Deep learning approaches in Finance: Forecasting volatility and enhancing Quantitative trading strategies

Marcelo Sardelich Nascimento

PhD Thesis

University of York

Computer Science

December 2019

Abstract

This thesis addresses practical, real-world problems in the financial services industry using Deep Learning architectures. The main focus is on advancing current approaches in the areas of Risk Management and Quantitative Trading. The former is concerned with the risk of investment, whereas the latter refers to identifying profitable investment opportunities. Both areas are crucial in assessing the profitability and risk levels of investments. This research has three important findings which contribute to the academic literature.

First, the impact of news headlines on forecasting short-term volatility is explored. A novel neural network architecture, the Multimodal Hierarchical Attention Network (MHAN) is introduced to learn the joint representation of multiple data modalities (prices and news). The architecture addresses two aspects which are essential for news representation: news relevance and news novelty. The importance of the components of MHAN is investigated using the ablation method, with different sentence encoders being used to represent text. The experimental results confirm that adding news headlines consistently improves volatility forecasting across different market sectors and that the MHAN model outperforms the widely used GARCH model.

Second, a novel graph neural network architecture, the Graph Transformer Network (GTN) is proposed to better deal with the Range trading strategy which profits from short-term distortions in stock prices. The relationship between stocks is represented by a graph. The effect of other stocks on the target stock range prediction is then investigated by injecting prior knowledge via a graph into the learning process. The proposed approach is evaluated on a large number of stocks. Experiments confirm the profitability of the strategy and demonstrate that the GTN outperforms current state-of-the-art graph networks.

Third, Deep Reinforcement Learning is successfully applied to the pair trading strategy. This thesis presents the Investment Strategy with the Investors' Preferences (ISIP) framework to integrate optimal portfolio allocations with a risk management component. This component targets a constant level of risk pre-set by the investor. The experimental results confirm that the proposed framework improves the performance of existing approaches and is effective at restricting portfolio volatility.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank the following people, without whom I would not have been able to complete this research.

The Artificial Intelligence Group (AIG) at University of York, especially to my supervisor Dr Suresh Manandhar, whose insight into the subject matter guided the research. Special thanks to Alexandros Komninos, who brought me to the “dark side” of Representation Learning. Thanks also for the long, passionate hours discussing and teaching me the subject.

My dear friends at the university, who have supported me and had to put up with my stresses and moans! Here, special thanks to Carlo Ottaviani, Mattew Prinold, Mudita Sharma and Seb Marshal.

My biggest thanks to Yauheniya Shynkevich, sorry for being even grumpier than normal whilst I wrote this thesis. Thanks for all your support, without which I would have stopped these studies. You have been astonishing, and I will now dedicate all my time in travelling around the world with you <3

Declaration of Authorship

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

The first paper, presented in Appendix A, resulted from the study of sentiment analysis in the financial domain. This research is published at the Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018):

Marcelo Sardelich and Dimitar Kazakov. "Extending the Loughran and McDonald Financial Sentiment Words List from 10-K Corporate Filings using Social Media Texts" Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). ACM, 2018, Miyazaki, Japan.

This research had the direction of using a pipeline approach for volatility forecasting. Thus, by using the Sentiment Lexicon we would be able to classify the sentiment of any news. We then would use this sentiment to improve volatility predictions. With the advancements in deep learning, it soon became clear that we could directly learn a mapping from the news and price to the volatility, and also take into account important aspects of news modelling, such as news relevance and novelty. The following preprint resulted from the investigation of short-term volatility forecasting using deep learning approaches:

Marcelo Sardelich, Suresh Manandhar. "Multimodal deep learning for short-term stock volatility prediction" (ArXiv 2018). preprint arXiv:1812.10479.

Part I

Introduction

Chapter 1

Introduction

The financial industry has experienced persistent growth over the last few decades and has matured into one of the most competitive fields across the globe. Financial markets are influenced by the combined actions of multiple market participants and are increasingly complex to model. The financial industry keeps pace and quickly adapts to advances in technology and its capabilities. The introduction of electronic trading platforms has changed forever the way market participants negotiate and execute trades. Improvements in the speed of information transfer have led to the development and adoption of high-frequency trading. Advances in computational processing resources have allowed for continual progress in financial market modelling. The increase in the amount of data generated on a daily basis and available for analysis serves as the main driver for deriving meaningful information from it. Active development in the area of Artificial Intelligence (AI) has provided opportunities for advancing the predictive capabilities of financial forecasting methodologies.

Markets are characterised by complex behaviour and modelling them is a challenging task. Historically, statistical and econometric models have been used to describe stock prices. Forecasting involves dealing with issues of non-stationarity, noise, a high degree of uncertainty, hidden relationships and often unstructured irregular data. In recent decades, fast development in the area of AI and its constituent Machine Learning (ML) has brought new opportunities to address these types of problems. In ML, a computer system can learn from data and experience how to perform a task after being given a set of examples of the correct behaviour. Specialists need to decide which information is relevant so that the system will learn and generalise its responses to new inputs. Typically, the representations for ML systems contain vectors in a space of arbitrary features. Recently, the application of Deep Learning (DL) algorithms has attracted public attention. DL refers to learning representations of the data using multiple levels of abstraction. Thus, the first layers of the neural network capture representations very close to the input data. Once these networks are stacked, higher levels of abstraction can be learnt. A final prediction layer is then used map the input to the output straight from the data, without any dependence on human-designed features.

In financial markets, technical analysts often rely on technical indicators to predict future market behaviour. Technical indicators are computed from price time series using domain knowledge and are employed as the input variable of different trading strategies. However, the process of engineering technical indicators is laborious. By applying DL methods to learn directly from the price time series, we can avoid the burdensome process of engineering technical indicators. The possibility of using DL to tackle problems of volatility forecasting and designing quantitative trading strategies is investigated in this thesis. No domain knowledge is required to engineer features to train a DL model because the model learns to map raw data directly to the desired outcomes. This research focuses on practical applications of DL to advance state-of-the-art techniques for Risk Management and Quantitative Trading.

Risk Management refers to the process of identifying, assessing and controlling risks, defined as the effect of uncertainty. It ensures that the risks are managed proactively. When considering the financial market risk of holding a stock, this risk arises from adverse movements in the stock price. Price volatility is commonly accepted as a measure of market risk. Quantitative Trading consists of identify opportunities to profit on the markets. When a trading strategy is evalauted, it is important to account for both the strategy's profitability and its risk.

1.1 Research Context and Aims

The research presented in this thesis has a strong focus on designing neural network architectures and frameworks for specific tasks in the financial industry. The methods used to achieve this goal are based on representation learning [22], where we employ supervised and reinforcement learning. In finance, there are many asset classes, e.g. Fixed Income (bonds) or Foreign Exchange (currency). This thesis focuses exclusively on equity assets, also know as stocks and shares. The tasks investigated in this thesis meet the necessities of two broad areas in the financial industry, Risk Management and Quantitative Trading. Generally speaking, the first area focuses on predicting the risk of an investment. Volatility, as measured by the standard deviation of stock returns,[1] is often taken to represent an asset's risk [103]. Thus, the first task studied in this thesis is short-term (one-day ahead) volatility forecasting. The Quantitative Trading area involves learning patterns in different sources of data (e.g. financial, economic and news articles) in order to identify profitable investment strategies. The task focuses on how data can be used to buy and sell stocks.

In order to further develop the performance of current solutions for tasks in the areas of Risk Management and Quantitative trading, the aims of this thesis are to:

1. Propose a multimodal neural network architecture that can use information from stock price time series (price modality) and news articles (textual modality) for the task of short-term volatility prediction. In particular, the architecture aims to learn a joint representation [15] of both modalities (news + prices) in an end-to-end fashion. Moreover, the proposed architecture should be able

[1] A return is the percentage change in price

to model three important aspects: (1) news relevance (2) news novelty, and (3) the fact that news items are events, which are different from the regular stock price time series, forming an irregularly spaced time series [54].

2. Propose neural networks that can inject the relational bias in a graph into the learning process. The motivation for encoding prior knowledge in the form of a relational bias is to increase sample efficiency and potentially achieve better performance. In general, the use of prior knowledge is particularly important for tasks that use financial data; unlike images or text, no additional data can be labelled in the case of a financial time series. Recently, Kipf and Welling [106] proposed Graph Convolutional Networks (GCN) in a setting where the data itself is represented by a graph (e.g. protein-protein interaction networks and knowledge graphs). In this thesis, we introduce a novel graph neural network, the Graph Transformer Network (GTN). Even though the stock time series data is not structured in the form of a graph, we apply GTN in a setting where each stock is a node of a graph and the relationships between stocks are edges. In particular, we investigate the use of relational bias in the range prediction task by evaluating the performance of the range trading quantitative strategy.

3. Investigate the use of Deep Reinforcement Learning (RL) in the context of quantitative trading. In particular, we are the first to approach the pair trading strategy [71] using RL. Importantly, aiming to meet the needs of the financial industry, this thesis proposes flexible Deep RL architectures that do not need to be retrained when the number of stocks under analysis changes. One particular drawback of employing the Supervised Learning (SL) paradigm to quantitative trading is having to rely on a pipeline approach in order to convert predictions into trading actions. To make it clear, when using SL for quantitative trading a time-series history is used as input in order to train a regression (price return) or classification (price return direction) problem. An additional layer of logic is then necessary to convert predictions into trading actions. Common to many studies, this layer of logic comes in the form of a fixed trading rule: If the prediction is "up" buy and if the prediction is "down" sell short.[2] Thus, using SL for quantitative trading means relying on a *history-prediction-action* pipeline. The main motivation for using RL is to effectively avoid this pipeline approach. Rather than first solving the prediction problem using SL and relying on a fixed rule to convert the prediction into actions, we employ Deep RL to learn a direct mapping from price history to trading actions. In particular, we consider general Deep RL methods capable of learning discrete [131] and continuous actions [115] and evaluate these methods in the specific context of the pair trading strategy.

[2]To the reader not familiar with the term sell short, this type of trade will be explained in detail below. For the moment, consider that selling short is a type of order that allows a profit when the price of a stock falls.

In the next section, we put into context the financial problems addressed in this thesis and the respective methods.

1.1.1 Research in the Context of Risk Management

Forecasting stock market volatility has attracted a great deal of interest in many areas of the financial services industry, including investment banking, commercial banking and insurance [158]. After the financial crisis of 2008, the interest in forecasting volatility was scaled on the regulatory side. Governments around the world imposed risk-based capital adequacy, precise the one-day regulatory Value at Risk (VaR) [1]. As a consequence , the financial industry demands accurate models to forecast short-term volatility, precisely one-day ahead.

From the economics community viewpoint, considerable effort has been focused on proposing parametric models for short-term volatility forecasting. The relevance of these efforts was recognised with the Nobel Memorial Prize in Economic Sciences (2003) to Robert F. Engle and Clive Granger for methods of analysing volatility, in particular, the Autoregressive Conditional Heteroscedasticity (ARCH) model [56]. This model captures many stylised facts, among them the fact that volatility is time-varying, exhibits a significant auto-correlation and tends to cluster; therefore, periods of high (or low) volatility persist for a long time. Nowadays, the ARCH family of models are a de facto standard for volatility forecasting and widely adopted in the financial services industry. Moreover, it has been shown that they are hard to beat [84].

The Machine Learning community has concentrated on considering textual data for volatility forecasting. In a seminal work, Kogan et al. [107] introduces a corpus based on the 10-K annual reports,[3] and their experiments show that textual data helps to predict volatility. This finding was corroborated by other research works [194, 183, 137, 152] which approach the volatility forecasting problem using multiple Natural Language Processing (NLP) techniques using the same 10-K corpus.

One limitation of these studies is that they rely on a pipeline approach for the volatility prediction. By using this approach, first a sentiment classifier is independently trained in order to predict whether a report is "good" or "bad". Then, at a later stage, this sentiment is merged with the stock price features. One clear limitation of the pipeline approach is that the sentiment classifier error propagates to the volatility prediction task. More importantly, from the financial industry viewpoint, previous studies are of limited applicability. Since each company releases only one 10-K report per year, previous studies only focus on *long-term* volatility forecasting (one quarter or one year ahead).

In order to meet financial industry needs, this thesis focuses on *short-term* (one-day ahead) volatility forecasting. To reach this aim, we had to first consider a corpus

[3]The Securities Exchange Commission (SEC) mandates that corporations fill in annual reports known as Form 10-K. Section 7, known as Management's Discussion and Analysis (MD&A), was used as a corpus. All reports are available to the public on the SEC's web site.

where text items are released at a higher frequency than the annual frequency of the 10-K reports. This led us to the first research contribution: In this thesis, we introduce a comprehensive corpus of approximately 150,000 news headlines, compiled at individual stock level and covering stocks in a broad range of sectors. Aiming to foster research in this area, we have made our financial news corpus freely available.

However, using news for short-term volatility prediction presents challenges not faced by previous studies. First, news can be released at any time. Therefore, we can have a situation where no news items are released for a given stock on a given day. In this sense, different from the regular time series of daily stock prices, news forms an irregularly spaced time series [54], where the time elapsed between observations is not constant. Second, rather than a single 10-K report per stock, many news items can be released during one day. To make things worse, some news is distracting; therefore, a proposed solution should be able to "distill" news items that really impact the market. Finally, another important aspect is the news novelty. To make it clear, when just released a news item can impact stock volatility. However, if the same news is repeated afterwards, there will be no impact, since the news has been already "priced in".

In order to tackle the challenges inherent in short-term volatility forecasting using price and news, in this thesis we introduce a novel neural network architecture called a Multimodal Hierarchical Attention Network (MHAN). In general, this architecture can be applied to forecasting problems where some modalities form an irregular time series (e.g. news or unforeseeable events). In particular, in the MHAN architecture irregularly or unevenly spaced time series are handled as part of the learning process; therefore, intervals that may have no textual modality data (e.g. days without news) are formulated as missing data. In other words, days without news are "marked" as absent, rather than just being skipped during the sequence encoding. Thus, the MHAN neural network is designed to learn a joint representation [15] of textual and price modalities in an end-to-end fashion.

Applying the MHAN architecture to short-term volatility forecasting led to our first research question (with experiments reported in Chapter 3):

Question 1: Can we improve the one-day ahead volatility prediction by adding textual data? Mainly, how does it compare to well-established econometric models that only use stock price data?

1.1.2 Research in the Context of Quantitative Trading Strategies

Quantitative trading strategies play an important role in the financial markets [182, 52]. A trading strategy (or trading system) is defined as follows [37, 182, 52]: (1) A set of *trading rules* to enter and exit trades, i.e. the point in time to trade the stock, and the trade type (buy, neutral or sell-short), and (2) a risk control mechanism.

Previous research [37, 89, 11] surveyed a large amount of studies focusing on predicting the stock return using AI techniques. These studies differ in terms of the input variables (e.g. by adding economic data), forecasting problem (i.e. regression for the stock return and classification for its direction) and forecasting methods (e.g. neural networks, random forest or genetic algorithms). However, once the forecasting model is trained, trading rules [37, 72, 12] are employed to evaluate the model's performance in the context of quantitative trading. To exemplify, let us consider that the model was trained on daily data using a binary classifier. Thus, the trading rules are as follows: (1) if the prediction is 1 enter a buy trade, otherwise enter a sell-short trade, (2) exit all trade when the market closes. Similar trading rules also apply to three-classes problems (i.e. buy, sell-short or neutral) or in the case of stock return predictions (regression problem). Therefore, the main focus of these studies is to predict the stock price return.

However, predicting the stock price volatility (i.e. risk) is different from predicting stock price return. In the volatility case, we attempt to predict how stable the stock returns will be in the future, rather than the stock performance or trend direction. While volatility is persistent and presents significant auto-correlation, it is received wisdom in economic science that predicting the stock return using public information (such as news or past stock prices) is an extremely difficult task. This is attributed to the fact that any available information is assumed to be incorporated into the stock price almost immediately, leading to the so-called Efficient Market Hypothesis (EMH) [60, 61]. In fact, many studies have shown [177, 61, 60] that the auto-correlation of stock returns is usually not statistically different from zero, suggesting that stocks prices follow a random walk.

In light of the EMH, we decided to adopt a conservative approach. Rather than attempting to predict the stock return (or return direction), which is not a feasible task according to EMH, we focused on market-neutral trading strategies, also called statistical arbitrage [98, 71, 182]. Broadly speaking, market neutral strategies attempt to profit from short-term price distortions, and the performance is expected to be independent of market direction [108, 98].

To this end, this thesis investigates two trading strategies: Range trading [127] and pair trading [71]. The range trading is explained as follows. First, a range (or a band) is predicted within which the price is expected to oscillate. Then, it follow the trading rule: if, in the next day, the stock price expands beyond the predicted range *in any direction*, it enters a trade contrarian to the current movement. This is done expecting the price has less ground to go beyond the predicted range. Therefore, we wait until the market closes to exit the trade. In a similar way, pair trading is also based on a rationale of a short-term distortion in prices, but relies on the idea of time series cointegration [79]. Therefore, if two stocks (a pair) share common (stochastic) trends, short-term divergences between the two stocks are expected to correct to their long term equilibrium.

Importantly, we observe that by focusing our efforts on the two strategies above, namely range trading and pairs trading, we approach the challenge of "profiting from the stock market" without having to predict the mean or direction of stock returns.

Effectively, both strategies discussed in this thesis can be profitable even in the case when stock prices follow a random walk process.

Even though previous studies [127, 193, 77] show that the High-Low strategy is profitable, they evaluate it on a small number of stocks. By contrast, in this thesis we use a higher number of stocks covering a broader range of sectors (precisely, 388 stocks and 11 sectors). We also evaluate the strategy for a large portfolio,[4] taking into account not only the strategy's profitability, but also for its risk.

Another limitation of these studies is that they do not take into account the effect of other stocks on the range prediction. A straightforward way to account for this effect is to make use of a neural network architecture that operates on the concatenation of the time series of all stock, i.e. the individual target stock and the remaining stocks, in order to learn a joint representation [15]. However, this architecture is not scalable, because by increasing the number of stocks we would need more data to discriminate the relationships. This problem is aggravated by the fact that deep learning is, still, a "data hungry" approach with low sample efficiency [125].

Recently, a class of neural networks, generally called Graph Neural Networks (GNN) [106, 81, 190, 18], have been proposed to model structured data (e.g. knowledge bases). Even though stock data is not structured in a graph, this thesis applies graph neural networks in a market graph setting. In this graph, each stock is represented by a node and the pairwise relationships between stocks by binary edges, i.e. one in the presence of a relationship and zero in the absence of one. Therefore, the market graph is used in this thesis in the context of relational bias, where different relationships can be readily encoded using prior knowledge about the stock market.

Aiming to push the performance in financial prediction further, we introduce a novel graph neural network, the Graph Transformer Network (GTN). The network is general, but applied and evaluated in this thesis on a range prediction task. More specifically, the GTN operates on the concatenation of stock time series, but also considers the market graph, i.e. the presence or absence of relationships between stocks, as part of the learning process.

As opposed to the Graph Convolutional Network (GCN) proposed in [106], the GTN uses an attention mechanism to learn the influences between a stock and its neighbourhood. In this sense, GTN is similar to Graph Attention Networks (GAT) proposed in [190]. However, our attention mechanism is inspired by the scaled dot-product attention proposed in [188], while GAT uses an additive attention. More importantly, by applying GTN to the range prediction task, our experiments show that the GTN outperforms GAT.

When expressing the relationship between stocks in a graph, we consider two extreme cases: (1) all stocks are related to each other (fully connected graph), or (2) the stock is not influenced by other stocks (identity graph). Case (1) is equivalent

[4]To the reader not familiar with the concept of portfolio, given a number of assets it defines the percentage of wealth that should be allocated to or invested in each asset.

to learning a model without injecting any bias, while (2) reduces the model to a setting where the effect of other stocks is not taken into account. However, between both extremes there are many ways to bias the learning process, e.g. by considering a market graph where only correlated stocks are connected [124].

As mentioned, by biasing our models using prior knowledge in a market graph, we expect to improve sample efficiency. On the other hand, a mismatched bias can also be detrimental to the learning process [18] by imposing constraints which do not reflect the true stock dynamics. Thus, the advantages and disadvantages of training the model using a market graph as a prior can be summarised in the following research question, with experiments reported in Chapter 4:

Question 2: In terms of risk-adjusted profitability, is there any advantage in biasing the range prediction model via a graph?

One clear benefit of using GTN is that we can analyse the attention weights after training. Common sense would suggest that some stocks are more sensitive to their neighbourhood than others. Deep learning works as a kind of a black-box model; therefore, attention weights have been used to help with the interpretability of deep learning models [74, 157]. Aiming the applicability in the financial markets of our GTN, guided to the following research question (with experiments reported in Chapter 4):

Question 3: How can the analysis of the attention weight matrix be applied to provide useful information for the financial services industry?

Even though the two trading strategies investigated in this thesis (Range trading and Pairs trading) are broadly classified as market-neutral strategies, they have different characteristics.

In the range strategy, we enter and exit the trade on the same day. In the Pairs trading strategy [71] a trade is entered when the pair history widens more than a given triggering value, and is only exited when the pair history converges to its long-term equilibrium. Specifically, Gatev et al. [71] proposes a triggering value of two standard deviations of the pair history with the trading rule as follows: if a pair history is above (below) two standard deviation they sell-short (buy) the pair. In particular, different from the Range strategy, it has been shown [71] that this convergence takes a long time, with an average trade horizon of 3.75 months.

Pairs trading has been extensively studied in the financial literature ([50, 24, 51, 69, 31, 114, 33, 95, 99, 59, 199]). Some research proposes different triggering values or attempts to optimise it. However, all studies rely on a trading rule to enter and exit the trade.

In this thesis, we approach the pair trading strategy from a completely different angle. We propose model-free RL [173, 131, 115]. Rather than using a trading rule to decide on when to enter and exit the trades and the trade type (i.e. buy or sell the pair short), we attempt to learn a mapping from the pair history to the trading actions.

To the best of our knowledge, we are the first to approach the pair trading strategy using model-free RL, i.e. in a “learning to trade” way. Here, the trading rules used in previous studies are abstracted away and replaced by a *pair trader agent*, which learns to trade by interacting with an environment.

In addition, because pair history is a continuous variable we use a class of recently proposed algorithms called Deep RL. More specifically, the Deep Q-Network (DQN) [131] and Deep Deterministic Policy Gradient (DDPG) [115]. The motivation for using these algorithms is that they achieve state-of-the-art performance in playing games [131] and in control tasks such as legged locomotion and driving vehicles [115].

On the other hand, the main motivation for using RL rather than SL, for the pair trading strategy is based on the long horizon of its trades, which is in the order of months. In other words, once we enter a trade, the pair takes a long time to converge to its long-term equilibrium. Therefore, on its way to the convergence, the immediate reward, as measured by the daily profitability, is less relevant. In other words, the effect of a trade (action) today takes a long time to mature in the future. This type of effect can be captured in the RL setting, since the goal is to optimise the cumulative discounted reward in a given horizon. By contrast, the SL optimisation only accounts for the immediate reward. Another limitation of SL is that it relies on a trading rule. However, since both paradigms (RL and SL) involve different optimisations, we define the following research question:

Question 4: In the specific case of pair trading, where trades take a long time to mature, does Reinforcement Learning provide better performance than Supervised Learning?

In line with previous studies, we also evaluate the pair trading strategy for a large number of stocks, i.e. in the context of a portfolio. Nonetheless, investors have different tolerances for risk and interest, with respect to the stocks they want to trade. In order to meet these necessities, we designed a flexible Deep RL architecture that is trained only once, and at execution time can be employed to output the optimal allocations for portfolios of any size and structure. Therefore, we avoid the problem of having to retrain our Deep RL models every time the number of stocks in the portfolio changes.

In addition, we introduce a framework, the *Investment Strategy with Investors’ preferences* (ISIP), which integrates the optimal portfolio allocations, as represented by a decision-making component, with a Risk management component. Thus, the framework was designed to keep the portfolio (i.e. the risk measure) at a level the investor is comfortable with. Broadly speaking, the ISIP operates in the following input-output space: Given the tolerance to risk and the portfolio constituent stocks as inputs, it outputs optimal portfolio allocations. This is done without having to retrain the time-consuming Deep RL models if the investors have portfolio of different sizes.

1.2 Research Contributions

This thesis addresses practical problems in the financial services industry and pushes forward the performance of state-of-the-art research in the area. By pursuing this goal, we contribute a number of tools that can be used in problems beyond the financial industry. Below we describe these contributions and how they were applied in this thesis.

- A comprehensive corpus available for conducting further research. This corpus contains 150,000 headlines of news articles released by Reuters from 2007 to 2017. It is compiled at the individual level for fifty stocks, with ten stocks for each of the following sectors: Consumer Staples, Energy, Financial, Healthcare and Utilities. This language resource is applied to the problem of short-term volatility prediction (Chapter 3).
- A sentiment Lexicon that consists of a dictionary with unigrams and bigrams with "good" and "bad" connotation. Lexicons are usually applied to directly classify the sentiment of a text when labelled data is not available[5].
- The MHAN is a neural network which can be applied to general forecasting problems with irregularly-spaced time series (e.g. web logs). In particular, we used this architecture for short-term (one-day ahead) volatility forecasting which takes textual and stock price modalities as input (Chapter 3).
- A graph neural network, the Graph Transformer Network (GTN), which can be employed in two types of settings: (1) when the problem structure can be encoded as a graph, and (2) when the data itself is represented by a graph (e.g. protein-protein interaction networks and knowledge graphs). The GTN learns the importance between two nodes using an attention mechanism. In this thesis, we applied GTN to range forecasting and range trading strategy evaluation. Thus, we use the GTN in settings where we encode the relationship between stocks in a graph and apply GTN in the context of quantitative trading, specifically in the range trading strategy (Chapter 4).
- A RL environment for pair trading. The implementation follow the widely used OpenAI Gym specification [30]. We employed this environment to train our Deep RL models, but the environment is model-agnostic and therefore, can be used by any RL model proposed by the research community. The environment was employed to train our Deep RL architectures (Chapter 5).
- A Deep RL architecture that can be applied to the general problem of portfolio allocations. Importantly, our architecture is decentralised. The models are trained only once and at execution time can be consumed for the optimisation

[5]This additional language resource was developed during the early stages of the PhD program and accepted for publication. However, we decided to add this in the Appendix A. In this thesis, this research is not taken as one of our three main contributions.)

of portfolios of *any size*. Thus, time-consuming Deep RL models do not have to be retrained to match the preferences of each investor with regards to the number of stocks he/she is interested in trading. We apply our Deep RL architecture to a quantitative trading strategy called Pairs trading (Chapter 5).

- A framework, *Investment Strategy with Investors' Preferences* (ISIP), which integrates the optimal portfolio allocations of a given model (i.e. the decision-making component) with a risk management component. At a high level, the ISIP operates in the following input-output space: given the investor risk tolerance and the portfolio constituent stocks as inputs, it outputs the portfolio weights adjusted to target a constant level of volatility, which is preset by the investor. Thus, the risk management component is designed to counter fluctuations in volatility. We apply this framework to the pair trading strategy, where the decision-making component is represented by the Deep RL model described above (Chapter 5).

1.3 Research Scope

This thesis contributes to research on volatility forecasting and quantitative trading strategies.

The quantitative trading strategies investigated in this thesis attempt to profit from short-term stock price distortions and are part of a class of market-neutral trading strategies, also known as statistical arbitrage. These strategies do not depend on predicting the direction of the stock price. Therefore, directional trading strategies are not examined in this thesis.

Trading strategies and volatility forecasting models can cover different asset classes, such as stocks, commodities, foreign currencies and fixed income. The scope of this thesis is limited to the stock market only. The application of the models proposed in this thesis to other asset classes will be investigated in future work.

Information about the state of financial markets comes from multiple data sources, including, but not limited to, time series of stock prices; time series of stock market indices; variables describing companies' earnings and cash flow; variables describing the current state of the economy, such as unemployment rate and inflation; and, financial textual data including news articles, press releases, financial statements, and official regulatory reports. These rich sources of data can potentially be used to derive meaningful information about future market performance. The models designed in this thesis only employ information extracted from datasets of historical daily stock prices and financial news headlines published about stocks. Thus, employing other data sources or intraday data is out of the scope of this thesis.

AI relies on the ability of a computer system to learn from data and experience. AI techniques have successfully addressed multiple real-world problems. These techniques include, but are not limited to, Bayesian inference, evolutionary computing and fuzzy logic. The scope of this thesis is restricted to the use of several DL approaches including SL and Deep RL.

1.4 Thesis Outline

Chapter 2 presents background concepts in the financial literature used throughout the thesis. These include econometric models for the volatility, self-financing portfolios, and Fama-French 3-factor model [62] for performance attribution analysis.

Chapter 3 investigates the impact of news headlines on the short-term (one-day ahead) volatility prediction. Previous studies [107, 194, 183, 137, 152] focus only on *long-term* volatility prediction (over a quarter or a year). In order to meet the demands of the financial industry, this chapter concentrates on the problem of *short-term* volatility prediction. To tackle this problem, a novel neural network architecture, the MHAN was introduced, which learns the joint representation of multiple data modalities in an end-to-end way. The MHAN architecture is then applied to jointly represent both textual modality (headlines of news articles) and stock price modality for one-day ahead volatility forecasting. Moreover, the MHAN also considers two essential aspects to model news in financial applications: (1) *news relevance*, which uses an attention mechanism, called News Relevance Attention (NRA), to aggregate all news released on a given day into a single representation, and (2) *news novelty*, which encodes the past history of the daily news and operates on top of NRA representations. Different sentence encoders are investigated and sentence representations transferred from other tasks. Experiments show that sentence representations trained on a natural language inference (NLI) task [29] achieve the best results in terms of transfer accuracy; this result is in line with previous studies [41]. However, these representations do not perform better than sentence encoders fine-tuned for the volatility task. Moreover, different ablations show that the News Relevance Attention (NRA) outperforms previous alternatives [49, 145]. These make use of averaging as the aggregation method, rather than an attention mechanism, and are not able to capture news relevance. Finally, experiments across different market sectors show that adding textual data consistently improves the performance of one-day ahead volatility predictions compared to using price only data.

Chapter 4 deals with the range trading strategy which attempts to profit from short-term distortion in prices within a trading day. In this strategy, the trading rules are directly given in terms of the stock range prediction. More specifically, next day prediction sets the range within which the price is expected to oscillate. If the price expands beyond this range in any direction, a trade is entered, as follows: short sell (buy) the stock if the price cross above (below) the range. Previous studies on range strategy [127, 193, 77] are extended by (1) evaluating a larger number of stocks, (2) taking into account not only the profitability, but also the risk side of the range trading strategy, (3) taking into account the effect of other stocks in a target stock range prediction, and (4) proposing a novel graph neural network, the Graph Transformer Network (GTN). The GTN is then applied to inject prior knowledge, encoded in a graph, to the stock range prediction problem. Using the prior knowledge that stocks in the same sector are closely related, a sector graph is introduced: stocks are nodes, and the edges have value one if two stocks are within the same sector and zero otherwise. In order to evaluate the advantages of using prior knowledge in the range prediction problem, results using the sector graph are contrasted with a fully connected graph, in which all stocks are related to each other. A further evaluation setting contrasts GTN with the current state-of-the-art graph network, Graph Attention Networks (GAT) [190]. Overall, experiments for a portfolio with a large number of stocks corroborate previous findings [127, 193, 77] showing the range trading strategy is profitable. However, different from the assumptions in the literature [193], the sectoral results show the range strategy is not sensitive to the level of volatility. Experimental results demonstrate that GTN outperforms the alternative GAT for the range prediction problem; indicating the potential of the GTN architecture proposed in this thesis. Substantially, analysis using a statistical method robust to multiple pairwise comparisons definitely shows that there are clear advantages in biasing the range predictions using the sectoral graph proposed in this thesis.

Chapter 5 presents the *Investment Strategy with Investors' Preferences* (ISIP) framework and investigates the use of model-free Deep RL in the pair trading strategy. The ISIP framework is unique in the sense that it integrates optimal portfolio allocations with a risk management component, where this component dynamically leverages the portfolio at times of low market volatility and scales down at times of high market volatility. In other words, the risk management component attempts to target a constant level of risk pre-set by the investor. Similar to previous studies [95, 33, 191, 69], the cointegration method [79] is applied at the pairs screening step in order to find two stocks that move in tandem. Nonetheless, different from previous studies [50, 24, 51, 69, 31, 114, 33, 95, 99, 59, 199], the pair trading strategy is approached using model-free Deep RL [131, 115], where a pair history (i.e. observations) is directly mapped to a trade (i.e. actions). Thus, rather than relying on trading rules, a *pair trader agent* learns to trade by interacting with a trade environment, which is introduced in this thesis. Experiments show that the Deep RL method improves on the performance of the pipeline approach common to previous studies [11], where first a supervised learning method is used to predict the return direction and then a trading rule is put in place to evaluate the trading strategy. In addition, results show the effectiveness of the ISIP framework in restricting the portfolio volatility to a level that the investor is comfortable with.

Chapter 6 provides a summary of the findings of the thesis, addresses the research questions and proposes future work.

Part II

Background

Chapter 2

Background concepts in Finance

2.1 Volatility: The Risk Side

This section reviews background concepts related to volatility. These concepts are used in Chapter 3, which deals with volatility forecasting models using price and news articles as data sources.

2.1.1 GARCH Model

On a given day t, the stock price return r_t is given in terms of the difference in prices p over one day period. That is,

$$r_t = \ln \frac{p_t}{p_{t-1}}.$$

A widespread econometric model used to forecast volatility of price returns is the Generalised Autoregressive Conditional Heteroscedasticity (GARCH) [56, 26]. The GARCH(p,q) model is specified in terms of the lagged parameters p and q. Hansen and Lunde [84] compared GARCH(p,q) with 330 different econometric volatility models showing that they are not significantly better than GARCH(1,1). In this thesis, we use GARCH(1,1) forecasts as a baseline to evaluate our volatility models. Henceforth, we refer to GARCH(1,1) simply as GARCH.

The GARCH model considers that price returns r_t have a time-varying volatility σ_t, and uses the following specification:

$$r_t = \mu + \epsilon_t \tag{2.1}$$

$$\epsilon_t = \sigma_t z_t \tag{2.2}$$

$$\sigma_t^2 = a_0 + a_1 \epsilon_{t-1}^2 + b_1 \sigma_{t-1}^2, \tag{2.3}$$

where μ is a constant (return drift) and z_t is a sequence of i.i.d. random variables, where the distribution is assumed to be a Gaussian noises with mean zero and unit variance. In the GARCH model the conditional mean return described in Equation (2.1) has a constant value, but its volatility σ_t is time-dependent.

Forecasting

The h-day variance forecast conditioned on information at time t can be computed from Equation (2.3) recursively. For h=1 we have

$$\widehat{\sigma}_{t+1}^2 = a_0 + a_1 \epsilon_t^2 + b_1 \widehat{\sigma}_t^2 \tag{2.4}$$

One interesting property of the GARCH model is that for long horizons the volatility forecast reverts to its *unconditional volatility* σ_u. That is [207]:

$$\begin{aligned} &\lim_{h\to\infty} \widehat{\sigma}_{t+h} = \sigma_u, \quad \text{where} \\ &\sigma_u = \sqrt{a_0/(1-a_1-b_1)} \end{aligned} \tag{2.5}$$

We observe that previous studies, which also use news and prices for volatility forecasting, focus on long horizons (e.g. quarterly or annual), where the models are evaluated using *unconditional* volatility. By contrast, this thesis proposes neural network architectures for short-term volatility forecasting (specifically one-day ahead) and uses the GARCH *conditional* forecasts in Equation (2.4) for evaluation purposes.

Evaluation

Let σ_{t+1} denote the ex-post "true" daily volatility at a given time t. The performance on a set with N daily samples can be evaluated using the standard Mean Squared Error (MSE) and Mean Absolute Error (MAE)

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(\widehat{\sigma}_{t+1} - \sigma_{t+1})^2 \tag{2.6}$$

$$MAE = \frac{1}{N}\sum_{t=1}^{N}|\widehat{\sigma}_{t+1} - \sigma_{t+1}| \tag{2.7}$$

Additionally, following [7], the models are also evaluated using the coefficient of determination R^2 of the regression

$$\sigma_{t+1} = a + b\widehat{\sigma}_{t+1} + e_t \tag{2.8}$$

where

$$R^2 = 1 - \frac{\sum_{t=1}^{N} e_t^2}{\sum_{t=1}^{N}\left(\widehat{\sigma}_{t+1} - \frac{1}{N}\sum_{t=1}^{N}\widehat{\sigma}_{t+1}\right)^2} \tag{2.9}$$

One of the challenges in evaluating GARCH models is the fact that the ex-post volatility σ_{t+1} is not directly observed using daily prices. In other words, we need intraday price returns to estimate the daily volatility. One potential solution would be to directly use the next day squared returns as a proxy for the variance. However, it was shown [7] that even in cases where the price returns follow a GARCH process, using square returns leads to a very low coefficient of determination.

In the next section, we present volatility estimators that are more efficient than squared returns, these estimators are used in Chapter 3 as a proxy for the ex-post daily volatility.

2.1.2 Price Range as Proxy for Volatility

Let O, H, L, C be the open, high, low and close prices of a stock on a given day. Assuming that the daily price follows a geometric Brownian motion with zero drift and constant daily volatility σ, Parkinson [141] derived the first daily volatility estimator:

$$\widehat{\sigma}_{PK} = \sqrt{\frac{\ln\left(\frac{H}{L}\right)^2}{4\ln(2)}} \tag{2.10}$$

which represents the daily volatility in terms of its price range, and contains information about the price path during the day. Parkinson's volatility estimator was extended by Garman-Klass (GK) [70] by incorporating additional information about the opening (O) and closing (C) prices and is defined as:

$$\widehat{\sigma}_{GK} = \sqrt{\frac{1}{2}\ln\left(\frac{H}{L}\right)^2 - (2\ln(2)-1)\ln\left(\frac{C}{O}\right)^2} \tag{2.11}$$

The relative noise of different estimators $\hat{\sigma}$ can be measured in terms of relative efficiency to the daily volatility σ and is defined as

$$e\left(\widehat{\sigma^2}, \sigma^2\right) \equiv \frac{Var[\sigma^2]}{Var[\widehat{\sigma}^2]} \tag{2.12}$$

where $Var[\cdot]$ is the variance operator. It follows directly from Equation (2.2) that the squared return has efficiency 1 and is therefore very noisy. Molnár and Molnar [132] shows that the Parkinson ($\widehat{\sigma}^2_{PK}$) volatility estimator has 4.9 relative efficiency and Garman-Klass ($\widehat{\sigma}^2_{GK}$) 7.4. Additionally, all described estimators are unbiased.

