FINANCIAL ASSET PRICE PREDICTION WITH GRAPH NEURAL NETWORK-BASED TEMPORAL DEEP LEARNING MODELS

A Thesis

by

Yasin Uygun

Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Computer Science

Özyeğin University
February 2024

FINANCIAL ASSET PRICE PREDICTION WITH GRAPH NEURAL NETWORK-BASED TEMPORAL DEEP LEARNING MODELS

Approved by:

Asst. Prof. Dr. Emre Sefer, Advisor
Department of Computer Science
Özyeğin University

Asst. Prof. Dr. Enis Kayış
Department of Industrial Engineering
Özyeğin University

Prof. Dr. Bertan Badur
Department of Management Information Systems
Boğaziçi University

Date Approved: January 5, 2024

To my family

ABSTRACT

Multiple financial asset price changes over time can be characterized by their complex nature and the interdependence between them. More traditional forecasting approaches may overlook the interdependencies among these assets, since they may not fully consider the spatial-temporal dependencies between them. Graph Neural Networks (GNNs) have emerged as a powerful tool for modeling complex relational dependencies across various data types, recently taking more attention for their remarkable performance in areas such as social network analysis and traffic forecasting. The high capability of GNNs is mainly due to their permutation-invariance, and local connectivity. However, their application in asset price prediction remains relatively unexplored.

Here, we investigate GNNs' effectiveness in forecasting multiple financial asset prices jointly, specifically focusing on Foreign Exchange (Forex) and cryptocurrency markets. We employ two spatio-temporal GNN frameworks: MTGNN and StemGNN where both are recognized for their state-of-the-art performance in forecasting multivariate time series. Both models are uniquely capable of transforming time series data into graphs, and capturing both spatial and temporal dependencies. Both methods significantly outperform LSTM in predicting financial asset prices, especially in highly volatile markets such as cryptocurrencies. However, the performance difference between these GNN-based multivariate approaches and LSTM is less obvious for the Forex market which is less volatile than cryptocurrencies. The ability to model interdependencies between multiple assets both temporally and spatially is the main reason StemGNN and MTGNN outperform LSTM. Through a series of experiments and backtesting strategies, we assess the predictive power

and profitability of these models in portfolio construction. Our code can be found at `https://github.com/seferlab/temporal_gnn`.

ÖZETÇE

Çoklu finansal varlıkların zaman içindeki fiyat değişimleri, bu varlıkların karmaşık doğaları ve birbirlerine bağımlılıkları ile tanımlanabilir. Geleneksel tahmin yaklaşımları, bu varlıklar arasındaki uzaysal-zamansal bağımlılıkları tam olarak hesaba katmadığı için, varlıklar arasındaki bu bağımlılıkları göz ardı edebilir. Çizge Sinir Ağları (GNN), çeşitli veri türlerindeki karmaşık ilişkisel bağımlılıkları modellemek için güçlü bir araç olarak öne çıkmış ve son zamanlarda sosyal ağ analizi ve trafik tahmini gibi alanlarda gösterdikleri olağanüstü performanslarla dikkat çekmiştir. GNN'lerin bu yüksek kapasitesi, özellikle permutasyona karşı değişmezliği ve yerel bağlantıları sayesindedir. Ancak, varlık fiyat tahminindeki uygulamaları hala yeterince araştırılmamıştır.

Bu çalışmada, özellikle Döviz (Forex) ve Kripto Para piyasalarını ele alarak, çoklu finansal varlık fiyatlarının tahmininde GNN'lerin etkinliğini incelemekteyiz. Çok değişkenli zaman serilerinin tahmininde en gelişmiş performanslarıyla bilinen iki uzaysal-zamansal GNN yapısı olan MTGNN ve StemGNN'i çalıştırıyoruz. Her iki model de zaman serisi verilerini çizgeye dönüştürme ve hem uzaysal hem de zamansal bağımlılıkları yakalama konusunda benzersiz yeteneklere sahiptir. Özellikle Kripto Para gibi son derece değişken piyasalarda bu iki yöntem de finansal varlık fiyatı tahmininde LSTM'den belirgin şekilde üstün performans göstermektedir. Bununla birlikte, GNN tabanlı çok değişkenli yöntemler ile LSTM arasındaki performans farkı, Kripto Para piyasasına göre daha az değişken olan Döviz piyasası için daha az belirgindir. Çoklu varlıklar arasındaki karşılıklı bağımlılıkları hem zaman hem de uzay boyutunda modelleyebilme yetenekleri, StemGNN ve MTGNN'nin LSTM'den üstün performans göstermesinin ana sebebidir. Burada, bir dizi deney ve geriye dönük test stratejisi aracılığıyla, bu modellerin portföy oluşturmadaki tahmin gücünü ve

karlılığını inceliyoruz. Kodlarımız şu adreste yer almaktadır: `https://github.com/seferlab/temporal_gnn`.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Emre Sefer, for all his support, patience and understanding over the past 2.5 years. His helpful comments and guidance made this thesis possible.

I would also like to thank my family, for their love and support throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The prediction of asset prices, including stocks, through artificial intelligence-based trading systems has been a subject of study for nearly three decades. In contemporary financial markets, the scope of tradeable instruments extends beyond stocks to include Forex instruments [5], cryptocurrencies [6], options [7], Exchange-Traded Funds (ETFs) [8], commodities [9], etc. Correspondingly, the role of artificial intelligence-based trading systems has grown in significance and functionality across diverse global markets [9].

Deep learning methods have recently exhibited superior performance in various classification and prediction tasks compared to more traditional machine learning models like Support Vector Machines (SVMs). Notably, deep learning excels in image processing tasks such as segmentation and classification, marking a significant departure from conventional methodologies [10]. A parallel trend has emerged in financial asset price prediction tasks. While traditional time-series models like Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN) have been employed, the application of these techniques in financial prediction lags behind their prevalence in other artificial intelligence areas such as computer vision [11, 12] and Natural Language Processing [13, 14].

Graphs model a set of items as vertices, and the relationship between them as edges. Graphs have recently attracted more attention due to their significant expressive capabilities. For instance, relationships between humans can be modeled as social networks [15]. Similarly, the relationship between companies' trade relationships can be modeled as a supply chain graph [16]. Graphs are a common non-Euclidean data

structure for machine learning problems. To apply deep learning-based techniques to nonlinear graph data, Graph Neural Networks (GNNs) have been developed. The utilization of GNNs in modeling complex data relationships has been showing significant promise across distinct domains [17, 18, 19, 20], mainly due to their remarkable performance. Historically, the application of GNNs in forecasting multivariate time series has been largely confined to specific domains such as power systems [21] and traffic forecasting [18]. However, they haven't been studied extensively in predicting asset prices in financial markets.

Financial markets are characterized by their dynamic and interdependent nature, where the price movements of multiple assets can affect each other. For instance, in the Cryptocurrency market, Bitcoin price has a direct effect on the price of alternative cryptocurrencies such as Ethereum, Solana, Litecoin, etc [22]. Such dependencies also exist in the reverse direction; a surge in Ethereum price may cause sudden decreases in Bitcoin price, especially if the Ethereum price change is due to the movement of Bitcoin investments into Ethereum. In another example, Forex currencies also affect each other. For instance, EUR/USD will increase if the European Central Bank increases the interest rate higher than the United States Central Bank [23]. These examples present a potential use of GNNs, which are capable of capturing the joint patterns of multiple different assets price movements.

Here, we focus on the application of GNNs, specifically, MTGNN [3] and StemGNN [4], in forecasting the prices and price directions of multiple financial assets jointly, particularly in Foreign Exchange (Forex) and Cryptocurrency markets. The goal is to evaluate their predictive power in various market conditions, such as bull and bear markets, and assess their profitability through portfolio construction and backtesting [24]. The outcomes of these methods will be compared with those of baseline methods, such as LSTM [25], to demonstrate the relative effectiveness of GNNs in financial forecasting.

In this study, we focus on evaluating and analyzing two commonly used GNNs to predict asset price over time: MTGNN [3] and StemGNN [4] which are particularly designed to handle multivariate time-series data. Both GNNs are uniquely capable of transforming time series data into graphs, and capturing both spatial and temporal dependencies. Among them, MTGNN will focus on an automated extraction of unidirectional relationships between financial assets. Their proposed novel mix-hop propagation and dilated inception layers are important in jointly handling temporal convolution, graph convolution, and graph learning steps. On the other hand, StemGNN learns temporal dependencies and inter-series correlations jointly in the spectral domain. Graph Fourier Transform (GFT) and Discrete Fourier Transform (DFT) layers are combined to model temporal relationships. Once the data is processed at these layers, predictions are made over the next convolutional and sequential learning modules.

According to the detailed experiments, the GNN models, particularly MTGNN, outperformed the remaining methods in terms of both MAPE for price prediction and in terms of Accuracy, Precision, Recall, and F_1 score for price direction prediction. While MTGNN's improvement is statistically significant in both Forex and Cryptocurrency markets, the improvement has even more pronounced in the highly volatile Cryptocurrency market compared to the LSTM model. In terms of portfolio-based backtesting performance, MTGNN has on-par performance with LSTM in the less volatile Forex market, while generally outperforming the rest of the portfolio strategies even though they did not consistently result in significantly profitable cases in simulated trading scenarios. However, LSTM-based portfolio strategy significantly outperformed all the other models in the high-volatile Cryptocurrency market. Such portfolio-based results indicate the existence of a complex and nonlinear relationship between forecasting accuracy and financial performance, suggesting that higher accuracy does not necessarily lead to higher financial returns.

