGCRN and STVNN models can perform under extreme market conditions, aiming to improve strength when faced by real-world applications by running systematic stress tests and building robust loss functions to handle outliers and market crises.

We also aim to research methods to reduce its computational cost and then to explore deeper how both AGCRN and STVNN can learn across many different timescales, which is how financial markets operate. We also want to study how these models will react when we increase the number of assets.

We also aim to explore creating portfolios from predictions of STVNN using spanning trees. We may also use the mixture of the predictions of STVNN and AGCRN to get portfolios characterized with the highest returns with good risk exposure.

In addition, we plan to extend the use of these models beyond assets to include fixed income, commodities, real estate, or emerging markets. As this might build real-time adaptive portfolio mechanisms that actively optimize trading and allocation as markets change, creating a dynamic and self-updating investment strategy.

Lastly, we aim to increase the transparency and interpretability of graph-based deep learning by developing built-in explainability modules for visualizing network changes, highlighting important features, and clarifying how model decisions evolve over time. In order to make them more acceptable to asset managers.

In conclusion, this thesis demonstrated that integrating spatiotemporal neural networks with graph theory provides a powerful new lens for financial econometrics. By treating markets as dynamic, interconnected systems, the models developed herein moved beyond the limitations of traditional methods to deliver superior predictive accuracy and tangible portfolio performance. While significant challenges in robustness, cost, and interpretability remain, the path forward is clear. This research lays a foundation for a new class of adaptive, network-aware models poised to enhance how investors understand risk and allocate capital in an increasingly complex financial world.

Bibliography

Andersen, T. G., Bollerslev, T., Diebold, F. X., and Labys, P. (2001). The distribution of exchange rate volatility. *Journal of American Statistical Association*, 96:42–55.

Andersen, T. G., Bollerslev, T., Diebold, F. X., and Labys, P. (2003). Modeling and Forecasting Realized Volatility. *Econometrica*, 71:579–625.

Asai, M. and McAleer, M. (2015). Forecasting co-volatilities via factor models with asymmetry and long memory in realized covariance. *Journal of Econometrics*, 189(2):251–262.

Bai, L., Yao, L., Li, C., Wang, X., and Wang, C. (2020). Adaptive graph convolutional recurrent network for traffic forecasting. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 17804–17815.

Barndorff-Nielsen, O. E. and Shephard, N. (2002). Econometric analysis of realized volatility and its use in estimating stochastic volatility models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(2):253–280. Introduces realized volatility measures using high-frequency data, derives asymptotic distributions, and applies to stochastic volatility model estimation.

Barrat, A., Barthelemy, M., Pastor-Satorras, R., and Vespignani, A. (2004). The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences*, 101(11):3747–3752.

Bauer, G. H. and Vorkink, K. (2011). Forecasting multivariate realized stock market volatility. *Journal of Econometrics*, 160(1):93–101.

Belabbes, K., El Hachloufi, M., Guennoun, Z. E. A., and El attar, A. (2024). Uncertain portfolio optimization problems: Systematic review. *Statistics, Optimization & Information Computing*, 13(3):961–976.

Best, M. J. (2010). *Portfolio Optimization*. Chapman and Hall, Boca Raton, FL. University of Waterloo, Ontario, Canada.

Black, F. and Litterman, R. B. (1991). Asset allocation: Combining investor views with market equilibrium. *The Journal of Fixed Income*, 1(2):7–18.

Bollerslev, T. (2019). Market efficiency and return predictability. *Econ. 471/571 Lecture Notes, Duke University*.

Bollerslev, T., Engle, R. F., and Wooldridge, J. M. (1988). A capital asset pricing model with time-varying covariances. *Journal of Political Economy*, 96(1):116–131.

Bollerslev, T. et al. (2016). Forecasting realized covariances and implied correlations. *Journal of Financial Econometrics*.

Brini, A. and Toscano, G. (2024). SpotV2Net: Multivariate intraday spot volatility forecasting via vol-of-vol-informed graph attention networks. *International Journal of Forecasting*.

Brito, D. S., Medeiros, M. C., and Ribeiro, R. M. (2023). Forecasting large realized covariance matrices: The benefits of factor models and shrinkage. *Journal of Financial Econometrics*, 22(3):696–728.

Bucci, A. (2020). Realized volatility forecasting with neural networks. *Journal of Financial Econometrics*, 18(3):502–531.