Many alternative estimators to daily volatility have been proposed in the literature. However, experiments in [132] rate the Garman-Klass volatility estimator as the best based only on open, high, low and close prices. Given the high efficiency of $\widehat{\sigma}_{GK}$, it is used as the target variable to train our models. However, we evaluate our models using both $\widehat{\sigma}^2_{PK}$ and $\widehat{\sigma}^2_{GK}$.

2.2 Quantitative Trading Strategies: The Profitability Side

In this section we review concepts in the financial literature that are related to the quantitative trading strategies investigated in this thesis, namely range trading in Chapter 4 and pair trading in Chapter 5.

2.2.1 Long and Short Trades

An order is an instruction to buy or sell a stock at a specified price. A trade is the execution of the order. Unless specified, in this thesis, we use the word trade and position interchangeably. A long trade (also know as going long) is initiated by buying a stock. A short trade (also known as going short or short-sell or short selling) is initiated by selling a stock that was not previously bought. This is done by borrowing a stock and selling it to the market. A short-sell is implemented as follows:

1. Investor A (the short-seller) borrows n stocks from Investor B
2. Investor A sells the stocks on the market, hoping the price will fall
3. After achieving profit goals or in order to cut losses, investor A *covers* the short position by buying the n stocks from the market.
4. Finally, the n stocks are returned to Investor B, and Investor A incurs a profit or loss at no cost to Investor B.

Typically, the stock lender (Investor B) has a long-term view on the stock appreciation and is interested in lending the stock to the short-term Investor A, for an agreed period, in exchange for a fee (i.e. the borrowing cost). On the other hand, Investor A sells stocks that he/she does not own expecting that they can be bought at a lower price in the future.

Overall, a long position is used to bet that the stock price will go up and the short position, also called a bearish position, that the stock price will go down (bullish position). Thus, short selling is a way to profit in a falling market. However, it presents unique risks [55, 103]. To exemplify, by buying a stock, the maximum loss is limited to the initial investment. Conversely, because the stock price can continue to go up, theoretically, there is no limit to the amount that can be lost in a short position.

2.2.2 Self-financing and Buy and Hold portfolios

In order to emphasise that the portfolio allocations can vary over time, we use the superscript t to refer to periods and the subscript i to assets. A portfolio is a collection of assets. On a given day t the portfolio allocations (or weights) $w^t = [w_1^t, \cdots, w_N^t]^\intercal$ represents the percentage (or proportion) of wealth invested on each of the N assets. The portfolio return is given by

$$R^t = \sum_{i=1}^{N} w_i^t \cdot R_i^t, \tag{2.13}$$

where R_i^t is the i^{th} asset return.

The Buy and Hold (BAH) portfolio simulates the performance of a naive trading strategy that equally distributes the same amount of wealth between all stocks. This portfolio considers only long trades and the weights are given by:

$$\begin{aligned} w_i^t = w = 1/N &\quad \textit{all weights are equal} \\ \sum_{i=1}^{N} w_i = 1 &\quad \textit{sum all weights is one} \end{aligned} \tag{2.14}$$

Because the weights in a BAH portfolio do not change over time, it reflects the performance of a passive investment strategy. This is in contrast with active trading strategies that dynamically decide on what stocks to go long or short on. Giving these characteristics, the BAH portfolio is commonly used as a baseline to evaluate the performance of active trading strategies.

A *long and short* (LS) portfolio is composed of long and short positions. The main advantage of LS compared to long only portfolios (e.g. BAH), is that the short positions can reduce exposure to the market. In fact, both market-neutral strategies are investigated in this thesis; range trading and pair trading are evaluated for portfolios with a large number of stocks and having long and short positions.

The *self-financing* portfolio is an LS portfolio where all the long trades are funded by short trades, such that the sum of all weights is zero [10, 5, 62]. That is,

$$\begin{aligned} \sum_{i=1}^{N} w_i^t \cdot \mathbb{1}\left(w_i^t < 0\right) = -\sum_{i=1}^{N} w_i^t \cdot \mathbb{1}\left(w_i^t \geq 0\right) &\quad \textit{sum of long and short weights are equal} \\ \sum_{i=1}^{N} w_i^t = 0 &\quad \textit{sum of all weights is zero} \end{aligned} \tag{2.15}$$

Here, we have the number of long positions $N_l^t = \sum_{i=1}^{N} \mathbb{1}\left(w_i^t \geq 0\right)$ plus the number of short positions $N_s^t = \sum_{i=1}^{N} \mathbb{1}\left(w_i^t < 0\right)$ equal to the total number of stocks N in the portfolio, i.e. $N = N_l^t + N_s^t$.

The self-financing portfolio can be better understood in terms of its implementation. First, the stocks to go short are borrowed and immediately sold on the market (short-sell trade). In this transaction, cash is received in exchange for the sold stocks. Then, this same cash is used to buy the stocks to go long. Since the proceeds of the short-sale are used to purchase the stocks, no investment is needed in order to build a self-financing portfolio, and a self-financing portfolio is also called a *zero-investment portfolio* [5, 62]. It is worth noting that zero-investment portfolios largely differ from long only portfolios, such as BAH. Since positions are self-funded, stocks can be bought in an amount larger than the available wealth [98]. Another important fact is that a zero-investment portfolio uses short-selling; therefore, there is no theoretical limit to the loss.

In this thesis, we propose different models for range trading and pair trading strategies. At a high-level, given a portfolio of stocks, a model defines which stocks go long (positive weights) and which short (negative positions). However, in our studies this portfolio is always zero-investment, i.e. the sum of all weights is zero and the percentage of wealth allocated to long and short positions is equal. In addition, when evaluating our portfolios we consider not only the profitability side, but also the risk. To this end, the portfolio weights are dynamically scaled to target the same level of ex-ante volatility σ_{target}. This scaling method is discussed in the next section.

2.2.3 Position Sizing

In the context of trading strategies, it is important to evaluate portfolios using an integrated framework where the performance is evaluated in the light of risk. According to Cavalcante et al. [37], apart from trading recommendations, real-world applications should consider a *money management* mechanism, which is usually neglected by studies in the AI community.

> The money management mechanism. . . manages the position size, i.e. the amount of resources to be used in a trade considering the total capital available and the risk involved in the trade. . . Some work proposes an intelligent forecasting mechanism, but the majority provide no rules to negotiate in the market or even to manage investment risks. – Cavalcante et al. [37].

The relevance of this mechanism is also discussed in [182], and in the context of high-frequency trading strategies, in [4].

In this thesis, we adopt the *volatility targeting* method [96, 34] as a money management mechanism, i.e. in order to manage the amount of capital/wealth allocated to a trade. Thus, once the zero-investment portfolio of each model is obtained, we scale down the portfolio at times of high volatility (high risk) and leverage at times of low volatility. Effectively, the portfolio is always targeting a constant level of ex-ante volatility σ_{target}. Below we provide details on the volatility targeting formulation [96, 34].

Given a portfolio allocation w^t, collect T historical samples of portfolio returns, defined by:

$$R^{t-t'} = \sum_{i=1}^{N} w_i^t \cdot R_i^{t-t'}, \quad \text{with } t' \in \{1, \cdots, T\} \tag{2.16}$$

After the T samples of historical returns are obtained, the portfolio volatility $\hat{\sigma}^t$ is estimated using exponentially decaying weights. Therefore, the volatility is updated based on its previous estimate and the previous portfolio return R^{t-1} using the recurrence:

$$\left(\hat{\sigma}^t\right)^2 = (1-\lambda)\left(\hat{\sigma}^{t-1}\right)^2 + \lambda\left(R^t\right)^2, \quad \text{with } 0 < \lambda \leq 1 \tag{2.17}$$

Finally, the positions are sized using the same scaling factor γ^t. That is,

$$\begin{aligned} \gamma^t &= \frac{\sigma^{\text{target}}}{\hat{\sigma}^t} \\ w_i^t &\leftarrow \gamma^t \cdot w_i^t, \quad \text{with } i \in \{1, \cdots, N\}. \end{aligned} \tag{2.18}$$

In our experiments, all portfolios are sized on a daily basis, in order to manage the risks of the trading strategies. As in [34, 96], we keep the risk, as measured by the volatility, at constant annualised volatility of $\sigma_{\text{target}} = 10\%$, and we also use the same parameters $T = 207$ (i.e. $\sim$ 1-year), and $\lambda = 0.994$ (i.e. 90-day half-life) for the formulations above.

Importantly, in our experiments we show that the volatility targeting method is definitely effective in managing the trading strategies risks. Even though the portfolio is sized using an ex-ante volatility, as denoted in Equation (2.18), the *realised* volatility (ex-post) is very close to the target of 10%.

2.2.4 Common Factors in Stock Returns: Fama-French model

The difference in average returns across stocks has been widely studied in the financial community. In a seminal work, Banz [16] showed that a portfolio of small market capitalisation stocks (small caps) tends to perform better than a portfolio of large caps. Similarly, Hurst et al. [96] and Carhart [35] provide strong evidence of market momentum, i.e. stocks with the biggest returns over a year (i.e. winner stocks) continue to outperform past losers.

Fama and French [62] propose that stock and portfolio returns can be fully explained in terms of the three factors regression:

$$R_i^t - R_f^t = \alpha_i + \beta_i \left[R_M^t - R_f^t\right] + s_i SMB^t + h_i HML^t + \epsilon^t, \tag{2.19}$$

where

R_i	$\equiv$ stock/portfolio return
R_f	$\equiv$ risk-free asset return
R_M	$\equiv$ market portfolio (highly diversified)
SMB	$\equiv$ *S*mall *M*inus *B*ig portfolio
HML	$\equiv$ *H*igh *M*inus *L*ow portfolio

In the equation above, *SMB* and *HML* are returns of zero-investment portfolios built on deciles of stocks ranked by market capitalisation and book-to-market ratio, respectively. The regression intercept (alpha) captures the part of the portfolio/stock return that cannot be explained in terms of risk exposure.

Using the specification above, Fama and French [62] show that for a broad range of U.S. stocks alpha is not statistically different from zero and the regression presents a high coefficient of determination R^2. In other words, since the explanatory factors are shared among stocks the risk is undiversifiable. For this reason, these factors are also called common risk factors [62, 58].

Apart from explaining asset returns, the Fama-French three factors (FF3) have been widely used to study performance attribution in active investment management. Considering a trading strategy as a *black-box* model, performance attribution quantifies how much of the trading strategy could be replicated by simply taking passive exposure to FF3 factors. More specifically, if alpha in Equation (2.19) is statistically zero, it is possible to mimic the same stream of returns as the black-box model by taking exposure to common risk factors. In this case, the black-box trading strategy neither diversifies the risk nor discovers any additional pattern, and it is deemed unskilled. Another important point is that β, s and h, i.e. the level of exposure to risk factors, distils the sources of profitability. Surprisingly, experimental results in [64] show that the aggregate return of the U.S. mutual funds industry has alpha close to zero and $\beta = 1$ with very few funds presenting any statistically positive alpha. Similar results corroborating the poor performance of the investment industry are also reported in [17].

In this thesis, we regress the returns of our portfolios on the FF3 factors aiming to evaluate whether our models only learn patterns in returns that are known to earn significant profits.

Part III

Contributions

Chapter 3

Multimodal Deep Learning for Short-term Volatility Prediction

3.1 Introduction

Natural Language Processing (NLP) has increasingly attracted the attention of the financial community. This trend can be explained by at least three major factors. The first factor is the business perspective. This refers to the economics of gaining a competitive advantage using alternative sources of data going beyond historical stock prices; thus, trading by analysing market news automatically. The second factor is major advancements in the technology to collect, store and query massive amounts of user-generated data almost in real-time. The third factor refers to the progress made by the NLP community in understanding unstructured text.

Over the past decade the number of studies using NLP for financial forecasting has experienced exponential growth. According to Xing et al. [200], up to 2008, less than five research articles were published per year using both "stock market" and "text mining" or "sentiment analysis" as keywords. In 2012, this number increased to slightly more than ten articles per year. The numbers available for 2016 indicate this has increased to sixty articles per year.

The ability to mechanically harvest sentiment from text using NLP has shed light on conflicting theories of financial economics. Historically, there has been two differing views on whether disagreement between market participants induces more trades. The "non-trade theorem" [129] states that assuming all market participants have common knowledge about a market event, the level of disagreement between participants does not increase the number of trades, but only leads to a revision of market quotes. By contrast, the theoretical framework proposed in [85] advocates that disagreement between market participants increases trading volume. Using textual data from Yahoo and RagingBull.com message boards to measure the dispersion of opinions (positive or negative) among traders, it was shown [9] that disagreement between users' messages helps to predict subsequent trading volume and volatility. A similar relationship between disagreement and increased trading volume was found using Twitter posts [170].

Textual analysis is adding to the theories of medium-term/long-term momentum and reversal in stock markets [189]. The unified Hong and Stein model[1] [91] proposes that investors underreact to news, causing slow price drifts, and overreact to price shocks not accompanied by news, hence inducing reversals. The theoretically predicated behaviour between price and news has been empirically supported using financial media headlines [38, 28] and the Consumer Confidence Index® [8] published by The Conference Board [134]. Similarly, negative sentiment has been shown to be a good predictor of price returns and trading volumes [178].

Accurate models for forecasting both price returns and volatility are equally important in the financial domain. Volatility measures the degree of oscillation of an asset during a given time period and is related to the second moment of the price return distribution. In general terms, forecasting price returns is relevant in making a speculative decision, while the volatility, on the other hand, measures the risk of these decisions. On a daily basis, financial institutions need to assess the short-term risk[2] of their portfolios. Measuring risk is essential for the purposes of regulatory capital disclosures required by banking supervision bodies. Moreover, to maintain the risk within acceptable levels, it is useful to dynamically adjust position sizing according to market conditions.

Although, it is crucial to predict the short-term volatility from the financial markets application perspective, much of the current NLP research on volatility forecasting focuses on volatility prediction for very long-term horizons (see [107, 194, 183, 137, 152]). Predominately, these build on extensions of the bag-of-words representation, which has the main drawback of not capturing word order. Financial forecasting, however, requires the ability to capture semantics that is dependent on word order. For example, the headline *"Qualcomm sues Apple for contract breach"* and *"Apple sues Qualcomm for contract breach"* trigger different responses for each stock and for the market aggregated index. However, they share the same bag-of-words representation. Additionally, these studies use features from a pretrained sentiment analyis model to train the financial forecasting model. A key limitation of this approach is that it requires a labelled sentiment dataset and error propagation is not end-to-end.

In this work, we address the limitations of existing research in volatility prediction in the following manner:

1. To aid short-term daily volatility prediction we developed a financial news corpus. We compiled this corpus at the individual stock level, comprising Reuters news headlines of 50 stocks in 5 diversified sectors, with a total of

[1]The *gradual information diffusion* model of Hong and Stein considers two types of economic agents, namely "Newswatchers" and "Momentum traders", where three assumptions are made: 1) "Newswatchers" realise part of the public information and privately adjust their models, which are only based on macroeconomic and company specific forecasts. 2) "Momentum traders" only trade based on past price performance. 3) Private, rather than public, information diffuses gradually, since each agent has a different time frame to adjust their models. These assumptions about market agents are enough to model the relationship between news and long-term trends or short-term reversals.

[2]Usually, this risk is the conditional volatility for the next trading day

146,783 samples (2007–2017). We also collected daily stock prices from Yahoo Finance website for the 50 stocks.

2. We propose an end-to-end multimodal model that can jointly learn from daily stock prices and company news.
3. We investigate whether the textual mode is complementary or redundant for the short-term volatility prediction problem. Our results indicate that textual mode is complementary and improves the forecasting accuracy.
4. We evaluate how transferable the representations learnt in two different NLP tasks are to the specific problem of volatility forecasting.
5. We propose a hierarchical attention mechanism to effectively weight the most relevant news from the large amount of news released on a given day.

3.2 Related work

One previous study [107] incorporates sections from Form 10-K[3] to predict volatility, twelve months after the report is released. They train a support vector regression model on a bag-of-words feature vector (weighted by term frequency). This work was extended by employing *Loughran-McDonald Sentiment Word Lists* [121] containing words grouped by their sentiment categories (positive, negative and neutral) [194, 183, 137, 152]. In this later research, the bag-of-words representation of each 10-K document is further expanded by adding the top k most similar words. Additionally, dimensionality reduction methods such as PCA are applied to get improved results [152]. Primarily, only long-horizon volatility predictions (one year [194, 183, 137] or quarterly [152]) are addressed. Methods that combine both text and market price features train separate models for each modality [137, 152, 15].

In the context of predicting price direction (rather than the volatility), [25] builds an index of collective mood states by counting average Twitter word occurrences in the Profile of Mood States (POMS). The index is then used to predict the Dow Jones Index direction. Similarly, [160] makes use of handcrafted text representations including term count, noun-phrase tags and extracted named entities. Finally, an extension of Latent Dirichlet Allocation (LDA) is proposed in [136] to learn a joint latent space of topics and sentiments.

Our deep learning model is similar to existing work on predicting price direction [49, 145]. However, neither leverage both news and price data in a joint model. Instead, only text data is employed for predicting price direction. Ding et al. [49] preprocess headline news using *Stanford OpenIE* to generate triples that are fed into a Neural Tensor Network [169] to create the final headline representation. Pinheiro and Dras [145] pre-train a character-level embedding in an unsupervised manner and

[3]US based companies are enforced by the Securities and Exchange Commission (SEC) to file Form 10-K reports on an annual/quarterly basis. These forms provide an overview of the company's business and financial health. A Form 10-K example can be found here

a sequence model is used to learn the headline representation. Both studies [49, 145] average all headline representations on a given day, rather than attempting to weight the most relevant ones. In this work, we propose a joint text and stock learning model that employs a hierarchical attention mechanism to automatically weight the news relevance conditional on the stock data.

Despite the fact that end-to-end deep learning models have attained state-of-the-art performance, the large number of parameters make them prone to overfitting. Additionally, end-to-end models are trained from scratch requiring large datasets and computational resources. Transfer Learning (TL) alleviates this problem by adapting representations learnt from a different and potentially weakly related source domain to a new target domain.

In this work, we consider TL in our experiments for two main reasons. First, it addresses the question of whether our proposed dataset is suitable for end-to-end training, as the performance of the transferred representations can be compared with end-to-end learning. Second, it is still to be investigated which source domain transfers better to the specific forecasting problem. Recently, the NLP community has focused on universal representations of sentences [41, 92], which are dense representations that carry the meaning of a full sentence. In this work, we investigate the suitability of transfering the sentence encoders trained in the Stanford Natural Language Inference (SNLI) [29] and Reuters RCV1 [111] datasets to the volatility forecasting task.

3.3 Background

3.3.1 Sequence models

We start this section by reviewing the Recurrent Neural Network (RNN) architecture and its application to encode a sequence of words.

An RNN is capable of handling variable-length sequences, this being a direct consequence of its recurrent cell, which shares the same parameters across all sequence elements. In this work, we adopt the Long Short-Term Memory (LSTM) cell [90] with forget gates f_t [73]. The LSTM cell is endowed with a memory state that can learn representations that depend on the order of the words in a sentence. This makes LSTM more fit to find relations that could not be captured using standard bag-of-words representations.

Let $x_1, x_2, \cdots, x_T$ be a series of observations of length T, where $x_t \in \mathbb{R}^{d_w}$ and d_w is the embedding dimension[4]. In general terms, the LSTM cell receives a previous hidden state h_{t-1} that is combined with the current observation x_t and a memory state C_t to output a new hidden state h_t. This internal memory state C_t is updated

[4]We use the subscript w to refer to "words" in the sense that to encode a headline with T words we learn a sentence representation. In general, d_w can also represent any dimension, e.g. if x refers to an input vector of price returns, then d_w is the size of this vector. In this case, we would be interested in encoding a history with T price observations.

depending on its previous state and three modulating gates: input, forget and output. Formally, for each step t the updating process goes as follows (see Figure 3.1 for a high level schematic view): First, we calculate the input i_t, forget f_t and output o_t gates:

$$i_t = \sigma_s \left(W_i x_t + U_i h_{t-1} + b_i\right) \tag{3.1}$$

$$f_t = \sigma_s \left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{3.2}$$

$$o_t = \sigma_s \left(W_o x_t + U_o h_{t-1} + b_o\right) \tag{3.3}$$

where σ_s is the sigmoid activation. Second, a candidate memory state $\widetilde{C}_t$ is generated:

$$\widetilde{C}_t = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \tag{3.4}$$

Now we are in a position to set the final memory state C_t. Its value is modulated based on the input and forget gates of Equation (3.3) and is given by:

$$C_t = i_t \odot \widetilde{C}_t + f_t \odot C_{t-1} \tag{3.5}$$

Finally, based on the memory state and output gate of Equation (3.3), we have the output hidden state

$$h_t = o_t \odot \tanh\left(C_t\right) \tag{3.6}$$

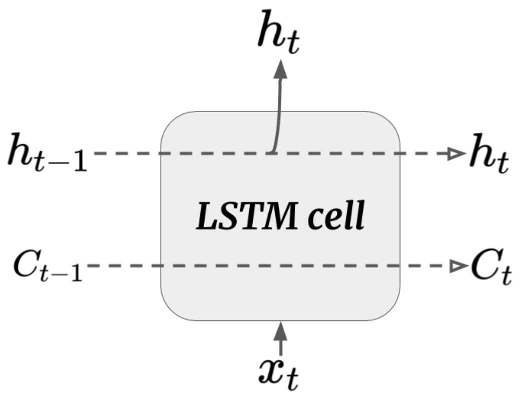


Figure 3.1: **Schematic view of a LSTM cell**. The observed state x_t is combined with previous memory and hidden states to output a hidden state h_t. The memory state C_t is an internal state, therefore, not part of the output representation. An LSTM network is trained by looping its shared cell across all sequence lengths.

Regarding the trainable weights, let n be the LSTM cell number of units. It follows that Ws and Us matrices of the affine transformations have $n \times d_w$ and $n \times n$ dimensions, respectively. Its bias terms bs are vectors of size n. Consequently, the total number of parameters is $4(nd_w + n^2 + n)$ and does not depend on the sequence number of time steps T.

Given a sequence of words $\{w_t\}_{t=1}^{T}$ we aim to learn the word's hidden state $\{h_t\}_{t=1}^{T}$ in a way that each word captures the influence of its past and future words.

The Bidirectional LSTM (BiLSTM) proposed in [161] is an LSTM that “reads” a sentence, or any sequence in general, from the beginning to the end (forward) and the other way around (backward). The new state h_t is the concatenation

$$h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}] \tag{3.7}$$

where

$$\overrightarrow{h_t} = \text{LSTM}\left(w_1, \cdots, w_T\right) \tag{3.8}$$
$$\overleftarrow{h_t} = \text{LSTM}\left(w_T, \cdots, w_1\right) \tag{3.9}$$
$$\tag{3.10}$$

Because sentences have different lengths, we need to convert the T concatenated hidden states of the BiLSTM into a fixed-length sentence representation. One straightforward operation is to apply any form of pooling. Attention mechanism is an alternative approach where the sentence is represented as a weighted average of hidden states where the weights are learnt end-to-end.

In the next sections we describe sentence encoders using pooling and attention layers.

BiLSTM max-pooling

The max-pooling layer aims to extract the most salient word features throughout the sentence. Formally, it outputs a sentence vector representation $S_{MP} \in \mathbb{R}^n$ such that:

$$S_{MP} = \max_{t=1}^{T} h_t \tag{3.11}$$

where h_t is defined in Equation (3.7) and the max operator is applied over the time steps dimension. Figure 3.2 illustrates the BiLSTM max-pooling (MP) sentence encoder.

The efficacy of the max-pooling layer is assessed in a large number of NLP studies. Lai et al. [109] employs a max-pooling layer on top of word representations and argues that it performs better than mean pooling. Experimental results in [41] show that among three types of pooling (max, mean and last[5]) the max-pooling provides the most universal sentence representations in terms of transferring performance to other tasks. Grounded in these studies, in this work, we choose the BiLSTM max-pooling as our pooling layer of choice.

BiLSTM attention

Attention mechanisms were introduced in the deep learning literature to overcome some of the simplifications imposed by pooling operators. When humans read a

[5]The “last” polling is a simple operator that takes only the last element of the T hidden states to represent a sentence.

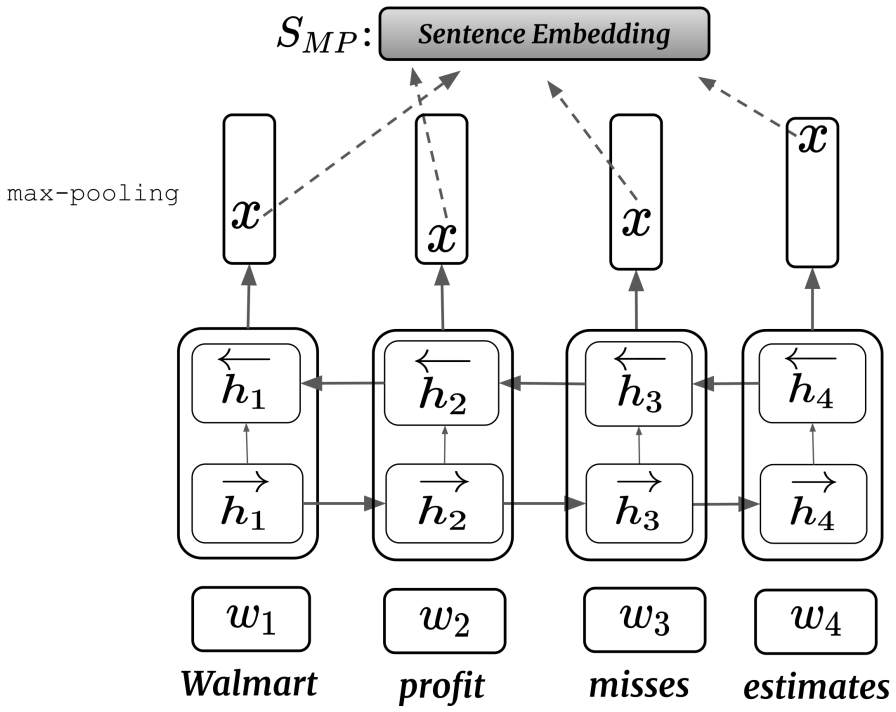

S_{MP}: Sentence Embedding
max-pooling
x
$\overleftarrow{h_1}$ $\overleftarrow{h_2}$ $\overleftarrow{h_3}$ $\overleftarrow{h_4}$
$\overrightarrow{h_1}$ $\overrightarrow{h_2}$ $\overrightarrow{h_3}$ $\overrightarrow{h_4}$
w_1 w_2 w_3 w_4
Walmart profit misses estimates

Figure 3.2: **BiLSTM max-pooling**. The network performs a pooling operation on top of each word hidden state.

sentence, we are able to spot the most relevant parts in a given context and disregard information that is redundant or misleading. The attention model aims to mimic this behaviour. A self-attentive sentence [112, 120, 116] assigns different weights to hidden states and converts all hidden states into a single vector representation.

Considering the word hidden vectors set $\{h_1, \cdots, h_T\}$ where $h_t \in \mathbb{R}^n$, the attention mechanism is defined by the equations:

$$\begin{aligned} \tilde{h}_t &= \sigma\left(W h_t + b\right) \\ \alpha_t &= \frac{\exp(v^{\intercal} \cdot \tilde{h}_t)}{\sum_t \exp(v \cdot \tilde{h}_t)} \\ S_{A_w} &= \sum_t \alpha_t h_t \end{aligned} \tag{3.12}$$

where $W \in \mathbb{R}^{d_a \times n}$, $b \in \mathbb{R}^{d_a \times 1}$, and $v \in \mathbb{R}^{d_a \times 1}$ are trainable parameters.

We can see that the sentence representation S_{A_w} is a weighted average of the hidden states. Figure 3.3 provides a schematic view of the BiLSTM attention, where we describe the attention in Equation (3.12) as a two layer model with a dense layer (d_a units) followed by another dense layer that predicts α_t (single unit).

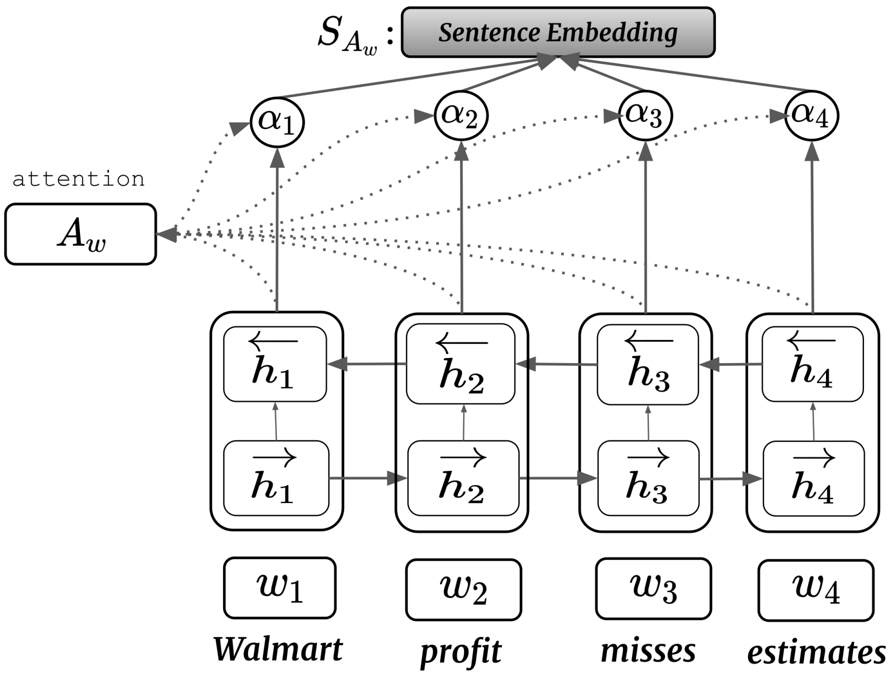


Figure 3.3: **BiLSTM attention**. The specific example encodes a headline from our corpus.

3.4 Methodology

In this section, we first briefly introduce our dataset. We then present our neural architecture that incorporates the aspects of relevance and novelty – crucial to correlate news with market variables. Finally, we introduce the idea of learning a global model, i.e. valid for all stocks, that was designed to capture part of the idiosyncrasies of each stock.

3.4.1 News Headlines Corpus

We compiled a corpus covering a broad range of 5 business sectors (Consumer Staples, Energy, Utilities, Healthcare and Financials) consisting of 50 stocks with news headlines grouped at stock level. A total of 150,000 headlines were collected from the Reuters Archive. Importantly, our dataset spans 10 years of news data (2007-2017), a period large enough to capture different market regimes. Details on the corpus compilation process are described in the next section. Below, we highlight its main features:

- **Comparison with 10-K dataset**: Our corpus is composed of daily financial news articles. Thus, it also includes headlines surrounding earnings release dates. Thus, it complements the 10-K dataset used in previous volatility studies. As an illustration, the headlines *"Walmart warns that strong U.S. dollar will cost \$15B in sales"* and *"Procter & Gamble Co raises FY organic sales growth forecast after sales beat"* are typical content of the "Management's Discussion and Analysis of financial conditions and results of operations" (MD&A)

section of 10-K reports. The main advantage is that news are released in a frequency higher than in the 10-K reports, where only one report is released per year for each stock.

- **Realistic timezone adjustments**: Since our corpus contains news from around the world, we converted all release timestamps to the New York stock exchange time zone and grouped news into the following time categories: {`before market`, `during market`, `after market`, `holidays`, `weekends`}. This procedure allows us to use the method proposed in [9] to avoid unrealistic model performance. That is, news released after the market closes is shifted by one day, thus, not used to predict next day price returns.

- **Objective news**: As a financial agency, Reuters mainly reports news related to corporate events (e.g. lawsuits, mergers & acquisitions, research & development investments) and economic numbers. This being factual information. On the other hand, user-generated content such as Twitter and message boards (as in [9, 170]) tends to be more subjective or to express transitory affective states. This behaviour is reflected in the distribution of release time (see Table 3.3). In our corpus, a high concentration of news is released before the market opens (55% on average). By contrast, using a corpus compiled from message boards posts, [9] found a high occurrence of news during market trading hours, which they assert indicates noisy comments from day traders.

Universe of stocks

The first step in compiling our corpus was to choose the constituent stocks. We found that Exchange Traded Funds (ETF)[6] provide a mechanical way to aggregate the most relevant stocks in a given business sector. There are many investment companies that offer ETFs, but we chose SPDR Sector Funds in our work since SPDR is the largest provider of sector funds in the United States. We included in our analysis the top 5 sector ETFs by financial trading volume (as of Jan/2018). Among the most traded sectors we also filtered out the sectors that were similar to each other. For example, the Consumer Staples and Consumer Discretionary sectors are both part of the parent Consumer category. For each of the top 5 sectors we selected its top 10 stock holdings, which are deemed to be relevant stocks. Table 3.1, details the sectors and respective stocks.

[6]An ETF is a fund that owns assets, e.g. stock shares or currencies, but, unlike mutual funds are traded in stock exchanges. These ETFs are extremely liquid and track different investment themes.

Sector ETF	Constituent Stocks
Consumer Staples (XLP)	Procter & Gamble (PG), Coca-Cola Company (KO), PepsiCo (PEP), Walmart (WMT), Costco Wholesale Corporation (COST), CVS Health Corporation (CVS), Altria Group (MO), Walgreens Boots Alliance (WBA), Mondelez International (MDLZ), Colgate-Palmolive (CL),
Energy (XLE)	Exxon-Mobil (XOM), Chevron (CVX), ConocoPhillips (COP), EOG Resources (EOG), Occidental Petroleum Corporation (OXY), Valero Energy Corporation (VLO), Halliburton Company (HAL), Schlumberger Limited (SLB), Pioneer Natural Resources (PXD), Anadarko Petroleum Corporation (APC)
Utilities (XLU)	NextEra Energy (NEE), Duke Energy (DUK), The Southern Company (SO), Dominion Energy (D), Exelon Corporation (EXC), American Electric Power Company (AEP), Sempra Energy (SRE), Public Service Enterprise Group (PEG), Consolidated Edison (ED), Xcel Energy (XEL)
Healthcare (XLV)	Johnson & Johnson (JNJ), UnitedHealth Group (UNH), Pfizer (PFE), Merck & Co. (MRK), Medtronic (MDT), Amgen (AMGN), Abbott Laboratories (ABT), Gilead Sciences (GILD), Eli Lilly (LLY), Bristol-Myers Squibb (BMY)
Financials (XLF)	Berkshire Hathaway (BRK-A), JPMorgan Chase (JPM), Bank of America Corporation (BAC), Wells Fargo (WFC), CitiBank (C), Goldman Sachs Group (GS), U.S. Bancorp (USB), Morgan Stanley (MS), American Express (AXP), PNC Financial Services Group (PNC)

Table 3.1: **Sectors and respective constituent stocks**. For each sector we selected the top 10 stock holdings (as in January 2018). Stock codes in parentheses.

Table 3.2 shows random samples taken from our dataset.

Individual stocks headlines

We link a given news to an individual stock if it explicitly mentions the stock name or any of its surface forms. As an illustration, in order to collect all news for the stock code PG, *Procter & Gamble* company name, we search all the headlines with any of these words: `Procter&Gamble` *OR* `Procter and Gamble` *OR* `P&G`. In this example, the first word is just the company name and the remaining words are the company surface forms.

Date and time	Headline
2011-12-13 00:18:39 EDT	*Valero reports power outage at Port Arthur refinery*
2007-04-17 08:54:27 EDT	*Wells Fargo profit rises 11 pct on commercial loans*
2017-12-14 14:40:31 EDT	*Perrigo lines up bid for Merck's consumer health unit*
2007-01-03 10:27:42 EDT	*UPDATE 1-Bear Stearns ups Merck to outperform*
2010-02-23 13:35:11 EDT	*Exxon Mobil says remains bullish on Nigeria*
2016-09-22 15:32:13 EDT	*Texas regulators express "deep concern" over NextEra deal*
2008-10-14 08:30:00 EDT	*Smart For LifeTM Now Available on Costco.com*

Table 3.2: **Random samples from our dataset**. Note the factual/objective characteristic of our corpus, where typical news do not carry any sentiment connotation.

We automatically derived the surface forms by starting with a seed of surface forms extracted from the DBpedia Knowledge Base (KB). We then applied the following procedure:

- Associate each company name with the KB entity unique identifier.
- Retrieve all values of the `wikiPageRedirects` property. The property holds the names of different pages that points to the same entity/company name. This step sets the initial seed surface forms.
- Manually filter out noisy property values. For instance, from the Procter & Glamble entity page we were able to automatically extract `dbr:Procter_and_gamble` and `dbr:P_&_G`, but had to manually exclude the noisy associations `dbr:Female_pads` and `dbr:California_Natural`.

The result of the steps above is a dictionary of surface forms denoted by wd_{sc}.

Headlines retrieval

Our corpus is built at stock code level by collecting headlines from the Reuters Archive. This archive groups the headlines by date, starting from 1 January 2007. Each headline is a html link (`<a href>` tag) to the full body of the news, where the *anchor text* is the headline content followed by the release time. For example, the page dated 16 Dec 2016 has the headline *"Procter & Gamble appoints Nelson Peltz to board 5:26PM UTC"*.

For each of the 50 stocks (5 sectors times 10 stocks per sector) selected using the criteria described in Section 3.4.1, we retrieved all the headlines from the Reuters Archive raging from 01/01/2007 to 30/12/2017. This process takes the following steps:

- For a given stock code (sc) retrieve all surface forms wd_{sc}.
- For each day, store only the headlines content matching any word in wd_{sc}. For each stored headline we also store the time and timezone.
- Convert the news date and time to Eastern Daylight Time (EDT)[7].
- Categorise the news release time for each date. We consider the following category set: {`before market`, `during market` , `after market`, `holidays`, `weekends`}. `during market` contains news between 9:30AM and 4:00PM. `before market` before 9:30AM and `after market` after 4:00PM.