Our main contributions can be summarized as follows:

- We show that GNNs can outperform more conventional deep learning time series models such as LSTM [26] in predicting financial asset prices due to their ability to capture complex inter-series relationships in multi-asset data.
- We expand the application of GNNs to the financial forecasting domain, a domain that is traditionally dominated by more conventional time series models.
- We show that GNNs can be used to learn latent dependencies among multiple assets, even though multi-asset price data does not have an explicit graph structure.

1.1 Related Work

In recent years, the surge in computational capacity has led to the emergence of newer deep learning-based approaches in finance. Existing studies, such as [27, 28, 29, 30, 31, 32, 33], have explored deep learning approaches for forecasting stock markets, leveraging events data, financial time-series datasets, and news integration. Relevant work, such as [32], [33], and [34] incorporate technical analysis indicators into prediction models, employing feedforward neural networks, CNNs, and transformers with two-dimensional matrix characterizations of technical analysis datasets. Notably, deep learning demonstrates accurate learning and generalization across buy and sell time points over extended testing horizons in these studies.

Asset price prediction over time via deep learning techniques can be seen as an emerging topic. Before the emergence of deep learning-based methods, most of the time-series prediction methods were statistical techniques. The auto-regressive integrated moving average (ARIMA) [35], gaussian processes (GP) [36], and vector auto regression (VAR) [37] could be seen as statistical methods to model time-series

data. While ARIMA and GP are quite common in time series prediction mainly because of their highly interpretable and simple nature, they do not extend robustly to multivariate time-series data, and they depend strongly on the stationary nature of time-series data. Similarly, although VAR is extensively used for its ability to model the interdependencies among multiple time series, it is inherently linear and may not effectively capture the complex, nonlinear dynamics often present in financial time series.

Deep learning methods can capture the nonlinear nature of the data as well as not rely on the stationarity of the data. The existing time-series deep learning techniques could be divided into 2 parts: 1-Univariate price prediction where univariate methods analyze each asset's time-series dataset individually without taking into account the dependencies among multiple different assets data [38]. For instance, FC-LSTM [39] predicts the future value of a univariate time series with LSTM and fully connected layers. On the other hand, SMF [40] enhances the LSTM model by partitioning the cell states of a given series into a collection of distinct frequency parts via the Discrete Fourier Transform. Lastly, N-BEATS [41] comes up with a deep learning system for the basis expansion of univariate time-series data via multiple fully connected layers. 2- Multivariate price prediction which treats a set of multiple time series as a joint entity [42, 43, 44]. For instance, TCN [45] applies dilated convolutions to model high-dimensional time-series data. LSTNet [46] integrates both recurrent neural network (RNN) and convolution neural network (CNN) to infer local dependencies between multiple time-series datasets. CNNs summarize the interactions between multiple variables into a global latent state, so they won't be able to exploit the hidden dependencies among variable pairs.

Graph neural networks (GNNs) are quite successful in modeling spatial correlations between network vertices. GNNs are effective at various types of problems such as graph classification, vertex clustering, edge inference, and vertex classification.

GNNs assume a node state is dependent on its neighbor states. Different GNN types have been proposed to model such spatial dependency [47, 48, 49]. GNNs learn a vertex's high-level embedding by combining the knowledge from the vertex's neighbors. Recently, spatiotemporal graph neural networks have been proposed in which input is a multivariate time series dataset with a dependency graph describing the relationships between multiple variables in multivariate time series. In these spatiotemporal GNNs, temporal dependencies between historical points are modeled by RNN [50, 51] or CNN [52, 53], whereas spatial dependencies between vertices are modeled by graph convolution.

Among the existing spatiotemporal GNNs, DCRNN [51] integrates spatial and temporal relations in the convolutional recurrent neural network to predict traffic. ST-GCN [54] integrates gated temporal convolution and graph convolution via spatiotemporal convolutional blocks. GraphWaveNet [55] captures spatiotemporal relationships by combining graph convolution with dilated casual convolutions. Among more recent methods, StemGNN [4] captures multivariate and temporal relationships jointly in the spectral domain. On the other hand, MTGNN [3] utilizes a graph learning module to infer unidirectional correlations between variables. To model temporal and spatial dependencies, it has novel dilated inception and mix-hop propagation layers.

CHAPTER II

MULTIVARIATE TIME SERIES FORECASTING

Multivariate time series forecasting involves predicting future values of multiple time-dependent variables. The subsequent sections will delve into neural network-based methods for tackling this problem.

2.1 Long Short-Term Memory (LSTM)

LSTM networks are a special kind of Recurrent Neural Network (RNN), specifically designed to address the shortcomings of traditional RNNs. RNNs are inherently suited for sequential data processing, and are widely chosen for tasks like time series analysis, speech recognition, and language modeling. However, they are limited by the vanishing/exploding gradient problem.

During the training of traditional RNNs, gradients are propagated back through each time step of the input sequence. This problem arises because, as the sequence length increases, these gradients tend to either vanish (become very small) or explode (become very large), making it challenging to learn long-term dependencies [56].

To overcome this challenge, LSTMs were introduced [57]. The proposed architecture was a unique arrangement of gates and a cell state, which allowed the network to effectively retain information over long sequences, thereby mitigating the vanishing/exploding gradient issue. These gates can learn which data in a sequence is important to remember and which to discard. They enable the network to make better predictions based on long-term dependencies.

2.1.1 Architecture of LSTM

LSTMs tackle the vanishing/exploding gradient problem through a specialized structure comprising input gates, forget gates, output gates, and a cell state, as shown in Figure 1. This structure enables the effective capture of long-term dependencies in data.

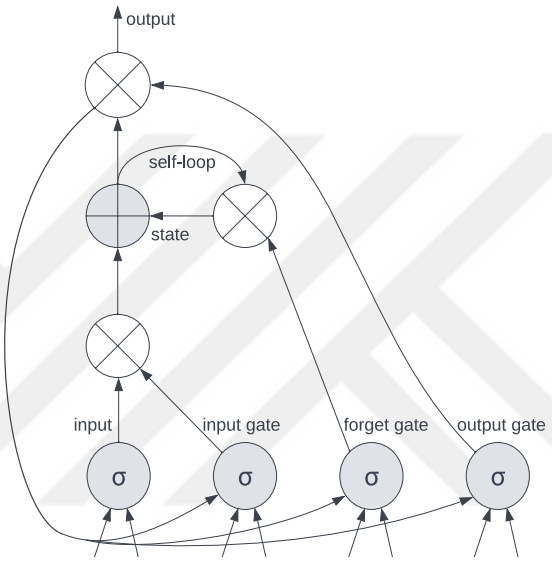

Figure 1: Structure of a Basic LSTM Cell (Adapted from Goodfellow et al. [1])

The LSTM cell comprises:

- **Cell State (C_t):** Acts as the network's memory, carrying relevant information across the sequence.
- **Input Gate (i_t):** Decides the extent of new information to be added to the cell state.
- **Forget Gate (f_t):** Determines the information to be removed from the cell state.
- **Output Gate (o_t):** Controls the output based on the cell state.

The operations within an LSTM cell are governed by the following equations, where σ denotes the sigmoid function, and W and b represent the weights and biases, respectively:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{1}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{4}$$

In these equations, W_i, W_f, W_o, and W_c are weight matrices, while b_i, b_f, b_o, and b_c are bias vectors. The input gate (i_t) and forget gate (f_t) use the sigmoid function to regulate the flow of information.

LSTMs process data in a sequential manner. At each time step t, their cell state (C_t) is updated based on the current input (x_t) and the previous state (C_{t-1}), modulated by the input and forget gates. The output gate (o_t) then determines the final output of the cell based on the updated cell state.

One of the most significant advantages of LSTM networks is their ability to capture long-term dependencies. Thanks to the forget gate, LSTMs can selectively maintain or erase information.

Overall, the architecture of LSTMs provides a robust framework for processing sequential data, by mitigating the problem of vanishing/exploding gradients and enabling the capture of long-term dependencies in data.

2.1.2 LSTMs in Time Series Forecasting

LSTM networks have gained significant traction in time series forecasting due to their ability to model complex temporal dependencies. They can learn and remember temporal information over extended sequences. They excel at learning and retaining information over long sequences, distinguishing them from traditional models like

ARIMA or GARCH, which may struggle with non-linear relationships and long-term context.

While LSTMs can be effective in univariate time series forecasting, multivariate forecasting can involve complex relationships between different time-dependent variables. LSTMs can process multiple input variables. However, their standard form struggles with modeling inter-series dependencies, which can be a critical aspect in complex scenarios like financial forecasting.

Furthermore, hybrid models combining LSTMs with other techniques such as attention mechanisms [58] or convolutional layers [59] have been explored. These approaches aim to enhance the LSTM's ability to handle complex interrelationships in multivariate time series, offering a promising direction for improving forecast accuracy in challenging domains.

In summary, LSTMs are a widely preferred powerful tool for time series forecasting. However, their limited capability in modeling inter-series dependencies in complex scenarios like financial forecasting necessitates further research and development of advanced hybrid models and alternative methods.