Bucci, A., Ippoliti, L., and Valentini, P. (2022). Comparing unconstrained parametrization methods for return covariance matrix prediction. *Statistics and Computing*, 32(90).

Bucci, A., Palma, M., and Zhang, C. (2024). Geometric Deep Learning for Realized Covariance Matrix Forecasting. Available at arXiv: https://arxiv.org/abs/2412.09517.

Callot, L. A. F., Kock, A. B., and Medeiros, M. C. (2017). Modeling and Forecasting Large Realized Covariance Matrices and Portfolio Choice. *Journal of Applied Econometrics*, 32(1):140–158.

Cao, D., Wang, Y., Duan, J., Zhang, C., Zhu, X., Huang, C., Tong, Y., Xu, B., Bai, J., Tong, J., and Zhang, Q. (2020). Spectral temporal graph neural network for multivariate time-series forecasting. In *Advances in Neural Information Processing Systems*, volume 33, pages 17766–17778.

Cavallo, A., Sabbaqi, M., and Isufi, E. (2024). Spatiotemporal covariance neural networks. In *Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2024*, volume 14942 of *Lecture Notes in Computer Science*, pages 1834–1854. Springer.

Chauhan, G. S. (2011). Modeling of extreme market risk using extreme value theory (evt): An empirical exposition for indian and global stock indices. *IIM Indore Management Journal*, 3(1):23–35.

Chekhlov, A., Uryasev, S., and Zabarankin, M. (2005). Drawdown measure in portfolio optimization. *International Journal of Theoretical and Applied Finance*, 8(1):13–58.

Chen, L., Xie, L., and Zhang, Y. (2018). Deep learning for financial time series forecasting: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8):3924–3938.

Chen, Q., Ding, R., Mo, X., Li, H., Xie, L., and Yang, J. (2024). An adaptive adjacency matrix-based graph convolutional recurrent network for air quality prediction. *Scientific Reports*, 14(1):4408.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, Cambridge, MA, 3rd edition.

Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7(2):174–196.

Danko, J., Šoltés, V., and Bindzár, T. (2022). Portfolio creation using graph characteristics and testing its performance. *Montenegrin Journal of Economics*, 18(1):7–17.

Di Clemente, A. and Romano, C. (2021). Calibrating and simulating copula functions in financial applications. *Frontiers in Applied Mathematics and Statistics*, 7:642210. Published: 23 March 2021.

Diebold, F. X. and Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business and Economic Statistics*, 13:253–263.

Dryden, I. L., Koloydenko, A., and Zhou, D. (2009). Non-Euclidean Statistics for Covariance Matrices, with Applications to Diffusion Tensor Imaging. *The Annals of Applied Statistics*, 3(3):1102–1123.

Dryden, I. L. and Mardia, K. V. (2016). *Procrustes analysis*, chapter 7, pages 125–173. John Wiley and Sons, Ltd.

Elnabarawy, I., Jiang, W., and II, D. C. W. (2020). Survey of privacy-preserving collaborative filtering. *arXiv preprint arXiv:2003.08343*.

Engle, R. F. (2002). Dynamic conditional correlation: A simple class of multivariate garch models. *Journal of Business & Economic Statistics*, 20(3):339–350.

Fama, E. and French, K. (1993). Common Risk Factors in the Returns on Stocks and Bonds. *Journal of Financial Economics*, 33:3–56.

Fama, E. F. (1965). The behavior of stock-market prices. *Journal of Business*, 38(1):34–105.

Fama, E. F. and French, K. R. (1992). The cross-section of expected stock returns. *The Journal of Finance*, 47(2):427–465.

Fan, J., Li, Y., and Yu, K. (2012). Vast volatility matrix estimation using high-frequency data for portfolio selection. *Journal of the American Statistical Association*, 107(497):412–428.

Fedorowicz, K. and Łopatka, A. (2022). The importance of investment in the development of enterprises in poland in 2009–2018. *Procedia Computer Science*, 207:4171–4179. Open access article, published under CC BY-NC-ND license.

Fischer, T. and Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669.

Gao, J., Huang, S., and Li, Y. (2020). Adaptive models for dynamic financial time series. *Financial Analytics Journal*, 7(2):45–62.

Giacomini, R. and White, H. (2006). Tests of conditional predictive ability. *Econometrica*, 74:1545–1578.

Gribisch, B., Hartkopf, J. P., and Liesenfeld, R. (2020). Factor state–space models for high-dimensional realized covariance matrices of asset returns. *Journal of Empirical Finance*, 55:1–20.