The time categories prevents data leakage and, consequently, unrealistic predictive model performance. In general, news released after 4:00PM EDT can drastically change market expectations impacting close to close prices returns. Following [9] method, news issued after 4:00PM (`after market`) are grouped with the pre-market (`before market`) on the consecutive trading day. Table 3.3 show the final distribution of news per time category.

Sector ETF	before market	during market	after market
Consumer Staples	54%	31%	15%
Energy	44%	36%	20%
Utilities	58%	31%	11%
Healthcare	55%	28%	17%
Financials	63%	24%	13%
total	84,556	40,996	21,231

Table 3.3: **Distribution of headlines per sector according to market hours**. The majority of the 146,783 headlines are released before 9:30AM (`before market`). The category `after market` includes news released after 4:00PM EDT. We count the categories `holiday` and `weekend` as `before market` since they impact the following working day. In addition, the categorisation is performed for each date. A detailed explanation of each category is given in Section 3.4.1.

3.4.2 Auxiliary Tasks for TL

This section provide details on the architectures employed to train the two source domain tasks, namely Text Categorisation and Natural Language Inference. It is

[7]The timezone of the New York Stock exchange

worth noting that we do *not* attempt to beat the results proposed for the two source tasks, but to only train the sentence encoders that are consumed in the target task (i.e. volatility forecasting) as fixed features.

Text Categorisation task

The Reuters Corpus Volume I (RCV1) contains 806,791 news articles annotated by topic using a hierarchical structure with 126 categories [111], where a news article can be assigned to more than one category.

Since RCV1 is not released with a standard train, validation, test sets, we separated 70%, 15% and 15% for the respective sets. With respect to the labels, we disregarded 23 labels that were assigned to any news. Moreover, we found many underrepresented labels (some with only 12 samples). Given the very small number of samples of the original fine-grained structure, we group into the same label all categories below the second hierarchical level. For example, given the root node CCAT (Corporate) we grouped C151 (ACCOUNTS/EARNINGS), C1511 (ANNUAL RESULTS) and C152 (COMMENT/FORECASTS) into the direct child node C15 (PERFORMANCE). Using this procedure the original 103 categories were reduced to 55, with the less represented class having approximately thousand samples.

Figure 3.4, shows the architecture for the text categorisation task. On the bottom of the architecture S_e receives word embeddings and outputs a sentence vector S. The S vector pass through a fully connected (FC) layer with sigmoid activation function that outputs a vector $\hat{y} \in \mathbb{R}^{55}$ where $\hat{y}_j \in [0, 1]$.

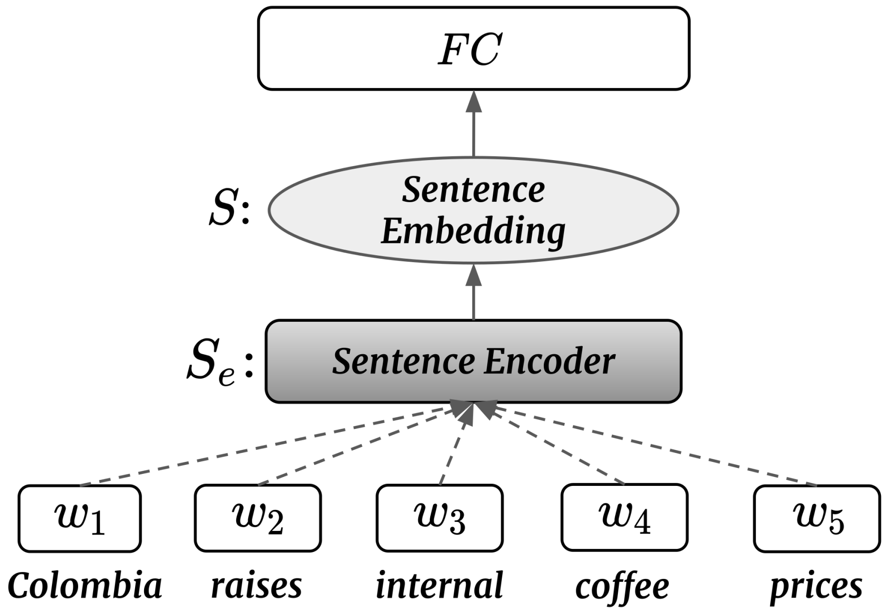


Figure 3.4: **RCV1 text categorisation architecture**. The sentence encoder S_e maps word emebddings w_i to a sentence vector S and the last FC layer has a sigmoid activation function.

Since a sample can be assigned to more than one label we train our models under the assumption that each label is independent but not mutually exclusive. Thus, the loss per sample is the average log loss across all labels:

$$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{55} \left(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right) \tag{3.13}$$

where the index i runs over the elements of the predicted and true vectors.

Finally, given the high label imbalance, we select the model with highest validation set F_1 score, defined by

$$F_1 = 2\frac{precision \cdot recall}{precision + recall}, \tag{3.14}$$

where $precision = tp/(tp + fp)$ and $recall = tp/(tp + fn)$ are expressed in terms of the number of true positive (tp), false positive (fp), and false negative (fn).

3.4.3 Natural language Inference task

The Stanford Natural Language Inference (SNLI) dataset [29] consist of 570,000 pairs of sentences. Each pair has a premise and a hypothesis, manually labelled according to one of the three labels: *entailment*, *contradiction*, or *neutral*. The dataset presents many desired properties. The labels are equally balanced, as opposed to the RCV1 dataset. Additionally, language inference is a complex task that requires a deeper understanding of the sentence meaning making this dataset suitable for learning supervised sentence encoders that generalise well to other tasks [41].

Figure 3.5, describes the neural network architecture. After each premise and hypothesis is encoded into S_p and S_h, respectively, we have a fusion layer. This layer has no trainable weights and just concatenates each sentence embedding. Following [41], we add two more matching methods: the absolute difference $|S_p - S_h|$ and the element-wise $S_p \odot S_h$. Finally, in order to learn the pair representation, S_{ph} is fed into a FC layer with rectified linear unit (ReLU) activation function, which is expressed as $f(x) = \max(0, x)$. The last softmax layer outputs the probability of each class. We observe that the sentence encoder of the premise and hypothesis share the same weights. Thus, the sentence encoder can be transferred to other tasks unambiguously.

3.4.4 Problem Statement

Henceforth, we use the superscript *sc* to index stocks and the subscript t periods in days. Our goal is to learn the mapping from historical multimodal data available up to day t to predict the next day volatility σ_{t+1}. To this end, we use a *sliding window* approach with window size T. That is, for each stock a sample on day t is expressed

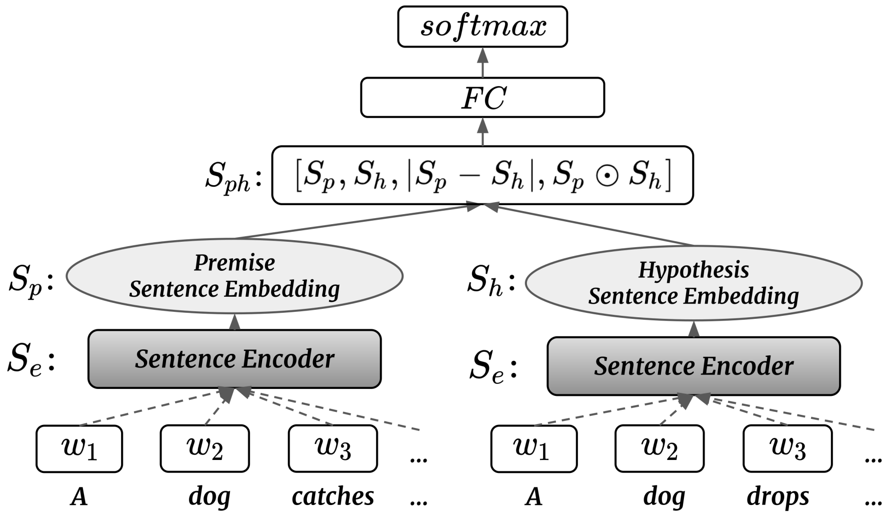


Figure 3.5: **Natural Language Inference task architecture**. Note that the sentence encoder S_e is shared between the premise and hypothesis pair. The FC layer learns the representation of the sentence pair and the final softmax layer asserts the output of the 3 possible labels, i.e. [*entailment*, *contradiction*, *neutral*], sums to one.

as a history of prices P_t^{sc} and headlines N_t^{sc} (textual mode). The price history is defined by the sequence

$$P_t^{sc} = \left[DP_{t-T+1}^{sc}, \cdots, DP_t^{sc}\right], \tag{3.15}$$

where the Daily Price (DP) is defined in terms of the Open, High, Low, Close (OHLC) prices of each stock.

$$DP_t^{sc} = \left[\frac{O_t^{sc}}{C_{t-1}^{sc}} - 1, \frac{H_t^{sc}}{C_{t-1}^{sc}} - 1, \frac{L_t^{sc}}{C_{t-1}^{sc}} - 1, \frac{C_t^{sc}}{C_{t-1}^{sc}} - 1\right] \tag{3.16}$$

The headlines history is defined by the sequence

$$N_t^i = \left[DN_{t-T+1}^i, \cdots, DN_t^i\right], \tag{3.17}$$

where daily news $DN_{t'}^{sc}$ is a vector representing all news released on a given day t'.

Finally, given its efficiency as a daily volatility proxy, we consider the Gaman-Klass estimator in Equation (2.11) as our target variable.

3.4.5 Global Features and Stock Embedding

Given the price and news histories for each stock we could directly learn *one model per stock*. However, this approach suffers from two main drawbacks. First, stocks are expected to share global patterns that are not captured in a single stock model. Second, since our price data is sampled on a daily basis, each stock model would be trained relying on a very small sample size.

In this work, we propose a method that learns a *global model* and is implemented using the following methods:

- **Multi-Stock batch samples**: Since our models are trained using Stochastic Gradient Descent, we propose at each mini-batch iteration to sample from a batch containing the history of any stock. As a consequence, the mapping between volatility and multimodal data is now able to learn global patterns. Moreover, adopting this approach increases the total number of training samples by a multiple of 50 (number of stocks).
- **Stock Embedding**: Using the Multi-Stock batch samples, we tackle the problem of modelling global features. However, it is reasonable to assume that stocks have part of their dynamic driven by idiosyncratic factors. In order to incorporate information specific to each stock, we propose to equip our model with a "stock embedding" mode that is learnt jointly with price and news modes. That is to say, we leave the task of distinguishing the specific dynamic of each stock to be learnt by the neural network. Specifically, this stock embedding is modelled using a one-hot encoding I_t^{sc}, indicating the stock of each sample.

Formally, we can express the "one model per stock" approach as the mapping

$$\begin{aligned}\sigma_{t+1}^{sc} = f^{sc}(&DN_{t-T}^{sc}, DN_{t-T+1}^{sc}, \cdots, DN_t^{sc};\\ &DP_{t-T}^{sc}, DP_{t-T+1}^{sc}, \cdots, DP_t^{sc})\end{aligned} \tag{3.18}$$

where $DN_{t'}^{sc}$ is a fixed-vector representing all news released on a given day for the stock sc[8] and $DP_{t'}^{sc}$ is defined in Equation (3.16).

The global model used in this work learns a single mapping f that at each mini-batch iteration randomly aggregates samples across all the universe of stocks and is expressed as

$$\begin{aligned}\sigma_{t+1}^{sc} = f(&DN_{t-T}^{sc}, DN_{t-T+1}^{sc}, \cdots, DN_t^{sc};\\ &DP_{t-T}^{sc}, DP_{t-T+1}^{sc}, \cdots, DP_t^{sc};\\ &I_t^{sc})\end{aligned} \tag{3.19}$$

The global model paradigm: Is it practically useful?

In this section we discuss some apparent limitations of the global model presented in the previous section, and propose solutions to make the model practically useful.

As in Equation (3.19), once the global model has been trained considering a given set of stocks, each stock prediction is computed independently for each stock in our set of stocks. More specifically, the prediction of each stock sc depends on three representations (or fixed-length vectors): 1) The stock news history (DN^{sc}), 2) The

[8]It will become clear in the next section how this news representation is modelled.

stock price history (DP^{sc}) and 3) the stock embedding, which is modelled using the stock one-hot encoding (I^{sc}).

Note that in contrast with the individual stock model, where one model is independently trained for each stock, we train a single global model. The motivation behind the global model is clear: To propose an architecture that, at least in principle, is capable of learning patterns common to the stock market as a whole, without scarifying the possibility of learning dynamics specific to each stock (i.e. the idiosyncratic factors).

Even though the global model is designed to capture patterns shared among stocks, it can not be consumed to predict the dynamics of stocks that are not part of the initial set of stocks used during training. This happens because each stock has its own one-hot encoding. To make this limitation clear, let us consider an universe of three stocks ($\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$) with the following one hot encodings: $I^{\mathcal{A}} = \{[1, 0, 0]\}$, $I^{\mathcal{B}} = \{[0, 1, 0]\}$ and $I^{\mathcal{C}} = \{[0, 0, 1]\}$. In the global model setting the training takes into account the three stocks data. following question emerges: How can we make a prediction for a stock $\mathcal{D}$ that is not part of the initial universe of stocks? In other words: How practically useful is the global model or how "global" is the proposed model? Note that in this case $\mathcal{D}$ can not be represented using the initial one hot encoding.

To address this inherent limitation of the global model we propose the following solution: To assign the stock embedding of $\mathcal{D}$ to a closest match. Thus, a good candidate for this match would be to assign $\mathcal{D}$ to the average embedding of stocks in the same sector.

For example, if $\mathcal{D}$ is a stock in the Energy sector, as well as, $\mathcal{A}$ and $\mathcal{C}$ we would assign the $\mathcal{D}$ embedding to the average of $\mathcal{A}$ and $\mathcal{C}$ embedding[9]. In this solution, once the $\mathcal{D}$ stock embedding is assigned, the prediction still uses the the stock X news representation ($DN^{\mathcal{D}}$) and price representation ($DP^{\mathcal{D}}$).

By using this assignment solution, if the global model is trained for a broad range of sectors, as is the case throughout the thesis, the global model is practically useful to predict the behaviour of stocks that are not part of the initial set of stocks used for training. In other words, we achieve a "truly global" model.

As a matter of fact, in Chapter 4 we extend the concept of stock embedding by proposing neural networks architectures where the relationship among stocks is injected to the training process in the form of a graph. Even though this formalism is general, we provide experimental results where the relationship among stocks is represented in the form of a sectoral graph (A_s). In addition, in Chapter 5 we make use of a more general sector embedding, to propose Deep RL architectures that can be deployed for a portfolio containing any number of stocks. That being said, once trained for a given set of stocks and using a broad range of sectors, our models can be readily deployed for predictions involving virtually any stock.

[9]Note the average here is taken with respect to stock embedding representations (dense vectors), as opposed to the one hot encoding (sparse vectors).

3.4.6 Multimodal Hierarchical Attention Network

In this section we describe the proposed Multimodal Hierarchical Attention Network (MHAN).

First, each headline released on a given day t is encoded into a fixed-size vector S using a sentence encoder.[10] Second, our daily News Relevance Attention (NRA) mechanism, which pays attention to each news item based on its content, converts a variable number of n headlines released on a given day t into a single Daily News (DN) vector. This step is depicted in Figure 3.6. We observe that DN accounts for the overall effect of all news released on a given day. Finally, we consider the temporal effect of the past T days of market news and price modes jointly, a process shown in Figure 3.7. For each stock the temporal encoding for news is denoted by Market News MN_t^{sc} and for price by Market Price MP_t^{sc}. These are a function of the past T Daily News representations $[DN_{t-T}^{sc}, \cdots, DN_t^{sc}]$ (**Text mode**) and Daily Prices features $[DP_{t-T}^{sc}, \cdots, DP_t^{sc}]$ (**Price mode**), where each Daily Price $DP_{t'}^{sc}$ feature is given by Equation (3.16) and the $DN_{t'}^{sc}$ representation is calculated using the daily **News Relevance Attention**. After the temporal effects of T past days of market activity were already encoded into the Market News MN_t^{sc} and Market Price MP_t^{sc}, we concatenated feature-wise MN_t^{sc}, MP_t and the **Stock embedding** E^{sc}. The stock embedding E^{sc} represents the stock code of the sample on a given day t. Finally, we have a Fully Connected (FC) layer that learns the **Joint Representation** of all modes. This fixed-sized joint representation is fed into a FC layer with linear activation, which predicts the next day volatility $\hat{\sigma}_{t+1}$.

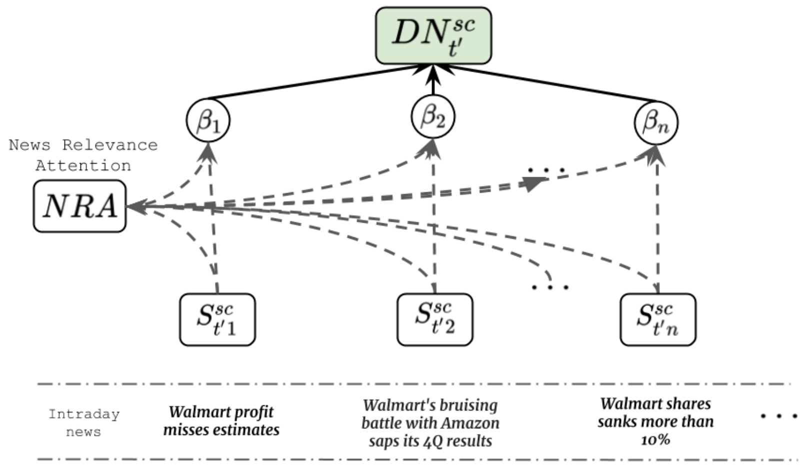


Figure 3.6: **Daily News Relevance Attention**. The figure illustrates a day where three news items were released about the Walmart company. After the headlines are encoded into a fixed-size representation S, the daily News Relevance Attention (NRA) converts all sentences into single vector representation of all Daily News DN by attending each headline based on its content.

[10]The headline encoding is learnt end-to-end from the headline word embeddings or transferred from the TL tasks as fixed features.

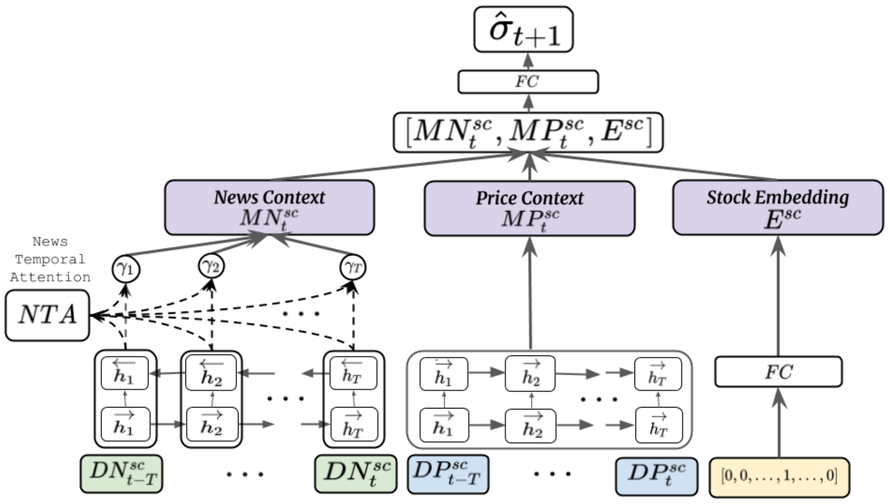


Figure 3.7: **Hierarchical Neural Network architecture**.

Below, we detail, for each mode separately, the layers of our hierarchical model.

Text mode

1. **Word Embedding Retrieval**
   Standard embedding layer with no trainable parameters. It receives a vector of word indices as input and returns a matrix of word embeddings.

2. **News Encoder**
   This layer encodes all n news on a given day t and outputs a sequence of news representations $[S_{t1}^{sc}, \cdots, S_{tn}^{sc}]$. Each encoded sentence has dimension d_S, which is a hyperparameter of our model. We evaluate our models considering sentence encoders trained end-to-end, using the BiLSTM attention (Section 3.3.1) and BiLSTM max-pooling (Section 3.3.1) architectures, and also transferred from the RCV1 and SNLI as fixed features.

3. **Daily News Relevance Attention**
   Our proposed news relevance attention component. The attention mechanism is introduced to tackle information overload. It was designed to "filter out" redundant or misleading news and focus on relevant stories based solely on the news content. The layer outputs a single vector representation DN from a set of n headlines on a given day t denoted by $\{S_{t1}^{sc}, \cdots, S_{tn}^{sc}\}$. Fully expressed, the daily presentation DN of all n news may be given by:

$$\begin{aligned} \tilde{S}_{ti} &= \sigma\left(W_R S_{ti}^{sc} + b_R\right) \\ \alpha_{ti} &= \frac{\exp(v_R^\intercal \cdot \tilde{S}_{ti})}{\sum_t \exp(v_R \cdot \tilde{S}_{ti})} \\ DN_t^{sc} &= \sum_{i=1}^{n} \alpha_{ti} S_{ti}^{sc} \end{aligned} \tag{3.20}$$

This layer is shared among all days t and for all stocks. The NRA has trainable weights W_R, b_R and v_R. Figure 3.6, illustrates our relevance attention. Note that this layer was deliberately developed to be invariant to headlines permutation, as is the case with the linear combination formula above. The reason is that our price data is sampled daily and, as a consequence, we are not able to discriminate the market reaction for each intraday news.

4. **News Temporal Context**
   Sequence layer with daily news embeddings DN_t^{sc} as time steps. This layer aims to learn the temporal context of news, i.e. the relationship between the news at day t and the T past days. It receives as input a chronologically ordered sequence of T past Daily News representations $[DN_{t-T}^{sc}, \cdots, DN_t^{sc}]$ and outputs the news mode encoding Market News $MN_t^{sc} \in d_{MN}$. The sequence with T time steps is encoded using a BiLSTM attention, with News Temporal Attention (NTA). This layer was designed to capture the temporal order that news is released in and the current **news novelty**, i.e. news that was repeated in the past can be "forgotten" based on the modulating gates of the LSTM network.

Price mode

5. **Price Encoder**
   Sequence layer analogous to the **News Temporal Context**, but for price mode. The input is the ordered sequence Daily Prices $[DP_{t-T}^{sc}, \cdots, DP_t^{sc}]$ of size T, where each element of the price feature is defined in Equation (3.16). In particular, the architecture consists of two stacked LSTM's, the first outputs a hidden vector that takes the temporal context into account for each price feature time step. Then these hidden vectors are again passed to a second independent LSTM. The layer outputs the price mode encoding Market Price $MP_t^{sc} \in d_{MP}$. This encoding is the last hidden vector of the second LSTM Market.

Stock embedding

7. **Stock Encoder**
   Stock dense representation. The layer receives the discrete encoding I_t^{sc} indicating the sample stock code passes through a FC layer and outputs a stock embedding E_{sc}.

Joint representation

8. **Merging**
   Feature-wise News, Price and Stock modes concatenation. No trainable parameters.

9. **Joint Representation Encoder**
   FC layer of size d_{JR}.

3.4.7 Multimodal Learning with Missing Modes

During the training we feed into our neural model the price, news and stock indicator data. The price and stock indicator modes data occur on all days. However, at the individual stock level some companies are not covered by the media. This feature imposed challenges on our multimodal training, since neural networks are not able to handle missing modes without special intervention. A straightforward solution would be to consider only days with news released, disregarding the remaining samples. However, this approach has two main drawbacks. First, the "missing news" does not happen at random, neither are they attributable to measurement failure, as is for example, the case of multimodal tasks using mechanical sensors data. Conversely, as highlighted in [38, 28] the same price behaviour results in distinct market reactions when accompanied or not by news.[11] In other words, specifically in financial forecasting problems the absence or existence of news is highly informative.

A number of methods were proposed in the multimodal literature to effectively treat informative missing modes or "informative missingness" [15]. In this work, we directly model missing modes using the method initially proposed in [118, 117] for clinical data and applied in the context of financial forecasting in [3]. Specifically, we implement the *Zeros & Imputation* (ZI) method [117] in order to jointly learn the price mode and news relationship across all days of market activity.

In ZI implementation, before the daily news sequence is processed by the text temporal layer (described in Item 4) we input a 0 vector for all time steps with missing news and left the news encoding unchanged otherwise. This step is called zero imputation. In addition, we concatenated feature-wise an indicator vector with value 1 for all vectors with zero imputation and 0 for the days with news.

As described in [3], the ZI method endows a temporal sequence model with the ability to learn different representations depending on the news history and its relative time position. Moreover, it takes into account the current and past news informative missingness. Finally, we observe that the learnt positional news encoding works differently from a typical "masking", where days without news are not passed through the LSTM cell. Masking the time steps would be losing information about the presence or absence of news concomitant with prices.

3.5 Experimental Results and Discussion

We evaluate our hierarchical neural model from three perspectives:

1. The potential improvements of our proposed NRA as compared to other solutions proposed in previous works.

[11]Experimental results [38, 28] demonstrate that large price dislocations in the absence of news tends to revert and continue the movement (momentum) when driven by news.

2. Does textual data improve volatility forecasting?
3. The importance of the different sentence encoders. Mainly, how end-to-end training compares to transferring the sentence encoder from our two auxiliary TL tasks.

3.5.1 Training Setting

During the training of our hierarchical neural model described in Section 3.4.6, we took special care to guard against overfitting. To this end, we completely separate 2016 and 2017 as the test set and report our results on this "unseen" set. The remaining data is further split into training (2007 to 2013) and validation (2014 to 2015). We set the sliding window T to 252 days, i.e. one year of historical data. Model convergence during training was monitored in the validation set. We monitored the validation score of our model at the end of each epoch and stored the network weights if the validation scores improved between two consecutive epochs. Additionally, we used mini-batch SGD with an Adam optimiser and early stopping with patience set to eight epochs.

The hyperparameter tuning was performed using grid search on the validation set – the search space and best parameters are reported in Appendix B.1. In addition, all the training related to the TL tasks and its results are detailed in Appendix B.2.

3.5.2 Stocks Universe Result

In order to evaluate the contributions of each component of our neural model described in Section 3.4.6, we use ablation techniques. The minus(plus) signs means the component was removed(added) to our architecture. We report our results using the following baselines:

1. **- News (unimodal price only)**: For this baseline we completely remove any textual data as input. Using this ablation we aim to evaluate the influence of news on the volatility prediction problem.
2. + **News (End-to-end Sentence Encoders) - NRA**: This baseline ablates our proposed News Relevance Attention (NRA) component and instead makes use of the same averaging method [49, 145], where all fixed-sized headline representations on a given day are averaged, without taking into account the relevance of each news item. We evaluate this baseline for both BiLSTM attention (Att) and BiLSTM max-pooling (MP) sentence encoders. Here, our goal is to assess the true contribution of our NRA component in the case that SOTA sentence encoders are taken into account.
3. + **News (End-to-End W-L Att Sentence Encoder)** + **NRA**: The Word-Level Attention (W-L Att) sentence encoder implements an attention mechanism directly on top of word embeddings. Thus, because the attention is

permutation invariant, this baseline does not consider the order of words in a sentence.

4. + **News (TL Sentence Encoders) + NRA**: Makes use of the sentence encoders of our two auxiliary TL tasks as fixed features. This baseline aims to address the following questions: Which dataset and models are more suitable to transfer to our specific volatility forecasting problem?; How do End-to-End models, which are trained on top of word embeddings, perform compared to sentence encoders transferred from other tasks?

Table 3.4 summarises the test scores for the ablations discussed above. In addition, the statistical significance of these results is provided in Appendix B.3 (see Table B.3).

All stocks		
Model	MSE	MAE
- News (price only unimodal)†	2.140E-05	3.093E-03
+ News (BiLSTM Att) - NRA	2.078E-05	3.037E-03
+ News (BiLSTM MP) - NRA	2.077E-05	3.031E-03
+ News (TL Reuters RCV1 BiLSTM MP) + NRA	2.037E-05	3.020E-03
+ News (TL Reuters RCV1 BiLSTM Att) + NRA	2.023E-05	3.011E-03
+ News (W-L Att)†† + NRA	2.006E-05	2.947E-03
+ News (TL SNLI BiLSTM Att) + NRA	1.986E-05	2.926E-03
+ News (TL SNLI BiLSTM MP) + NRA	1.974E-05	2.918E-03
+ News (BiLSTM MP) + NRA	1.904E-05	2.851E-03
+ News (BiLSTM Att) + NRA	**1.898E-05**	**2.823E-03**

Table 3.4: **Model architecture ablations and sentence encoders comparisons**. Best result, in bold, considers our full-fledged architecture (described in Section 3.4.6). The results in the first row do *not* consider news as input data. The minus sign means that a component of our network architecture was ablated (i.e. removed) and the plus sign that it was added. The second and third rows report results replacing our proposed NRA by a News Averaging component as in [49, 145]. † indicates our model was trained using only the price mode. †† highlights that the sentence encoder Word-Level Attention (W-L Attention) does not take into consideration the headline words order. Table B.3 provides a test of significance for the difference in performance between the best result, in bold, and all other models.

Experiments show that the best model is + News (BiLSTM Att) + NRA. This model is trained end-to-end and uses our full-fledged architecture with both price and news modalities. The second best model, i.e. + News (BiLSTM MP) + NRA, ranks slightly lower and only differs from the best model in terms of the sentence encoder. The best model uses a sentence encoder with attention layer (as defined in Section 3.3.1) and the latter a max-pooling layer (as defined Section 3.3.1), where both layers operates on top of the LSTM hidden layers and uses pre-trained word embeddings. However, as reported in Table B.3, the difference in MSE and MAE for these models is not statistically significant. More specifically, at the 1% level of significance, we can *not* reject the hypothesis that the model performances are equal.

Importantly, our experiments show that using news and price (multimodal) to predict the volatility improves the scores by 11% (MSE) and 9% (MAE), as compared with a model which considers only price features as explanatory variables (first row in Table 3.4). In addition, as reported in Table B.3, the difference between the MSE and MAE of these models is statically significant (p $<$ 0.01).

When comparing the performance of End-to-End models and the TL auxiliary tasks the following can be observed:

1. The models using two SOTA sentence encoders (specifically, BiLSTM Att and BiLSTM MP), which uses pre-trained word embeddings as input, perform better than transferring sentence encoder from both auxiliary tasks.
2. Even though SOTA sentence encoders perform better than those transferred from other tasks, the same does not hold for models trained using a simpler sentence encoder (specifically, WL-Att). In other words, considering the appropriate TL task, it is preferable to transfer a SOTA sentence encoder, which was trained on a larger dataset, than learning a less robust sentence encoder in an end-to-end fashion.
3. Initially, we thought that because the RCV1 is a financial domain corpus it would demonstrate a superior performance when compared to the SNLI dataset. However, the SNLI transfers better than RCV1. One possible explanation is that the text categorisation task (RCV1 dataset) is not able to capture complex sentence structures at the level required to perform natural language inference (SNLI dataset). The fact that SNLI achieves the best performance for the volatility prediction task corroborates the findings in [41], where it was shown that the SNLI dataset provides also superior performance, but for NLP tasks that are not related to financial prediction.

Experimental results in Table 3.4 also demonstrate that our proposed NRA outperforms the News Averaging method proposed in previous studies [49, 145]. This happens even when evaluating our NRA component in conjunction with the elementary W-L Att sentence encoder. Importantly, as reported in Table B.3, this difference in performance between our best model and the models proposed in the literature is statistically significant (p $<$ 0.01). In other words, our results show a clear advantage of using an attention mechanism to discriminate noisy news from impacting news and the effectiveness of our NRA as compared to other methods proposed in the literature.

Stock embedding – Additional analysis

In Section 3.4.5 we discussed the advantages of the global model paradigm. This model makes use of multi-stock batch samples and stock embedding during the training process. In particular, we highlighted that the global model benefits from effectively increasing the number of training samples. However, by using a stock embedding we also increase the dimensionality of the global model. In this section, we provide additional analysis and experimental results on the impact of the stock embedding dimensionality on the global model performance.

We start by analysing the relative dimension of the three vectors that are concatenated in order to learn the joint representation. As depicted in Figure 3.7 (purple boxes), these vectors are: News Context MN, Price Context MP and Stock Embedding E. Based on the hyperparameters tuning reported in Appendix B.1 (see Table B.1), they have the following dimensions: $d_{MN} = 512$, $d_{MP} = 128$ and $d_E = 16$.

The first point to note is that we need a higher dimension to encode News (d_{MN}), as compared to Prices (d_{MP}). This is agreement with the intuition that News have a higher degree of freedom (e.g. the order of words in a headline and the own complexity of the grammar). However, we also note that the increase in the global model dimensionality by using a stock embedding is very small, as compared to News and Prices. More specifically, the stock embedding only increases the dimension of the joint representation by $16/(512 + 128) = 2.5\%$. This observation shows that the increase in the global model dimensionality brought by the stock embedding component is relatively small.

Even though the impact in terms of dimension is small, it is still to be quantified if in terms of performance (MSE and MAE) the increase of the stock embedding dimensionality outweights the advantages of using a global model, which can effectively be trained using the samples of all stocks. To this end, we contrast our best model (last row in Table 3.4), which has an optimal stock embedding dimension[12] ($d_E = 16$), with three models: *no increase* in dimension ($d_E = 0$), a *moderate increase* in dimension ($d_E = 32$) and an *extreme increase* in dimension ($d_E = 1024$).

More specifically, by a model with no increase in dimension ($d_E = 0$) we mean to completely remove from our global model the increase in dimensionality attributed to the stock embedding. Thus, we disregard the one-hot encoding input (bottom yellow box in Figure 3.7), and, as a consequence, the representation related to the stock embedding (rightmost purple box).

Finally, since our goal is to isolate the effect of the stock embedding dimension d_E, we keep all other hyperparameters, including $d_{MN} = 512$ and $d_{MP} = 128$, as in our best model.

Table 3.5 reports experimental results of the impact of the stock embedding dimensionality on the global model performance. In addition, the statistical significance of these results is provided in Appendix B.3 (see Table B.4)

We observe that according to results reported in Table B.4, the differences in mean performance between the model with optimal stock embedding dimension and each of the three models are statistically significant. In other words, we reject the hypothesis that the differences in mean are equal ($p < 0.01$). This being true for both the mean MSE and mean MAE.

[12]As discussed above, this dimension is selected using hyperparameter search

Importantly, as reported in the first row ($d_E = 0$), we can see that the performance of our global model is degraded by completely removing the stock embedding[13]. Thus, if the stock embedding dimension is properly tuned using hyperparameter search, the increase in global model dimensionality is compensated by the difference in performance. In other words, the outperformance of a model with optimal stock embedding dimension, as compared to a model that does not use stock embedding, outweighs the increase of the global model dimensionality.

However, we can also conclude that if this dimension is not properly tuned the stock embedding component can be detrimental to the global model architecture. A moderate increase in dimensionality (as reported in the second row) slightly degrades the performance (for both MSE and MAE). However, when we artificially push the stock embedding dimension to extreme values (third row), as compared its optimal value (last row in bold), it would be preferable to completely disregard the stock embedding (as reported in the first row).

All Stocks		
Our full-fledged model with ...	MSE	MAE
no stock embedding† ($d_E = 0$)	1.982E-05	2.935E-03
a moderate increase in the stock embedding dimension ($d_E = 32$)	1.959E-05	2.869E-03
an extreme increase in the stock embedding dimension ($d_E = 1024$)	2.038E-05	3.094E-03
optimal stock embedding dimension†† ($d_E = 16$)	**1.898E-05**	**2.823E-03**

Table 3.5: **The impact of the stock embedding dimensionality on our global model architecture.** † indicates that we completely remove from our global model (last row in bold) the increase in the dimensionality attributed to the stock embedding. †† indicates the optimal stock embedding dimension ($d_E = 16$) is obtained by performing hyperparameter search on the validation set.

Comparisons with GARCH

Having analysed our best model and its sensitivity to the stock embedding dimension, we now turn to its comparative performance with respect to the widely regarded GARCH model described in Section 2.1.1. We assess our model performance relative to GARCH using standard loss metrics (MSE and MAE), but as in econometric studies, we also use the regression-based accuracy specified in Equation (2.8), and measured in terms of the coefficient of determination R^2. In addition, we evaluate our model across two different volatility proxies: Garman-Klass ($\widehat{\sigma_{GK}}$) (Equation (2.11)) and Parkinson ($\widehat{\sigma_{PK}}$) (Equation (2.10)). As discussed in Section 2.1.1 both estimators are reliable, and efficient proxies of next day volatility in the lack of intraday data. Finally, it is worth noting that our models were trained using the Garman-Klass estimator as target variable.

Table 3.6 reports the comparative performance between our best model (+ **News BiLSTM (Att)** + **NRA**) and GARCH, as well as, the Price only (unimodal) model.

[13]Note that lower MSE (or MAE) means better performance.

In addition, the statistical significance of these results is provided in Appendix B.3 (see Table B.5).

All Stocks				
Model	Vol Estimator	R^2	MSE	MAE
GARCH	$\widehat{\sigma_{GK}}$	0.357	2.46E-05	3.16E-03
	$\widehat{\sigma_{PK}}$	0.329	2.57E-05	3.20E-03
Our Model: Price (Unimodal)	$\widehat{\sigma_{GK}}$	0.384	2.14E-05	3.09E-03
	$\widehat{\sigma_{PK}}$	0.350	2.36E-05	3.29E-03
Our Model: Price + News	$\widehat{\sigma_{GK}}$	**0.455**	**1.90E-05**	**2.82E-03**
	$\widehat{\sigma_{PK}}$	**0.410**	**2.09E-05**	**2.98E-03**

Table 3.6: **Our volatility model performance compared with GARCH**. Best performance in bold. Our model has superior performance across the three evaluation metrics and taking into consideration the state-of-the-art volatility proxies, Garman-Klass ($\widehat{\sigma_{GK}}$) and Parkinson ($\widehat{\sigma_{PK}}$). Table B.5 provides a test of significance for the difference in performance between the best result, in bold, and the two other models.