2.2 Graph Neural Networks (GNNs)

Real-world data is often naturally represented as graphs. Social networks, recommendation systems, and biological networks are just a few examples of domains where data inherently exhibits graph structures. Graph data is inherently non-Euclidean and exhibits unique characteristics that distinguish it from traditional Euclidean spaces, such as the 2D or 3D Cartesian coordinate systems commonly used to represent data. In Euclidean spaces, properties like distance, angles, and geometric relationships are well-defined. However, graphs deviate from these familiar concepts. They introduce complexities, including variations in the number of neighbors for each node, the

presence of different edge types, and often sparse connectivity patterns. These characteristics make it challenging to directly apply conventional neural networks like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) to graph-structured data, since such neural networks are ill-equipped to directly capture the relationships and dependencies among nodes in a graph. This led to attempts at extending and generalizing traditional neural networks to handle graphs [60, 61].

GNNs [2] address these challenges by operating directly on the graph, utilizing its connectivity structure and node features to learn a representation of the data. This capability has driven a surge in the application of GNNs across diverse fields, including social network analysis and protein-protein interaction modeling [62]. Since their inception, GNNs have evolved significantly into many different forms. They encompass various architectures that utilize different techniques in neural networks, namely, recurrent GNNs, convolutional GNNs, graph autoencoders, and spatial-temporal GNNs, as categorized in [63].

Next, we will delve into the basic functioning of GNNs in their first proposed version [2], which we will refer to as the vanilla GNN. Then, we will extend to cover specialized GNN architectures such as MTGNN [3] and StemGNN [4], both of which are subjects of our financial analysis.

2.2.1 Vanilla GNN

The vanilla GNN is designed to capture the state embeddings of nodes, representing the information of a node's local neighborhood in a graph. The learning process for these embeddings is structured as follows:

- *Initialization:* Each node is initialized with a state derived from its features.
- *State Update:* The state of a node is iteratively updated through a *local transition function* f_W, which incorporates the node's features as well as the features and states of its adjacent nodes. This function f_W is consistent across all nodes.

- *Output Computation:* After many iterations, which allow the node states to converge, an *output function* g is applied to the state of each node to generate an output value. This value is then utilized for various downstream tasks, such as node classification or regression.

- *Training:* The model is trained by adjusting the parameters of f_W and g to minimize the loss function. For supervised learning tasks, an example loss function is typically a sum of the squared differences between the predicted and target values for each node, as shown below:

$$\text{Loss} = \sum_{i=1}^{p} (t_i - o_i)^2 \tag{5}$$

In Equation 5, p represents the number of nodes that have supervised target information. For each node i, t_i denotes the target value, which could be continuous in regression tasks. The term o_i represents the output of the GNN for node i, as computed by the output function g.

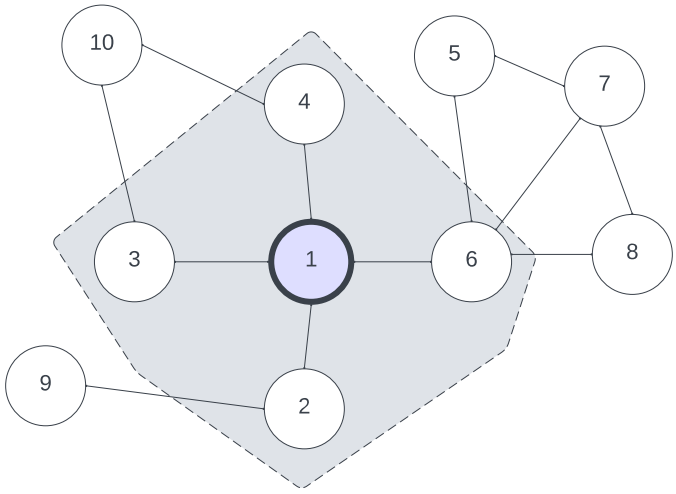

Figure 2: The state of node 1 is updated based on its own features and the aggregated features of its neighbors. (Adapted from Scarselli et al. [2])

Figure 2 depicts node 1 and its neighborhood. The shaded area represents the aggregation process where the information from neighboring nodes and the edges

connecting to node 1 are considered to update its state. Here, node 1's state is updated by passing the features of the node, along with those of its adjacent nodes and edges, into f_W.

The vanilla GNN model is a foundational approach that paved the way for more complex GNN architectures. It provides a structured way to learn how to encode graph topology into state embeddings of nodes, which can then be used for various predictive and classification tasks.

Although it excels at handling relational dependencies, it is designed to work with a well-defined graph structure, making it less suitable for scenarios where the graph structure is unknown beforehand, like in the case of multivariate time series data. This problem led to the development of GNN frameworks that can learn graphs from latent relationships between individual time series such as MTGNN [3] and StemGNN [4].

2.2.2 MTGNN

MTGNN (Multivariate Time Series Forecasting with Graph Neural Networks) is a deep learning framework specifically designed to forecast multivariate time series without predefined graph structures [3]. It treats individual time series as nodes within a graph, interconnected by edges that represent the hidden dependencies and interactions between the series. This approach allows MTGNN to learn the graph structure directly from the data, by uncovering latent relationships.

The MTGNN architecture integrates learning of the graph structure with training of the graph neural network in an end-to-end fashion, ensuring that the graph it constructs is optimized for the prediction task at hand.

As depicted in Figure 3, MTGNN comprises three core components: graph learning layer, graph convolution (GC) modules, and temporal convolution (TC) modules.

Initially, the graph learning layer initializes random node embeddings and learnable model parameters. The TC and GC modules, interleaved within the architecture,

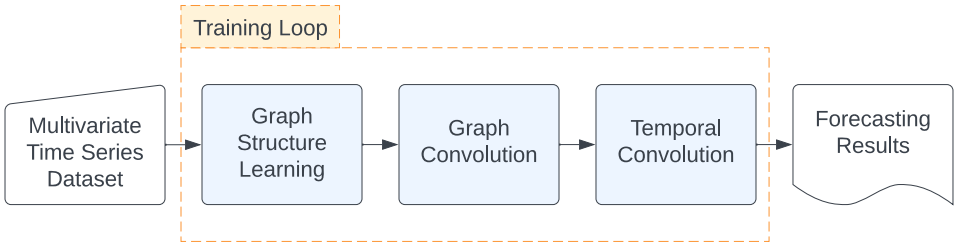

Figure 3: Simplified MTGNN Workflow (Adapted from Wu et al. [3])

incorporate residual and skip connections to prevent the vanishing gradient problem. Throughout each epoch, these modules process the node embeddings, thereby capturing temporal and spatial dependencies. Subsequently, the model calculates the stochastic gradient from the loss, which is then used to update both the model parameters and the node embeddings.

The graph learning layer constructs an adjacency matrix to capture the complex relationships within the multivariate time series data. It updates this matrix using the node embeddings across epochs. One challenge is that as the number of time series increases, the computational cost for calculating similarities between pairs of nodes rises quadratically. To mitigate this problem, the layer samples the top-k closest nodes as neighbors for each node, thereby sparsifying the adjacency matrix. Moreover, this layer learns unidirectional relationships within the multivariate time series data, acknowledging that influences among series can be asymmetric. In the adjacency matrix A it constructs, for any positive entry A_{ij}, indicating an influence from series i to series j, the corresponding entry A_{ji} is set to zero.

The graph convolution modules exploit spatial dependencies encapsulated in the graph structure using two novel mix-hop propagation layers. Mix-hop propagation enables the learning about neighbors at different hops through the employment of higher-order message passing, which allows for longer-range interactions [64]. However, aggregating neighborhood information on different degrees of nodes can potentially introduce noise. MTGNN's novelty in mix-hop propagation lies in maintaining

the original states of the nodes. In turn, this approach helps preserve local information as the nodes learn from a deep neighborhood. For this, the layers employ two steps: (1) recursively propagating node information throughout the graph to capture a deep neighborhood, and (2) selectively filtering the gathered information at each hop to prevent the propagation of noise and mitigate the over-smoothing that often accompanies an increase in the number of hops.

The temporal convolution modules capture temporal dependencies by employing a series of modified 1D convolutional filters. These modules consist of two dilated inception layers, which incorporate dilated convolutions to extend the range of temporal patterns the model can capture without increasing its depth or the size of its filters. Here, choosing an appropriate kernel size poses a challenge; larger kernels might miss subtle short-term patterns, whereas smaller ones might not adequately capture longer-term sequences. MTGNN addresses this dilemma through an inception strategy, which involves concatenating the outputs from a diverse array of kernel sizes. Specifically, it utilizes filters of sizes 1×2, 1×3, 1×6, and 1×7, a configuration to support a range of periodicities characteristic of temporal data—such as weekly and monthly cycles—are effectively represented. By adopting this approach, MTGNN aims to balance granularity with the ability to recognize and integrate long-term temporal sequences, thus enhancing its forecasting capabilities for complex time series data. This end-to-end process iteratively refines the model, culminating in the generation of forecasting results.

2.2.3 StemGNN

StemGNN (Spectral Temporal Graph Neural Network) presents a different approach to multivariate time series forecasting [4]. This approach, akin to MTGNN, simultaneously captures intra-series temporal correlations and inter-series relationships, yet

does not rely on predefined graph topologies. Diverging from MTGNN and similar models, StemGNN operates within the spectral domain. It utilizes the Graph Fourier Transform (GFT) to learn inter-series correlations and the Discrete Fourier Transform (DFT) to learn temporal dependencies, integrating these processes into a unified end-to-end framework. This spectral domain focus offers enhanced clarity in the spectral representations of data.