Grinold, R. C. and Kahn, R. N. (2000). *Active Portfolio Management: A Quantitative Approach for Producing Superior Returns and Controlling Risk*. McGraw-Hill, 2nd edition.

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273.

Guerrón-Quintana, P. A., Khazanov, A., and Zhong, M. (2023). Financial and macroeconomic data through the lens of a nonlinear dynamic factor model.

Finance and Economics Discussion Series 2023-027, Board of Governors of the Federal Reserve System.

Gundersen, K. S., Lunde, A., and Oeftiger, B. (2024). Testing for time-varying nonlinear dependence structures: Regime-switching and local gaussian correlation. *Scandinavian Journal of Statistics*, 51(3):1012–1060.

Gunjan, A. and Bhattacharyya, S. (2022). A brief review of portfolio optimization techniques. *Artificial Intelligence Review*, 56(5):3891–3940. Published online: 15 September 2022.

Halbleib-Chiriac, R. and Voev, V. (2011). Modelling and Forecasting Multivariate Realized Volatility. *Journal of Applied Econometrics*, 26:922–947.

Hansen, P. R., Lunde, A., and Nason, J. M. (2011). The Model Confidence Set. *Econometrica*, 79(2):435–497.

Hartkopf, B. and Lillo, P. (2020). Forecasting network structures: evidence from financial returns. *Quantitative Finance*, 20(7):1089–1103.

Hautsch, N., Kyj, L. M., and Oomen, R. C. A. (2012). A blocking and regularization approach to high-dimensional realized covariance estimation. *Journal of Applied Econometrics*, 27(4):625–645.

Higham, N. J. (1988). Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, 103:103–118.

Hortúa, H. J., Mora-Valencia, A., and Vanegas, E. (2025). Stock Return Predictability on S&P 500 ETF: A Deep Learning Approach. *Annals of Data Science*.

Hull, B. and Qiao, X. (2017). A practitioner's defense of return predictability. *Journal of Portfolio Management*, 43(3):60–76.

Jin, X. et al. (2021). *Advanced Methods for Realized Covariance Estimation*. Financial Econometrics Publishing.

Jonnalagadda, J. and Hashemi, M. (2023). Long Lead ENSO Forecast Using an Adaptive Graph Convolutional Recurrent Neural Network.

Kanungo, P. K. (2025). Time series forecasting in financial markets using deep learning models. *World Journal of Advanced Engineering Technology and Sciences*, 15(1):709–719.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Kumar, P. N., Umeorah, N., and Alochukwu, A. (2024). Dynamic graph neural networks for enhanced volatility prediction in financial markets. *Mathematics*, 11(8):1298.

Laurent, S., Rombouts, J. V., and Violante, F. (2010). On the forecasting accuracy of multivariate garch models. *Journal of Business & Economic Statistics*, 28(4):504–522.

Ledoit, O. and Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411.

Lewis, R. (2024). 7 Safe Investments: Major Types and Why They're Low-Risk. *Business Insider*.

Lucchetti, R., Nicolau, M., Palomba, G., and Riccetti, L. (2024). Reconciling tracking error volatility and value-at-risk in active portfolio management: A new frontier. *Computational Economics*. Accepted for publication.

López de Prado, M. (2018). *Advances in Financial Machine Learning*. John Wiley & Sons, Hoboken, New Jersey.

Mainik, G., Mitov, G., and Rüschendorf, L. (2015). Portfolio optimization for heavy-tailed assets: Extreme risk index vs. markowitz. *Journal of Empirical Finance*, 32:115–134.

Mantegna, R. (1999). Hierarchical structure in financial markets. *European Physical Journal B*, 11(1):193–197.

Marconi, B. A. (2025). Time series foundation models for multivariate financial time series forecasting. *arXiv preprint arXiv:2507.07296*.

Markowitz, H. (1952). Portfolio selection. *Journal of Finance*, 7(1):77–91.

Merton, R. C. (1980). On estimating the expected return on the market: An explanatory investigation. *Journal of Financial Economics*, 8:323–361.

Michaud, R. O. (1989). The Markowitz Optimization Enigma: Is 'Optimized' Optimal? *Financial Analysts Journal*, 45(1):31–42.

Millington, T. and Niranjan, M. (2021). Construction of minimum spanning trees from financial returns using rank correlation. *Preprint submitted to Elsevier*.