We can see that our model provides a superior performance as compared to GARCH for both volatility proxies. It is worth noting that evaluating the GARCH model relying on standard MSE and MAE error metrics should be taken with a grain of salt. Andersen and Bollerslev [7] provides the background theory and arguments supporting R^2 as the metric of choice in evaluating the predictive power of a volatility model. In any case, the outperformance of our model with respect to GARCH is valid for all three metrics, namely R^2, MSE and MAE. Importantly, as reported in Table B.5, this outperformance is statistically significant ($p < 0.01$).

3.5.3 Sector-level Results

Sectors are expected to have different risk levels, in the sense that each sector is driven by different types of news and economic cycles. Moreover, by performing a sector-level analysis we were initially interested in understanding if the outperformance of our model with respect to GARCH was the result of a learning bias to a given sector or if, as turned out to be the case, the superior performance of our model was spread across a diversified portfolio of sectors.

In order to evaluate the performance per sector, we first segregated the constituent stocks for each sector in Table 3.1. We then computed the same metrics discussed in the previous section for each sector individually.

Table 3.7 reports our experimental results segregated by sector. We observe that the GARCH model accuracy, measured using the R^2 score, has a high degree of variability between sectors. For example, the accuracy ranges from 0.15 to 0.44 for the HealthCare and Energy sectors respectively. This high degree of variability is in agreement with previous results reported in [152], which focus on long-term (quarterly) volatility predictions. Although the GARCH accuracy is sector-dependent, without any exception, our model using price and news as input clearly outperforms GARCH sector-wise. This fact allow us to draw the following conclusions:

- The core aggregated performance reported in Table 3.6 is not composed of a mix of outperforming and underperforming sector contributions; rather, the aggregated effect is composed of outperformance across all sectors. This fact provides strong evidence that our model is more accurate than GARCH.
- The proposed Global model approach discussed in Section 3.4.5 generalises well, i.e. the patterns learnt are not biased to a given sector or stock.

One of the limitations of our work is to rely on proxies for the volatility estimation. Although these proxies are handy if only daily price data is available, by having high frequency data we would be able to evaluate our model in a way much closer to the true daily latent volatility. For example, in evaluating the GARCH performance for the Yen/Dollar exchange rate [7] reports R^2 values of 0.237 and 0.392 using hourly and five minutes intraday returns, respectively. In a similar way, we argue that intraday data would ameliorate the learning process. The reason for this is that the NRA component would have a smaller number of news items to pay attention to and it would arguably help to learn the most important headlines. In other words, the higher the data frequency, the easier it is to learn the "wording" that makes a market volatile. Although we show that the relevance attention is key, using the very low-frequency daily data we are only measuring the aggregate effect of all news on the one-day-ahead volatility prediction.

3.6 Conclusion

In this chapter, we studied the joint effect of stock news and prices on the daily volatility forecasting problem. To our knowledge, this work is one of the first studies aiming to predict short-term (daily) rather than long-term (quarterly or yearly) volatility, taking news and price as explanatory variables and using a comprehensive dataset of news headlines at the individual stock level.

Our hierarchical end-to-end model benefits from state-of-the-art approaches to encode text information and to deal with the two main challenges in correlating news with market reactions: news relevance and novelty. That is, to address the problem of how to pay attention to the most important news based purely on its content (*news relevance attention*) and to take into account the temporal information of past news (temporal context). Additionally, we propose a multi-stock mini-batch + stock embedding method suitable to model commonality among stocks.

The experimental results show that our multimodal approach outperforms the GARCH volatility model, which is the most prevalent econometric model for daily volatility predictions. The outperformance is sector-wise and demonstrates the effectiveness of combining price and news for short-term volatility forecasting. The fact that we outperform GARCH for all analysed sectors confirms the robustness of our proposed architecture and evidences that our global model approach generalises well.

We ablated different components of our neural architecture to assess its most relevant parts. To this end, we replaced our proposed *news relevance attention* layer, which aims to pay attention to the most important news items on a given day, with a simpler architecture proposed in the literature, which averages the daily news. We found that our attention layer improves the results. Additionally, we ablated all the architecture related to the news mode and found that news enhances the forecasting accuracy. Importantly, both improvements are statistically significant ($p < 0.01$).

Finally, we evaluated different sentence encoders, including those transferred from other NLP tasks, and concluded that they achieve better performance compared to a plain Word-level attention sentence encoder trained end-to-end. However, they do not beat state-of-the-art sentence encoders trained end-to-end.

In order to contribute to the literature of Universal Sentence Encoders, we evaluated the performance of transferring sentence encoders from two different tasks to the volatility prediction problem. We showed that models trained on the Natural Language Inference (NLI) task are more suitable to forecasting problems than a financial domain dataset (Reuters RCV1). By analysing different architectures, we showed that a BiLSTM with max-pooling for the SNLI dataset provides the best sentence encoder.

Model	Vol Estimator	R^2	MSE	MAE
		Consumer Staples		
GARCH	$\widehat{\sigma_{GK}}$	0.173	2.01E-05	2.63E-03
	$\widehat{\sigma_{PK}}$	0.155	2.08E-05	2.70E-03
Our Model: Price (Unimodal)	$\widehat{\sigma_{GK}}$	0.194	1.93E-05	2.67E-03
	$\widehat{\sigma_{PK}}$	0.176	2.04E-05	2.82E-03
Our Model: Price + News	$\widehat{\sigma_{GK}}$	**0.224**	**1.80E-05**	**2.48E-03**
	$\widehat{\sigma_{PK}}$	**0.201**	**1.90E-05**	**2.61E-03**
		HealthCare		
GARCH	$\widehat{\sigma_{GK}}$	0.150	2.20E-05	3.05E-03
	$\widehat{\sigma_{PK}}$	0.138	2.33E-05	3.09E-03
Our Model: Price (Unimodal)	$\widehat{\sigma_{GK}}$	0.186	2.01E-05	3.01E-03
	$\widehat{\sigma_{PK}}$	0.164	2.24E-05	3.21E-03
Our Model: Price + News	$\widehat{\sigma_{GK}}$	**0.258**	**1.76E-05**	**2.74E-03**
	$\widehat{\sigma_{PK}}$	**0.225**	**1.96E-05**	**2.90E-03**
		Financials		
GARCH	$\widehat{\sigma_{GK}}$	0.274	2.02E-05	3.14E-03
	$\widehat{\sigma_{PK}}$	0.250	2.17E-05	3.18E-03
Our Model: Price (Unimodal)	$\widehat{\sigma_{GK}}$	0.326	1.77E-05	3.10E-03
	$\widehat{\sigma_{PK}}$	0.290	2.03E-05	3.32E-03
Our Model: Price + News	$\widehat{\sigma_{GK}}$	**0.373**	**1.65E-05**	**2.84E-03**
	$\widehat{\sigma_{PK}}$	**0.332**	**1.86E-05**	**3.00E-03**
		Energy		
GARCH	$\widehat{\sigma_{GK}}$	0.443	4.38E-05	4.24E-03
	$\widehat{\sigma_{PK}}$	0.412	4.52E-05	4.27E-03
Our Model: Price (Unimodal)	$\widehat{\sigma_{GK}}$	0.440	3.60E-05	4.13E-03
	$\widehat{\sigma_{PK}}$	0.406	3.98E-05	4.34E-03
Our Model: Price + News	$\widehat{\sigma_{GK}}$	**0.538**	**3.04E-05**	**3.72E-03**
	$\widehat{\sigma_{PK}}$	**0.495**	**3.38E-05**	**3.88E-03**
		Utilities		
GARCH	$\widehat{\sigma_{GK}}$	0.167	1.71E-05	2.75E-03
	$\widehat{\sigma_{PK}}$	0.154	1.75E-05	2.77E-03
Our Model: Price (Unimodal)	$\widehat{\sigma_{GK}}$	0.145	1.40E-05	2.56E-03
	$\widehat{\sigma_{PK}}$	0.128	1.51E-05	2.75E-03
Our Model: Price + News	$\widehat{\sigma_{GK}}$	**0.225**	**1.24E-05**	**2.34E-03**
	$\widehat{\sigma_{PK}}$	**0.193**	**1.34E-05**	**2.51E-03**

Table 3.7: **Sector-level performance comparison.**

Chapter 4

Graph Transformer Network for Trading the Price Range of Stocks

4.1 Introduction

In this chapter, we focus on proposing network architectures that take into account the *joint* effect of other stocks on each stock forecast, and on investigating the benefits of these networks architectures in the context of quantitative trading strategies.

In Chapter 3 we saw that the volatility is persistent and as such, highly predictable across stocks and sectors. For daily predictions the volatility is efficiently proxied using the Parkinson estimator in Equation (2.10). This estimator depends on the logarithm difference between high and low prices. A natural question that arises is whether the same applies to the high and low price levels individually, and whether these predictions are useful for building profitable trading strategies. An appealing aspect of daily high and low prices is its markedly lower volatility compared to widespread closing price returns[1] [77]. This happens because stocks, which are traded in exchanges with well-defined daily opening and closing times, present a *U-shaped* intraday volatility pattern [198] – prices at the beginning and the end of the day are more volatile than the reminder of the trading session. In other words, since high and low prices can occur at any time during the day, return measurements involving high and low prices are not concentrated on volatile periods and are consequently less noisy. This is in contrast to closing price returns, where returns are taken with respect to the most volatile periods of the day. Thus, it is expected that ML models optimised to learn the less noisy high (or low) price series are less prone to overfitting. Surprisingly, with few exceptions (e.g. [127, 193, 77]), the vast majority of financial forecasting papers focus on (directional) predictions using closing price return. We attribute this to the fact that closing return predictions can be easily converted into a trading strategy, i.e. go long or short[2] based on the "up" and "down" predictions. In contrast, how to exploit the range predictions for trading is less clear. Thus, the use of range predictions in the context of trading, as proposed initially in

[1]The daily high (or low) price returns are computed using the current high (or low) price against any of the previous Open, High, Low, Close (OHLC) prices.

[2]To the reader not familiar with long and short trades they are discussed in Chapter 2

[127], was much less explored. In essence, the range trading strategy provides ways of profiting from the high and low price predictions.

Recently, Graph Neural Networks (GNN) [106, 81, 190, 18] architectures have been proposed to leverage the power of deep learning in structured data (e.g. Knowledge bases).

Even though stock data is not structured in a graph, we use GNN in a market graph setting, where each stock is represented by a node and the pairwise relationships between stocks by edges. Overall, by using a market graph we bias our deep learning models using prior knowledge about stock relationships.

The relational bias inherit of GNN has many desirable properties:

- Flexibility in representing different relationships, i.e. stocks can be related based on historical correlation or just because they are part of the same sector.
- Permutation invariance. It is usually the case that entities (i.e. stocks) do not have a natural ordering – except when confronted with their relations.
- Generalisation. Layers are reused (shared) across all nodes.

One of the main advantages of GNN is increasing sample efficiency, which is achieved by constraining relations between entities. In order to make this statement clear, let us consider an MLP architecture for a typical financial forecasting problem. It receives the representation for each of the n stocks as input[3] and outputs the target stock price. Although each stock prediction does not depend on the order of the $n-1$ stocks time series, the MLP architecture conceives each possible input order as fundamentally different from the others.[4] Thus, worst case scenario, it requires $(n-1)!$ sample pairs to learn an order invariant mapping. By contrast, a GNN aggregates the entities based on stock pairs that hold a relationship, and then assigns a new representation using a symmetric function. This process, which can be seen in general as *aggregation* and *averaging*, enforces the desired order invariance and is introduced via a graph. Unfortunately, in the financial forecasting setting the relationship between the entities (stocks) is not explicit[5]. Nevertheless, by using GNNs in context of the market graph, as proposed in this chapter, we can inject the relational bias using domain knowledge.

The main question we aim to address in this chapter is the real advantages of using prior knowledge in the context of trading strategies. In addition, we have the following main contributions:

1. We propose a novel graph neural network, the Graph Transformer Network (GTN). This network can be applied to general problems. In particular, experiments show that GTN performs better than Graph Attention Networks (GAT)

[3]For example, each time a series feature is encoded using an RNN.

[4]In other words, given a set of initial NN weights, stock index permutations map to different predictions.

[5]This is contrary to, for example, social networks or chemical graphs problems, where the relationships are given by friends lists or atoms attraction rules, respectively.

[190] for the trading range strategy. The GAT is the current state-of-the-art for many structured datasets.

2. We introduce a sectoral graph A_s to bias our deep learning models, and experiment shows that there are clear performance improvements if this type of prior knowledge is employed during the training process.

3. We evaluate the range trading strategy for a large number of stocks, where previous studies are restricted to a small number of stocks [127, 193], or considered a version of the range strategy with only long positions [77]. In addition, the strategy is investigated taking into account the risk involved in the trades (as detailed in Section 2.2.3).

4. By conducting a return attribution analysis we show that the range trading strategy is not only profitable (3.11 Sharpe ratio) but also its performance cannot be explained in terms of the market performance (as a whole) or other factors that are know to be profitable (as reviewed in Section 2.2.4). This analysis was not conducted in previous studies.

4.2 Related Work

Our work is strongly connected with the following areas: financial graphs, graph networks and range trading. We dedicate a section to each of these areas.

4.2.1 Financial Graphs

Mantegna [124] introduces graphs as a tool to analyse the relationships in the financial markets. He first estimates the distance between a pair of stocks directly from the stock returns correlation matrix. The Minimum Spanning Tree (MST)[6] of the correlation matrix is then introduced as a way find hierarchical clusters for a given set of stocks.

A large number of studies have followed this seminal work, with distinct goals in mind. From the viewpoint of different ways of measuring distance (or relatedness) between stocks, Fiedor [66] proposes information-theoretic metrics and distances based on Granger causality (expressed as directed graphs), discussed in [202]. Huang et al. [94] uses an interbank graph to investigate systemic risk of financial institutions and finds that nodes with large centrality[7] tend to contribute to financial crisis spillovers. Similarly, Battiston et al. [19] shows that the dynamics of densely interconnected banks can be employed to detect crisis tipping points. The presence of communities in the pairwise stocks correlation graph and the dynamics of its topological properties, such as graph modularity, is studied in [167]. The authors show that modularity alone summarises the whole graph structure and that this structure

[6]An MST forms a connected graph (with no loop) where all stocks are linked with the minimum possible total edge (distance) weight.

[7]In general, centrality quantifies the most influential nodes in the graph.

drastically changes during periods of economic crisis, with stocks moving away or disconnecting from former communities (clusters).

Tse et al. [184] proposes a market index that select stocks with a large number of connections. Then, two stocks are considered connected, i.e. edge value different from zero, if the correlation is above a given threshold value. They conclude that the market is highly dominated by stocks in the financial sector. This study is extended in [53] by proposing different ways of composing market indices based on graph properties, where they show that some indices perform better during turbulent periods.

In broad terms, all studies described above focus on: (1) investigating general properties of the the stock market structure (e.g. number of clusters and modularity), or (2) As in [184, 53], on creating hand-crafted indices based on the graph properties and only then evaluating the performance of these indices. By contrast, our thesis employ graphs as a way of biasing our models, where these models are optimised for a specific task, i.e. end-to-end.

4.2.2 Graph Networks

The use of GNNs for financial forecasting and trading is in its infancy. The first research in this area (in October 2018) [39] consider a *shareholding ratio* graph. The task is the binary classification of next day open to close direction of Chinese stocks. Their best reported model, trained "end-to-end" and using a Graph Convolution Network (GCN) [106] architecture, achieved 57% (directional) accuracy. We note that the evaluation concerns only accuracy, i.e. no trading performance metrics are evaluated for any trading strategies built on the prediction. Moreover, a comparison with baselines provides evidence that a model endowed with relational bias performs better than a standard sequence model operating on the price feature vector. Although results are promising, the model was evaluated in a very short period of approximately two months (from 10/2017 to 12/2017). One limitation of this study is that, different than our study, they do not compare a setting where all stocks can potentially relate to each other, as represented by a fully-connected graph.

Feng et al. [65] (March 2019) introduces two relational biases: the sector-industry graph (edge value one if companies share the same sector-industry and zero otherwise) and the Wikidata (knowledge) graph. In the latter, predicates (relations) are extracted from two stocks (both object and subject). These relationship are types like OWNED_BY or FOUNDED_BY.

Similar to our work, they also propose a mechanism to learn the relationship between stocks dynamically (i.e. an attention mechanism). However, different than in our study, they do not compare their performance with a SOTA graph neural network, the Graph Attention Network (GAT) [190]. In addition, Feng et al. [65] attention mechanism is very similar to GAT, in the sense that both use an additive attention that operates on the concatenation of two node representations. Thus, the investigation of the attention proposed in [65] will be the subject of future work.

4.2.3 Range Trading Strategy

The use of daily high and low price predictions to build trading strategies was first proposed in [127]. They use intra-day data (at 15 minute frequency) to decide the best time to enter long and short positions during trading hours[8]. Even though experimental results indicate profitability, the evaluation is restricted to only two stocks traded on the Brazilian stock exchange (BOVESPA) and only three months of data (May 2008 to December 2008). von Mettenheim and Breitner [193] investigate the same trading strategy, but for five stocks and two market indices (S&P 500 and Russell 2000) during a period under different market regimes (2011 and 2012). A larger universe of stocks, precisely, 370 stocks, was analysed in [77], but for a trading strategy that only consider long trades.

In our research, we contribute to the literature of range trading as follows:

1. We extend the research reviewed in this section by proposing NN architectures that incorporate the effect of other stocks on a given target stock (range) prediction.

2. In contrast with shallow neural network architectures in previous studies, we approach the problem not only using advanced sequence encoders, but also by attempting to improve the performance using prior knowledge about the stock relationships.

3. We investigate the same trading strategy in [127, 193], but evaluate its performance for 11 sectoral indices, involving a total of 388 stocks traded in North America. Thus, overcoming the limitations of the long-only strategy proposed in [77] and at the same time, assuring the robustness of the range trading strategy through a large number of samples (stocks).

4. We contribute to understanding the strategy's sources of profitability by studying its dependence on volatility levels. In this way, we investigate questions raised in previous research.

4.3 Background

In this section, we review GNN architectures for unsupervised, supervised and semi-supervised settings.

Formally, these architectures account for the relational bias by means of an *attributed graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, V)$, where $\mathcal{V}$ is a vertex set $\mathcal{v}_i$, $i = 1, \cdots, n$ and the set of linkages between nodes pairs is denoted by $\mathcal{E} := \{\mathcal{e}_{i,j} \mid \mathcal{v}_i, \mathcal{v}_j \in V\}$. For an unweighted graph these linkages are expressed by an adjacency matrix $A \in \{0,1\}^{n \times n}$ such that $a_{ij} = 1$ if $\mathcal{e}_{i,j} \in \mathcal{E}$ and zero otherwise. Finally, for each node we attribute a d-dimensional vector $v_i \in R^d$ and the graph representations/attributes is denoted by matrix $V \in \mathbb{R}^{n \times d}$.

[8]We will soon explain the range trading strategy in details

Given $\mathcal{G}$ above, a GNN approximates the mapping $f : V \mapsto V'$ from the original graph attributes $V \in \mathbb{R}^{n\times d}$ to new graph attributes $V' \in \mathbb{R}^{n\times d'}$ using a neural network.

In the supervised learning setting, a set of node labels $Y := [y_1, \cdots, y_n]^\intercal$ is provided and a final prediction layer is added at the end of the architecture to minimise the supervised loss. Importantly, irrespective of the learning setting, the learnt graph attributes are expected to have the same topological structure as the graph.

4.3.1 Unsupervised Graph Embedding

In the unsupervised setting, a graph embedding learns node representations (graph attributes) V taking into account the graph relations, which are represented by the graph edges. In broad terms, the learning process involves iteratively updating each node representation v_i by aggregating (AGG) and averaging (AVG) the connected nodes representations. Formally, the k^{th} iteration approximates the mapping composition AVG $\circ$ AGG : $V^{(k-1)} \mapsto V^{(k)}$. At the node level this is expressed as:

$$v_{\mathcal{N}(i)}^{(k-1)} = \text{AGG}\left(\{v_j^{(k-1)} \mid \forall j \in \mathcal{N}(i)\}\right) \quad \text{and} \tag{4.1a}$$

$$v_i^{(k)} = \text{AVG}\left(v_i^{(k-1)}, v_{\mathcal{N}(i)}^{(k-1)}\right) \tag{4.1b}$$

where the neighbourhood function $\mathcal{N}(i)$ is the set of nodes adjacent to the i^{th} node[9] and captures the *local* topological structure of the graph.

It is shown [159, 144, 81, 106] that repeating the iterative procedure above generates an (unsupervised) graph embedding. In other words, points $v_i \in \mathbb{R}^d$ are close to each other for neighbour (adjacent) nodes and distant otherwise; this holds true even in the case where the V matrix is initialised randomly. Many aggregate, average and neighbourhood functions have been proposed in the literature. DeepWalk [144] considers the neighbourhood as a sample containing all node representations visited by a random walk rooted at a target node i. This set of node representations forms a context "tokens" and is concatenated (at the aggregation step) in order to approximate the likelihood of observing the target node i given its context (average function). Similarly, Hamilton et al. [81] proposes GraphSAGE which aggregates the first-neighbour set $\mathcal{N}_1(i) := \{j \in \mathbb{N}^+ \mid a_{ij} \neq 0\}$ by means of max. or mean pooling functions. The node representation $v_i^{(k-1)}$ is then concatenated with the pooled representation and fed into an MLP layer, which predicts the next representation $v_i^{(k)}$ interactively, as in Equation (4.1b).

[9]Note that in order to increase generalisation the own node is not part of the neighbour set.

4.3.2 Graph-Based Learning

In a nutshell, graph-based learning leverages the information contained in a graph by assuming that nodes that are connected will share the same label. Typically, only a subset of instances are labelled and the goal is to learn the function $Y = f(V)$ that maps feature space V to node labels Y in a transductive or inductive semi-supervised setting.[10] To this end, in [205, 21, 196] a Laplacian regularisation loss is proposed and defined by:[11]

$$\mathcal{L}_g := f(V)^\intercal \triangle_u f(V) = \frac{1}{2}\sum_{i,j=1}^{n} a_{ij}\|f(v_i) - f(v_j)\|^2 \tag{4.2}$$

where $\triangle_u = D - A$ is the (unnormalised) Laplacian matrix of a graph with a diagonal degree matrix D with elements $d_{ii} = \sum_j a_{ij}$. The total loss $\mathcal{L}$ to be optimised is the contribution of the supervised loss $\mathcal{L}_s$ and the graph loss above weighted by a regularisation factor λ, that is:

$$\mathcal{L} = \mathcal{L}_s + \lambda\mathcal{L}_g$$

The graph loss can be directly interpreted in terms of the summation in Equation (4.2). Since the adjacency matrix elements a_{ij} are positive by definition, and the summation involves squared terms, it follows that the loss is always positive. Therefore, a minimisation is attained if and only if nodes close to each other, i.e. with large a_{ij} value, have representations drifted to the same label, that is $a_{ij} \to 1 \iff f(v_i) \approx f(v_j)$. This desired property is called *Smoothness of the graph Lapalcian* [192] and enforces that the values y_i assigned to each node (labels) varies smoothly around adjacent nodes. Additionally, we can see from the graph regularization definition in Equation (4.2) that the lowest eigenvalue of the Laplacian matrix has an associated eigenvector f_u that minimises the graph loss[12]. Thus, even in the total absence of labels y_i we can still make use of the Laplacian decomposition to aggregate nodes that share similar labels, this being the core idea behind spectral clustering algorithms [135, 192].

4.3.3 Convolution on Graphs

Convolutional networks on graphs take advantage of the Laplacian smoothness property described above to convolve filters that capture the local connection structure

[10]Note since only a fraction of node labels are provided, it is expected that knowing the graph structure, which includes labelled and unlabelled nodes, improves the learning process.

[11]The derivation of the last summation equality can be found in [192]

[12]Note if f_u is an eigenvector of the Laplacian matrix $\triangle$ it follows that $\mathcal{L}_g := f_u(V)^\intercal \triangle f_u(V) = \lambda f_u(V)^\intercal f_u(V) = \lambda$, where λ is the decomposition eigenvalue and we assume the eigenvectors are normalised to one. Finally, we observe that the Laplacian matrix is symmetric, implying that all its eigenvalues are positive.

of each node. The spectral convolution on a graph with signal $V \in \mathbb{R}^{n \times d}$ is defined in [32, 82] as:

$$f(W, V) = (UWU^{\intercal})V \tag{4.3}$$

where the filtering operation f is parameterised by the diagonal matrix $W \in \mathbb{R}^{n \times d}$ (one filter weight per node), and U is the eigenvectors matrix of the normalised graph Laplacian $\Delta := I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ with eigendecomposition $\Delta = U\Lambda U^{\intercal}$. Applying convolutional filters directly to attributed graphs removes the need to explicitly rely on a regularisation loss term L_g (Equation (4.2)), which was formerly added to the supervised loss L_s to inject the relational bias. However, the convolution above depends on a full decomposition of the Laplacian matrix, which makes the operation unfeasible for large graphs. Moreover, its computation is expensive, since it depends on dense matrix-matrix multiplications, even for graphs where the adjacency matrix A is very sparse. To alleviate this problem, Defferrard et al. [45] suggest approximating the filter W as a function of Laplacian eigenvalues Λ by means of a Chebyshev expansion. By making use of identity $\Delta^k = (U\Lambda U^{\intercal})^k = (U\Lambda^k U^{\intercal})$ for the diagonalised Laplacian matrix Δ, this approximation allows us to rewrite the spectral convolution in Equation (4.3) entirely in terms of the *rescaled* Laplacian matrix $\tilde{\Delta} = 2\Delta/\lambda_{max} - I$, where λ_{max} is the largest eigenvalue of the normalised Laplacian matrix. In this way, we have the K-localised spectral convolution proposed in [45]:

$$f(\tilde{\Delta}, V) \approx \sum_{k=0}^{K} (w_k T_k(\tilde{\Delta}))V \tag{4.4}$$

where $T_k(x)$ are Chebyshev polynomials and w_k its coefficients. The convolution approximation is called K-localised because it depends only on powers of the Laplacian, and consequently, the k^{th} term accounts for a neighbour k-hops away from a target node, where the expansion is truncated up to K terms. Effectively, by increasing the number of terms in the expansion we increase the neighbourhood set of the target node. Experimental results in [45] demonstrate the ability of K-localised spectral convolutions to learn the graph node features.

4.3.4 Graph Convolutional Networks

Kipf and Welling [106] propose the Graph Convolution Network (GCN) composed of graph convolutional layers *linear* with respect to the graph Laplacian. Compared with K-localised convolutions in [45], in the GCN, the influence of the K^{th}-order neighbour is achieved by *stacking* a number K of 1^{st}-order (linear) layers, rather than using the explicit parametrisation of the Chebyshev expansion with K terms in Equation (4.4). Specifically, the GCN layer maps a signal $V \in \mathbb{R}^{n \times d}$ to a convolved signal $V' \in \mathbb{R}^{n \times d'}$, where d' represents the number of filters or feature maps. Considering only the first-order approximation of Equation (4.4) and under mild assumptions (e.g. reducing the number of free parameters), Kipf and Welling [106] propose the following graph convolutional function:

$$V' = f(A, V) = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}VW, \tag{4.5}$$

where $W \in \mathbb{R}^{d\times d'}$ is a matrix of (trainable) filter parameters, $\tilde{A} = A + I$ is the (renormalised) adjacent matrix, i.e. the adjacency matrix of an undirected graph G with added *self*-connections (loops), and $\tilde{d}_{ii} = \sum_j \tilde{a}_{ij}$ the respective diagonal elements of the degree matrix. Finally, using the function above, a k-layers GCN has the following layer-wise forward rule:

$$V^{(k+1)} = \sigma\left(f(A, V^{(k)})\right) = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}V^{(k)}W^{(k)}\right) \tag{4.6}$$

Here, $V^{(0)} := V$, i.e. the initial node representation (or feature vector), and σ denotes an activation function (e.g. relu or sigmoid).

The GCN largely simplifies previous methods to learn graph node features. It is also has the advantage of being expressed in simple matrix computations. Finally, the architecture is an approximation of convolutional operators on graphs.

4.3.5 Graph Attention Networks

Inspired by attention mechanisms for sequence tasks [112, 120, 116, 14], the Graph Attention Network (GAT) is proposed in [190]

The GAT is defined as a sole layer called the *graph attention layer* and proposes an additive attention mechanism similar to Bahdanau et al. [14] to learn the node representations.

The task is to learn a mapping between the input node features $\{v_1, \cdots, v_n\}$, $v_i \in \mathbb{R}^d$ and the new node features (of potentially different dimension) $\{v'_1, \cdots, v'_n\}$, $v'_i \in \mathbb{R}^{d'}$. In order to achieve expressive power, the authors propose, as an initial step, to apply a linear transformation to every input node feature expressed as

$$k_i = Wv_i, \tag{4.7}$$

where $W \in \mathbb{R}^{d\prime\times d}$ is a learnable parameter.

A shared *pairwise attention* $f_a := \mathbb{R}^{d'} \times \mathbb{R}^{d'} \mapsto \mathbb{R}$ then computes the influence of node j on node i with attention coefficients denoted by [190]:

$$e_{ij} = f_a(k_i, k_j) \tag{4.8}$$

Importantly, this pairwise attention injects the graph structure into the GAT architecture, which *only* calculates e_{ij} for nodes j that are in some neighbourhood of node i [190]

A normalisation on the attention coefficients in Equation (4.8) is then proposed for all node neighbourhoods. Using the attention weight matrix notation of the previous section, this normalisation is given by [190]:

$$[\mathcal{A}]_{ij} = \mathrm{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{j' \in \mathcal{N}(i)} \exp(e_{ij'})}, \tag{4.9}$$

Explicitly, GAT proposes a *pairwise dot product* attention f_a with learnable vector $q \in \mathbb{R}^{2d'}$ expressed as [190]

$$e_{ij} = \sigma\left(q^{\intercal} \cdot [k_i \mid\mid k_j]\right), \tag{4.10}$$

where σ represents a non-linear activation (the LeakyReLU [122]), and $||$ is the concatenation operation between a pair of node features. Expanded out, the GAT attention weight matrix may then be denoted by:

$$\mathcal{A}_{ij} = \frac{\exp\left(\sigma\left(q^{\intercal} \cdot [k_i \mid\mid k_j]\right)\right)}{\sum_{j' \in \mathcal{N}(i)} \exp\left(\sigma\left(q^{\intercal} \cdot \left[k_i \mid\mid k_{j'}\right]\right)\right)} \tag{4.11}$$

After the attention weight matrix is obtained, the new features of the i^{th} node are given by the weighted average:

$$v'_i = \sigma\left(\sum_{j \in \mathcal{N}(i)} \mathcal{A}_{ij} k_j\right) \tag{4.12}$$

Multi-head extension. The authors consider the influence of different aspects of the node features by extending the mechanism to employ *multi-head attention* [188] – in its early incarnation, also called “multiple hops of attention” [116]. Specifically, H independent attention mechanisms are employed, implying that the h^{th}-head has its own independent learnable weights W^h and q^h, as defined in Equations (4.7) and (4.10), respectively. Additionally, the feature of each head, noted ${v'}_i^h$, uses the same transformation of Equation (4.8), but without any activation – delayed to the final heads aggregation step. Finally, the authors propose to average all heads, rather than concatenating as in [188]. For the H-heads attention case, the new feature of the i^{th} node is expressed as:

$$v'_i(H\text{-heads}) = \sigma\left(\mathrm{MEAN}_h\left(\{{v'}_i^1, \cdots, {v'}_i^H\}\right)\right) \tag{4.13}$$

One of the main advantages of GAT compared to the GCN is the flexibility of learning different importances for the neighbour nodes via an attention mechanism. Moreover, these importances are adaptive, since they depend on the own node features (self-attention). However, this mechanism attends to only the nodes connected to a given node i. Precisely, the summations in Equations (4.11) and (4.12) run over the nodes $j \in \mathcal{N}(i)$. Because the neighbourhood varies for each node and is not a fixed-size, a GATHER operation has to be performed to collect the neighbourhood feature vectors for each node i.

In the next section, we present the novel Graph Transformer Network (GTN), which can be expressed entirely in terms of sparse matrix multiplications, as in the GCN, but is still able to discriminate the self and neighbour node contributions. At the same time, it benefits from the attention mechanism, as in GAT, but without the computational limitations of gathering the neighbour vectors for each of the n graph nodes. Thus, speeding-up computations via highly efficient matrix libraries [126].

4.4 Methodology

4.4.1 Universe of Stocks

We start by discussing the universe of stocks under analysis and its classification structure.

In this chapter, we extended the initial universe of stocks considered in Chapter 3. First, we consider all eleven sectors from each SPDR Sector fund and all stock holding of each sector. Using this procedure, we have a total of 388 stocks aggregated into 11 sectors[13]. Second, we use a more fine-grained classification structure, where we made use of the *Morningstar Global Equity Classification Structure* [153], described in the next section.

Sector and Industry information

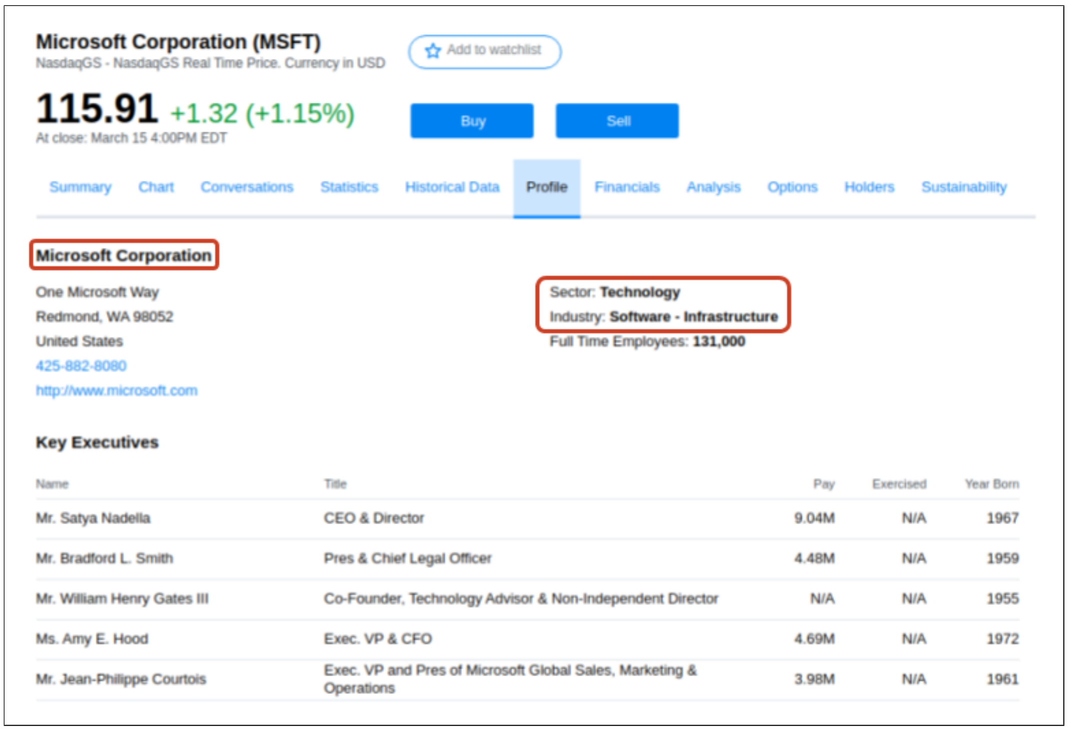
In the Morningstar Global Equity Classification Structure [153] each stock is mapped into one of 213 industries, which, in turn, are folded into a higher level representing 11 sectors. Each industry group reflects the underlying business of the company and is based on publicly available information, e.g. annual reports and Security Exchange Commission (SEC) Form 10-Ks. Additionally, companies that have more than three sources of revenue, without any clear dominant stream, are assigned as a diversified industry [153]. To provide an example of the classification structure [153], the Utilities sector is broken down into the following industries: Electric Utilities, Gas Utilities, Water Utilities and Diversified Utilities.

Yahoo Finance adopts the Moringstar classification structure, therefore, the metadata is publicly available, where we extracted the following fields: company name, sector and industry groups, for the 388 stocks in our universe. An example of Microsoft metadata is shown in Figure 4.1.

4.4.2 Graph Transformer Networks

In this section, we describe the novel Graph Transformer Network (GTN) which adapts the Transformer [188] attention mechanism to model graph structures.

[13]We have the following sectors: Basic Materials, Consumer Cyclical, Consumer, Defensive, Communication Services, Energy, Financial Services, Healthcare, Industrials, Real Estate, Technology, Utilities



Microsoft Corporation (MSFT)
NasdaqGS - NasdaqGS Real Time Price. Currency in USD

115.91 +1.32 (+1.15%)
At close: March 15 4:00PM EDT

Microsoft Corporation

One Microsoft Way
Redmond, WA 98052
United States
425-882-8080
http://www.microsoft.com

Sector: **Technology**
Industry: **Software - Infrastructure**
Full Time Employees: **131,000**

Key Executives

Name	Title	Pay	Exercised	Year Born
Mr. Satya Nadella	CEO & Director	9.04M	N/A	1967
Mr. Bradford L. Smith	Pres & Chief Legal Officer	4.48M	N/A	1959
Mr. William Henry Gates III	Co-Founder, Technology Advisor & Non-Independent Director	N/A	N/A	1955
Ms. Amy E. Hood	Exec. VP & CFO	4.69M	N/A	1972
Mr. Jean-Philippe Courtois	Exec. VP and Pres of Microsoft Global Sales, Marketing & Operations	3.98M	N/A	1961

Figure 4.1: **Microsoft profile web page at Yahoo Finance**. Screenshot of MSFT (Microsoft) profile tab. The red boxes highlight the HTML tags: company name (left) as well as sector and industry groups (right).

In line with previous research, we make use of an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, V)$ (with n nodes) to inject the *relational bias*, where each node attribute/feature vector v_i is represented in a compact form by $V \in \mathbb{R}^{n \times d}$. Our goal is also to learn the mapping $f : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d'}$ from the initial node features V (randomly initialised or not) to new node features (of potentially different dimension d') V', where the learning process is expected to produce node features that capture the graph structure.

Goals. We designed the GTN with the following goals in mind:

G.1 *Generalise* the *fixed* node weighting scheme of the GCN to an attention mechanism, where weights depend on the node features V.