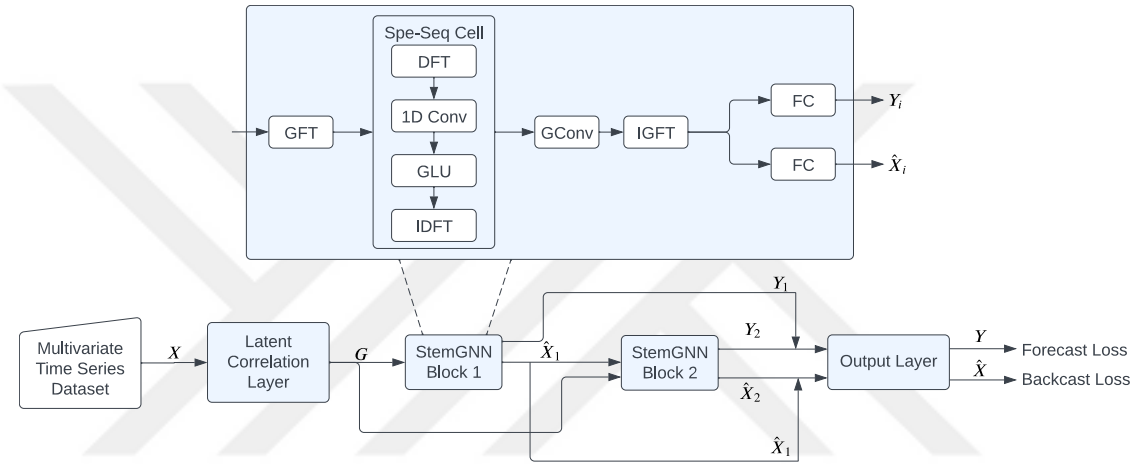

Figure 4: Overview of StemGNN Architecture (Adapted from Cao et al. [4])

Figure 4 illustrates the StemGNN process, starting with the input of multivariate time series data, X. StemGNN converts this input into a graph, learns relationships from this graph, and generates forecasts.

The initial step involves the Latent Correlation Layer, whose task is uncovering the underlying relationships among time series variables. This layer treats each time series as a distinct graph node and seeks to learn the edges that represent the interdependencies between them. While such edges can sometimes be predefined based on domain knowledge, StemGNN provides a way to learn these relationships directly from the data. It utilizes a Gated Recurrent Unit (GRU) to capture temporal features at each timestamp t, and then employs a self-attention mechanism over the GRU's final hidden state to capture the latent correlations between node pairs. The output is a weight matrix that serves as the adjacency matrix, encapsulating the strength of

connections in the resultant graph. The resulting graph, G, is then processed through two successive StemGNN blocks, interconnected by skip connections to allow building deeper models.

StemGNN blocks aim to model the spatial and temporal dependencies. Each of them begins with the application of GFT to convert the graph into a spectral matrix representation. Following this, the Spectral Sequential (Spe-Seq) Cell employs DFT for temporal dependency analysis, transforming time-domain signals into the frequency domain. Subsequently, the combination of 1D convolution and Gated Linear Units (GLU) facilitates learning temporal feature representations in the frequency domain. These are then reconverted into the time domain using the Inverse DFT (IDFT). The Spectral Graph Convolution (GConv) in each block captures spatial dependencies through graph convolution filtering of the spectral representation, followed by the Inverse GFT (IGFT) to revert to the time series representation in the original space.

The network's output layer consists of GLU and fully connected sub-layers, producing two output types: forecasting outputs, Y, for predicting future values, and backcasting outputs, $\hat{X}$, for reconstructing past values. The final loss function of the network is derived from the combined losses of these outputs. Overall, StemGNN's architecture offers a comprehensive method for jointly addressing spatial and temporal dependencies in multivariate time series within the spectral domain.

2.2.4 GNNs for Multivariate Time Series Data

For multivariate time series data, GNNs emerge as powerful forecasting tools, particularly well-suited for intricate and interdependent domains such as financial markets. Their strength lies in their ability to construct representations for each individual time series that go beyond merely encoding historical values, as they effectively capture both spatial and temporal dependencies. For example, in financial forecasting,

GNNs can adeptly capture how changes in one asset's performance relate to changes in others.

In essence, the unique structural characteristics and capabilities of GNNs position them as highly effective tools for forecasting in complex and interconnected domains like financial markets. They offer insights and forecasting accuracy that may not be attainable with traditional time series models alone.

CHAPTER III

EXPERIMENTAL SETUP

3.1 Dataset Description and Preparation

The study utilizes two distinct datasets to evaluate the efficacy of GNNs in varying market conditions: Cryptocurrency and Foreign Exchange (Forex) datasets. We obtained the price time series of both datasets by using the yfinance library [65]. The Cryptocurrency dataset comprises daily closing prices from October 2016 to October 2022 for 20 assets selected from the 50 highest market capitalization cryptocurrencies, based on data availability. This dataset includes FIL-USD, ETH-USD, BTC-USD, and 17 other assets. The Forex dataset includes daily closing prices from May 2006 to October 2023 for 10 major currency pairs. The selected Forex pairs are: AUD-USD, TRY-USD, EUR-USD, and 7 other pairs. For both datasets, included assets are shown in Table 1. Compared to cryptocurrencies, the Forex market is characterized by relatively lower volatility.

Table 1: Crytocurrency and Forex assets used in our experiments.

Asset Type	Used Assets
Cryptocurrencies	FIL-USD, MKR-USD, USDT-USD, TRX-USD, ETH-USD, XMR-USD, EOS-USD, XTZ-USD, ETC-USD, MIOTA-USD, XRP-USD, ADA-USD, XLM-USD, DOGE-USD, MANA-USD, LINK-USD, BCH-USD, BNB-USD, BTC-USD, LTC-USD
Forex	AUD-USD, TRY-USD, RUB-USD, INR-USD, EUR-USD, GBP-USD, CHF-USD, CAD-USD, CNY-USD, JPY-USD

Augmented Dickey-Fuller (ADF) tests [66] applied to both datasets indicated non-stationarity in the raw price data. To address this, we transformed both price datasets

into return ratios RR_t, calculated by using the following formula:

$$RR_t = \frac{P_t}{P_{t-h}} \tag{6}$$

where P_t represents the price at time t, and h denotes the horizon. For the Forex market, the horizon is set to 5 days to align with its weekday-only trading, while for cryptocurrencies, a 7-day horizon is used, reflecting their trading on every day. This transformation aims to stabilize the mean and variance, rendering the series more suitable for predictive modeling. Unlike the conventional return calculation, which often includes subtracting 1 from the ratio, this study employs a direct ratio approach. The decision to omit the traditional subtraction of 1 was based on empirical observations where models trained on the unadjusted data (return ratio) outperformed those trained on the adjusted data (conventional returns). Following this transformation, the return ratios of the assets' close prices were merged into multivariate data. It is important to note that some assets, such as FIL-USD or MKR-USD, have less historical data compared to others like ETC-USD or BTC-USD. To handle this discrepancy, the unavailable prices were backfilled. We acknowledge that this decision to backfill may introduce potential biases into the dataset. However, discarding the initial days with missing data would reduce the dataset's size, thereby limiting the learning potential of the models due to fewer samples. Therefore, we opted for a simple backfilling approach to impute the missing data.

In terms of a deep learning baseline, we used Long Short-Term Memory (LSTM) which serves as the baseline model in this study. To enable the models for optimal performance, hyperparameter optimization was performed using the Hyperopt [67] library in Python.

For the purpose of price prediction, we applied a sliding window approach to form the input variable. Each window comprised a sequence of prices $[P_{t-s+1}, P_{t-s+2}, \ldots, P_t]$, which was used to forecast the target price P_{t+h}, where s is a hyperparameter representing the sequence length.

3.2 Performance Evaluation

We assess and compare the performance of StemGNN, MTGNN, and LSTM baseline via traditional machine learning metrics and financial metrics. In terms of traditional metrics, we use Accuracy, Precision, Recall, and F_1 score to evaluate direction-based decisions such as buy, sell, and hold. These metrics are defined below:

$$\text{Accuracy} = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C}(TP_i + TN_i + FP_i + FN_i)} \tag{7}$$

$$\text{Precision} = \frac{\sum_{i=1}^{C}(\text{Precision}_i \times \text{Support}_i)}{\sum_{i=1}^{C} \text{Support}_i} \tag{8}$$

$$\text{Recall} = \frac{\sum_{i=1}^{C}(\text{Recall}_i \times \text{Support}_i)}{\sum_{i=1}^{C} \text{Support}_i} \tag{9}$$

$$\text{F}_1 \text{ Score} = \frac{\sum_{i=1}^{C}(\text{F}_1 \text{ Score}_i \times \text{Support}_i)}{\sum_{i=1}^{C} \text{Support}_i} \tag{10}$$

where C represents the total number of classes, which is 3 in our case for buy, hold, and sell. For each class i, the terms TP_i, TN_i, FP_i, and FN_i denote the counts of true positives, true negatives, false positives, and false negatives, respectively. The term Support_i indicates the size of class i. The precision, recall, and F_1 score for each class i are calculated as:

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i} \tag{11}$$

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i} \tag{12}$$

$$\text{F}_1 \text{ Score}_i = \frac{2 \times \text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{13}$$

We evaluate the price prediction performance by Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), and Root Mean Square Error (RMSE)

which are defined as follows:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|A_t - F_t|}{|A_t|} \tag{14}$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|A_t - F_t| \tag{15}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(A_t - F_t)^2} \tag{16}$$

where A_t represents the actual value and F_t is our forecast.

Lastly, by comparing their error measures across ten different training sessions, we calculated p-values to evaluate the statistical significance of the models' prediction performance.