Mincer, J. A. and Zarnowitz, V. (1969). The Evaluation of Economic Forecasts. In *Economic Forecasts and Expectations: Analysis of Forecasting Behavior and Performance*, NBER Chapters, pages 3–46. National Bureau of Economic Research, Inc.

Newman, M. E. J. (2010). *Networks: An Introduction*. Oxford University Press, Oxford, UK.

Palomar, D. P. (2025). *Portfolio Optimization: Theory and Application*. Cambridge University Press.

Patton, A. J. and Sheppard, K. (2009). *Evaluating Volatility and Correlation Forecasts*. in T. Mikosch, J.P. Kreiß, A. Richard and G.T. Andersen (*eds.*), Handbook of Financial Time Series, Springer Berlin Heidelberg.

Pemmaraju, S. and Skiena, S. (2022). *Computational Discrete Mathematics: Combinatorics and Graph Theory*. Cambridge University Press, Cambridge, UK, 2nd edition.

Pozzi, F. et al. (2013). Spread of risk across financial markets: better to be disconnected? *Scientific Reports*, 3:1665.

Prim, R. C. (1957). Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401.

Protter, P. (2005). *Stochastic Integration and Differential Equations*. Springer, 2nd edition.

Riccetti, L. (2012). Using tracking error volatility to check active management and fee level of investment funds. *Global Business and Economics Review*, 14(2):145–166.

Riccetti, L. (2013). A copula–garch model for macro asset allocation of a portfolio with commodities: An out-of-sample analysis. *Empirical Economics*, 44(3):1315–1336.

Rozemberczki, B., Scherer, P., He, Y., Panagopoulos, G., Riedel, A., Astefanoaei, M., Kiss, O., Beres, F., Lopez, G., Collignon, N., and Sarkar, R. (2021). Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, page 4564–4573.

Salo, A., Doumpos, M., Liesiö, J., and Zopounidis, C. (2023). Fifty years of portfolio optimization. *European Journal of Operational Research*, 318:1–18.

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, 19(3):425–442.

Sharpe, W. F. (1966). Mutual fund performance. *The Journal of Business*, 39(1):119–138.

Silva, V. D., Oliveira, P., and Sousa, P. (2020). Modeling systemic risk with spatiotemporal neural networks: A case study. *Quantitative Finance*, 20(9):1407–1424.

Sims, C. A. (1980). Macroeconomics and reality. *Econometrica*, 48(1):1–48.

Tumminello, M. et al. (2005). Tools for filtering information in complex systems. *PNAS*, 102(30):10421–10426.

Tumminello, M., Lillo, F., and Mantegna, R. N. (2010). Correlation, hierarchies, and networks in financial markets. *Journal of Economic Behavior & Organization*, 75(2):224–252.

Wang, X., Yu, L., and Chen, J. (2020). Modeling spatial dependencies in financial networks. *Quantitative Finance*, 20(5):735–748.

Wang, Y. (2024). *Network Representations for Multivariate Time-series with Applications in Finance*. PhD thesis, University College London. PhD thesis.

West, D. B. (2001). *Introduction to Graph Theory*. Prentice Hall, 2nd edition.

West, K. (1996). Asymptotic inference about predictive ability. *Econometrica*, 64:1067–1084.

Yang, S. and Bristol, T. (1999). Neural network approach to stock price prediction. *Decision Support Systems*, 27(3):199–224.

Yin, J. et al. (2023). Graph neural networks in financial modeling. *Journal of Financial Data Science*.

Yu, H., Luo, S., and Jiang, X. (2019). Spatiotemporal graph convolutional neural networks for financial market prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 31(10):3997–4009.

Yu, Y., Wang, X., and Liu, X. (2018). High-dimensional covariance matrix estimation: Challenges and advances. *Statistical Modelling*, 18(4):301–318.

Yule, G. U. (1927). On a method of investigating periodicities in disturbed series, with special reference to wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A*, 226:267–298.

Zhang, C., Cucuringu, M., and Zohren, S. (2025). Graph-based methods for forecasting realized covariances. *Journal of Financial Econometrics*, 23(2):nbae026.

Zhang, H. (2023). Limitations and critique of modern portfolio theory: A systematic review. *Journal of Financial Economics and Management Research*, 12(3):45–67.

Zhang, L. (2021). Applying spatiotemporal graph neural networks for global market analysis. *Journal of Financial Data Science*, 3(2):22–35.

Zhang, Y. (2018). Deep neural networks in financial modeling: Opportunities and challenges. *Quantitative Finance*, 18(3):409–432.