G.2 *Discriminate contributions* from *self* (or target) node i to its *neighbours* j (in the averaging step). In this way, enhancing GCN and also targeting the performance improvements reported in GraphSAGE [81] and GAT [190].

G.3 *Computational efficiency.* As in GCN, our architecture can be solely represented in terms of matrix multiplications, which can be implemented using highly optimised and parallel matrix multiplication libraries. This is in contrast to previously proposed architectures, e.g. GAT and GraphSAGE, which rely on individual computations for each node v_i, i.e. they cannot be presented in a "packed" matrix form.

GCN weighting scheme. We start by denoting the GCN's *graph convolution layer* in a way that allows us to easily identify its terms. That is [106]:

$$V' := \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}VW\right) = \sigma\left(\overline{V}W\right), \quad \text{where} \tag{4.14a}$$

$$\overline{V} = \mathcal{A}^u V \quad \text{and} \tag{4.14b}$$

$$\mathcal{A}^u = g(A) = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} = \begin{cases} \frac{1}{\sqrt{d_{ii}d_{jj}}} & \text{if } a_{ij} \neq 0 \text{ (connected);} \\ 0 & \text{if } a_{ij} = 0 \text{ (\textit{not} connected)} \end{cases} \tag{4.14c}$$

Here, the GCN only (trainable) parameter is a shared (to all nodes) *graph projection matrix* $W \in \mathbb{R}^{d \times d'}$ that acts on the "averaged" node features $\overline{V}$. This average is computed by the product of the *fixed (unnormalised) weight matrix* $\mathcal{A}^u$ with elements a^u_{ij}, which quantifies the influence of neighbour nodes j's on a given self (target) node i. We use the superscript u to highlight the lack of node-wise normalisation, i.e. $\sum_j a^u_{ij} \neq 1$.

The GCN has the following characterists:

P.1 Injects *graph structure* (i.e. relational bias) via the (one-hop) adjacency matrix A. Using Equation (4.14b), if a given node j is not connected to node i, i.e. $a_{ij} = 0$, its contribution is efficiently filtered out in the matrix multiplication.

P.2 Average representations $\overline{V}$ can be efficiently computed using sparse-dense matrix multiplications (time complexity linear in the number of graph edges $|\mathcal{E}|$).

P.3 The (fixed) weights are given by the linear approximation of (localised) *spectral convolutions on graphs* [106, 45].

P.4 If a set of graph nodes have the *same degree* (of importance) d, their *weights are uniform*, i.e. $a_{ij} = 1/d$.

P.5 Even though the weights depend on the degree, during the averaging step, no distinction can be made between the self node i and its neighbours js.

GTN layer – Attention-agnostic. In order to generalise the GCN (Item G.1), without sacrificing its computational efficiency (Item G.3), we propose a convolutional layer that follows the same structure as the GCN. In its attention-agnostic form, it computes the new node features:

$$V' = \sigma\left(\overline{V}W\right), \quad \text{where} \tag{4.15a}$$

$$\overline{V} = \mathcal{A}V \quad \text{and} \tag{4.15b}$$

$$\mathcal{A} = g(A, V) = \begin{cases} a_{ij} & \text{if } a_{ij} \neq 0 \text{ (nodes are connected);} \\ 0 & \text{if } a_{ij} = 0 \text{ (nodes are \textit{not} connected)} \end{cases} \tag{4.15c}$$

Here, the attention weight matrix $\mathcal{A} = g(A, V)$ is learnable and depends explicitly on V. However, as in the GCN, it still injects the graph structure via matrix A (see Item P.1). Additionally, because we only modify the attention mechanism, we call the layer *convolutional*, as detailed in Item P.3.

Given the general architecture above, our problem boils down to proposing an attention mechanism (as denoted in Equation (4.15b)) that can discriminate the contributions from self and neighbour nodes during the learning process (Item G.2) and also can compute new node features efficiently (Item G.3). To this end, we propose the *Graph Scaled-Dot Product attention*, which closely follows the Transformer [188], but endows it with the following properties:

- Invariance under permutation of node's neighbours. A graph has no natural ordering; therefore, the influence of neighbour nodes should be computed by a symmetric function, i.e. invariant to the order of its input.
- Trainable *self* and *neighbour* embedding, here called *graph role embedding.* This embedding will discriminate between the two contributions (as detailed in Item G.2).
- Graph structure injected via A matrix and conforming to Equation (4.15c)

GTN layer. The *graph transformer layer* operates on a graph, where relations are expressed by adjacency matrix $A \in \{0,1\}^{n \times n}$, by mapping node features $V \in \mathbb{R}^{n \times d}$ to new node features $V' \in \mathbb{R}^{n \times d'}$. As illustrated in Figure 4.2, the layer computes the influence of neighbour nodes js on a given target node $i = \{1, \cdots, n\}$, by first projecting neighbour node features into keys (K), in a step called *key-value pairs*, and then *querying* the node i (self) projected value, noted q_i, against all neighbour keys. Importantly, all node queries are computed simultaneously and represented by the query matrix Q. Fully expanded out, the new node features V' may then be denoted by:

$$Q = (V + e_{\text{sf}})W^Q \tag{4.16a}$$

$$K = (V + e_{\text{ns}})W^K \tag{4.16b}$$

$$V' = \sigma\big(\underbrace{\text{softmax}(\frac{QK^\intercal}{\sqrt{d_k}} \odot A)}_{\mathcal{A}=g(A,Q,K)} VW\big), \tag{4.16c}$$

where the softmax function normalises all weights of the i^{th} node, and we have the following trainable parameters: 1) (attention) projection matrices $W^Q \in \mathbb{R}^{d \times d_k}$ and $W^K \in \mathbb{R}^{d \times d_k}$, 2) (graph) weight matrix $W \in \mathbb{R}^{d \times d'}$, and 3) vector embeddings[14] *self* $e_{\text{sf}} \in \mathbb{R}^d$ and *neighbours* $e_{\text{ns}} \in \mathbb{R}^d$. Additionally, the element-wise product in

[14]The + operator in Equations (4.16a) and (4.16b) represents a Matrix-vector sum that makes use of *broadcast operations*. Specifically, before computing the sum, the vector is converted into a matrix by stacking row-wise n copies of the same vectors. This broadcast operation is efficiently implemented (in terms of time and space complexities) by many libraries (e.g. Numpy, TensorFlow, Theano and PyTorch). Note that n is the matrix number of rows.

Equation (4.16c), noted $\odot$, ensures the attention weight matrix $\mathcal{A}$ is sparse (satisfying Item G.3), and in our model the graph structure is injected via A matrix (as described in Item P.1). As a consequence, effectively neighbour nodes js not connected to node i are not attended, i.e. have zero attention weights.

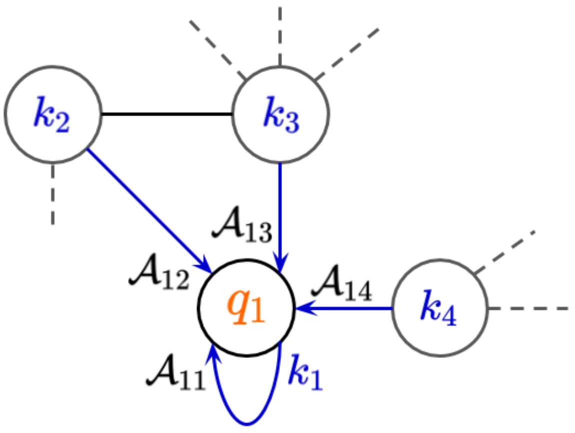


Figure 4.2: **GTN – Graph scaled dot-product attention**. For illustration purposes the *querying* is performed at node $i = 1$, which has 3 first-neighbours. The goal is to learn the 4 attention weights $\mathcal{A}_{1j}$ with $j = \{1, 2, 3, 4\}$, which includes the self-loop relation (curved arrow). In this architecture we only show the weights different than zero ($A_{ij} \neq 0$). The *compatibility* between the node 1 features and its neighbours is computed by the vector dot-product $q_1 \cdot k_j$ that, apart from a normalisation constant, is given by Equation (4.16c). In practice, Equation (4.16c) computes the neighbourhood attention weights for all n graph nodes "in one go" (via matrix multiplication), and the whole graph attention weight pairs are noted $\mathcal{A}$.

One important aspect of our architecture are the trainable vector embedding e_{sf} and e_{ns}. Both are introduced to discriminate the contributions between the node self-connection and its neighbours. Because nodes can have different roles depending on where they are being queried or sending keys, we call the GTN embedding a *graph role embedding*. Below, we compare our graph role embedding with a positional embedding.

Our graph role embedding versus positional embedding. The positional embedding is an integral part of Transformer architecture [188]. This encoding is represented by a (learnable) embedding matrix,[15] where each row indicates the position of the input vector v_i in a sequence of size n [188]. Specifically, as in GTN, the positional embedding matrix is added to the input matrix V, i.e. $V \leftarrow V + \text{Embedding matrix}$ (as defined in Equation (4.16a)). However, our network aims

[15] A fixed (rather than learnable) positional encoding, represented by a sinusoidal function was also proposed for the Transformer. However, experiments show that both fixed and learnable encodings produce a similar performance [188].

to describe *graph mappings*, where the n nodes have no natural ordering. Thus, by using the same positional encoding, which is designed for order dependent sequences, we would not be exploiting the permutation invariance. Thus, rather than adding a vector to each token, as in the Transformer, our graph role embedding adds the same embedding vector to all neighbours. This implies that the attention function has the desired graph symmetry, i.e. it is invariant to permutations of neighbour node features.

In addition, the two embedding vectors of our *graph role embedding* account for the two different roles that a node plays, as detailed in Item G.2. That is, given the node feature v_i, when computing the influence of the node on itself the query q_i carries information about the self embedding e_{sf}. However, when the same node assumes the role of neighbour its feature v_i is passed using a different embedding (as denoted in Equation (4.16a)).

Finally, it is worth noting that the graph role embedding is trainable. Thus, its self and neighbours' embedding can converge to similar representations during the learning process. However, our experiments show that these representations are distinct. To make it clear, we show that the attention weight of the node on itself (self) is much greater than against its neighbours. This evidence the relevance of the graph role embedding to the GTN architecture.

GTN layer – Multi-head attention. In the multi-head attention, each head focus on different aspects/views of the input features [116, 188]. Additionally, it has been shown that multi-head architectures stabilise neural networks training [188]. To benefit from this, we extend the Transformer [188] multi-head formulation to our graph layer. Below we describe the multi-head version of GTN, as illustrated by Figure 4.3.

In order to consider different views of the node input data (i.e. attention heads), we compute H independent attention matrices $\mathscr{A}_h = g(A, Q_h, K_h)$, where Q_h and K_h are given in terms of V and the graph role embeddings as in Equations (4.16a) and (4.16b), respectively.

Once the H attention matrices are obtained, the new head node feature, denoted head_h, is computed by multiplying each head attention matrix by VW, as denoted i81n Equation (4.15b). At this stage, we could employ *averaging*, as proposed in the GAT architecture [190], to aggregate all H heads contributions. However, as noted in [188], the averaging process inhibits *jointly* attending to information at different positions (here nodes) of each of the H sub-spaces.

Thus, by using the heads aggregation proposed in [188], with (trainable) *output* projection matrix $W^O \in \mathbb{R}^{hd' \times d'}$, the *H-heads graph transformer layer* new node features may then be given by:

$$V'_{H\text{-heads}} = \sigma\left(\mathrm{Concat}(\mathrm{head}_1, \cdots, \mathrm{head}_H)W^O\right), \quad \text{where} \tag{4.17a}$$

$$\mathrm{head}_h = \mathcal{A}_h V W \tag{4.17b}$$

and the (learnable) h^{th}-head attention weight matrix $\mathcal{A}_h$ is computed by projecting V into Q_h and K_h using the trainable weights $W_h^Q \in \mathbb{R}^{d\times d_k}$ and $W_h^K \in \mathbb{R}^{d\times d_k}$. That is:

$$Q_h \leftarrow (V + e_{\text{sf}})W_h^Q \tag{4.17c}$$

$$K_h \leftarrow (V + e_{\text{ns}})W_h^K \tag{4.17d}$$

$$\mathcal{A}_h = g(A, Q_h, K_h) = \mathrm{softmax}\left(\frac{Q_h K_h^\intercal}{\sqrt{d_k}} \odot A\right) \tag{4.17e}$$

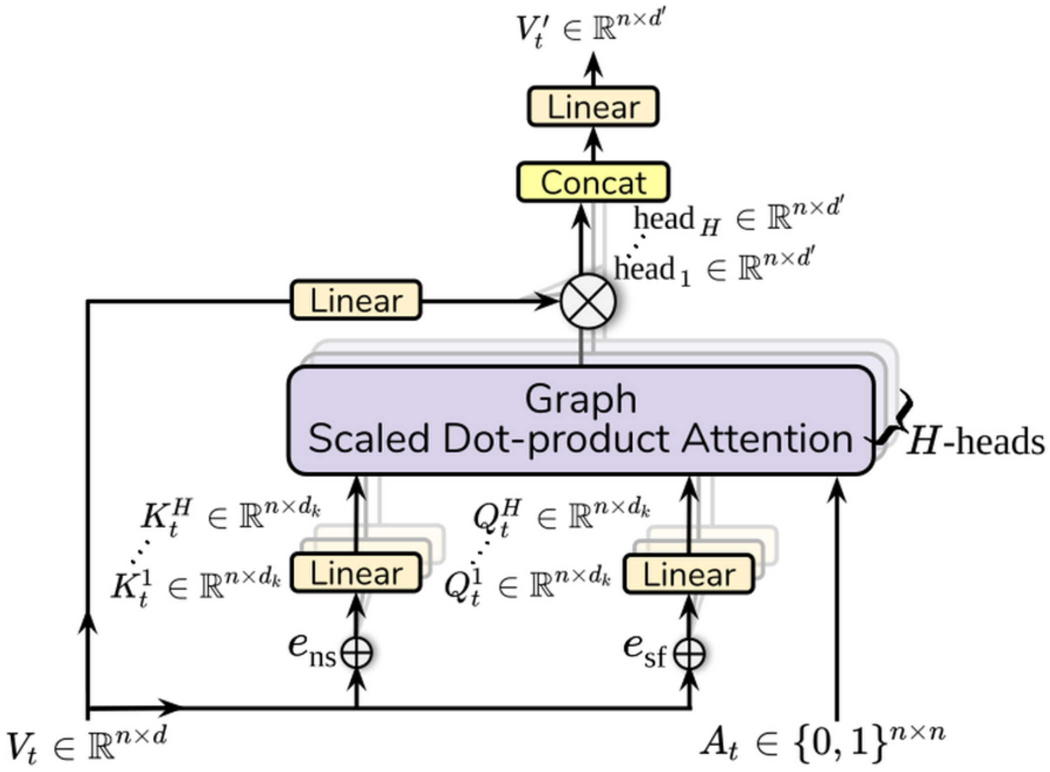

$V'_t \in \mathbb{R}^{n\times d'}$
Linear
Concat
head$_H \in \mathbb{R}^{n\times d'}$
head$_1 \in \mathbb{R}^{n\times d'}$
Linear
Graph Scaled Dot-product Attention
H-heads
$K_t^H \in \mathbb{R}^{n\times d_k}$
$K_t^1 \in \mathbb{R}^{n\times d_k}$
Linear
$Q_t^H \in \mathbb{R}^{n\times d_k}$
$Q_t^1 \in \mathbb{R}^{n\times d_k}$
Linear
e_{ns}
e_{sf}
$V_t \in \mathbb{R}^{n\times d}$
$A_t \in \{0,1\}^{n\times n}$

Figure 4.3: **GTN – H-heads graph transformer layer**. From bottom (input) to up (output) the entire layer is represented by the mapping $(V_t, A_t) \mapsto V'$. The Graph scaled dot-product attention (purple) computes H attention matrices *in parallel*, where the h^{th} attention matrix $\mathcal{A}_h = g(A, Q_h, K_h)$ is denoted in Equation (4.17e). Each parallel *head$_h$* is a weighted average of the (projected) input node features V (as in Equation (4.17b)), and this projection is shared among all graph nodes. The final linear transformation operates on the (row-wise) head's concatenation (as noted in Equation (4.17a)) and outputs one feature per graph node.

We observe that, in contrast to the Transformer architecture [188], the (trainable) graph weight matrix W, as in the single-head case, is *shared between all heads and consequently all nodes*. The main reason to adopt this approach is to to keep our architecture *convolutional*, since the W term comes from the linear approximation a (localised) spectral convolution on graphs (as explained in Item P.3), which in turn, requires the architecture to follow the attention-agnostic formulation in Equation (4.15).

In our experiments we employ the H-head graph transformer layer discussed in Equation (4.17) with $H = 3$ parallel heads and consider for each head the influence of 2-hops neighbours', by *stacking* two 3-head graph transformer layers. To stack these layers, we set the feature dimension of the output V' equal to the input V, i.e. $d' = d$. According to Equation (4.17), this implies an output projection matrix $W^O \in \mathbb{R}^{hd\times d}$ and a graph projection weight $W \in \mathbb{R}^{d\times d}$.

4.4.3 Time Series Prediction with Relational Information

We consider a general setting for multivariate time series prediction using graph neural networks. In this setting, the prediction function is approximated by a neural network.

The main idea is to represent each time series as a graph node and the explanatory variables as graph attributes. Using graph networks prior knowledge about the time series relations is represented by the graph adjacency matrix A.

Importantly, as in the financial markets applications, we consider the case where these relationships adapt or vary over time. This makes the framework flexible enough to accommodate the temporal structures reviewed in Section 4.2.1.

We start by representing the *input feature* of the j^{th} time series at time t by $X_t^j = [x_{t-T+1}^j, \cdots, x_t^j]^\intercal \in \mathbb{R}^{T\times d}$, where T is the history period (or number of tokens) and d_i the input feature dimension. In order to jointly model n sequences, we pack all features into a single *joint input feature* tensor:

$$\mathcal{X}_t = \left[X_t^1, \cdots, X_t^n\right]^\intercal \in \mathbb{R}^{n\times T\times d} \tag{4.18}$$

Following the formulation in Section 4.3, we inject the relational inductive bias, i.e. relational information, via an attributed graph, where each node is associated with one input feature (e.g. each node is paired with a constituent stock of a market index or portfolio). However, we now consider a (time-dependent) graph $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t, V_t)$, where nodes (or vertices) are stationary[16] but the node relations, expressed by the adjacency matrix $A_t \in \{0,1\}^{n\times n}$, and its attribute/features V_t depend on each time t. With added bias, our goal is to learn the *graph prediction function*:

$$\hat{y}_{t+1} = f_P\left(\mathcal{X}_t, A_t\right), \tag{4.19}$$

[16]Meaning no nodes are added or dropped from the original graph over time.

that maps the feature space to the target labels space, where the ground-truth label is represented by the vector $y = \left[y_t^1, \cdots, y_t^n\right] \in \mathbb{R}^n$ (one target label per node). This prediction function can be learnt using a neural network with trainable weights by minimising a loss function $\mathcal{L}(\hat{y}, y)$.

General neural network architecture

As illustrated in Figure 4.4, the graph prediction function $f_P : \mathbb{R}^{n \times T \times d_i} \times A_t \in \{0,1\}^{n \times n} \to \mathbb{R}^n$ can be approximated using a (general) network architecture with the following components:

GA.1 **Sequence Encoder:** A single encoder is employed for all n nodes (or time series), i.e. the (trainable) weights are shared among all nodes. By sharing the weights we make the full architecture scalable for graphs with a large number of nodes. The only restriction being that the layer output is a vector, which in a packed form is represented by the matrix V_t.

GA.2 **Graph Network**: Injects the relational bias. Given its input and output dimensions $V \to V'$, this component accommodates any GNN layer described in Section 4.3, including the novel GTN.

GA.3 **MLP**: Standard final layer that converts the learnt representations of each node (in d'-dimensional embedding space) into a n-dimensional vector (one target per node). As in the sequence encoder this layer is shared between all nodes (or stocks).

4.4.4 Price Range Prediction with Relational Information

Our goal is to predict the high and low price of a portfolio consisting of n stocks leveraging the graph structure of its stocks. To this end, we associate each stock with a graph node and the relational bias is injected via the adjacency matrix $A_t \in \{0,1\}^{n \times n}$, which express the presence or absence of relationships among the stocks at a given time t.

The price range architecture is similar to the general one described in the previous section, but employs multi-task learning. Below, we describe the setting and then present the full price range architecture.

Multi-task Component

In contrast to previous research [77, 193, 127], where a model is trained for high and low prediction separately, we employ a multi-task setting with hard parameter sharing [36, 156, 203]. Thus, we are able to share the network part, which learns the representations, while keeping independent *task-specific* layers.

Apart from halving the computational time necessary to train each prediction task independently, by using multi-tasking learning we can benefit from the following:

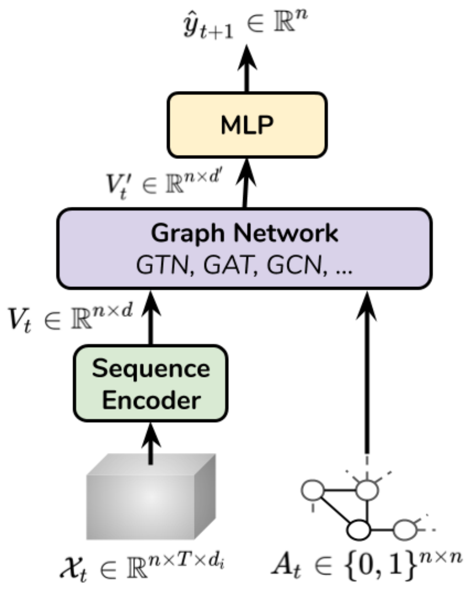

$\hat{y}_{t+1} \in \mathbb{R}^n$
MLP
$V'_t \in \mathbb{R}^{n \times d'}$
Graph Network
GTN, GAT, GCN, ...
$V_t \in \mathbb{R}^{n \times d}$
Sequence Encoder
$\mathcal{X}_t \in \mathbb{R}^{n \times T \times d_i}$
$A_t \in \{0,1\}^{n \times n}$

Figure 4.4: **Time series prediction with relational information – General architecture.** A schematic view of the three layers the entire general architecture (detailed in Items GA.1 to GA.3). At the bottom of the architecture, the input is composed of the 3-d tensor $\mathcal{X}_t$, representing all n time series input features, and the graph adjacency matrix A (representing the nodes pairwise relationship). The output vector $\hat{y}_{t+1}$ represents one target per node.

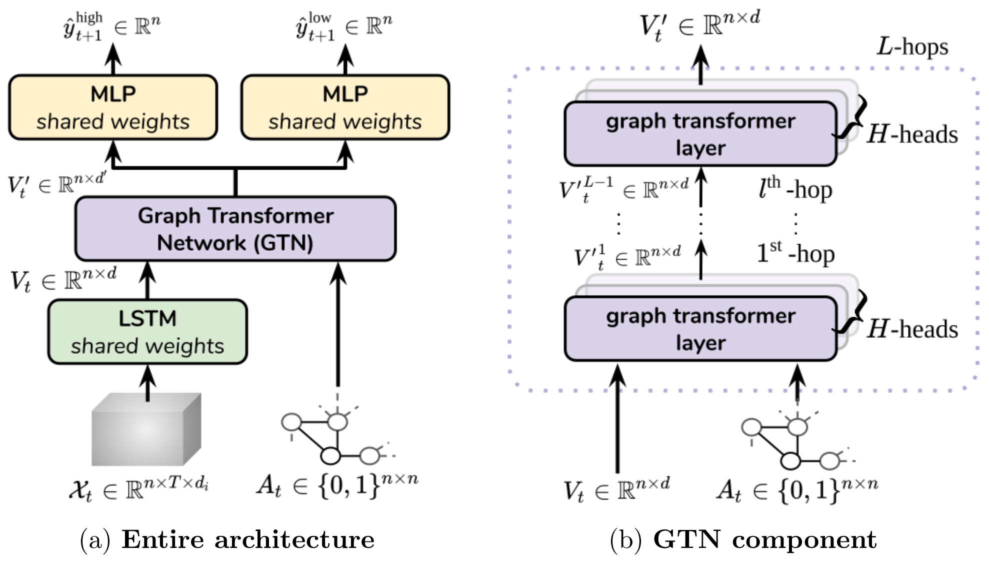
1) A natural fit – both tasks are closely related (financial prediction) and expected to benefit from the same representation [36, 156] 2) Generalisation – during optimisation representations are biased to prefer all tasks jointly, therefore, are expected be useful for "unseen" tasks from the same domain [20], and 3) Attention focusing – financial data is very noisy; thus, learning both tasks provides additional evidence on the relevant features, i.e. those that effectively benefit all tasks, making noisy signals irrelevant [36, 156].

Loss. In order to (jointly) learn the two tasks, the multi-task loss has to account for each task loss, here, denoted $\mathcal{L}$ (i.e. same function for both tasks). This loss is computed for the high prediction task using the ground-truth labels $y^{\text{high}}_{t+1} \in \mathbb{R}^n$ (one per node) and $y^{\text{low}}_{t+1} \in \mathbb{R}^n$ for the low prediction task. Additionally, we consider the high and low tasks equally important for trading purposes – meaning the task losses are equally weighted – and, as in other chapters, we use the (square) L2-loss function. Thus, the *price range neural network* is trained by minimising the *multi-task loss*:

$$\begin{aligned}\mathcal{L}_{MTL} &= \frac{1}{2}\left[\mathcal{L}\big(\hat{y}^{\text{high}}_{t+1}, y^{\text{high}}_{t+1}\big) + \mathcal{L}\big(\hat{y}^{\text{low}}_{t+1}, y^{\text{low}}_{t+1}\big)\right] \\ &= \frac{1}{2}\frac{1}{N}\frac{1}{n}\left[\sum_{t=0}^{N} \|\hat{y}^{\text{high}}_{t+1} - y^{\text{high}}_{t+1}\|_2^2 + \sum_{t=0}^{N} \|\hat{y}^{\text{low}}_{t+1} - y^{\text{low}}_{t+1}\|_2^2\right], \qquad (4.20)\end{aligned}$$

where N is the number of training samples and $\|\cdot\|_2$ the L2-norm operator.[17]

Multi-task Graph Neural Network Architecture


$\hat{y}^{\text{high}}_{t+1} \in \mathbb{R}^n$
$\hat{y}^{\text{low}}_{t+1} \in \mathbb{R}^n$
MLP shared weights
MLP shared weights
$V'_t \in \mathbb{R}^{n \times d'}$
Graph Transformer Network (GTN)
$V_t \in \mathbb{R}^{n \times d}$
LSTM shared weights
$\mathcal{X}_t \in \mathbb{R}^{n \times T \times d_i}$
$A_t \in \{0,1\}^{n \times n}$
$V'_t \in \mathbb{R}^{n \times d}$
L-hops
graph transformer layer
H-heads
$V'^{L-1}_t \in \mathbb{R}^{n \times d}$
l^{th}-hop
$V'^{1}_t \in \mathbb{R}^{n \times d}$
1^{st}-hop
graph transformer layer
H-heads
$V_t \in \mathbb{R}^{n \times d}$
$A_t \in \{0,1\}^{n \times n}$

(a) **Entire architecture** (b) **GTN component**

Figure 4.5: **Price range prediction with relational information – Architecture.** *Left:* The entire network is composed of three layers (detailed in Section 4.4.4). The input (bottom) consists of the n stocks input features (3-d tensor $\mathcal{X}_t$) and the graph adjacency matrix A (expressing the pairwise stock relationships). The Graph network (purple) converts the input graph into another graph with new node features V'_t, and its constituent layers are detailed on the right. Each MLP (orange) represents the *task-specific* layers and outputs the next period high and low prices for all n stocks. *Right:* Detailed view of the GTN. Each of the L layers have H heads (discussed in Equation (4.17)). Each head is expected to *attend to different aspects/views* of the same input nodes, with the aggregated output of H-heads discussed in Equation (4.17a). By *stacking* graph layers we increase the first-neighbours, represented by A_t, to l-neighbours/hops.

The architecture is depicted in Figure 4.5a. The task-specific layers (in orange) output the targets (high and low prices) for all n nodes, and the remaining components, i.e. Sequence encoder (green) and Graph Network (purple), learn representations common to both tasks. Below we describe all components.

Sequence Encoder: At a given time t, we feed the i^{th} stock history (of length T) to a LSTM network (with share weights), which outputs a sequence with T hidden states. We then take the last hidden state h_T to represent the encoded stock node feature, i.e. $v_i = h_T$. We denote this pooling operation $LAST$. In a compact matrix form the component (shown in green) may then be expressed as:

$$V_t = \text{LAST}(\text{LSTM}(\mathcal{X}_t)) \tag{4.21}$$

[17]Even though we explicitly denote all constants, they are omitted by some authors. This is due the fact that the constant c can be absorbed into the learning rate constant η: $W \leftarrow W + \eta\nabla(c\cdot\mathcal{L}) = W + (\eta \cdot c)\nabla(\mathcal{L})$.

Graph Network: Injects the relational bias via the graph adjacency matrix A. In our experiments, the novel GTN is implemented by stacking the two graph transformer layers in Equation (4.17), both with three heads and denoted $\mathrm{GT}_{\text{3-heads}}$. The layers are depicted in Figure 4.5b, where L=2 and H=3. This component (in purple) is then computed by:

$$\begin{aligned} V_t' &= \mathrm{GTN}(V_t, A^t) && (4.22) \\ &= \mathrm{GT}_{\text{3-heads}}(\mathrm{GT}_{\text{3-heads}}(V_t, A^t)) && (4.23) \end{aligned}$$

Task-specific layers: These two FC layers, denoted $\mathrm{FC}^{\text{high}}$ and FC^{low} output the high and low prediction for all nodes, i.e. weights are shared between the n nodes. Note each layer has shared weights. The two components (yellow) are then given by:

$$\hat{y}_{t+1}^{\text{high}} = \mathrm{FC}^{\text{high}}(V_t') \tag{4.24a}$$

$$\hat{y}_{t+1}^{\text{low}} = \mathrm{FC}^{\text{low}}(V_t') \tag{4.24b}$$

4.4.5 Range of Stocks Trading

In this section we describe how we exploit the high and low predictions to trade stocks.

We employ the range trading strategy as proposed in [127, 193]. The daily OHLC prices of given stock is denoted by $y_t^{\text{open}}, y_t^{\text{high}}, y_t^{\text{low}}, y_t^{\text{close}}$. The trade gain/loss is denoted by Δ.

At a given day t the strategy is defined by the following rules [127, 193]:

TS.1 if $y_t^{\text{low}} < \hat{y}_t^{low}$ and $y_t^{\text{high}} > \hat{y}_t^{high}$ then $\Delta = \hat{y}_t^{high} - \hat{y}_t^{low}$ (intraday price exceeds the predicted (daily) range. Profitable long or short trade)

TS.2 if $y_t^{\text{low}} < \hat{y}_t^{low}$ and $y_t^{\text{high}} < \hat{y}_t^{high}$ then $\Delta = y_t^{\text{close}} - \hat{y}_t^{low}$ (intraday price exceeds the predicted low, but not the high. Entry long trade at $\hat{y}_t^{low}$ and exit at y_t^{close})

TS.3 if $y_t^{\text{low}} > \hat{y}_t^{low}$ and $y_t^{\text{high}} > \hat{y}_t^{high}$ then $\Delta = y_t^{\text{high}} - y_t^{\text{close}}$ (intraday price exceeds the predicted high, but not the low. Entry short trade at $\hat{y}_t^{high}$ and exit at y_t^{close})

Trading strategy phenomenon. In essence, the proposed trading strategy attempts to profit from the fact that stock markets exhibit *excess volatility* [179, 165]; here, meaning that the intraday price swings exceed the daily range predicted by the model. Because of this characteristic, the trading strategy is non-directional. In other words, we initiate trade when the price during the day expands beyond the limits stipulated by the range model in *any* direction. By doing so, we hope the price will revert, rather than continuing in the same direction as the breakout. Thus, we wait until the market closes to profit on the reversion. Whether the "reverts to mean" hypothesis is supported or not by the available data is indicated by the strategy performance on the test data.

The main advantage of the range trading strategy is that the trades are initiated and terminated during the same trading session. As an intraday trading strategy, its investor is *not* subject to the *gap risk*. The so-called gap risk happens when there are adverse announcements overnight, when the stock exchange is shut, and the opening price is substantially lower (or higher) than the previous day's closing price – note the gap occurs *with no trade in between both prices*.

Limitations of using daily OHLC data for performance evaluation. Relying only on OHLC data imposes limitations on the performance evaluation of the trading strategy. When the Item TS.1 is triggered, i.e. the price exceeds the band both sides, we cannot decide the trade type (i.e. long or short). In other words, we are sure about the gain, $\Delta = \hat{y}_t^{high} - \hat{y}_t^{low}$, as defined in Item TS.1, though we cannot determine if the trade is long (low price was hit first) or short (high price was hit first). Another limitation is that without intraday data we cannot determine if there is more than one trade during the day; in days of intense market swings it could be the case that the trade is triggered more than once and this can not be determined using OHLC daily data. These two limitations, namely the trade type and the number of occurrences for rule Item TS.1, lead to assumptions, which are describe below.

Assumptions. When rule Item TS.1 is triggered we cannot determine the trade type, i.e. long (positive) or short (negative) or the number of times the trade occurred within the same day. Thus, if rule Item TS.1 is triggered we make the following assumptions: (1) we adopt a conservative approach and assume that only one trade was triggered per day, and (2) we randomly assign half of the positions as long and the other half as short. This way, we avoid any bias to the long trade side, which would result in better performance for the up trending stock market in the test period. Note that (2) is only implemented when rule is Item TS.1.

Based on the assumptions above we can determine: (1) the trade type, (2) the entry and exit prices, and (3) the trade return. In the next section, we provide details on how the range trading strategy is evaluated in the portfolio context.

4.4.6 Portfolio Evaluation

Depending on the range prediction of each model under evaluation, we will have different positions on a given day t, where the position to be taken on each stock can be long (1), short (-1) or neutral (0). The number of long positions is denoted by n_t^l and the number short positions by n_t^s such that $n = n_t^l + n_t^s$. These positions form a portfolio with weighs denoted by $w_t = \left[w_t^1, \cdots, w_t^n\right] \in \mathbb{R}^n$.

Self-financing portfolio

In this chapter we evaluate the portfolio performance considering that all available wealth is *equally distributed* among the stocks to go long and the stock to go short, and that we have a self-financing portfolio (as described in Section 2.2.2). Then the portfolio weights are given by:

$$w_t^i = \begin{cases} \dfrac{1}{n_t^l}, & \text{for long positions;} \\ -\dfrac{1}{n_t^s}, & \text{for short positions;} \\ 0, & \text{for neutral positions} \end{cases} \tag{4.25}$$

Position sizing

Now we apply a money management mechanism to define how much of the available wealth should be allocated on each day. As discussed in Section 2.2.3, we adopt the *volatility targeting* method proposed in [96, 34] as a money management mechanism. By using the proposed mechanism we scale down the portfolio at times of high volatility (high risk) and leverage at times of low volatility. Effectively, the portfolio is always targeting a constant level of ex-ante volatility σ_{target}, which is set to 10% per year.

4.5 Experimental Results and Discussion

4.5.1 Training Setup

In our experiments we guard against overfitting and unrealistic trading performances (caused by training data leakage). Thus, we completely separate the data from 2016 and 2017 and assign it to the test set. Unless otherwise noted, results are reported on this “unseen” test set. The remaining data is further split into training (2007 to 2013) and validation (2014 and 2015), where we set the sliding window 252 days, i.e. one year of historical data, to estimate the volatility and adjust the size of each position. We monitor the validation loss at the end of each epoch and employ early-stopping with patience set to eight epochs, meaning that if the validation loss does not improve for eight consecutive epochs we abort training. If the early-stopping is not triggered, we continue the training up to a maximum of sixty five epochs.

The neural network weights are updated during the training process using mini-batch SGD [27] with Adam [105] optimiser.

We tune the hyperparameters using grid search. The search space and best parameters are reported in Appendix C.1.

4.5.2 Price Input Features

The price input features for the i^{th} stock on a given day t is expressed as:

$$x_t^i = \Bigg[\overbrace{\frac{y_{t+1}^{i,\text{open}}}{y_t^{i,\text{open}}}, \frac{y_{t+1}^{i,\text{open}}}{y_t^{i,\text{high}}}, \frac{y_{t+1}^{i,\text{open}}}{y_t^{i,\text{low}}}, \frac{y_{t+1}^{i,\text{open}}}{y_t^{i,\text{close}}}}^{\text{Gap return } t+1}, \overbrace{\frac{y_t^{i,\text{close}}}{y_{t-1}^{i,\text{open}}}, \frac{y_t^{i,\text{close}}}{y_{t-1}^{i,\text{high}}}, \frac{y_t^{i,\text{close}}}{y_{t-1}^{i,\text{low}}}, \frac{y_t^{i,\text{close}}}{y_{t-1}^{i,\text{close}}}}^{\text{Closing return } t} \Bigg] - 1 \quad (4.26)$$

In terms of implementing the trading strategy in "live trading", we need to wait for the market to open in order feed the opening price to our models. This same setting was also used in previous studies [127, 77].

4.5.3 Financial Market Graphs

In our experiments, we inject the relational bias using two graphs represented by the Sectoral (A_s) and (A_J) adjacency matrices, described below:

- Sectoral graph (A_s): This graph is built based on the sector of each stock, where we use the classification structure with 11 sectors detailed in Section 4.4.1. Thus, if two stocks are in the same sector the edge value, represented by A_s elements, is one and it is zero otherwise.
- Fully connected graph (A_J): We consider all stocks connected to each other. More specifically, this graph is represented by the all ones matrix J.

We observe that because each stock is allocated to only one sector, the sector super(graph) is composed of 11 subgraphs (one per sector), where stocks in a given subgraph, i.e. in the same sector, are connected to each other, but to no other stock in the (super)graph A_s.