3.2.1 Portfolio Metrics

In addition to the traditional machine learning metrics, we evaluate the financial performance of the models. The backtesting training protocol, depicted in Figure 5, starts with a hyperparameter search step where all available time series data, excluding the final 104 weeks is used. This data is partitioned into a training set comprising 75% and a validation set forming the remaining 25%. This step selects a hyperparameter set that gives the lowest loss on the validation set. Subsequently, the model undergoes retraining on progressively diminishing subsets of the trailing data. In each iteration, the model is trained on the available data and then employed to predict the forthcoming week's return ratios. This approach emulates a rolling window that moves forward one week at a time, where each week the amount of data excluded from the end of the series diminishes by one week, until all 104 weeks are consumed.

An equally weighted long-short portfolio was constructed based on the models' predictions to simulate real-world trading scenarios. We assume that we start with \$100 cash at the beginning and the trading commission is assumed to be 0.5% per transaction since we are trading only liquid Forex and Cryptocurrency names. We

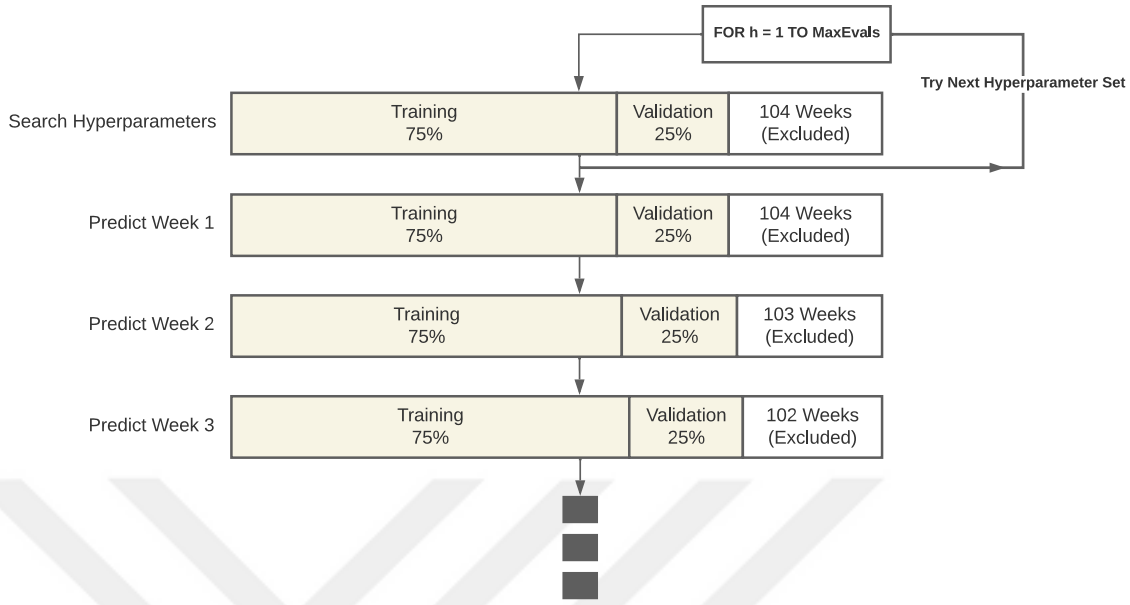

Figure 5: Data Partitioning During Backtesting

evaluate the portfolio performance using the Annualized Return, Annualized Volatility, Sharpe Ratio (SR), and Max Drawdown (MD). Let P_i denote the vector of the last closing prices of the assets for the i-th week (e.g., Friday for the Forex market, Sunday for the Cryptocurrency market), w_j represent the equal weight of the j-th asset, N be the number of invested assets, Σ be the covariance matrix of asset returns, $\bar{R}$ be the sample mean of weekly returns, and r_f be the risk-free rate corresponding to the three-month U.S. Treasury Bill rates during the trading period [68]. The portfolio metrics are defined as follows:

$$\text{Weekly Return } (R_i) = \frac{P_i}{P_{i-1}} - 1 \tag{17}$$

$$\text{Asset Weight } (w_j) = \frac{1}{N} \tag{18}$$

$$\text{Annualized Return } (AR) = w^T \bar{R} \times 52 \tag{19}$$

$$\text{Annualized Volatility } (AV) = \sqrt{w^T \Sigma w \times 52} \tag{20}$$

$$\text{Sharpe Ratio } (SR) = \frac{AR - r_f}{AV} \tag{21}$$

$$\text{Maximum Drawdown } (MD) = \frac{LowestValue – PeakValue}{PeakValue} \tag{22}$$

Each model's predictions were utilized to construct a long-short portfolio. This portfolio was rebalanced weekly for 104 weeks, based on the model's latest predictions. For a given asset, a long position was initiated only when the model projected an increase in its price. Conversely, a short position was taken in anticipation of a price decline as predicted by the model. In instances where the model did not forecast a definitive increase or decrease, the portfolio's position in that asset was liquidated, and the investment was reverted to cash. See Figure 6 for the overview of this process.

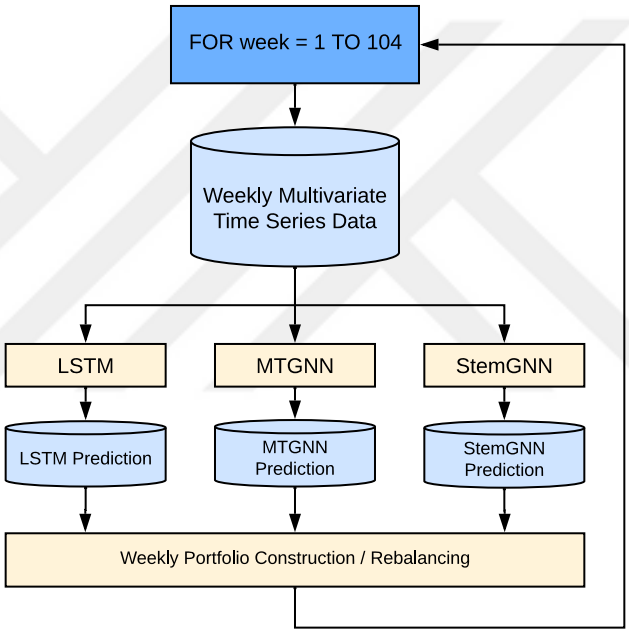

Figure 6: Overview of the Weekly Model Portfolio Rebalancing Process

In addition to these model-specific portfolios in Figure 6, two baseline portfolios were created for comparative statistical analysis: 1- A random long-only portfolio, which was rebalanced weekly with randomly selected assets; and 2- an "All-asset Equal Portfolio", which involve equal investment across all assets and had the advantage of incurring no transaction costs since there is no rebalancing.

To evaluate the statistical significance of results, throughout the 104-week simulation, we compared the profits of all portfolios weekly. Subsequently, p-values were calculated for all possible portfolio pair combinations, such as Random vs MTGNN,

Random vs StemGNN, StemGNN vs LSTM, etc. to assess the statistical significance.

3.3 Computational Performance

We evaluated the computational performance of MTGNN, StemGNN, and LSTM to assess their suitability for deployment. The training times of models can vary based on the used hyperparameter sets. To account for these variations in training times, our comparative analysis was conducted during the hyperparameter optimization phase. Subsequently, we measured the running times for searching 100 different hyperparameter sets using Hyperopt. Each model was trained for 30 epochs, notably, training was halted before convergence at epoch 30, resulting in the hyperparameter search that identifies the most promising configurations in the early training stages. In the weekly simulations, we employed these selected hyperparameters for training beyond the initial 30 epochs, continuing until convergence, guided by an early stopping approach. The experiments were conducted on Google Colab [69], utilizing Nvidia A100 40 GB GPUs.

CHAPTER IV

RESULTS AND DISCUSSION

4.1 Comparison via Traditional Metrics

Tables 3-4 present the performance of methods on the testing set using MAPE, MAE, and RMSE. The mean and standard deviation are calculated by averaging over 10 iterations.

Table 2: The breakdown of MAPE, MAE, and RMSE results (Mean ± Standard Deviation) for return ratio prediction, calculated over 10 iterations. Bold values represent the methods with the best performance.

Table 3: Forex

Model	**MAPE (%)**	**MAE**	**RMSE**
LSTM	1.707 ± 0.173	0.017 ± 0.002	0.154 ± 0.0004
MTGNN	$\mathbf{1.309 \pm 0.111}$	$\mathbf{0.013 \pm 0.001}$	$\mathbf{0.021 \pm 0.002}$
StemGNN	1.850 ± 0.165	0.021 ± 0.002	0.223 ± 0.001

Table 4: Cryptocurrency

Model	**MAPE (%)**	**MAE**	**RMSE**
LSTM	10.982 ± 0.250	0.104 ± 0.002	0.153 ± 0.002
MTGNN	$\mathbf{6.174 \pm 0.187}$	$\mathbf{0.061 \pm 0.002}$	$\mathbf{0.084 \pm 0.002}$
StemGNN	8.050 ± 0.224	0.076 ± 0.002	0.108 ± 0.003

Even though GNN approaches MTGNN and StemGNN outperform LSTM significantly for more volatile Cryptocurrency price prediction, LSTM performs slightly better than StemGNN in less volatile Forex market. Out of all models, MTGNN performs significantly best in both markets. However, as discussed in the later sections, the order of models' prediction errors does not directly translate into their financial profitability, which suggests that different models may have captured more profitable opportunities despite their overall higher errors relative to each other.