4.5.4 Evaluation – Models and Baselines

We compare the performance of the range trading strategy discussed in Section 4.4.5 using 5 models and 4 baselines. For the 5 models, we have 4 graph models, which inject the graph information using the relational biases A_s and A_J described above, and one LSTM model, which disregard all graph information. Below, we describe the 5 five models and, subsequently, the 4 baselines:

- GTN A_s: The novel GTN with the A_s graph structure. The entire multi-task graph neural network architecture is depicted in Figure 4.5a, where we use $H = 3$ heads and stack $L = 2$ *graph transformer layers*, as illustrated in Figure 4.5b. Each graph transform layer has independent (trainable) weights, as described in Equation (4.17).
- GTN A_J: The GTN model described above, but with the A_J graph structure.

- GAT A_s: The same general multi-task graph neural network architecture used in the GTN models and depicted in Figure 4.5a. However, we replace the graph network component (in purple) with GAT [190], as detailed in Section 4.3.5. More specifically, the i^{th} row of the output matrix V_t' (as depicted in Figure 4.5a) is computed as in Equation (4.13). To make the comparisons fair, we also use $H = 3$ heads, stack $L = 2$ *graph attention layers*, and independent (trainable) weights.

- GAT A_J: The GAT model described above, but with the A_J graph structure.

- LSTM: We remove any component that leverages the graph information. Thus the input consists of the price history only. More specifically, we consider $V_t' = V_t$ in Figure 4.5a. That is remove: (1) the graph component (in purple), and (2) the adjacent matrix A_t input, at the bottom of the architecture.

- BAH (gap risk): As a baseline, we consider the Buy and Hold (BAH) trading strategy that *buys* a given stock and *holds* it during the test period. Note that there are no short selling trades. In particular, for each day t the BAH return is denoted by $y_t^{close}/y_{t-1}^{close} - 1$.

- BAH (no gap risk): As above, there are no short trades for this baseline. However, it considers buying a given stock at opening price and selling it at closing price, implying that strategy returns are denoted by $y_t^{close}/y_t^{open} - 1$. We propose this strategy because, as in the range strategy, trades are initiated and terminated during the same trading session and as such are not subjected to the gap risk (discussed in Section 4.4.5).

- S&P500 (gap risk): We compute the returns as in the BAH (gap risk) strategy, but use Dow Jones S&P 500 Index® prices. Here, the motivation is to extend the analysis beyond our universe of 388 stocks, by employing an index the gauges the 500 largest stocks by market capitalisation.

- S&P 500 (no gap risk): We compute the returns as in the BAH (no gap risk) strategy, but use S&P 500 Index prices.

The five models above (i.e. GTN A_s, GTN A_J, GAT A_s, GAT A_J, and LSTM) are optimised independently using the setting discussed in Section 4.5.1. As detailed in Section 4.4.6, the performance of each model is evaluated considering its corresponding self-financing portfolio.

4.5.5 Metrics

After training, we evaluate the models and baselines using the following metrics (as defined in Table 4.1): Sharpe ratio, MAR ratio (both *risk-adjusted* metrics), CAGR, Mean, Std. Dev., Min and Accuracy. Below we describe how we report these metrics.

Metric	Formula	Explanation
% Mean	$\mu(S) = \frac{1}{N}\sum_{t=1}^{N} r_t$	Average of the (percentage) return samples $S = \{r_1, r_2, \cdots, r_N\}$ (one sample per day)
% SD	$\sigma(S) = \sqrt{\frac{\sum_{i=1}^{N}(r_t-\mu)^2}{N-1}}$	Standard deviation of N (percentage) daily return samples.
% Min	$\text{MIN}(S)$	Worst (or minimum) daily (percentage) return, where $S = \{r_1, r_2, \cdots, r_N\}$.
CAGR	$\left(\prod_{t=1}^{N}(1+r_t)\right)^{(252/N)}-1$	Compound Annual Growth Rate (CAGR) converts the performance to yearly.
Sharpe ratio[†] [163]	$\left[\frac{\mu(S)-r_f}{\sigma(S)}\right] \cdot \sqrt{252}$	Risk-adjusted metric. The denominator measures risk and the numerator return (profitability). We discount the return by the risk free r_f to account for the opportunity cost of not holding a risk free government bond.
MAR ratio	$\frac{\text{CAGR}}{\text{MDD}}$	Managed Account Reports (MAR) ratio. The risk is quantified by the Maximum Drawdown (MDD) defined as *the worst loss in successive declines* from peaks to troughs and denoted by MDD $= -\min_{\tau\in(0;N)}(\min_{t'\in(0;\tau)} R(t',\tau))$, where $R(t',\tau)$ is the compound return between t' and τ.
Accuracy	$\frac{\text{No. of profitable trades}}{N}$	The numerator is equal to number of days market went up and the strategy went long (TP) + number of days market went down and the strategy went short (TN).

Table 4.1: **Evaluation metrics definition.** N is the number of samples. [†] we use the US 1-month T-Bill return (debt) for the risk free rate r_f, as indicated in Fama-French data library.

4.5.6 Global Self-financing Portfolio Results

Table 4.2 shows the experimental results for the self-financing portfolio of the five (5) models (top) and the (4) baseline portfolios. We have the following findings:

- The range trading stratgey is profitable. Its annual performance of 33.71% (mean GACR of all models) is double the S&P 500 index (16.77%). The Sharpe ratio is an impressive 1.82 points higher than the S&P500 index, and similar results apply to the BAH portfolio, which is a long only, consisting of the same 388 stocks under evaluation for all models.
- The novel Graph Neural Network (GTN) with sectoral graph information (GTN A_s) presents the best performance in terms of risk-adjusted performance metrics (Sharpe and MAR) with significant performance gains against the recently proposed GAT (approx. 0.30 points for both Sharpe and MAR ratios).

It is worth noting that GAT [190] is a state-of-the-art model for a broad range of graph problems.

- Better accuracy does not lead to better trading performance. We can see that the less profitable model (LSTM) has a Sharpe ratio 0.45 points worse than the best (GTN A_s), but its accuracy is on a par (approx. 63.82 versus 64.21), and similar results hold for the MAR ratio. However, as discussed above, the *utility of a trading strategy is better represented by risk-adjusted profits*, rather than accuracy.
- Graph structure largely improves profitability. This is shown by contrasting the LSTM (no added relational bias) against all other graph models. This holds true even taking into consideration the A_J graph, where all stocks are connected to all stocks.
- Apparently, the GAT architecture is less sensitive to the relational bias in terms of the Sharpe ratio (the difference is 0.04), but this is compensated when comparing the MAR ratio, where we have a gap in performance of 0.24. This outcome indicates the importance of using the appropriate bias, here by leveraging the connections between companies in the same sector, and confirms the sensitivity to the relational bias highlighted in [18].
- Similar to GAT, GTN performance degrades for the A_J graph though by a larger margin. Specifically, -0.41 (Sharpe) and -0.21 (MAR). We attribute this to the fact that GTN is not able to attend to the relevant features in the case where all stocks could be related to all stocks, i.e. the A_J graph. On the other hand, the simpler GAT attention mechanism is more effective in this environment. However, trained with an appropriate bias, we showed that the added complexity of GTN pays off. That said, we hypothesise that the GTN performance would improve by increasing the sample size (number of stocks or training years), i.e. more samples provide additional evidence to drop the weights of irrelevant stocks, i.e. those that do not change the target labels. This is equivalent to what happens when the sectoral graph is introduced.
- The test period was marked by market optimism. The S&P 500 index and the BAH portfolios (at the bottom of the table) both rose by approx. 15% per year (CAGR). In addition, the market rewards investors with an appetite for taking the gap risk – both ratios are higher for the series that includes the gap than those that do not.

Model Comparison – Additional Analysis

In order to further investigate the architectures evaluated in this work, we use the *model comparison* protocol suggested in [164, 154] and described in the following steps: (1) A specific metric is chosen to evaluate all k models, (2) This metric is

	Sharpe ratio	MAR ratio	CAGR	Mean	SD	Min	MDD	Accuracy
Models								
GTN A_s	**3.110**	**2.710**	**37.406**	**0.128**	0.649	-3.967	-13.804	**64.215**
GAT [190] A_s	2.815	2.400	33.258	0.116	0.638	-4.306	-13.858	63.221
GAT [190] A_J	2.779	2.644	32.763	0.115	0.652	-4.070	-12.392	63.419
GTN A_J	2.697	2.598	31.638	0.111	0.647	**-3.321**	**-12.176**	62.425
LSTM †	2.687	2.158	31.490	0.111	0.657	-3.583	-14.594	63.817
Baseline portfolios								
BAH†† (gap risk)	1.626	1.967	17.774	0.067	0.644	-3.576	-9.038	57.058
S&P500* (gap risk)	1.545	1.850	16.786	0.064	0.655	-3.643	-9.074	55.666
BAH (no gap risk)	1.540	1.801	16.725	0.064	0.649	-3.597	-9.286	55.268
S&P500 (no gap risk)	1.395	1.692	14.984	0.058	0.657	-3.903	-8.858	55.467

Table 4.2: **Portfolio daily return statistics**. † LSTM model do not consider any graph structure and all other models are graph neural networks.

computed for all models on a set n related problems, also called trials (3) A statistical test is applied to compute the mean rank of each model (4) Finally, a post-hoc null hypothesis test is conducted to confirm whether the difference among each model mean rank is statistically significant.

We now describe how the protocol above was applied to our work and show that the results corroborate the superior performance of GTN architecture. In terms of implementation, all statistical tests use the STAC platform described in [154].

For the metric of choice in step (1), we use the Sharpe ratio, with each stock in our portfolio as one trial, as defined in step (2). The Sharpe ratio computed independently for all $n = 388$ stocks for each of the $k = 5$ models is reported in Appendix C.3. We now proceed to step (3) and employ the non-parametric Friedman test [68] to compute the mean ranks. The result is reported in Table 4.3, where all mean ranks are significant (p<.01). In line with the aforementioned findings, it confirms that GTN A_s *is also the best model* (mean rank score of 2.698), and the difference between the top and second ranked models is larger than the subsequent differences (0.22 against an average difference of 0.12). At this stage, we proceed with step (4) and employ the null hypothesis test (H_0) proposed by Li [43]. We perform pair comparisons using a control model, here, the best model is GTN A_s. Specifically, we test the Null hypothesis (H_0): The mean rank of GTN A_s (control model) against each other group is equal (compared in pairs). The H_0 result is shown in Table 4.4 where the p-value is adjusted for multiple comparisons. We can see that the hypothesis is rejected, implying the rank differences are significant ($p < .05$). Thus, providing further evidence that the novel GTN A_s model *performs better than any of the other models under analysis.*

Model	Mean Rank
GTN A_s	**2.69845**
GAT [190] A_J	2.92268
GAT [190] A_s	2.95876
GTN A_J	3.13660
LSTM†	3.28351

Table 4.3: **Model Comparison – Ranking according to Friedman test**. The mean rank value (right) for each model (left). All results are significant (p $<$ .01) and ordered by rank position (top best to worst bottom). † LSTM model do not consider any graph structure and all other models are graph neural networks.

Comparison	Result	Statistic	Adjusted p-value
GTN A_s versus LSTM	H_0 is rejected	5.15377	.00000
GTN A_s versus GTN A_J	H_0 is rejected	3.85965	.00012
GTN A_s versus GAT [190] A_s	H_0 is rejected	2.29309	.02244
GTN A_s versus GAT [190] A_J	H_0 is rejected	1.97523	.04824

Table 4.4: **Model Comparison – Post-hoc control method.** The best model (GTN A_s), according to Friedman ranking in Table 4.3, is set as control model and compared against all other models. We use the method proposed in [43] that test the Null hypothesis (H_0): The mean rank of GTN A_s against each other groups *is equal* (compared in pairs).

Performance Attribution Analysis

In the previous section, we show that the GTN architecture with sectoral graph provides the best risk-adjusted performance. Figure 4.6 shows its cumulative performance compared with the baselines, i.e. the long-only portfolios (BAH) and market index (S&P 500). We can see that one dollar invested in the best model (blue line) turns into 1.85 dollars, while only approx. 1.25 dollars if invested in all other baselines (remaining lines). Moreover, qualitatively, it seems that the model performance is not correlated with the S&P 500 index. For example, from Jan/2016 to April/2016 the S&P 500 index gains were almost flat, compared with a relevant 20% gain (from 1 to 1.20 dollars) of the best model. Conversely, from Nov/2016 to Jan/2017 the index remains constant, i.e. with no gains, and the model suffers its worst consecutive decline (losing approximately 13%).

In order to better understand whether a given model performance can be attributed to market movements, we employ the FF3 factors regression [62]. The results can be interpreted in the following way: the true gains of a portfolio are measured in terms of (regression intercept) α, which is the returns (or gains) that cannot be explained in terms of exposure to common risk factors, as measured by the three

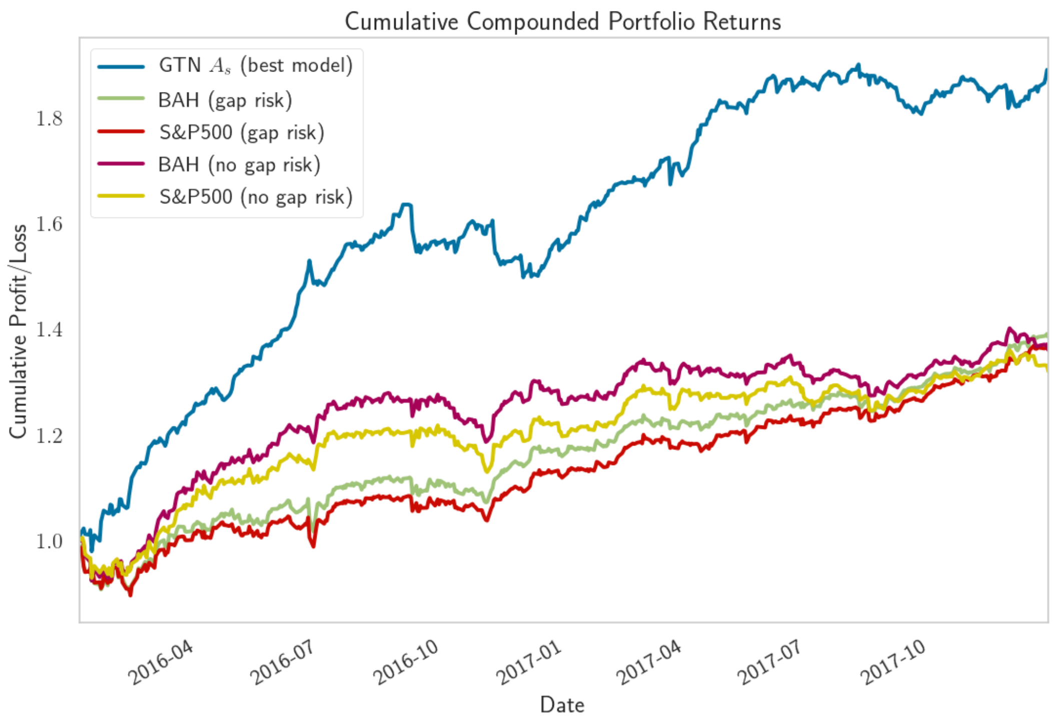


Figure 4.6: **Equity Curve**. The portfolio cumulative compounded return for each model under analysis and baseline portfolios.

regression coefficients (β). Table 4.5 shows the results of the FF3 factors regression using daily portfolio returns, ordered by α. As expected, all baseline portfolios αs (at the bottom) are not statistically different from zero and the high R^2 value shows that its gains can be largely explained by the three common risk factors. In particular by the high beta exposure to the market factor R_M. On the other hand, all models (at the top) have significant α (p $<$ .01) and its low R^2 values evidence that the model's returns can be specified in terms of a constant, i.e. do not depend on three β factors. Importantly, *in terms of alpha, the two best results are achieved by the GTN architecture*, using the sectoral (A_s) and fully connected (A_J) graphs, respectively. Moreover, in terms of compounded alpha returns, *the GTN yearly gains (36.585%) are 4.27% percentage points larger than GAT (32.3220%).*

Overall, the superior performance of GTN architecture is also confirmed by its alpha (return) value and its performance cannot be attributed to common risk factors, including the market portfolio (R_M), which captures the aggregate performance of the stock market as a whole.

4.5.7 Sectorial Self-financing Portfolio Results

In the previous sections, we compared the range trading strategy performance across five models. Moreover, the aggregated results were represented by the performance self-financing portfolio containing all 388 stocks. Here, we conduct a fine-grained analysis across sectors for the best performing model, namely GTN A_s. To this end,

	α	$\beta_{R_M-r_f}$	β_{SMB}	β_{HML}	R^2
Models					
GTN A_s	**0.1238** (.000)	**0.0870** (.050)	0.1425 (.020)	**-0.2437** (.000)	0.064
GTN A_J	0.1116 (.000)	0.0175 (.697)	0.1096 (.075)	-0.2510 (.000)	**0.051**
GAT A_s	0.1112 (.000)	0.0988 (.027)	0.1117 (.067)	-0.2617 (.000)	0.066
GAT A_J	0.1098 (.000)	0.0985 (.027)	**0.1241** (.042)	-0.2668 (.000)	0.071
LSTM	0.1090 (.000)	0.0555 (.214)	0.1041 (.089)	-0.2751 (.000)	0.062
Baseline portfolios					
BAH† (gap risk)	0.0032 (.590)	0.9183 (.000)	0.0147 (.250)	0.0787 (.000)	0.959
S&P500* (gap risk)	-0.0019 (.349)	0.9798 (.000)	-0.1287 (.000)	-0.0211 (.049)	0.994
BAH (no gap risk)	0.0176 (.349)	0.6680 (.000)	0.2012 (.000)	-0.0689 (.049)	0.591
S&P500 (no gap risk)	0.0110 (.559)	0.7155 (.000)	0.0606 (.133)	-0.1815 (.000)	0.593

Table 4.5: **Portfolio exposure to common risk factors**. Regression results of FF3 factors model [62, 63] for all 5 models under evaluation (top) and baseline (long-only) portfolios or market indices (bottom). We have the regression intercept α (first column) followed by 3 regression coefficients β's and Coefficient of determination R^2 in the last column. The regression p-values are in parentheses. Regression intercept α is the return after controlling for underlying portfolio risk factors, as measured by β's. The three risk factors are: 1) Excess Market risk ($R_m - R_f$, where R_f is the risk free rate) 2) Small (market capitalisation) minus Big (SMB) and 3) High (book-to-market ratio) minus Low (HML).

we first separate the stocks of a given sector. Then, for each sector we compute its corresponding self-financing portfolio and dynamically size its positions.

It is worth noting that we do not re-train the model for each sector, but all portfolios target the same level of volatility, which as in the global portfolio, is set to σtarget $= 10\%$.

Table 4.6 shows the portfolio performance in descending order by Sharpe ratio and as mentioned for the best model GTN A_s.

We can see that the top performers are Financial Services followed by Industrials, and the worst performers are Basic Materials and Communication Services. Although there is a reasonable variability of performance between sectors, the key driver is difficult to interpret. For example, on the one hand, Utilities and Communication services sectors have a similar performance (8.22% and 7.10%, respectively) and this is justified in terms of common economic drivers, i.e. both sectors are under the infrastructure umbrella, where dividends are less sensitive to economic growth and stocks less volatile. On the other hand, the Energy sector (CAGR 17.7%) has more than double the annual performance of Basic materials (CAGR 7.6%). However, both sectors are driven by commodity prices and global demand, and as such, expected to have a comparable performance. Rather than economic drivers, other research [77, 193] suggests that range strategy performance is driven by volatility. Quoting von Mettenheim and Breitner [193]: *"The core of our trading system is trading the high-low intraday range. For this to work, we need some amount of intraday volatility. We expect that volatile stocks will yield better results than comparatively "duller" stocks."*. In the same way, [77] proposes changes in the entry level of the strategy based on stock volatility. However, the research lacks a more detailed analysis to confirm the assumption that higher volatility is correlated with better performance.

Sectors	Sharpe ratio	MAR ratio	CAGR	% Mean	% SD	Min	Acc.	Number of stocks (N_s)
Financial Services	3.130	3.837	37.691	0.129	0.652	-4.048	67.197	52
Industrials	2.336	1.404	26.792	0.096	0.649	-4.059	64.215	56
Real Estate	2.317	2.326	26.542	0.096	0.653	-2.774	64.016	23
Technology	2.230	2.442	25.404	0.092	0.640	-3.245	61.034	47
Healthcare	2.160	1.451	24.491	0.089	0.642	-3.987	60.835	50
Consumer Cyclical	1.868	1.350	20.774	0.077	0.652	-3.360	58.449	55
Energy	1.623	2.680	17.734	0.067	0.661	-2.681	59.245	24
Consumer Defensive	1.298	0.972	13.828	0.054	0.641	-3.664	59.443	31
Utilities	0.812	0.537	8.221	0.034	0.639	-3.459	57.853	26
Basic Materials	0.755	0.415	7.580	0.031	0.649	-2.806	58.847	17
Communication Services	0.710	0.316	7.077	0.029	0.651	-2.748	55.467	7
All sectors	3.110	2.710	37.406	0.128	0.649	-3.967	64.215	388

Table 4.6: **Portfolio daily return statistics per sector**.

Figure 4.7 shows the Sharpe ratio (y-axis) against different levels of annualised volatility (x-axis) for each sector sample. Note we compute the *volatility inherent in each sector*, measured using a portfolio that buys and holds (BAH) the constituent stocks of a given sector during the test period. We can see that volatility mostly varies in the range 10% to 18%, with the Energy sector as an outlier (24% ann. volatility). Clearly, the Sharpe ratio is evenly spread around the mean value (1.78). Indeed, the very low regression coefficient of determination ($R^2 = 0.008$) rejects the hypothesis that performance variability can be explained in terms of volatility level, as suggested by previous research.

4.5.8 Attention Weights – Preamble

As described in Section 4.4.2, the attention weights are represented by an $n \times n$ matrix, where $n = 388$ is the number of stocks (or nodes) and each element $\mathcal{A}_{ij} \in [0, 1]$ measures the influence (or weight) of node j on node i. Moreover, for all nodes

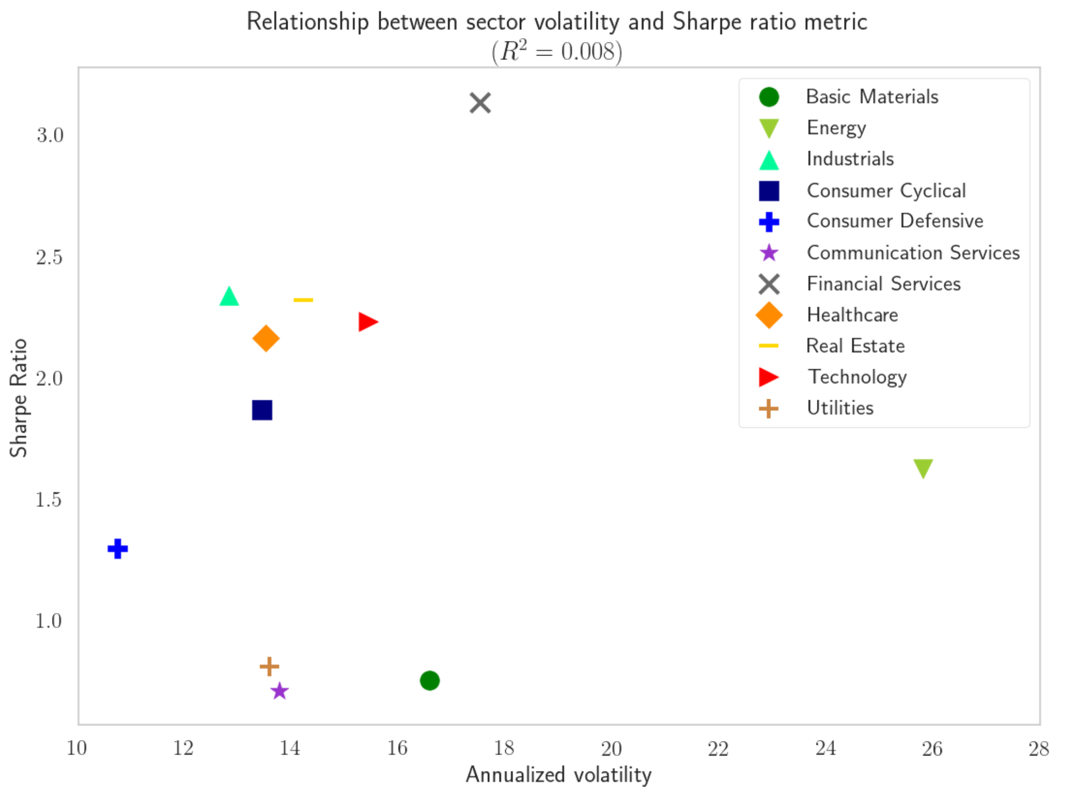


Figure 4.7: **Volatility elasticity of Sharpe ratio**.

$i = \{1, \cdots, 388\}$ the neighbour node weights (js) plus the self-connection (or self-loop) weight $\mathcal{A}_{ii}$ sum to one, i.e. they are normalised for each node.

Weights extraction

In this section we describe how we extract the attention weights of the GTN architecture. After the model is trained, each h-head attention matrix A_h can be directly computed from its queries Q_h, keys K_h, and the adjacent matrix A by using Equation (4.17e), i.e. $\mathcal{A}_h = g(A, Q_h, K_h)$. However, each head key and query depend in turn on graph attributes V_t and binary relationships A^t. Thus, we have one attention head for each day t, denoted $\mathcal{A}_h^t$. Finally, we observe that V_t varies over time because it is computed in terms of the stock history tensor $\mathcal{X}_t$ (Equation (4.21)).

Since the GTN model is trained with $H = 3$ heads $L = 2$-hops, we compute $\mathcal{A}_h^t$ above at the second hop or layer. This is done by considering $V_t = V'^1_t$ (illustrated in Figure 4.5b). Moreover, we aggregate all attention heads into a single attention weight matrix by using mean-pooling, denoted by $\mathcal{A}^t = \text{MEAN}\left(\{\mathcal{A}_1^t, \mathcal{A}_2^t, \mathcal{A}_3^t\}\right)$.

Experimental setting

Henceforth, we use the GTN A_s model for analysis and extract its sequence of attention weight matrices $(\mathcal{A}^1, \mathcal{A}^2, \cdots, \mathcal{A}^N)$, where $N = 503$ is the number of days in the test set and each matrix in the sequence is an aggregation of the $H = 3$ attention heads. As the sectoral adjacency matrix A_s is composed of 11 subgraphs (one per sector), any connection between stocks of different sectors has zero weight. Thus, sector results are computed using each subgraph independently.

4.5.9 Attention Weights – Graph Mapping Interpretation and Visualisation

The GTN attention can be seen as a graph edges mapping $f_{\mathcal{E}} : \{0,1\}^{n\times n} \to [0,1]^{n\times n}$ from an *undirected unweighted graph* with binary relations represented by A^t to a *weighted directed graph* with learnt edge weights represented by $\mathcal{A}^t$. In particular, *the edges mapping preserves the same graph structure*, i.e. the attention weights are learnt by attending the present relations ($A^t_{ij} = 1$) and dropping the absent ones ($A^t_{ij} = 0$), implying[18] that $\mathcal{A}^t_{ij} = 0 \iff A^t_{ij} = 0$.

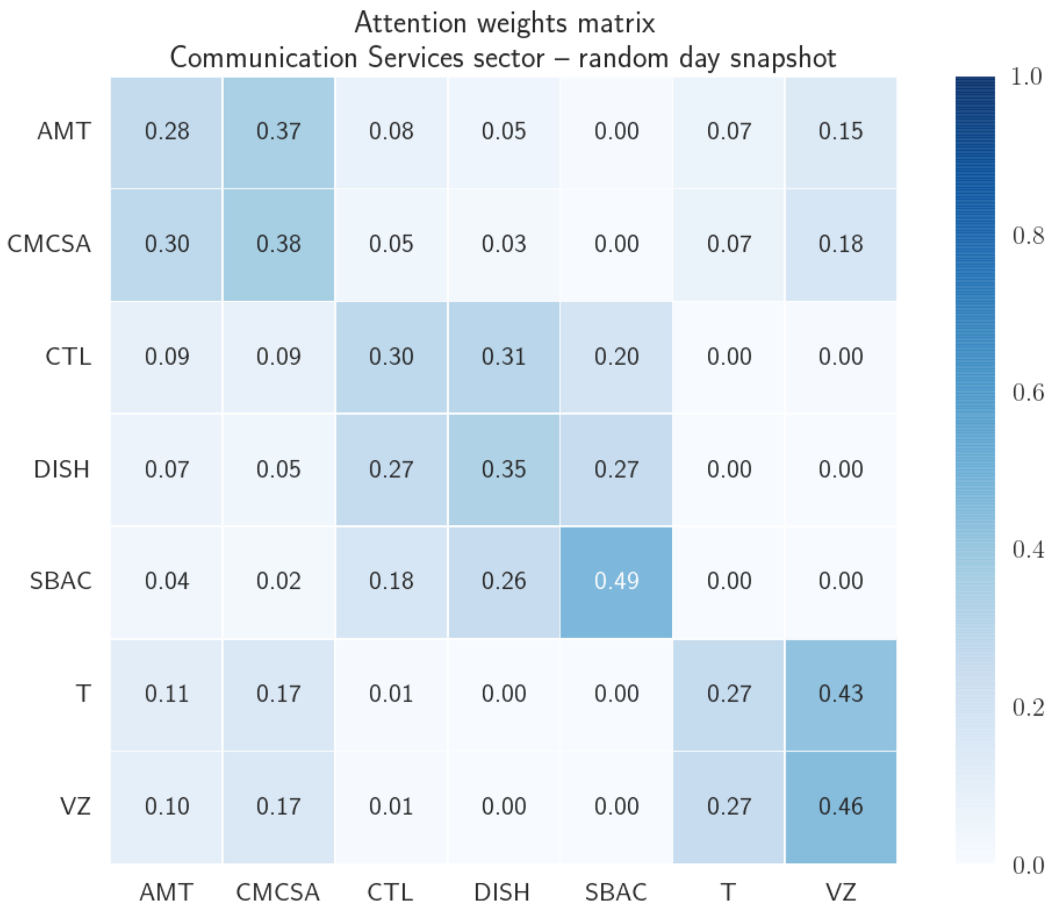
As the weights are extracted from a model that predicts the *future price* and the attention matrix $\mathcal{A}_t$ acts on *past price histories*, represented by V_t, the self-connection is associated with an auto-regressive coefficient and the neighbours with cross-regressive coefficients. That is, for each stock the self-connection attention weights of the diagonal measure the dependence on its own lagged values (auto-regressive), and the neighbour node weights, i.e. off-diagonal attention weights, the dependence on the lagged values of each other stock (cross-regressive).

In order to better understand the properties discussed above, Figure 4.8 shows the attention weight matrix for a random test set day and the Communication Services subgraph with seven stocks. As mentioned above, initially, the relational bias was injected into the GTN architecture by considering all seven nodes potentially interacting with each other in the sector subgraph, i.e. edge value one and zero for all other stocks in the A_s (super)graph. After training, the (attention weight) matrix $\mathcal{A}$ can be interpreted as the adjacency matrix of a weighted directed graph with the same seven nodes or stocks. We can see that *the influences (or weights) depend on the direction and as such are not symmetrical*, i.e. $\mathcal{A}_{ij} \neq \mathcal{A}_{ji}$. For example, to predict AT&T (T) more weight is put on Verizon's (VZ) past price history (0.43), than on its own past price history (0.27). Conversely, for the Verizon prediction almost half of the weight is allocated to its own past history (0.46), followed by 0.27 for AT&T (T). Additionally, for both AT&T and Verizon almost no weight is allocated to CenturyLink (CTL), Dish Network (DISH) and SBA Communications Corporation (SBAC) – meaning that these histories are irrelevant for the AT&T and Verizon predictions.

4.5.10 Attention Weights – Graph Self-connection and Neighbours

In Section 4.4.2 we introduced the *graph role embedding* and discuss its importance in discriminating between the self-loop connection and its neighbours. The self-(connection) weights are the diagonal elements of the A_s matrix and measure the influence of node past price history on itself.

[18] We observe that the presence or absence of relations is preserved because of the Hadamard product $\odot A^t$ in Equation (4.17e), which is effectively introduced in the novel GTN architecture.



	AMT	CMCSA	CTL	DISH	SBAC	T	VZ
AMT	0.28	0.37	0.08	0.05	0.00	0.07	0.15
CMCSA	0.30	0.38	0.05	0.03	0.00	0.07	0.18
CTL	0.09	0.09	0.30	0.31	0.20	0.00	0.00
DISH	0.07	0.05	0.27	0.35	0.27	0.00	0.00
SBAC	0.04	0.02	0.18	0.26	0.49	0.00	0.00
T	0.11	0.17	0.01	0.00	0.00	0.27	0.43
VZ	0.10	0.17	0.01	0.00	0.00	0.27	0.46

Figure 4.8: **Attention weight matrix snapshot**.

Note that if the weights to neighbour nodes are large compared to the self weight the stock is highly impacted by any shock coming from its neighbour nodes. Conversely, if the neighbour weights are negligible the stock dynamics is almost entirely described by its own past history. We conduct a cross-sectional analysis to highlight the difference between the self and neighbours weights across sectors. Below, we describe how we separate both effects and report our results.

For each sector subgraph, as described in Section 4.5.8, we separate the diagonal (sf) and off-diagonal (ns) matrix elements of all $N = 503$ days of the test set into the self-(connection) and neighbour weight samples set, denoted w_{sf} and w_{ns}, respectively. Note the number of weight samples in each set depends on the number of stocks in each subgraph n_s. More precisely, $|w_{\text{sf}}| = N \cdot n_s$ (diagonal) and $|w_{\text{ns}}| = N \cdot (n_s^2 - n_s)$ (off-diagonal). Finally, in order to make the self and neighbour statistics comparable, we downsample the neighbours set (ns) to the same size as the self (sf).

Figure 4.9 shows the weights histogram for the self (blue) and neighbours (green) weights set of each sector subgraph, with all sectors shown bottom-right. All weights are standardised (z-score), where the mean and std dev. are computed per sector and on the union of the self and neighbours sets. As shown in the bottom y-axis, the vertical dashed line represents zero standard deviations (SD) and all plots range from -2 SD to +2 SD. Additionally, histogram bins sum to one (probability). We start by analysing the aggregate "All sectors" weight distribution. We see that the concentration of small weights, i.e. with negative values below the mean, is higher for neighbours than self. That is, the neighbours' distribution peaks around small weight values. In addition, because the area under each distribution sums to one, this implies that large weight values have to be more frequent for the self-(connections). Overall, this indicates that on average the stock predictions are more driven by their own (self) past price (auto-regressive) than the past price of other (neighbour) stocks (cross-regressive). The same peak around small weights for the cross-regressive contribution can be seen for other sectors, but in different degrees. One exception is the Utilities sector, where the distinction between the peaks is less pronounced.

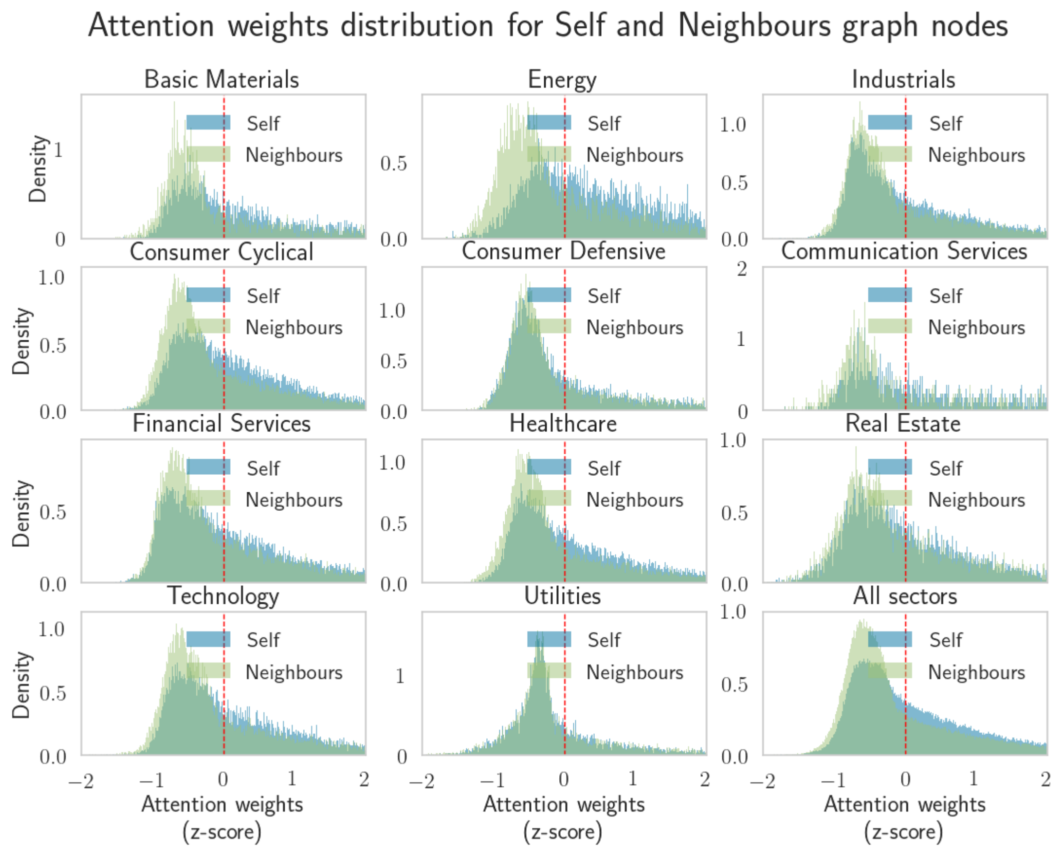


Figure 4.9: **Attention weights distribution**.

Table 4.7 shows the statistics for the distributions discussed above. For both self and neighbours weight sets, we can see that across sectors the Skewness (γ) is positive and the very large Kurtosis (κ) shows that extreme weights are higher than would be found in a Gaussian distribution, where γ and κ are equal to zero. These excess weights indicate that some stocks have a very large allocation for their own self-connection, and this characteristic can be seen in all sectors. Since each sector has a different number of stocks (as detailed in Table 4.6) the mean weight values of self and neighbours are not directly comparable across sectors. For example, if the allocation were equally-weighted we would have $1/17 = 0.059$ and $1/47 = 0.021$ for the Basic Material and Technology sectors, respectively. However, we correct this sample size effect by considering the ratio self / neighbours (shown in the last column of the table). Note that if all weights were evenly balanced across stocks the ratio would be one. The fact that the ratio is approximately 2, when all sectors are taken into account, and that it has a minimum of 1.51 across sectors, provides evidence that there is a clear distinction between the self-connection and neighbours weights. In this sense, the graph role embedding proposed in Section 4.4.2 for the GTN architecture helps to discriminate both contributions during the training phase.