4.2 Direction-Based Prediction Evaluation

We also evaluate the methods' performance in terms of their ability to predict the direction of price movements. While classifying the price change into "Buy", "Hold", and "Sell" categories, a "Hold" is considered when the price change in either direction remains within 1%. Otherwise, if the price rises by more than 1%, it is labeled as "Buy"; if the price decreases by more than 1%, it receives a "Sell" label. Tables 5-6 present the general and category-specific direction-based classification metrics for Forex forecasts, respectively. According to Table 5, StemGNN outperforms the remaining methods for all classification metrics, which was quite different than price prediction performance results in terms of MAPE. Each of the three methods exhibits difficulties with "Buy" and especially "Sell" decisions, while having strong "Hold" values. MTGNN's precision on "Buy" decisions is remarkably better than that of the other models, thereby allowing it to identify more opportunities to enter the market.

Table 5: Classification performance of models for Forex data. Bold values represent the methods with the best performance.

Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F_1 (%)**
LSTM	57.28	83.15	57.28	67.29
MTGNN	54.95	63.59	54.95	58.57
StemGNN	**62.14**	**87.79**	**62.14**	**71.67**

Table 6: The breakdown of Forex Buy/Hold/Sell Decision Metrics respectively. Bold values represent the methods with the best performance.

Model	**Precision (%)**	**Recall (%)**	**F_1 (%)**
LSTM	4.76/90.44/1.24	23.41/61.12/5.13	7.91/72.95/2.00
MTGNN	**23.81**/78.06/**8.07**	28.06/64.42/21.31	**25.76**/70.59/11.71
StemGNN	6.93/**95.77**/**8.07**	**30.77**/**65.21**/**31.71**	11.31/**77.59**/**12.87**

Similarly, Tables 7-8 display these classification metrics for Cryptocurrency forecasts. In the Cryptocurrency market, StemGNN demonstrates significant superiority over the other models in both tables. However, this superiority does not translate into its portfolio performance as detailed in the portfolio analysis.

Table 7: Classification performance of models for Cryptocurrency data. Bold values represent the methods with the best performance.

Model	Accuracy (%)	Precision (%)	Recall (%)	F_1 (%)
LSTM	38.88	65.25	38.89	46.99
MTGNN	39.85	42.75	39.85	39.84
StemGNN	**77.52**	**76.82**	**77.52**	**77.08**

Table 8: The breakdown of Cryptocurrency Buy/Hold/Sell Decision Metrics respectively. Bold values represent the methods with the best performance.

Model	Precision (%)	Recall (%)	F_1 (%)
LSTM	77.46/24.50/6.03	41.91/20.81/41.48	54.39/22.51/10.54
MTGNN	53.68/34.54/28.13	44.59/18.66/48.69	35.66/24.23/35.66
StemGNN	**80.63/54.62/80.71**	**83.86/44.59/81.67**	**81.68/48.69/81.68**

4.3 Backtesting Portfolio Analysis

A backtesting of equal-weighted portfolios, each starting with $100, was conducted based on the forecasts provided by the models over a span of 104 weeks in both the Forex and Cryptocurrency markets. To inject a realistic aspect of trading into the analysis, a transaction cost of 0.5% per trade was factored into the simulation, which is realistic given all names we trade are highly liquid. Figures 7-8 depict the wealth curves and the trajectories of portfolio capital throughout the backtesting period, offering the investment performance attributed to each method.

Table 9 provides detailed portfolio metrics derived from these trading simulations in Forex market. For the cryptocurrencies, Tables 10-11 offer the same metrics, categorizing them into bull and bear market periods, respectively. These periods are assumed based on the observed trends in Cryptocurrency prices, as depicted for various cryptocurrencies in Figure 10 over the simulation period from October 2020 to October 2022. The first year is characterized by a consistent upward trend, assumed to be a bull market, while the second year shows a consistent downward trend, indicative of a bear market.

Within these tables, return and volatility columns represent annualized return and

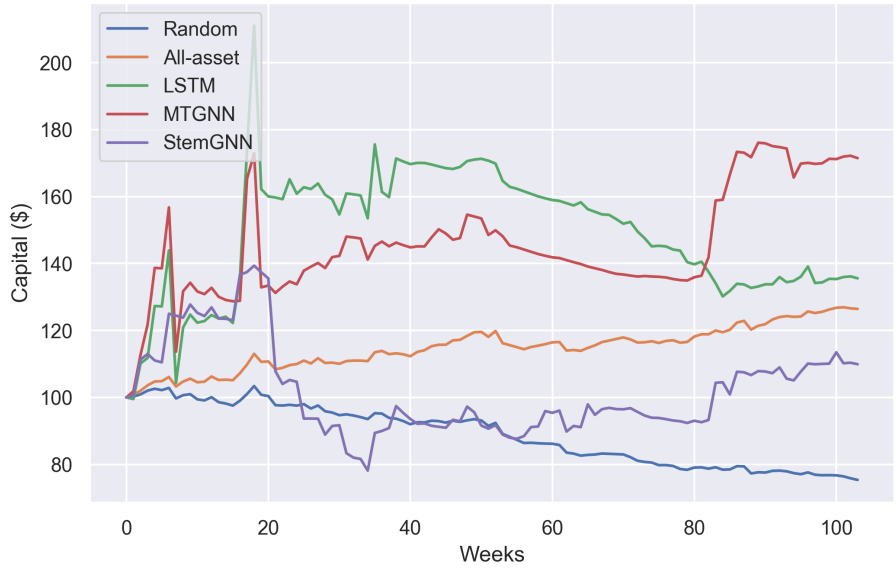

Figure 7: Forex

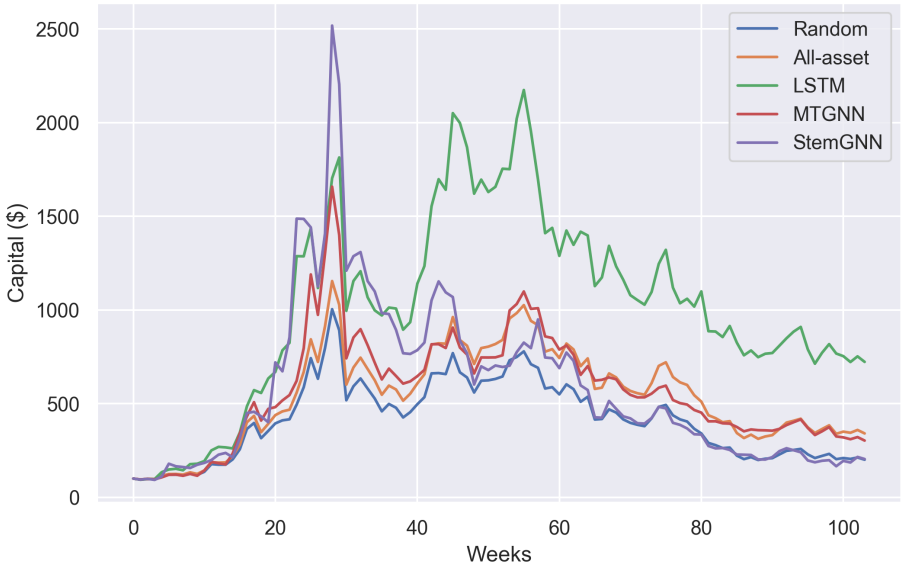

Figure 8: Cryptocurrency

Figure 9: Wealth curves for tested time-series GNN-based approaches.

volatility respectively whereas SR and MD define Sharpe Ratio and Max Drawdown respectively. The capital changes throughout the backtesting suggest a disconnection between prediction accuracy and trading performance. In the Cryptocurrency market,

LSTM's higher error rate did not preclude it from realizing the highest portfolio growth. Similarly, being the best model in all direction-based prediction metrics did not guarantee the highest profit for StemGNN. In the Forex market, MTGNN model demonstrates the highest annualized return while having a high volatility.

Table 9: Backtesting performance for Forex portfolios.

Portfolio	**Return (%)**	**Volatility (%)**	**SR**	**MD (%)**
LSTM	25.24	44.44	0.46	-38.33
MTGNN	35.70	40.61	0.76	-27.49
StemGNN	9.52	30.78	0.15	-43.93
Random	-14.00	7.22	-2.60	-27.12
All-asset	12.12	7.25	1.01	-4.95

Table 10: Backtesting performance for Cryptocurrency portfolios during bull market period.

Portfolio	**Return (%)**	**Volatility (%)**	**SR**	**MD (%)**
LSTM	363.74	120.69	3.00	-50.68
MTGNN	287.17	127.12	2.25	-63.41
StemGNN	335.72	181.91	1.84	-76.10
Random	249.4	108.26	2.29	-57.57
All-asset	276.74	108.44	2.54	-55.33

Table 11: Backtesting performance for Cryptocurrency portfolios during bear market period.

Portfolio	**Return (%)**	**Volatility (%)**	**SR**	**MD (%)**
LSTM	-70.60	61.27	-1.17	-67.20
MTGNN	-78.56	54.68	-1.46	-72.44
StemGNN	-95.00	75.28	-1.28	-82.54
Random	-99.74	59.59	-1.69	-74.45
All-asset	-73.25	59.65	-1.25	-82.54

4.4 Statistical Significance Analysis

The statistical significance of the results is analyzed in two parts: 1- the significance of prediction performance is assessed through two-sided independent t-tests between model prediction errors; and 2- the significance of portfolio performance is evaluated

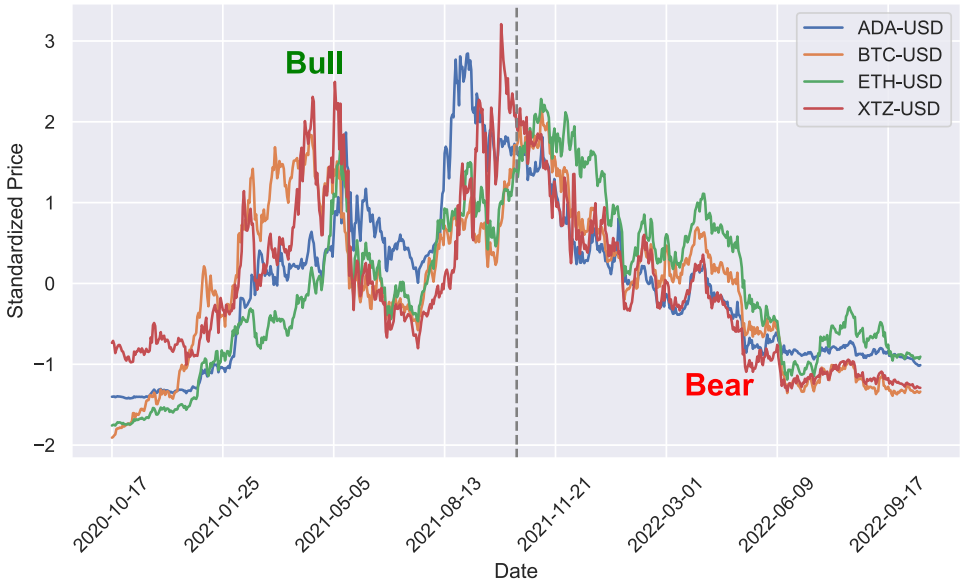

Figure 10: Assumed Bull and Bear Periods in the Cryptocurrency Market

through two-sided paired t-tests on the wealth curves of different portfolios across the 104-week simulation period. Tables 12-13 present these analyses for the Forex and Cryptocurrency datasets, respectively. A performance comparison between methods is considered to have a statistically significant difference if the p-value is less than 0.05. In the context of t-tests, a negative t-statistic indicates that the model listed first (or on the left side of a comparison pair) has a lower sample mean. This suggests better performance in the context of Table 12, which deals with prediction errors, and worse performance in Table 13, which pertains to portfolio capital.

The analysis of prediction errors shows a clear advantage for both GNNs over LSTM in almost all cases for both Forex and Cryptocurrency markets. The exception is the Forex market comparison between StemGNN and LSTM, where LSTM significantly outperforms StemGNN in terms of MAE and RMSE and has a marginally better MAPE. Additionally, MTGNN consistently has statistically significantly lower errors than StemGNN, positioning it as the best performing model in terms of prediction.

Table 12: Statistical significance of MAPE, MAE, and RMSE comparisons in Forex and Cryptocurrency markets.

		Forex		Cryptocurrency	
Metric	**Comparison**	t**-stat**	p**-value**	t**-stat**	p**-value**
MAPE	MTGNN vs StemGNN	-8.61 *	$< 10^{-5}$ *	-20.31 *	$< 10^{-5}$ *
	MTGNN vs LSTM	-6.11 *	$< 10^{-5}$ *	-48.66 *	$< 10^{-5}$ *
	StemGNN vs LSTM	1.90	0.07362	-27.61 *	$< 10^{-5}$ *
MAE	MTGNN vs StemGNN	-12.17 *	$< 10^{-5}$ *	-17.39 *	$< 10^{-5}$ *
	MTGNN vs LSTM	-5.70 *	$< 10^{-5}$ *	-47.46 *	$< 10^{-5}$ *
	StemGNN vs LSTM	5.22 **	$< 10^{-5}$ **	-30.08 *	$< 10^{-5}$ *
RMSE	MTGNN vs StemGNN	-263.10 *	$< 10^{-5}$ *	-22.98 *	$< 10^{-5}$ *
	MTGNN vs LSTM	-174.64 *	$< 10^{-5}$ *	-91.07 *	$< 10^{-5}$ *
	StemGNN vs LSTM	308.44 **	$< 10^{-5}$ **	-44.09 *	$< 10^{-5}$ *

*: First model is better. **: Second model is better.
No asterisk: No significant difference.

Table 13: Statistical significance of portfolio weekly capital comparisons in Forex and Cryptocurrency markets.

	Forex		Cryptocurrency	
Comparison	t**-stat**	p**-value**	t**-stat**	p**-value**
Random vs All-asset	-17.79 **	$< 10^{-5}$ **	-18.38 **	$< 10^{-5}$ **
Random vs LSTM	-30.51 **	$< 10^{-5}$ **	-18.25 **	$< 10^{-5}$ **
Random vs MTGNN	-26.03 **	$< 10^{-5}$ **	-14.01 **	$< 10^{-5}$ **
Random vs StemGNN	-9.60 **	$< 10^{-5}$ **	-5.99 **	$< 10^{-5}$ **
All-asset vs LSTM	-18.10 **	$< 10^{-5}$ **	-17.56 **	$< 10^{-5}$ **
All-asset vs MTGNN	-27.35 **	$< 10^{-5}$ **	-2.63 **	0.00973 **
All-asset vs StemGNN	7.86 *	$< 10^{-5}$ *	-1.49	0.14013
LSTM vs MTGNN	1.29	0.19981	15.57 *	$< 10^{-5}$ *
LSTM vs StemGNN	17.22 *	$< 10^{-5}$ *	10.66 *	$< 10^{-5}$ *
MTGNN vs StemGNN	21.61 *	$< 10^{-5}$ *	-0.81	0.41809

*: First portfolio is better. **: Second portfolio is better.
No asterisk: No significant difference.

Conversely, portfolio performance analysis paints a more complex picture. In the Forex market, LSTM and MTGNN show comparable performance, outperforming other models. Furthermore, the All-asset strategy demonstrates superior performance over StemGNN's portfolio, paralleling the prediction error analysis, which indicated LSTM's slight advantage over StemGNN in the Forex market. In the Cryptocurrency market, LSTM's portfolio performance is significantly better than all other strategies.

MTGNN outperforms both the Random and All-asset strategies, while being comparable to StemGNN. As in the Forex market, StemGNN shows the least favorable performance among the deep learning models, as it fails to surpass the simple All-asset strategy.

These disparities between prediction error significance and portfolio performance imply that statistically significant prediction accuracy does not necessarily translate to more effective portfolio management in financial markets. In such a domain, statistical differences do not fully encapsulate a model's practical utility. A model may not consistently excel across all metrics yet prove valuable if it can make slightly more accurate predictions at pivotal moments, potentially leading to significant financial gains. This aspect is particularly salient in highly volatile markets, such as cryptocurrencies, where the fortuitous timing of trades by less accurate predictors might yield better results than those by more accurate models. Thus, the practical implications of a model's performance, particularly in terms of risk management and return on investment, should also be carefully considered alongside statistical findings.

4.5 Computational Performance

The comparison of running times of MTGNN, StemGNN, and LSTM during hyperparameter optimization is presented in Table 14.

Table 14: Running times for MTGNN, StemGNN, and LSTM during hyperparameter optimization on Forex (10 assets × 4548 days) and Cryptocurrency (20 assets × 2189 days) datasets, with times noted in hours (h) and minutes (m).

Model	**Forex**	**Cryptocurrency**
LSTM	12m	5m
MTGNN	2h 8m	1h 10m
StemGNN	1h 8m	30m

GNNs exhibit significantly longer training times compared to LSTM. In practical scenarios involving weekly training, all training times can be deemed acceptable. However, for higher-frequency trading strategies, those such as daily or hourly trading,

the shorter running times of LSTM due to its simpler architecture could be more advantageous. This can allow a more extensive hyperparameter search for LSTM than the GNN models.

Overall, while GNNs demonstrate promising predictive capabilities, their higher computational demands may limit their applicability in certain high-frequency trading scenarios. Conversely, due to its lower computational overhead, LSTM presents a viable alternative in situations that require rapid model training and retraining.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

In this study, we have investigated the effectiveness of Graph Neural Networks (GNNs), specifically MTGNN and StemGNN, in forecasting financial asset prices in comparison to a more traditional deep learning time-series model such as LSTM. The study focused on two distinct financial markets: the foreign exchange (Forex) market and the Cryptocurrency market. In terms of machine learning-based performance evaluation, GNN models, particularly MTGNN, showed significant outperformance in both markets compared to LSTM. This is even more pronounced in the highly volatile Cryptocurrency market. On the other hand, our backtesting portfolio results presented a more complex picture. Despite their superior forecasting accuracy, GNN methods did not consistently convert better prediction performance into increased profitability in automated trading scenarios during backtesting. This underscores the complex and non-linear relationship between forecasting accuracy and financial performance, suggesting that higher accuracy does not necessarily lead to higher financial returns. Overall, this research contributes to the field of financial forecasting by expanding the use of GNNs, providing empirical evidence of GNNs' effectiveness in predicting financial asset prices, especially in the volatile Cryptocurrency market, and enhancing the understanding of the non-linear relationship between predictive accuracy and trading profitability.