Sectors	Self-(connection)				Neighbours				Ratio
	μ_{sf}	σ_{sf}	γ_{sf}	κ_{sf}	μ_{ns}	σ_{ns}	γ_{sf}	κ_{sf}	$\mu_{\text{sf}}/\mu_{\text{ns}}$
Healthcare	0.043	0.077	5.581	44.580	0.020	0.031	4.402	40.333	2.182
Basic Materials	0.121	0.156	2.762	9.327	0.056	0.084	2.722	11.510	2.164
Technology	0.042	0.078	6.126	50.188	0.021	0.031	5.315	66.610	2.009
Communication Services	0.237	0.245	1.449	1.327	0.125	0.158	1.727	3.182	1.905
Energy	0.076	0.076	3.364	20.931	0.041	0.054	3.330	21.403	1.878
Consumer Cyclical	0.033	0.050	6.299	69.756	0.018	0.025	4.116	44.187	1.868
Industrials	0.032	0.064	6.756	63.369	0.018	0.026	4.289	39.410	1.815
Utilities	0.059	0.122	5.016	27.661	0.038	0.051	4.630	31.976	1.568
Financial Services	0.029	0.045	6.375	68.582	0.019	0.026	3.993	37.898	1.546
Real Estate	0.066	0.097	4.632	28.379	0.043	0.049	3.656	32.482	1.528
Consumer Defensive	0.049	0.083	4.472	27.693	0.032	0.046	3.382	22.587	1.518
All sectors	0.050	0.092	5.285	36.701	0.027	0.047	5.390	49.317	1.814

Table 4.7: **Attention weights statistics – Self and Neighbour nodes breakdown**. The statistics of the histograms of standardised weights (z-score) shown in Figure 4.9. We compute the mean (μ), standard deviation (σ), skewness (γ) and kurtosis (κ) independently for the self-connection and neighbour nodes (Neighbours column) sample sets. The self-connection weight samples are the diagonal elements of the attention weight matrix $\mathscr{A}$ and measure the influence of the own (self) stock past history (auto-regressive). The Neighbour (columns 5 to 8) connections are the off-diagonal weights, and measure the influence between pairs of stocks (cross-regressive).

Moreover, the ratio measures the *self-connection relative strength* compared to neighbour nodes. Because the neighbours' weights contribute to the cross-regression terms, the larger the relative strength, the less affected the sector stocks are by each other's stock shocks. In the same way, the lower the index, the more integrated are the sector stocks. Based on the (self-connection) relative strength values we see that Consumer Defensive, Real Estate and Financial Services are more integrated and prone to shocks. On the other hands, stocks in Healthcare, Basic Materials and Technology have dynamics which are more regulated by their past auto-regressive terms, compared to other stocks cross-regressive terms. This result follows the intuition of the *systemic risk* in the Real Estate and Financial Services sectors, with a high level of debt. It also shows that stocks in Technology (e.g. Microsoft or Apple) and Heathcare (e.g. Pfizer and Abbott laboratories) have price dynamics more driven by idiosyncratic factors, like the products they are selling (software, medical equipment or drugs discovered).

4.6 Conclusion

In this chapter, we propose the novel Graph Transformer Network (GTN) which learns to attend to the features attributed to each node, taking into account the local graph structure, i.e. its relationships or edges. Similar to the Graph Attention Network (GAT) [190], which is state-of-the-art for graph problems, our architecture is able to differentiate between the contributions of its own node features (self-connection) and its neighbour nodes. However, we discriminate using the graph role embedding, introduced in this work. This embedding allows our attention mechanism to depend only on sparse matrix computations, which are highly efficient and optimised. In addition, the compact matrix formulation of GTN computes the attention weights for all graph nodes simultaneously, via matrix multiplication. This differs from GAT and all other work recently proposed in the literature (e.g. GraphSAGE [81]), where the pairwise attention weights are computed node by node

We evaluate different graph network architectures to the problem of predicting the price range of stocks. Moreover, we formulate this problem using multi-task learning, where both prices, i.e. High and Low, are predicted using two task-specific layers (one for the High price and another for the Low price). We then feed these predictions to the range trading strategy, recently proposed in the literature [127]. To the best of our knowledge, we are the first to use graph networks to tackle the problem of range prediction and its trading strategy profitability. Additionally, we conducted the broadest analysis of this problem considering 388 stocks from the widely traded NYSE stock market, where previous research considers only five stocks [193] or two stocks traded on the Brazilian stock exchange [127]. Our detailed sectoral analysis refutes the assumption of previous work [193, 77] that the strategy profitability depends on the volatility level. More specifically, we found a very low coefficient of determination $R^2 = 0.008$ for this relationship, which is not statistically different from zero ($p < 0.01$).

In our experiments, we consider two relational biases represented by A_s (sectoral graph) and A_J (fully connected graph). Overall, in line with previous research, our results confirm the high profitability of the range trading strategy. Moreover, using ablation methods, our results demonstrate the effectiveness of injecting the relational bias into the learning process. More specifically, we found that the performance of all graph network architectures is better than that of LSTM (without graph structure). In the context of financial forecasting using graph neural networks, experimental results show the superior performance of the novel GTN, beating by a large margin the state-of-the-art GAT for multiple metrics. The GTN yearly returns are 4.3 percentage points larger than GAT. For the risk-adjusted Sharpe ratio and MAR ratio metrics performance gains are also expressive: 3.11 (GTN) against 2.81 (GAT) and 2.71 (GTN) against 2.40 (GAT). Using a robust model comparison method [43, 68], with p-values adjusted for multiple groups and trials, our experimental results rejects the hypothesis that GTN's superior performance is a statistical artefact. The top ranking of GTN is significant for any pair model comparison (p < 0.05).

Finally, a more detailed analysis of the attention weight matrix, which is an integral part of the end-to-end GTN architecture, contributes to the topic of model interpretation in machine learning. We show that the weights (edges) can be formulated in general as a directed graph, where the influence of stock j on i, as measured by A_{ij} element, is not equal to the influence of i on j (A_{ji}) – a fact also seen in the financial markets.

We also leverage the attention weights to quantify the sensitivity of a given stock to its peers' stocks, and given its practical implication for financial markets, we call this measure the Self-connection Relative Strength index. The higher the index, the less integrated the stock is in the graph, and, as such, less prone to price shocks coming from other stocks. Using this index we show that Real Estate and Financial stocks are more prone to systemic risks; they can trigger the collapse of their entire sector or even the economy.

Chapter 5

Pairs Trading with Deep Reinforcement Learning Agents

5.1 Introduction

Pair trading is a quantitative trading strategy popular among institutional investors and hedge funds [71, 83]. In essence, the strategy makes use of statistical methods to find a pair of stocks whose prices tend to move together, i.e. that have a long-term statistical relationship. Thus, if the relative price between the two stocks diverges in the short-term, we enter a *pair trade*, by buying the underpriced stock and selling-short the overpriced stock. This trade is then exited, i.e. reverted,[1] when prices return to their long-term equilibrium.

The pair trading strategy can be summarised in two independent steps. First, the *pair screening* step defines the pair formation method, where stocks with long-term equilibrium are grouped in pairs. Then, based on how far a pair is from its equilibrium, the *decision-making* step is a criterion for triggering pair trading. More specifically, when to open and close the positions and which stocks go long or short.

In a seminal study, Gatev et al. [71] proposes the Distance Method (DM) for the pair screening stage and a rule-based system for the decision-making step. More specifically, the distance $d(s_1, s_2)$ between stock s_1 and s_2 is computed using their normalised historical prices, where, at the pair screening step, only stocks s_1 and s_2 with minimal distance are grouped into pairs. Their rule-based system has three possible pair positions, namely $\{\text{long} = 1, \text{neutral} = 0, \text{short} = -1\}$, as follows (1) long pair position: opened by buying stock s_1 and selling short stock s_2, (2) short pair position: opened by selling short s_1 and buying s_2, and (3) neutral pair position: Do nothing. Thus, positions are opened based on a pre-specified triggering value with rules defined as follows: if the z-score of the distance is above (below) the triggering value of plus (minus) two standard deviations we have a long (short) pair position, otherwise we have a neutral position (i.e. no position is opened). Both long and short pair positions are then closed when the z-score is zero, in other words, when the distance converges to its historical long-term mean value.

[1]In order to close the opened positions we *cover* the short position by buying the stocks that were sold short, which are then returned to the stock lender, and selling the stocks that were bought in the long position.)

The pair trading strategy has gained widespread attention among researches with many studies proposing different methods for the pairs screening and decision-making steps ([50, 24, 51, 69, 31, 114, 33, 95, 99, 59, 199]). In particular, some authors make use of more sophisticated statistical methods at the pairs screening step, compared to DM [71] described above, employing the cointegration method [57, 79] or more recently the copulas method [101]. However, without exception, all these studies make use of a rule-based system at the decision-making step.

In our study, we approach the pair trading strategy from a different angle. We propose model-free Deep RL [173], where the actions a_i^t represent the pair positions on a given day t for a given pair i. Thus, rather than using a rule-based system that defines when to open and close the positions and which stock should go long or short, we propose an end-to-end model that maps pair history (here, observations) to position (actions).

In line with previous studies, we consider discrete actions $a_i^t \in \{\text{long} = 1, \text{neutral} = 0, \text{short} = -1\}$, but, also extend to continuous actions $a_i^t \in [-1, 1]$. In particular, for discrete actions we use the recently proposed Deep Q-Network (DQN) [131] algorithm, and for continuous actions the Deep Deterministic Policy Gradient (DDPG) [115]. Our choice is based on the fact that both algorithms learn optimal actions by interacting with the environment (i.e. they are model-free), and achieve state-of-the-art performance in playing games [131] and in control tasks such as legged locomotion and car driving [115].

5.1.1 Pair Trading: Portfolio Profitability and Risk

In this work, we evaluate the pair trading strategy for a portfolio of stocks. However, different from other studies, we also consider its risk. In this context, the profitability of the pair trading strategy is evaluated in the light of its risk.

To this end, we propose a novel framework where the *universe of stocks* in the portfolio and the investor risk tolerance are first-class elements, and treated as *investor preferences*. Importantly, in our framework the Deep RL architecture was designed in way that it is only trained once and at execution time can be consumed for portfolios of any size and structure.

Figure 5.1 provides a high-level view of our integrated framework applied to the pair trading strategy. Here, we review the framework at execution time. The input parameters are the risk tolerance σ_{target}, and the N stocks that the investor is interested in trading (i.e. the portfolio constituents). The output is the optimal vector of portfolio weights w^t, which is the allocations the investor is meant to follow for each stock on a daily basis. Thus, given N stocks, the pair screening step groups the stocks in N_p pairs. The history of all N_p pairs is then passed to a *pre-trained* Deep RL model, which outputs the actions for each pair independently. The last component adjusts the portfolio weights to conform to the volatility σ_{target}, pre-set according to investor preference. In particular, this adjustment takes into account current market conditions.

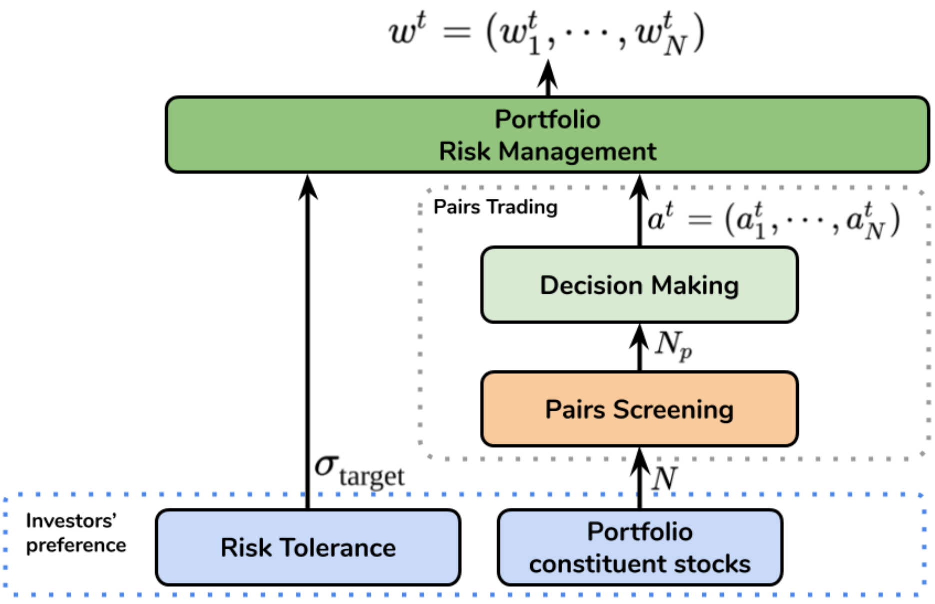

$w^t = (w_1^t, \cdots, w_N^t)$
Portfolio Risk Management
Pairs Trading
$a^t = (a_1^t, \cdots, a_N^t)$
Decision Making
N_p
Pairs Screening
σ_{target}
N
Investors' preference
Risk Tolerance
Portfolio constituent stocks

Figure 5.1: **Novel framework: Investment Strategy with Investors' preferences (ISIP)**

More recently, Deep RL frameworks have been proposed to tackle the general portfolio management problem in finance [201, 113, 100]. These studies are close to our work, in the sense that they also employ Deep RL to find optimal allocations, but differ from our framework in four relevant aspects: (1) They are restricted to portfolios with only long positions; (2) They do not consider the risk component of our framework, which is vital for portfolios with short positions; (3) They are not suitable for discrete actions because of the exponential growth of the combined action space. More specifically, this is a consequence of the centralised controller paradigm [149, 128], inherent in their architecture, where a single neural network outputs all N continuous actions (i.e. stock weights) *simultaneously*,[2] (4) Most importantly, given the centralised controller paradigm, their architecture can only operate on a *fixed* number of assets.

Thus, using their architecture means to re-train the whole model to match the preference of each investor, in regards to the number of stocks N that he/she is interested in trading. Given the intense computational time to train Deep RL models, this limits the usability of their framework.

Our research has the following main contributions:

1. Aiming for flexibility in portfolios of any size and structure, we propose a single-agent architecture, where the agent, called a pairs trader agent, is represented by a single Q-value network for discrete actions, or by a single actor-critic

[2]In other words, if we apply their framework (centralised controller) to discrete actions, the output layer has to be the size of the combined action space, which is given by $|A|^N$, where $|A|$ is the number of possible positions for each stock. Here, if we consider long, neutral or short we have, we would be talking about a network with 3^N units, which, for a very small number of stocks already becomes unfeasible to train

network for continuous actions. In our architecture, the learning remains independent or decentralised, because the pair action is only conditioned on its own individual history. However, particular to our framework, when interacting with the environment our pairs trader agent collects experiences from multiple pairs, and stores these experiences on a single *multi-pair replay buffer*. In terms of multi-agent learning, our architecture is related to the Independent Learning paradigm originally proposed by [176].

2. We propose a novel framework for portfolio optimisation using Deep RL, called Investment Strategy with Investors' preferences (ISIP). In this framework, profitability (reward) is integrated with risk. Thus, the investor defines the volatility level σ_{target}, i.e. the level of risk he/she can tolerate, and also the portfolio constituent stocks, i.e. the N stocks he/she is interested in trading. The efficacy of our framework in keeping the risk at acceptable levels is demonstrated by the *ex-post* volatility of our Deep RL portfolios. Even though the portfolio volatility is adjusted *ex-ante* to $\sigma_{\text{target}} = 10\%$ the ex-post portfolio volatility is 10.163%. In other words, we miss the 10% target by a negligible margin.

3. Apart from the fact that our Deep RL models can be deployed to manage the portfolio of investors with different preferences, we conduct a rigorous *performance attribution analysis*. This analysis was not conduced in other Deep RL studies [201, 113, 100], but is crucial to understand whether the model performance (i.e. its return) is correlated with market performance. Our experimental results show that our Deep RL models (both for continuous and discrete actions) definitely *learn a policy whose performance cannot be attributed to the market*. In other words, the performance of the Deep RL models are not correlated with market returns. In particular, after controlling for exposure to common risk factors [62, 63], the efficacy of our Deep RL model (continuous actions) has annual gains of approximately 26% for a risk level, as measured by its volatility, of 10%. The performance largely beats the best baseline, which is based on Supervised Learning (SL), and has annual gains of 14% for the same 10% risk level.

5.2 Related Work

We start this section by reviewing the previous research in pair trading strategy. As mentioned above, different from our Deep RL approach, previous studies all use a rule-based system to open and close the positions on each pair.

Do and Faff [50, 51] document a falling performance in pair trading since the seminal work of Gatev et al. [71]. However, they find that the strategy performs strongly during periods of turbulence, including the 2008 Subprime mortgage crisis. They also propose a pairs screening step based on the number of times the z-score distance crosses the zero value, where they use the same distance as in the Distance

Method (DM) proposed in [71]. Experiments evidence that their pair screening method improves performance compared to DM.

Vidyamurthy [191] proposes the use of the cointegration method for the pairs screening step. Thus, stocks are grouped in pairs if the cointegration test [57, 79] is accepted, i.e. if the test statistic is less than the critical value of 5%. The rules for opening and closing the positions are the same as DM, i.e. positions are opened at +2 or -2 standard deviations and closed at zero. Moreover, common to all work that uses the cointegration method, the z-score of the cointegration residual is used to measure how far a pair is from its long term equilibrium.

Bogomolov [24] evaluates the cointegration method and DM. His experiments corroborate the profitability of the pair trading strategy. However, the performance is reported only for the Australian Stock Market. Galenko et al. [69] also uses the cointegration method and evaluates the pair trading strategy for four exchange traded funds that track world stock market indexes. Their results indicate that pair trading is not only profitable for the US market, but also around the world. Jacobs and Weber [99] extend this analysis to 34 international markets and find that the pair trading strategy is persistently profitable.

Broussard and Vaihekoski [31] investigate the pair trading strategy profitability under different trading rules. More precisely, they vary the triggering value for opening the positions, usually, pre-set to +/-2 standard deviations. They found that the optimal trigger value (in-sample) achieved annualised returns of 12.5% (out-of-sample). Similar to our work, they also conducted a performance attribution analysis and found that pair trading profits are not related to market risk. In particular, this study only considers data from the Finnish stock market.

Huck and Afawubo [95] contrast the cointegration method with DM. Compared with other studies, they cover the largest number of stocks (500 North American stocks). As in our study, they provide a detailed evaluation of the pair trading strategy across sectors. Moreover, we use the same eleven sectors of their study. As in [31], they also experiment with different trigger levels (2 and 3 standard deviations), but extend the analysis by considering different lengths of historical data to perform the cointegration tests. They found that the cointegration method provides the strongest results (with average annual gains of ∼17%), where the DM generates insignificant returns. A similar performance is reported in [33], where the authors also use the cointegration method to investigate pair trading profitability, but on the Brazilian stock market.

In particular, we use the cointegration method at the pairs screening step. Our choice is mainly based on the exceptional performance of the cointegration method, as reported by the studies review above, and its wide usage in other areas of knowledge. In addition, by using the cointegration method, stocks are grouped using a more rigorous statistical hypothesis test compared to DM.

Puspaningrum et al. [148] investigates the relationship between the number of trades and the trigger value. Intuitively, the higher the trigger value (in terms of standard deviation) the higher the expected profit per trade, but the lower number of trades. Thus, they derive an analytical expression for the optimal trigger value by assuming a parametric AR(1) model for the cointegration residual, which as in other studies, quantifies how far a pair is from its long-term equilibrium. This study is relevant in the sense that it attempts to use a parametric model to optimise trading rules.

In a similar way, Fallahpour et al. [59] uses RL to optimise the trigger values. Similar to other studies, the trigger values are then used to open and close the positions. More specifically, they consider the trigger value as a discrete action, with values varying between zero and three standard deviations, in steps of 0.5 standard deviations. They also consider discrete actions for other parameters of the optimisation. In particular, a stop-loss parameter, and as in [95], a parameter controlling the length of historical data for the cointegration test. This stop-loss pre-sets a value to prematurely close the trade in case of losses and also takes discrete values based on the number of standard deviations. Importantly, we observe that even though they use RL, their approach is not applied to find optimal pair allocations, as in our study, but to find the optimal trigger value. This means that they follow the same rule-based approach at the decision-making step as in all studies reviewed above.

Tourin and Yan [180] propose an optimal stochastic control approach to the pair trading strategy. First, they assume that the pair price, noted $X(t)$, follows a *parametric* stochastic process. In particular, given its mean-reverting characteristic they assume an Ornstein-Uhlenbeck process [187] for the pair price. Then, they consider the problem of finding the optimal allocation for the pair, where the objective is to maximise, for a fixed time horizon H, the expected terminal wealth of this portfolio. Importantly, they obtain a closed form solution for the allocation. This allocation depends on the estimated coefficients of $X(t)$, such as the mean drift and speed of reversion of the pair price.

In general, the studies that employ stochastic control formulation, also consider a setting similar to ours. That is, they also optimise the allocations, which in our formulation are called actions, and they also consider a utility dependent on future rewards for the fixed time horizon. However, the main difference is that their approach is parametric. One clear advantage of the parametric approach is to obtain an analytical solution for the optimal allocation. However, the assumed process is usually a simplification of the real price process, which is usually governed by a much more complex dynamic. Moreover, particular to the financial markets, the estimated coefficients of the parametric process are strongly time-varying and highly unstable. In our study, we use a model-free Deep RL approach that learns optimal allocations by interacting with the environment. Thus, the model-free approach is expected to be more time-consuming, but has the advantage of not relying on a parametric model for the pair price.

Successful attempts have been made to use a model-free RL to trade single stocks, rather than pairs. Considerable research [133, 23, 46, 6] is based on the Direct RL algorithm introduced by Moody et al. [133]. One of the limitations of the Direct RL algorithm is that it only outputs discrete actions (i.e. long, neutral or short), but, more importantly, the method is less suitable for the pair trading strategy. This happens because the Direct RL optimisation only takes into account the *immediate reward*, as opposed to the Deep RL algorithms employed in our study, which optimise the *discounted future reward for a given horizon*. In other words, as opposed to single stocks, pair trading is based on convergence to the long-term equilibrium, which takes a very long horizon, on average 3.75 months [71]. In this scenario, considering only the immediate reward is of little value, since the effect of an action today will take a long time to mature and is only properly quantified taking into account future rewards.

More recently some studies [114, 199, 150] propose the use of the copulas method [101] at the screening step. In essence, the copulas are used in pair trading to describe the dependence between the two stocks, where stocks with high correlation are grouped in pairs. Compared to the cointegration method used in our study, the copulas method is computationally consuming, since it requires the estimation of the marginal distributions of each stock in the pair, and of the copula function, which is a *joint* cumulative distribution. Even though this method could potentially find different pairs, all studies that use copulas still use a triggering value to open and close the positions, a process that is abstracted away in our framework.

5.3 Background

5.3.1 Deep Reinforcement Learning

We consider a standard RL setup, where an agent interacts with an environment E in discrete time steps t. At any given time step t, the total return from a state s is defined as the sum of discounted future rewards over a horizon H and discount factor $\gamma \in [0, 1]$. That is:

$$R_{\text{total}}^{t} = \sum_{t'=t}^{H} \gamma^{t'-t} R(s^{t'}, a^{t'}) \tag{5.1}$$

In the above equation, the actions a are based on a policy π that maps states to actions. The goal in RL is to learn a policy that maximises the expected total return from an initial state distribution $p(s_1)$. Formally, this expectation (utility) is expressed as:

$$J = \mathbb{E}_{s_{t'} \sim \mathrm{E}, a_{t'} \sim \pi} \left[R_{\text{total}}^{1} \right] \tag{5.2}$$

Many RL algorithms make use of the action-value function Q which estimates the expected total return after taking action a^t in a state s^t and subsequently following the policy π, given by:

$$Q^{\pi}(s^t, a^t) = \mathbb{E}_{s_{t'} \sim \mathrm{E}, a_{t'} \sim \pi}\left[R^t_{\text{total}} | s^t, a^t\right], \quad \text{where } t' \geq t. \tag{5.3}$$

This action-value function can be expressed using a recursive relation known as the Bellman equation:

$$Q^{\pi}(s^t, a^t) = \mathbb{E}_{s^{t+1} \sim \mathrm{E}}\left[R(s^t, a^t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}\left[Q^{\pi}(s^{t+1}, a^{t+1})\right]\right], \tag{5.4}$$

where $R(s^t, a^t)$ is the return received after taking action a^t in a state s^t, also called the *immediate reward.* Assuming that the policy is deterministic the inner expectation over actions in the equation above can be avoided and, as a consequence, it is possible to learn the Q off-policy.

In this work, we consider only off-policy algorithms for both continuous and discrete action spaces.

Discrete actions

A popular off-policy algorithm is Q-learning [195], which uses a greedy policy, expressed as $a = \max_{a'} Q(s, a')$, to chose actions at each time step. Mnih et al. [131] introduces the deep Q-network (DQN), which adapts Q-learning and approximates a value function $Q(\cdot \mid \theta)$ using deep neural networks. The DQN optimal action-value function is learnt by minimising the loss:

$$\begin{aligned} \mathcal{L}(\theta) &= \mathbb{E}_{s^t, a^t, R, s^{t+1} \sim \mathcal{D}}[(Q(s^t, a^t \mid \theta) - y)^2], \\ \text{with } y &= R(s^t, a^t) + \gamma \max_{a^{t+1}} \bar{Q}(s^{t+1}, a^{t+1}), \end{aligned} \tag{5.5}$$

where $\bar{Q}$ is the *target network* with no trainable weights which are periodically updated with the most recent θ, and $\mathcal{D}$ is the *experience replay buffer* containing tuples (s^t, a^t, R, s^{t+1}) collected from the agent's experiences at each time step.

In contrast with supervised learning, where the targets (y) are fixed before training, here the targets depend on network weights. In this situation, the target network, where weights are only updated from time to time, proved to be a crucial component in stabilising DQN training [131]. In the same spirit, the replay buffer breaks the correlation between consecutive samples, thus reducing the variance of the updates. Note that the gradient updates are computed using past experiences (i.e. off-policy) which are reused during training. This dramatically increases the sample efficiency [131].

In terms of network architecture, the Q network receives as input a state s and outputs a single number for each valid action – what makes DQN only suitable for discrete action spaces (in our case buy, sell-short or hold a given stock pair).

Continuous actions

DDPG [115] is an off-policy algorithm that uses neural networks to approximate the performance of a deterministic policy [168]. The algorithm maintains both a critic $Q(s, a|\theta^Q)$ and an actor $\mu(s|\theta^\mu)$ network parametrized with weights θ^Q and θ^μ.

In practical terms, the critic loss is similar to DQN, though actions are based on the actor network output. That is, the loss target y in Equation (5.5) is replaced with:

$$y = R(s^t, a^t) + \gamma \bar{Q}(s^{t+1}, \mu(s^{t+1})) \tag{5.6}$$

Additionally, the actor is updated by maximising the expected return from the start distribution J (Equation (5.2)). Silver et al. [168] proved that the expected gradient of the policy's performance, called the deterministic policy gradient, is given by:

$$\nabla_{\theta^\mu} J = \mathbb{E}_{s^t \sim \mathcal{D}}[\nabla_a Q(s, a|\theta^Q)|_{s=s^t, a=\mu(s^t)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s^t}]. \tag{5.7}$$

We observe that because the first gradient concerns the actions a, DDPG is appropriate for continuous action spaces.

Similar to DQN, we see from the equations above that DDPG is trained by sampling trajectories (batches) from the replay buffer. Furthermore, it also makes use of target networks in order to stabilise training. However, it introduces the concept of "soft" target network updates, rather than periodically copying the weights as in DQN. This update follows the rule in Lillicrap et al. [115]:

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta, \tag{5.8}$$

where prime means target network weights, for both actor and critic, and τ is a hyperparameter. We observe that despite the fact that the weights of the target network are updated at each time step, because $\tau \ll 1$ the target network values change slowly.

5.3.2 Time series cointegration and long-term equilibrium

Cointegration measures the degree of long-term relationship between time series and was introduced to address the problem of spurious correlation in non-stationary time series. The problem arises when regressing the level of economic variables. Even though they may appear to be related based on standard statistical measures, they usually hold no relationship. Indeed, if two time series are both non-stationary then the hypothesis of no relationship is rejected even when none exists [78, 79].

Formally, considering N time series $\{y_i^t\}_{i=1}^N$ as elements of the Y_t $(N \times 1)$ vector, it is said that the time series are cointegrated of order d and d', denoted $y_t \sim CI\,(d, d')$, if

1. All y_t are integrated of order d, denoted $I(d)$, with $d \neq 0$[3].
2. There exists a vector $\beta = (1, -\beta_j, \cdots, -\beta_{N-1})$ with elements $\beta_j, ..., \beta_{N-1} \neq 0$ such that $z_t = \beta Y_t \sim I(d - d')$.

Here, β is the cointegrating vector. Typically, stock prices are *integrated of order one*, meaning $d = 1$ and $d' = 0$ above. That is, though the price series is not stationary the first difference of the logarithm of the price, also called the *log return*, is stationary. Additionally, we search for cointegration only between pairs of stocks, i.e. $N = 2$ above. It is worth noting that few time series are cointegrated. However, if two time series have a long-term relationship it is expected that a linear combination of the time series is stationary. This holds true even in the case where each time series is completely unpredictable and follows a random walk. Intuitively, cointegration can be understood in the following way: large relative price deviations between cointegrated stocks are expected to correct and revert to the mean in the long-term. This happens because the *difference* between the stock prices, as measured by the β coefficient, must be stationary. We observe that the cointegration is closely related to Granger causality, where all cointegrated series implies causality. Nevertheless, the cointegration does not estimate the direction of the causality, i.e. whether i causes j, j causes i, or the causality occurs in both directions.

In order to search genuine relationships (i.e. cointegration), as opposed to spurious correlation, we employ the following two step procedure [57]:

Proposition 1 (Engle-Granger two-step cointegration test)**.** *Considering two time series $y_{1,t} \sim I(1)$ and $y_{2,t} \sim I(1)$*

1. *Estimate the cointegration constant β of equation $y_{1,t} = \beta y_{2,t} + \epsilon_t$.*
2. *If the series are cointegrated the residual ϵ_t is stationary.*

The stationarity above is determined by applying the Augmented Dickey-Fuller (ADF) test on the regression residual. For a given time series ϵ_t we estimate the model:

$$\Delta\epsilon_t = \alpha + \beta t + \gamma\epsilon_{t-1} + \delta_1\Delta\epsilon_{t-1} + \cdots + \delta_{p-1}\Delta\epsilon_{t-p+1} + u_t. \tag{5.9}$$

Under the null hypothesis, i.e. the series needs to be differenced to make it stationary, $\gamma = 0$ and under the alternative hypothesis of stationarity $\gamma < 0$. The stationarity hypothesis is accepted by comparing the gamma t-value to the negative critical values for the ADF test. The rationale behind the stationarity test can be interpreted in terms of the ADF model in Equation (5.9). Intuitively, for $\gamma < 0$ any large positive (negative) ϵ_t contributes with a negative (positive) $\Delta\epsilon_t$ that forces the series in the

[3]A time series y_t is integrated of order d if they need d differences(s) to be stationary. A classical example is the stock price (time series). This time series is non-stationary. However, by taking the log difference of the price series we obtain the price returns (time series). This return (time series) is usually stationary, in the sense that, different than the prices, they usually oscillate around a zero mean. In this case we say the price time series are $I(1)$, and the return time series stationary (or $I(0)$).

opposite direction, i.e. forces it to revert to the mean. In particular, the random walk model $RW \sim I(1)$ has $\gamma = 1$ and $\alpha = \beta = \delta = 0$.

5.4 Methodology

In what follows we provide details of the three components of our ISIP framework in Figure 5.1, namely Pairs Screening (orange), Decision Making (light green), and Portfolio Risk Management (dark green).

As mentioned above, the ISIP framework integrates the optimal allocations of our Deep RL models with a portfolio risk management component. It considers the individual preferences of each investor concerning: (1) the level of volatility σ_{target} that he/she is comfortable in taking, i.e. the risk tolerance; (2) The N stocks he/she is interested in trading, i.e. the portfolio constituent stocks. At a high-level, given the preferences of an investor as input variables, our framework outputs the optimal vector of allocations $w^t = (w_1^t, \cdots, w_N^t)$.

5.4.1 Pairs Screening

Here we describe the pair screening component (orange) of our framework (ISIP) depicted in Figure 5.1. We start by discussing how we leverage information of the stock's classification structure in Chapter 4. We then describe in detail how the cointegration method is employed in our work to group two stocks by long-term relationship.

Since we are interested in stocks that are cointegrated, we need to apply the statistical test described in Section 5.3.2 to each potential stock pair. However, we would have to search a combinatorial space in the order of hundreds of thousands. In addition, the sector classification scheme is still very coarse. For example, the Technology sector places the companies IPG Photonics (a laser equipment producer) and Microsoft (software) under the same umbrella. Although at a high level both are "Tech" companies, their stocks are expected to have very different drivers, with IPG Photonics being more sensitive to oscillations in the price of the commodities necessary in manufacturing the laser equipment, while Microsoft is more sensitive to aggregated consumer spending. In order to achieve a more granular level of segmentation for the pairs screening process, we use the industry classification of each sector. The classification structure that we use is described in Section 4.4.1.

Industry heuristics for pairs screening

We propose a heuristic search that explores stock pairs limited to stocks of the same industry. This heuristic is based on the observation that stocks in the same industry are expected to share the same factors of production (e.g. natural resources) and demand drivers (e.g. consumer spending) [153]. In essence, the proposed heuristic trades off pair screening speed for optimal finding of inter-industry cointegration. Using this heuristic, the pairs screening process is divided into the following steps:

- *Set splits*: only test for cointegration using data from 2000 to 2015 (i.e. fitting step). Additionally, filter out any industry with fewer than 3 stocks.
- *Selection criteria*: search for cointegrated pairs from the set of all possible pairwise combinations of stocks within the same industry (search space) using the two-step process discussed in Proposition 1. First, s_1 and s_2 with price time series $p^t_{s_1}$ and $p^t_{s_2}$, respectively, the goal is to find pairs of stocks where the residual $\epsilon^t_i = \ln\left(p^t_{s_1}\right) - \beta_i \ln\left(p^t_{s_2}\right)$ is stationary. We use ADF in Equation (5.9) to test for stationarity using a significance level of 0.05. Here, the significance level sets the threshold for deciding if the pair is cointegrated or not. Therefore, if the residuals are stationary (p $<$ 0.05) the pair is selected. In particular, we use MacKinnon's critical values [123], with the number of lags in the ADF estimated using the Akaike Information Criterion (AIC) [2].
- *Pair residual series*: The i^{th} pair residual series ϵ^t_i is computed using β_i above (i.e. transform step). Note that we do not use the test set to estimate β_i, and it effectively avoids any training data leaking. As in all other studies using cointegration, the series ϵ^t_i measures the level of divergence between s_1 and s_2. In our study, the pair history h^t_i is a function of the pair residual series e^t_i and h^t_i is the input data for our neural network.

Table 5.1 highlights the outcome of the pairs screening process described above. Compared to the original search space, which included all possible pairs, by using our heuristic the search space was reduced by a factor of $\times 100$. Note that even considering intra-industry stocks, only about 20% of possible pairs have long-term relationships. The complete list of all 205 pairs is provided in Appendix D.2. A visual inspection provides evidence that the screening process produces fairly intuitive results. To cite a few, we have *Cisco Systems* x *Motorola* for the Technology sector and *Halliburton Company* x *Schlumberger* for the Oil sector. In fact, the Industrials sector is more diffuse, capturing pairs ranging from *Lockheed Martin* x *Northrop Grumman* to *FedEx* x *United Parcel Service (UPS)* and *Republic Services* x *Waste Management.*[4] Given the rich set of pairs covered and intuitive results, we argue that the heuristic search and the selection criteria proposed in this work is effective in discovering stocks that are "truly" related, as well as, delivers a highly diversified set of pairs.

5.4.2 Decision-making

This section provides details of the Decision-making component (light green) of our framework (ISIP) (see Figure 5.1). This is a core component that holds all neural

[4]Both companies offer waste collection and recycling services.

Sector	Number of Pairs (N_p)	Number of Stocks (N)	Size of Search Space
Industrials	61	38	216
Financial Services	40	32	156
Healthcare	34	30	147
Technology	28	25	95
Utilities	22	20	157
Consumer Cyclical	18	20	53
Energy	12	15	97
Basic Materials	6	6	15
Consumer Defensive	5	8	42
Real Estate	4	6	36
Communication Services	3	5	10
All sectors	*233*	*205*	*1024*

Table 5.1: **Number of cointegrated pairs and respective search space size**. Values are sorted by number of pairs N_p.

networks in our Deep RL architecture and the only component of our framework that is effectively trained.

We start by describing the reward and its structure. We then describe the Deep RL settings that inspired our architecture. Finally, we show how our agent, called the pairs trader agent, learns to trade *any* pair by interacting with the environment.

Reward structure

We consider a set of N_p stock pairs with price returns at a given time t denoted by the vector $\boldsymbol{r}^t := (r_1^t, r_2^t, \cdots, r_{N_p}^t)$ and actions by $\boldsymbol{a}^t := (a_1^t, a_2^t, \cdots, a_{N_p}^t)$, where a_i^t represents the percentage of wealth invested on the i-th pair. Our goal is to find an optimal policy $\pi^*(\boldsymbol{a}^t \mid \boldsymbol{h}^t)$, hence also called a *trading strategy*, that is conditioned on a history of past information $\boldsymbol{h}^t := (h_1^t, h_2^t, \cdots, h_{N_p}^t)$ or observations. To this end, we consider a sequential learning process that is reinforced by rewarding actions taken at each time t. Below, we detail the reward structure of our problem.