Although we obtained reasonable performance via spatio-temporal GNNs, additional enhancements could be integrated into our framework. In future work, one can also consider newly emerging transformer-based spatio-temporal architectures [70, 71]. We can investigate the methods to mitigate the effects of volatility

for improved forecasting performance, especially in high-volatility markets such as cryptocurrencies. Financial time series data suffers from a lot of noise influenced by a wide array of factors, such as politics and social media. Thus, the study's reliance on historical data may not account for future market conditions and may invalidate our findings across different market structures. The timing of trading decisions can affect the investment gains. Models with similar predictive power might yield disparate portfolio performances due to fortunate timing in critical trading decisions. Hence, the trading simulations conducted here may be influenced by the fortunate timing of one model over another. To address this, another enhancement could be to focus on different trading strategies instead of just long-short portfolio construction. In line with this, it would be worthwhile to explore trading strategies that focus on decorrelation from market trends [72], rather than solely relying on the accuracy of price prediction models. Lastly, for financial time series, only Forex and Cryptocurrency markets are examined in this study. Hence, generalizing our findings to other financial markets could also be another future work.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[2] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[3] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 753–763, 2020.

[4] D. Cao, Y. Wang, J. Duan, C. Zhang, X. Zhu, C. Huang, Y. Tong, B. Xu, J. Bai, J. Tong, *et al.*, "Spectral temporal graph neural network for multivariate time-series forecasting," *Advances in neural information processing systems*, vol. 33, pp. 17766–17778, 2020.

[5] M. Ayitey Junior, P. Appiahene, O. Appiah, and C. N. Bombie, "Forex market forecasting using machine learning: Systematic literature review and meta-analysis," *Journal of Big Data*, vol. 10, no. 1, p. 9, 2023.

[6] S. Freeda, T. C. E. Selvan, and I. Hemanandhini, "Prediction of bitcoin price using deep learning model," in *2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 1702–1706, 2021.

[7] M. Ge, S. Zhou, S. Luo, and B. Tian, "3d tensor-based deep learning models for predicting option price," in *2021 International Conference on Information Science and Communications Technologies (ICISCT)*, pp. 1–6, IEEE, 2021.

[8] W. Zheng, "Exchange-traded fund price prediction based on the deep learning model," in *2021 China Automation Congress (CAC)*, pp. 7429–7434, 2021.

[9] R. C. Cavalcante, R. C. Brasileiro, V. L. Souza, J. P. Nobrega, and A. L. Oliveira, "Computational intelligence and financial markets: A survey and future directions," *Expert Systems with Applications*, vol. 55, pp. 194–211, 2016.

[10] A. M. Ozbayoglu, M. U. Gudelek, and O. B. Sezer, "Deep learning for financial applications : A survey," *Applied Soft Computing*, vol. 93, p. 106384, 2020.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[14] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[15] E. Sefer, "Mocmin: convex inferring of modular low-rank contact networks over covid diffusion data," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 30, no. 7, pp. 2568–2585, 2022.

[16] S. M. Wagner and N. Neshat, "Assessing the vulnerability of supply chains using graph theory," *International Journal of Production Economics*, vol. 126, no. 1, pp. 121–129, 2010. Improving Disaster Supply Chain Management – Key supply chain factors for humanitarian relief.

[17] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[18] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *Expert Systems with Applications*, vol. 207, p. 117921, 2022.

[19] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5782–5799, 2023.

[20] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *Journal of Machine Learning Research*, vol. 24, no. 43, pp. 1–48, 2023.

[21] W. Liao, B. Bak-Jensen, J. R. Pillai, Y. Wang, and Y. Wang, "A review of graph neural networks and their applications in power systems," *Journal of Modern Power Systems and Clean Energy*, vol. 10, no. 2, pp. 345–360, 2021.

[22] C. M. Hafner, "Alternative assets and cryptocurrencies," *Journal of Risk and Financial Management*, vol. 13, no. 1, p. 7, 2020.

[23] M. Hanisch, "German and euro area economies will benefit from a us interest rate hike in the short term," *DIW Weekly Report*, vol. 8, no. 12, pp. 115–121, 2018.

[24] R. D. Arnott, C. R. Harvey, and H. Markowitz, "A backtesting protocol in the era of machine learning," *Available at SSRN 3275654*, 2018.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European journal of operational research*, vol. 270, no. 2, pp. 654–669, 2018.

[27] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Deep learning for event-driven stock prediction," in *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, p. 2327–2333, AAAI Press, 2015.

[28] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.

[29] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[30] C. Krauss, X. A. Do, and N. Huck, "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500," *European Journal of Operational Research*, vol. 259, no. 2, pp. 689–702, 2017.

[31] A. Yoshihara, K. Fujikawa, K. Seki, and K. Uehara, "Predicting stock market trends by recurrent deep neural networks," in *PRICAI 2014: Trends in Artificial Intelligence* (D.-N. Pham and S.-B. Park, eds.), (Cham), pp. 759–769, Springer International Publishing, 2014.

[32] O. B. Sezer, M. Ozbayoglu, and E. Dogdu, "A deep neural-network based stock trading system based on evolutionary optimized technical analysis parameters," *Procedia Computer Science*, vol. 114, pp. 473–480, 2017. Complex Adaptive Systems Conference with Theme: Engineering Cyber Physical Systems, CAS October 30 – November 1, 2017, Chicago, Illinois, USA.

[33] O. B. Sezer and A. M. Ozbayoglu, "Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach," *Applied Soft Computing*, vol. 70, pp. 525–538, 2018.

[34] T. Tuncer, U. Kaya, E. Sefer, O. Alacam, and T. Hoser, "Asset price and direction prediction via deep 2d transformer and convolutional neural networks," in *Proceedings of the Third ACM International Conference on AI in Finance*, ICAIF '22, (New York, NY, USA), p. 79–86, Association for Computing Machinery, 2022.

[35] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[36] R. Frigola, "Bayesian time series learning with gaussian processes," Ph.D. dissertation, Dept. Eng., St Edmund's College, Univ. of Cambridge, Cambridge, UK, 2015.

[37] E. Zivot and J. Wang, "Vector autoregressive models for multivariate time series," *Modeling financial time series with S-PLUS®*, pp. 385–429, 2006.

[38] W. M. J. Murphy and K. Chen, "Univariate vs multivariate time series forecasting with transformers," 2023.

[39] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, pp. 802–810, 2015.

[40] L. Zhang, C. Aggarwal, and G.-J. Qi, "Stock price prediction via discovering multi-frequency trading patterns," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2141–2149, 2017.

[41] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," in *International Conference on Learning Representations*, 2020.

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

[43] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 922–929, 2019.

[44] A. Loukas and N. Perraudin, "Stationary time-vertex signal processing," *EURASIP journal on advances in signal processing*, vol. 2019, no. 1, pp. 1–19, 2019.

[45] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[46] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, 2018.

[47] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. of ICML*, pp. 1263–1272, 2017.

[48] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *International Conference on Learning Representations*, 2019.

[49] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[50] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *Proc. of NIPS*, pp. 362–373, Springer, 2018.

[51] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations*, 2018.

[52] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proc. of IJCAI*, pp. 3634–3640, 2018.

[53] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. of AAAI*, pp. 3482–3489, 2018.

[54] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 3634–3640, International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[55] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 1907–1913, International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[56] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, pp. 1310–1318, Pmlr, 2013.

[57] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[58] X. Zhang, X. Liang, A. Zhiyuli, S. Zhang, R. Xu, and B. Wu, "At-lstm: An attention-based lstm model for financial time series prediction," in *IOP Conference Series: Materials Science and Engineering*, vol. 569, p. 052037, IOP Publishing, 2019.

[59] I. E. Livieris, E. Pintelas, and P. Pintelas, "A cnn–lstm model for gold price time-series forecasting," *Neural computing and applications*, vol. 32, pp. 17351–17360, 2020.

[60] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[61] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5115–5124, 2017.

[62] Z. Liu and J. Zhou, *Introduction to graph neural networks*. Springer Nature, 2022.

[63] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[64] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *international conference on machine learning*, pp. 21–29, PMLR, 2019.

[65] R. Aroussi, "yfinance: Yahoo! finance market data downloader." `https://github.com/ranaroussi/yfinance`, Accessed: January 4, 2024.

[66] Y.-W. Cheung and K. S. Lai, "Lag order and critical values of the augmented dickey–fuller test," *Journal of Business & Economic Statistics*, vol. 13, no. 3, pp. 277–280, 1995.

[67] J. Bergstra, D. Yamins, D. D. Cox, *et al.*, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," in *Proceedings of the 12th Python in science conference*, vol. 13, p. 20, Citeseer, 2013.

[68] U.S. Department of the Treasury, "Treasury bill rates data." `https://home.treasury.gov/resource-center/data-chart-center/interest-rates/TextView?type=daily_treasury_bill_rates`, Accessed: January 11, 2024.

[69] Google LLC, "Google colab." `https://colab.research.google.com`, Accessed: January 15, 2024.

[70] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G.-J. Qi, and H. Xiong, "Spatial-temporal transformer networks for traffic flow forecasting," *arXiv preprint arXiv:2001.02908*, 2020.

[71] A. Deihim, E. Alonso, and D. Apostolopoulou, "Sttre: A spatio-temporal transformer with relative embeddings for multivariate time series forecasting," *Available at SSRN 4404879*, 2023.

[72] O. Hubáček and G. Šír, "Beating the market with a bad predictive model," *International Journal of Forecasting*, vol. 39, no. 2, pp. 691–719, 2023.

VITA

Yasin Uygun received his B.Sc. degree in 2020 in Computer Engineering from Yildiz Technical University, Istanbul, Turkey. Then, he worked for approximately three years as a Data Engineer at an e-commerce company in Istanbul, focusing on big/fast data applications. In 2021, he started his M.Sc. program in Computer Science at Özyeğin University, Istanbul. Under the supervision of Dr. Emre Sefer, he has been focusing on the applications of Graph Neural Networks in financial markets.