Definition 1 (**Reward structure**)**.** We assume our actions are minuscule compared with the market size, therefore do not impact the stock price dynamics. The **local reward** is the return at $t+1$ of a strategy that invests a_i percentage of wealth on pair i at time t:

$$R_i^{t+1}(\boldsymbol{h}^t, a_i^t) := a_i^t r_i^{t+1}(\boldsymbol{h}^t), \tag{5.10}$$

where the *next period* price return of the the i-th pair, noted r_i^{t+1}, is a stochastic variable *not observed* by the agent and obtained from public stock data. Additionally, we assume that this next period return can be approximated as a function of the pairs' histories[5] up to a time t. Moreover, the **joint reward** is the exact expression

[5]For example, the next period return could depend on how far a pair is from the mean-reverting equilibrium and its history

of the portfolio return of N stock pairs and defined by

$$\mathscr{R}^{t+1}(\boldsymbol{h}^t, \boldsymbol{a}^t) := R_1^{t+1}(\boldsymbol{h}^t, a_1^t) + \cdots + R_N^{t+1}(\boldsymbol{h}^t, a_N^t) \tag{5.11}$$

where the i-th reward R_i^{t+1} is the local reward defined above.

By choosing the global/joint reward as the portfolio return above we have a natural way to express it in terms of individual/local contributions, where the local reward relevance comes directly from its additive composition to the global reward. It has been shown that the use of local reward information significantly reduces the number of samples required for training [13, 80] and increases model performance [48].

Importantly, regardless of any reward structure used during training, we evaluate our results using the global reward $\mathscr{R}$ in Equation (5.11), which measures the total portfolio return.

Deep RL settings

In this section we review the elements of the single and multi-agent settings that inspired our Deep RL architecture. Henceforth, the word agent refers to a "spread trader" which decides on pairs allocations (actions). In this context, multi-agent means each agent specialises in a more fine-grained structure. For example, a setting where each agent decides on the actions of only a single stock pair, implying that the number of agents is equal to the number of pairs ($N_p = 205$). Another example is a setting where agents are specialised in a sector, in this case we would have 11 agents.

Centralised controller. The *centralised controller* [149, 128] reduces to a single-agent setting that operates on a concatenation of all N observations $\boldsymbol{h}^t$ in order to learn a joint action $\boldsymbol{a}^t$. Although straightforward, the main limitation of this approach is the *exponential growth of the combined action space*, making it intractable as N increases.[6] Additionally, given its central nature, the reward cannot be decomposed into local contributions; therefore, only the global reward is used during training. Finally, it was shown [88] that a common situation in this setting is to learn a policy where one agent excels, while others remain unskilled, a behaviour also observed in [172] and called "lazy" learners.

Independent learners. In this multi-agent setting each agent is *trained independently* to optimise the *global (joint) reward*, but conditioned only on its own individual action-observation [176]. This setting presents two main drawbacks. First, because agents change their behaviour during training, each agent is confronted with a non-stationary learning problem, which violates the convergence guarantees of the

[6]To exemplify, if we consider the same discrete actions for each pair, i.e. buy, sell or hold, a centralised policy network would require an output layer with 3^N units, representing all possible joint actions of the N pairs.

decision making process [110, 186]. Second, because the environment is partially observed from each agent's perspective, changes in the rewards may be originated from other agents' unobserved behaviour. In other words, the agent struggles to learn a policy effectively because it cannot discriminate between its own actions and other agents' actions. This problem is called the *spurious reward signal* [48]. Moreover, similar to the centralised setting, the ability to learn also decreases with the number of agents[7].

Some solutions were introduced to alleviate the problem of spurious reward signals in the multi-agent setting. Proper and Tumer [147] proposes the design of individual reward functions, i.e. not shared between all agents, called *difference rewards*. In particular, they discount the reward received from the global reward if the individual agent did not contribute to the environment. This type of discount can also be framed as a counterfactual term [67]. Thus, the difference rewards allows the agent to understand if it was responsible for the feedback received. Devlin et al. [48] unifies difference rewards in the general framework of potential-based reward shaping. One of the advantages of potential-based functions that the policy learnt from the shaped reward is guaranteed to be from the same set of possible policies of the original reward [47]. However, they show that large improvements in the agent's convergence time and learning ability are achieved only when the potential functions are handcrafted for the specific problem domain.

Proposed Deep RL setting

In this work, we propose a single-agent network called the spread trader controller. Similar to the decentralised independent learners approach, this (single) controller receives only local observations h_i^t. However, it differs from the independent learners approach in the following aspects:

- We exploit the *natural decomposition of our reward structure*, as detailed in Section 5.4.2, and use only local rewards during the training process, rather than the global (joint) reward.
- While interacting with the environment, the collected experiences are stored into a *single buffer for all stock pairs*, here called the *multi-pair replay buffer*.

The main advantage of our setting is that it can deployed on portfolios of any size without having to retrain the model to comply with each investor's preferences. This important feature is because the actions that compose the portfolio depend only on local action-observations.

We introduce the multi-pair replay buffer to encourage *generalisation*. Since the network is trained taking into account experiences of multiple stock pairs, we hoped to learn the *genuine properties* that make distortions between any pair of stocks revert to the mean, rather than attempting to learn the idiosyncrasies of the reversion

[7]Intuitively, with more agents into play, the chances increase that the global reward received may not be related to the agent's own action-observation.

process for each stock pair. Additionally, compared to the N buffers of the Independent Learners setting, the multi-pair buffer is *richer in terms of distinct observations*. In fact, our architecture allows a maximum number of different observations in the replay buffer N times larger than in the centralised controller or independent learners settings.[8]

By training using only local rewards, we obliterate the previously mentioned spurious reward problem. We also observe that, in contrast to the centralised setting, the size of the action space remains the same for any number of pairs. In other words, the proposed solution does not suffer from the exponential growth of the combined action.

However, one clear limitation of our approach is that it considers only the individual pair history information to learn the pair action. Below, we explain how our setting, i.e. local observation-action and reward, is a good fit for the specific problem of pair trading.

Setting rationale and Specialisation. In Section 5.4.1, we saw that some stocks exhibit common trends, i.e. their price dynamics share (unobserved) latent factors. Thus, predictions at the individual stock level are expected to depend on the stock's own price history, as well as that of every other stock. However, specific to the pair trading problem, the history is given by the cointegrating residuals and we expect the cointegration residual to be free of temporal cross-correlation. In other words, though some individual stocks move together, after correcting for this effect, as in the cointegrating residual series, the distortion from the equilibrium of a given pair i is not expected to be related to another pair j.

As in Chapter 3, we introduce a *sector embedding* to the learning process. The embedding functionality is the same as the previous stock embedding, i.e. *to increase specialisation* of the single network, i.e. it adds a *role information* to the learning process.

Rather than N independent networks[9] (i.e. one per agent), our single network approximates the local reward in Equation (5.10) using a (single) function f. Thus,

[8]Considering the same actions for all settings, in the centralised controller and independent learners settings the maximum number of different experiences is limited to the number of days available in the training set, noted N_d, since the centralised controller observes the history of all stocks simultaneously and in independent learning we have an experience buffer for each stock. By contrast, in our approach this number is equal to $N \times N_d$, since the history of each pair is stored in the same experience replay buffer.

[9]Typically, though independent, these networks are trained using a certain degree of parameter sharing, as proposed in [172, 88, 80].

based on the characteristics described above, we have the following simplification:

$$\begin{aligned}
\hat{R}_i^{t+1}(\boldsymbol{h}^t, a_i^t) &:= a_i^t f_i(r_i^{t+1} \mid h_i^t, \cdots, h_N^t) && \textit{approx. of returns in Equation } (5.10) \\
&\approx a_i^t f_i(r_i^{t+1} \mid h_i^t) && \textit{no temporal cross-correlation} \\
&\approx a_i^t f(r_i^{t+1} \mid h_i^t, I) && \textit{single network with role information} \\
&:= \hat{R}^{t+1}(o_i^t, a_i^t), && \textit{only locally dependent}
\end{aligned} \tag{5.12}$$

where the observation o_i^t is the concatenation of the individual pair history h_i^t and the sector one-hot representation I_i. That is,

$$o_i^t := [h_i^t; I_i] \tag{5.13}$$

In the next section, we explain how our replay buffer is populated with experiences from different stocks, and the general interaction of the spread trader agent with the enviroment.

Agent-Environment interaction

In Definition 2 we detail our Agent-Environment interaction. The collection of experience from different stock pairs, which are stored in a shared buffer $\mathcal{D}$, is based on random selection at the environment reset stage. Additionally, we have a fixed horizon task, where the episode finishes after H steps (measured in days). In our experiments, we set H to one year (~ 252 trading days), to give enough time for pair convergence.

Definition 2 (**Agent-Environment interaction**)**.** We consider a single agent, the pairs trader, who experiences the activity of N_p pairs through the pair trading environment. The agent interacts with the environment on a daily basis in order to learn the percentage of wealth to allocate on *any* stock pair. The stages of the Agent-Environment interaction are:

1. *Environment reset*: At the beginning of each episode
   (a) Randomly choose a day index t and stock pair index i from the set of all available training dates and the universe of N_p pairs.
   (b) Assign a one-hot indicator I value to the selected stock pair sector.
   (c) The agent observes $o_i^t := [h_i^t; I_i]$.
2. *Steps loop*: For each day
   (a) Based on the last state, the agent takes an action a_i^t, denoted as the percentage of wealth to allocate to pair i. It then receives reward R_i^{t+1} (Equation (5.10)) and observes the new o_i^{t+1}.
   (b) Store experience tuple $e := (o_i^t, a_i^t, R_i^{t+1}, o_i^{t+1})$ in the *multi-pair replay buffer* $\mathcal{D}$

continue these steps for a fixed horizon of H days until the episode finishes. Then, repeat Environment reset for a total of M training episodes.

We observe that even though the Deep RL algorithms (DQN and DDPG in Section 5.3.1) considered in this work are implementation specific, the abstract Agent-Environment interaction proposed in our study for a universe with N_p pairs remains the same. In addition, we implemented the pair trading environment following the widely used specification OpenAI Gym [30]. Thus, our pair trading environment can be easily used by other researchers and plugged into any RL algorithm.

Neural network architectures

Figure 5.2, depicts the DQN that approximates the Q-value. The output vector represent the i^{th} the three possible pair allocations [long = 1, neutral = 0 and short = -1]. All formulation is presented in Section 5.3.1

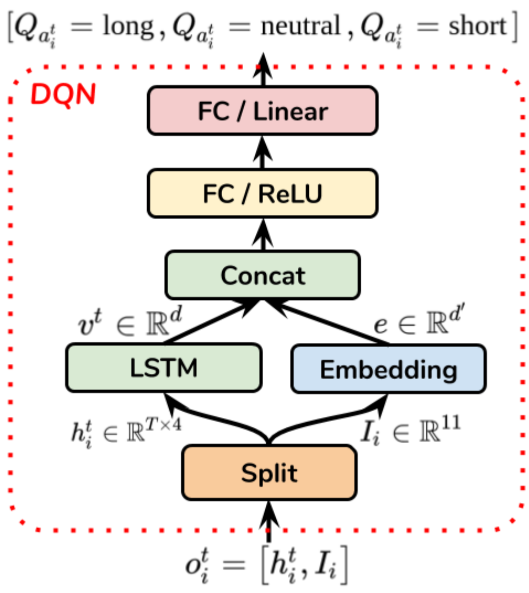

$[Q_{a_i^t} = \text{long}, Q_{a_i^t} = \text{neutral}, Q_{a_i^t} = \text{short}]$
DQN
FC / Linear
FC / ReLU
Concat
$v^t \in \mathbb{R}^d$
$e \in \mathbb{R}^{d'}$
LSTM
Embedding
$h_i^t \in \mathbb{R}^{T \times 4}$
$I_i \in \mathbb{R}^{11}$
Split
$o_i^t = [h_i^t, I_i]$

Figure 5.2: **Deep Q-Network (DQN) architecture.**. The network outputs the Q-value of the three possible actions (i.e. long, neutral, short) for a given state. The state is represented by a concatenation of i^{th} pair history and an one-hot vector indicating its sector (total of 11 possible sectors). The embedding layer (blue) retrieves a representation from the embedding matrix whose elements are trainable.

Figure 5.3, depicts the actor-critic architecture for the continuous actions DDPG algorithm described in Section 5.3.1.

Note that in the critic network the representation of the input state is concatenated with the action a_i^t using the Concat layer.

We observe that during training the agent receives observations o_i^t from the environment, and this process happens in a way that after one year a new random pair is selected (as detailed in Definition 2). Moreover, the LSTM encodes a pair history and is not shared between the actor and critic. The (trainable) embedding matrix is shared between both networks.

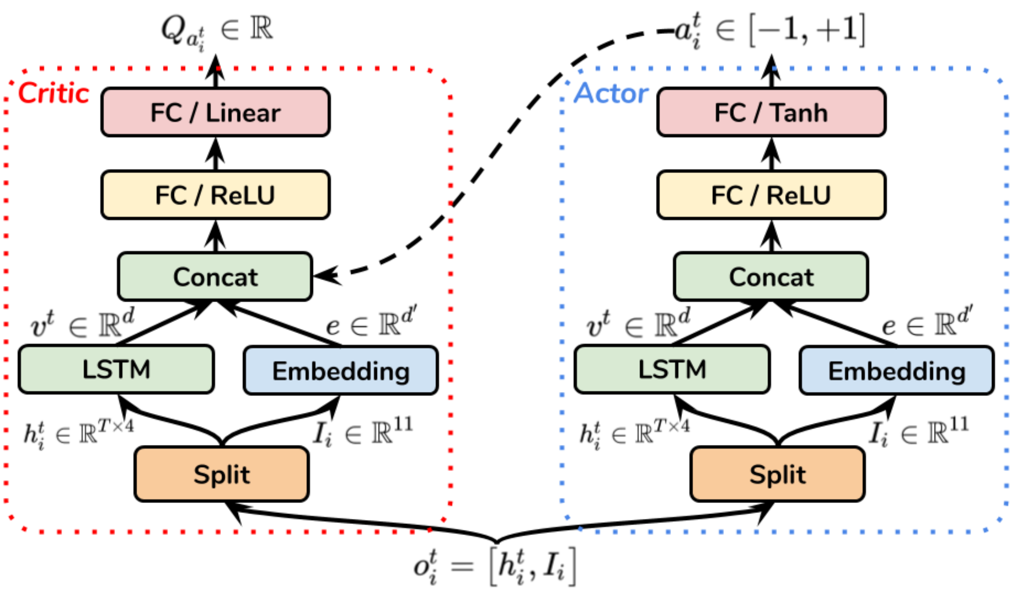


Figure 5.3: **DDPG Actor-Critic architecture.** Depict of Actor and Critic networks. In both networks the observation (or state) is represented by a concatenation of i^{th} pair history and an one-hot vector indicating its sector (total of 11 possible sectors). The embedding layer (blue) retrieves a representation from the embedding matrix whose elements are trainable, but shared between the Actor and Critic *Right:* Actor network. *Left:* Critic network. Note that in the Critic network the representation of the state is concatenated with the action a_i^t, which is given by the output of the Actor network.

In our study, we compare the performance of our DRL models with a baseline SL model. In what follows, we explain the SL setting.

5.4.3 Decision-making: SL Setting

In our study, we contrast our Deep RL models with a baseline SL model. This SL model is trained in a multi-class classification setting, and rely on the *history-prediction-actions* pipeline to convert the classification predictions into trade actions a_i^t.

History-prediction-actions pipeline

In the SL setting, the input history is as in the Deep RL setting. Precisely, the history is represented by the concatenation of pair history and its respective sector. The output prediction is based on the classification of the next day pair return, where, as in the DQN architecture, we consider three (3) classes: long, neutral and short.

The ground truth label is computed by discretising the continuous pair return. During the discretisation the width of the three bins is such that the number of samples for each class is equal. Each bin represent the short, neutral and long labels.

In order to convert the three class predictions into actions, after the model is optimised, we use the trading rules used in many studies:

1. If the prediction class is long then $a_i^t = +100\%$ (buy the pair)
2. If the prediction class is neutral then $a_i^t = 0\%$ (do not invest in the pair)
3. If the prediction class is neutral then $a_i^t = -100\%$ (short the pair)

Note that in the Deep RL setting the agent has to learn the action a_i^t, including its sign, which represents the trade type. In contrast, in the SL setting the allocation is given by a trading rule based on the classification predictions.

Neural network architecture

Figure 5.4 , depicts the neural network architecture for the pair return classification. Its output is the probability P of each class (long, neutral or short), where the top softmax layer (purple) ensures that the probability of all classes sums to one. The embedding layer (blue) retrieves a representation from the embedding matrix whose elements are trainable.

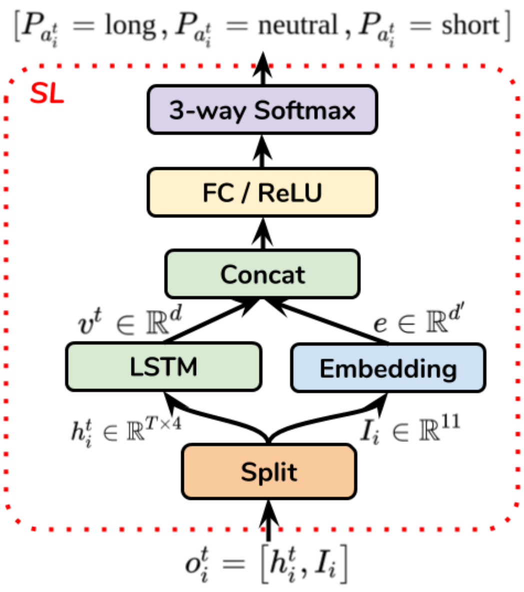

$[P_{a_i^t} = \text{long}, P_{a_i^t} = \text{neutral}, P_{a_i^t} = \text{short}]$
SL
3-way Softmax
FC / ReLU
Concat
$v^t \in \mathbb{R}^d$
$e \in \mathbb{R}^{d'}$
LSTM
Embedding
$h_i^t \in \mathbb{R}^{T\times 4}$
$I_i \in \mathbb{R}^{11}$
Split
$o_i^t = [h_i^t, I_i]$

Figure 5.4: **Baseline SL architecture.** Depiction of the neural network architecture employed to train a baseline SL model for the pair return classification. The input is as in the Deep RL setting, and is given by the i^{th} pair history and an one-hot vector indicating its sector (total of 11 possible sectors). The output is the probability P of each class (long, neutral or short), where the top softmax layer (purple) ensures that the probability of all classes sums to one. The embedding layer (blue) retrieves a representation from the embedding matrix whose elements are trainable. We observe that in the SL setting we have to rely on a trading rule to convert predictions to trade positions (as described in Section 5.4.3)

In the next section we discuss the risk management component of our ISIP framework.

5.4.4 Portfolio Risk Management

Here we describe the Portfolio Risk management component (dark green) of our framework (ISIP) depicted in Figure 5.1. This component is only used at execution time, when the allocations are adjusted to keep the volatility of the portfolio at the level pre-set by the investor's σ_{target}.

At execution time, i.e. after all models were optimised, the current optimal allocation a_i^t is computed independently for each pair $i = 1, \cdots, N_p$. Below we explain how these actions are obtained:

1. DRL setting: In the discrete actions case, the optimal action is chosen by taking the action that maximises DQN network output (as depicted in Figure 5.2). Therefore, it has three possible values: $a_i^t = +100\%$ (long), or $a_i^t = 0\%$ (neutral) or $a_i^t = -100\%$ (short). For the continuous actions case, we completely disregard the critic network in Figure 5.3, and the optimal action is given by the output of the actor network, where we have $a_i^t \in [-100\%, +100\%]$.

2. SL setting: In this baseline SL model the actions are given by the trading rules described in Section 5.4.3. The trading rules depend on the output of the neural network in Figure 5.4. After applying the trading rules the allocation can assume three possible values: $a_i^t = +100\%$ (long), or $a_i^t = 0\%$ (neutral) or $a_i^t = -100\%$ (short).

By using the procedure above, we obtain the vector of current (optimal) pair allocations $a^t = (a_1^t, \cdots, a_{N_p}^t)$. However, for any investment strategy the portfolio risk has to be computed in terms of the stock allocations, and defined by the vector $w^t = (w_1^t, \cdots, w_N^t)$. Below, we explain this procedure.

Obtaining stock allocations. The pair return R_i^t is computed in terms of the predicted allocation a_i^t as in the reward formula Equation (5.10):

$$R_i^t = a_i^t \cdot \left(R_{s_1}^t - \beta_i R_{s_2}^t\right) = \left(a_i^t\right) R_{s_1}^t - \left(a_i^t \beta_i\right) R_{s_2}^t, \tag{5.14}$$

where β_i is the pair cointegration coefficient, and $R_{s_1}^t$ and $R_{s_2}^t$ are returns of the pair constituent stocks s_1 and s_2, respectively. According to this equation, a_i percentage of wealth is allocated to stock s_1, while $(a_i\beta_i)$ is allocated to stock s_2. Each pair will contribute with one long and one short position. By repeating this procedure for all N_p pairs the stock allocation w_i^t is the sum of all contributions to stock i.

Once the vector of optimal stock allocations w^t is obtained, we apply a normalisation to ensure that the stock portfolio is self-financing. This type of portfolio is also used in [62, 58] and in the pair trading context in [71].

We observe that, as described in Section 2.2.2, in the self-financing portfolio, also called zero-investment, the sum of allocations is zero and no net-investment is need to enter these portfolios.

Normalisation of the stock allocations. After the portfolio allocations w_i^t are obtained, we apply the following normalisation:

$$w_i^t = \begin{cases} \dfrac{w_i^t}{\sum_{i=1}^N w_i^t \cdot \mathbb{1}\left(w_i^t \geq 0\right)}, & \text{for } w_i^t \geq 0, \text{ i.e. (positive) long weights ;} \\ \dfrac{w_i^t}{\sum_{i=1}^N \left|w_i^t \cdot \mathbb{1}\left(w_i^t < 0\right)\right|}, & \text{for } w_i^t < 0, \text{ i.e. (negative) short weights} \end{cases} \tag{5.15}$$

The denominator in the equation above computes the normalisation factor of the long and short positions independently. This implies the following summations

$$\begin{aligned} \sum_{i=1}^N w_i^t \cdot \mathbb{1}\left(w_i^t \geq 0\right) &= 1 && \textit{all weights of long positions sum to one} \\ \sum_{i=1}^N w_i^t \cdot \mathbb{1}\left(w_i^t < 0\right) &= -1 && \textit{all weights of short positions sum to minus one} \\ \sum_{i=1}^N w_i^t &= 0 && \textit{sum of all weights is zero} \end{aligned} \tag{5.16}$$

Here, we have the number of long positions $N_l^t = \sum_{i=1}^N \mathbb{1}\left(w_i^t \geq 0\right)$ plus the number short positions $N_s^t = \sum_{i=1}^N \mathbb{1}\left(w_i^t < 0\right)$ is equal to the total number of stocks in the portfolio $N = N_l^t + N_s^t$. Note that the number of long and short positions, i.e. which stocks go long or short and the weights, varies over time (here, daily) and entirely depends on each pair allocation a_i^t, given by the Deep RL models or the baseline SL model.

Position sizing

Once a self-financing portfolio is obtained using the normalisation described above we use the same position sizing mechanism employed in Chapter 4, and described in details in Section 2.2.3

This mechanism uses the *volatility targeting* method proposed in [96, 34]. By using the proposed mechanism we scale down the portfolio at times of high volatility (high risk) and leverage at times of low volatility. Effectively, the portfolio is always targeting a constant level of ex-ante volatility σ_{target}, where for evaluation purposes we set this target to 10% per year.

5.4.5 Walk-through Example

In what follows, we provide a simple example for the steps of our framework. For illustration purposes, we consider an investor interest in $N = 5$ stocks in the Technology sector and that the pairs screening component finds the following $N_p = 3$ pairs:

1. Adobe Systems (ADBE) x Autodesk (ADSK)

2. Autodesk (ADSK) x Intuit (INTU)

3. Citrix Systems (CTXS) x Oracle (ORCL)

At day t, we also consider the following situation:

1. The decision-making component (Deep RL) predicts the pair allocations/weights $a^t = (0.80, 0.40, -0.90)$

2. The pair's cointegration coefficients are $\beta = (1.10, 0.82, 1.0)$

3. The investor is a risk-taker, meaning she is comfortable with a volatility level of $\sigma_{\text{target}} = 15\%$ per year

First, we *convert* the pair weights (a_3^t, a_2^t, a_3^t) to stock weights (w_3^t, w_2^t, w_3^t). Thus, the pairs above contribute the following stock weights/allocations:

1. (long) $+$0.80 ADBE and (short) -(0.80 $\cdot$ 1.10) $=$ -0.88 ADSK

2. (long) $+$0.40 ADSK and (short) -(0.40 $\cdot$ 0.82) $=$ -0.41 INTU

3. (short) -0.90 CTXS and (long) -(-0.90 $\cdot$ 1.0) $=$ $+$0.90 ORCL

The *net weights of each stock* may then be given by $w^t = (0.80, -0.88 + 0.40 = -0.48, -0.41, -0.90, +0.90)$, for the stocks ADBE, ADSK, INTU, CTXS and ORCL, respectively. Thus, we have a total of $N = 5$ stocks with $N_l^t = 2$ long positions (ADBE and ORCL) and $N_s^t = 3$ short positions (ADSK, INTU and CTXS). Note that we have only 5 stocks, rather than two per pair (or 6 stocks), because the first and second pairs share the same ADSK stock. Also, because we went short on the last pair, i.e. $a_3^t < 0$, we go short on the first stock $s_1 =$ CTXS and long on the second stock $s_2 =$ ORCL.

Once the net weights were obtained, we *normalised* the long and short positions independently. Using Equation (5.15), we have the normalisation factors (equation denominator): $(0.80 + 0.90) = 1.70$ (for the long positions) and $(|-0.48| + |-0.41| + |-0.90|) = 1.79$ (for the short positions). Thus, the *normalised portfolio weights* may then be denoted by $w^t = (0.47, -0.27, -0.23, -0.50, 0.53)$, where we divided all short positions by 1.79 and the long ones by 1.70. As denoted in Equation (5.16), we observe that the sum of all stock weights is zero (i.e. zero net investment) and the long and short positions sum to one (in absolute value).

Now we apply *position sizing* to adjust the risk to a level the investor is comfortable with ($\sigma_{\text{target}} = 15\%$ per year). This is done using the risk tolerance component of our system. We collect a past history of portfolio performance, i.e. by computing the past (portfolio) returns using the same allocation $w^t = (0.47, -0.27, -0.23, -0.50, 0.53)$ and stock returns T days preceding the current day t. Here, we suppose that this computation leads to an *ex ante portfolio volatility* $\hat{\sigma}^t = 1.25\%$ per day or $1.25\% \cdot \sqrt{252} = 19.92\%$ per year. Thus, implementing the portfolio as it is would exceed the σ_{target} of 15% per year, set by the investor. In order to conform with acceptable levels of volatility, we reduce the portfolio allocation using Equation (2.18), and the *volatility-scaled weights* may then be $w'^t = (\sigma_{\text{target}} = 15\%/19.92\%) \cdot w^t = (0.753) \cdot (0.47, -0.27, -0.23, -0.50, 0.53) = (0.35, -0.20, -0.17, -0.38, 0.40)$.

The whole process described in this section can be summarised in the following composed mapping:

> from the *predicted pairs weights* $a^t = (0.80, 0.40, -0.90) \mapsto$ *predicted stock weights* $w^t = (0.80, -0.48, -0.41, -0.90, +0.90) \mapsto$ *volatility-scaled portfolio weights* $w'^t = (0.35, -0.20, -0.17, -0.38, 0.40)$,

where each stock weights corresponds to stocks ADBE, ADSK, INTU, CTXS and ORCL.

Importantly, note that the *weights normalisation* and *position sizing* processes *kept the same nature as the predicted allocation.* This holds true with respect to *which* stocks we should bet against (short) and in favour (long), as well as to its *relative proportion.* To make it clear, the model was more pessimistic about ADSK (w^t_2=-0.48) than INTU (w^t_3=-0.41). In other words, for each 100 pound sold of INTU we sell 117 pounds of ADSK (in proportion $w^t_2/w^t_3 = -0.48/-0.41 = 1.17$). Indeed, after sizing the portfolio allocations, the level of pessimism initially reflected in the model allocation remained the same since $w'^t_2/w'^t_3 = -0.20/-0.17 = 1.17$, and the same applies to all allocations within the long and short positions.

5.5 Experimental results and discussions

5.5.1 Training setup

In our experiments, to guard against overfitting, we completely separated data from 2016 to 2017 for the test set. Unless otherwise noted, results are reported on this "unseen" (test) set. The remaining data (2007 to 2015) is further split into training and validation sets, with 15% for validation and 85% for training. The daily portfolio volatility is estimated using a sliding window of 252 days and this estimation employed to adjust the size of each position, as described in Section 5.4.4. The network weights of the Deep RL networks are updated in order to maximise the episode reward and for the SL network, as is usually the case, to minimise the cross-entropy loss. We use mini-batch SGD [27] with Adam [105] optimiser to update the weights of all networks.

Each of the two Deep RL networks and the SL network were trained independently and the network hyperparameters tuned using grid search – the best hyperparameters and its respective search space are reported in Appendix D.1.

Below, we provide additional hyperparameters related to the Deep RL training. These hyperparameters were not tuned, but set to typical values suggested in the literature.

Actions Exploration In order to increase exploration, it is common to add noise to the action selection process of Deep RL algorithms during training. For the DQN we applied an ϵ-greedy policy [131], meaning that with $\epsilon = 10\%$ a random (discrete) action was selected, rather than the action that maximised the Q value. For the DDPG the noise was added directly to the actor policy [115]. We selected an action $a^t = \mu(s|\theta^\mu) + \mathcal{N}$, where $\mathcal{N}$ was a sample from an Ornstein-Uhlenbeck process [187] with $\theta = 0.15$ and $\sigma = 0.20$. This process is mean-reverting and generates actions that are temporarily correlated. Similar to the control systems experiments [115], here we use the same process for action exploration, because it avoids large changes to the portfolio weights.

Replay buffer warm-up In line with previous work [131, 115], before the network weights were updated we populated the replay buffer with 50,000 experiences (one per step) taken from random actions, and also set the the total buffer size to 1,000,000. Both values were set following [131]. We observe that during training, new experiences were added to the buffer and network weights updated from a batch of experiences, which were replayed from the buffer. Once buffer capacity was reached, the oldest experiences were dropped in a First In First Out (FIFO) process.

Target network update frequency. As mentioned in Section 5.3.1, we make use of target networks with no trainable weights to stabilise training. For the DQN we used a hard weight update at a 10,000 steps frequency, when the trainable network weights were copied to the target network [131]. The DDPG uses soft updates, where the target network weights are updated at each step, as denoted in Equation (5.8). We set the update parameter as $\tau = 0.001$, as suggested in [115].

Discount factor (γ) As above, for all experiments, we set the discount factor to the typical value of $\gamma = 0.99$.

5.5.2 State space features

As denoted in Equation (5.13), at a given time t, the agent observes the pair history h_i^t and its respective sector, represented by the sector one-hot vector I_i. To this end, we represent the pair history in terms of the regression residuals ϵ_i^t. As defined in Section 5.4.1, this residual series measures the divergence between stocks of a given pair. Thus, when the prices diverge in the short-term they are expected to converge in the long-term.

Moreover, we also add features in order to measure the speed of the divergence, as measured by the difference between residuals in time using periods of 1 day, 1 month ($\sim$ 20 trading days) and one year ($\sim$ 252 trading days). That is, the history of pair i at time t is represented by matrix

$$h_i^t = \left[x_{t-T+1}^j, \cdots, x_t^j\right]^\intercal \in \mathbb{R}^{T\times 4}, \tag{5.17}$$

where the *input feature vector* x_i^t is denoted by

$$x_i^t = \left[\epsilon_i^t \,,\, \epsilon_i^t - \epsilon_i^{t-1} \,,\, \epsilon_i^t - \epsilon_i^{t-20} \,,\, \epsilon_i^t - \epsilon_i^{t-252}\right] \tag{5.18}$$

Features normalisation. The four features above are standardised (z-score) independently over time, i.e. $x \leftarrow \frac{x-\mu}{\sigma}$. In order to avoid any training data leakage, we use the same protocol employed to calculate the β coefficients in Section 5.4.1. That is, compute the standardisation parameters μ (mean) and σ (standard deviation) using only data from the training set, which are used to transform, i.e. standardise, both training and validation and test data. Thus, effectively, the test data remains "unseen" for the parameters computation.

5.5.3 Evaluation – Models and baselines

We compare the performance of the portfolios based on Deep RL models with two baselines: BAH portfolio and SL model. The process used to obtain the vector of stock allocations w^t from the vector of pair allocations a^t is detailed in Section 5.4.4. Below, we describe the two Deep RL modes (DQN policy and DDPG policy), and, subsequently, the two baselines:

- DDPG policy: A long and short portfolio based on the DDPG algorithm. It uses the neural network architecture depicted in Figure 5.3. All details on the decision making process are described in Section 5.4.2
- DQN policy: A long and short portfolio based on the DQN algorithm. It uses the neural network architecture depicted in Figure 5.2. All details on the decision making process are described in Section 5.4.2 .
- SL: Different from the DRL models above, this baseline long and short portfolio relies on a history-prediction-actions pipeline. It uses the neural network architecture depicted in Figure 5.4. All details on the SL decision making process are described in Section 5.4.3.
- BAH: A long only portfolio that allocates the same amount of wealth among all stocks. Details on this baseline portfolio are described in Section 2.2.2.

5.5.4 Evaluation – Portfolio analysis and Metrics

After training, we evaluate all portfolios using the following metrics (as defined in Table 5.2): Sharpe ratio, MAR ratio (both *risk-adjusted* metrics), CAGR, Mean, Std, Min, MDD.

Metric	Formula	Explanation
% Mean	$\mu(S) = \frac{1}{N}\sum_{t=1}^{N} r_t$	Average of the (percentage) return samples $S = \{r_1, r_2, \cdots, r_N\}$ (one sample per day)
% SD	$\sigma(S) = \sqrt{\frac{\sum_{i=1}^{N}(r_t-\mu)^2}{N-1}}$	Standard deviation of N (percentage) daily return samples. This metric is also called *volatility.*
% Min	MIN(S)	Worst (or minimum) daily (percentage) return, where $S = \{r_1, r_2, \cdots, r_N\}$.
CAGR	$(\prod_{t=1}^{N}(1+r_t))^{(252/N)}-1$	Compound Annual Growth Rate (CAGR) converts the performance to yearly.
MDD	$-\min_{\tau\in(0;N)}(\min_{t'\in(0;\tau)} R(t',\tau))$, where $R(t',\tau)$ is the compounded return between t' and τ	Maximum Drawdown (MDD) is a way to quantify risk, and is defined as *the worst loss among successive declines from peaks to troughs.*
Sharpe ratio† [163]	$\left[\frac{\mu(S)-r_f}{\sigma(S)}\right]\cdot\sqrt{252}$	Risk-adjusted performance metric. The denominator measures risk and the numerator return (profitability). We discount the return by the risk free r_f to account for the opportunity cost of not holding a risk free government bond.
MAR ratio	$\frac{\text{CAGR}}{\text{MDD}}$	Managed Account Reports (MAR) ratio is a risk-adjusted performance metric, as in the Sharpe ratio above. However, the risk part (denominator) is assessed based on the Maximum Drawdown (MDD) (defined above). The numerator quantifies the profitability.

Table 5.2: **Evaluation metrics definition.** N is the umber of samples. † we use the US 1-month T-Bill return (debt) for the risk free rate r_f, as indicated in the Fama-French data library.

5.5.5 Global Portfolio Results

Before we report our results, we expect the long and short portfolios to perform better than the BAH portfolio, since in the long and short portfolios the allocations are actively managed, they change over time, as opposed to the passive BAH portfolio, which equally buys all stocks.

In addition, in the long and short portfolios the short positions can hedge against periods when the market is going down. Thus, the volatility of the long and short portfolios is expected to be lower compared to the BAH (long only) portfolio. This allows more exposure to the market (for the same level of target volatility) and potentially more gains.

We also *expect the Deep RL models (DDPG and DQN), to perform better than the SL model.* This is because we optimise Deep RL models based on the future discounted reward and the learning process is based on a direct mapping from history to actions; while the SL model uses a pipeline and are optimised based only on the "immediate" pair return.

Finally, because in the DDPG model the actions (pair weights) are continuous, it offers more flexibility than the DQN, with discrete Buy (100%), Hold (0%) or Sell (-100%) actions. In fact, if the DQN actions are optimal, the DDPG model can learn the same "full" (100%) allocation of the DQN model. Thus, having more flexibility, *we expect a superior performance from the DDPG compared to the DQN.*

Table 5.3 shows the experimental results for a portfolio with $N_s = 205$ constituent stocks, which was built considering the whole universe of $N_p = 233$ stock pairs. The first column shows the models for the following portfolios (top-bottom): 3 long and short portfolios, where the first two are based on Deep RL (DDPG policy and DQN policy) and the last on Supervised Learning (SL), followed by the (long only) BAH portfolio. The BAH portfolio has the same constituent stocks as the other portfolios.

Based on experimental results we have the following findings:

- The volatility (or standard deviation – SD) of all portfolios is *very close to the target volatility* $\sigma_{\text{target}} = 10\%$ annual. Precisely, $10.163\% \pm 0.23\%$ annually or $0.640\% \pm 0.015\%$ daily. Note that this volatility is the *realised volatility* during the test period. Since we adjusted the volatility (ex-ante) to target 10%, this result indicates that our *position sizing* component is effective in limiting the risk to its target, an investor-led level of risk, which was set to 10% for evaluation purposes. We can see that on average we underestimated the volatility, i.e. the portfolio has more exposure to than it should, since the realised portfolio volatility of 10.163% is on average larger than the target. However, we missed the 10% target by a very small margin of 0.163% (annual). Finally, because all models have a similarly realised volatility, we can directly compare the results across all absolute return metrics, namely CAGR, Mean, Min and Max, and DD.

- In general, we can see that the pair trading strategy, which the long and short portfolios are based on, is profitable. More specifically, the best model (Deep RL with continuous actions/allocations) has an annual (compounded) return of 29.016% being almost double the BAH portfolio return (15.599%). Similar results hold for the Sharpe ratio (2.599 against 1.528), which is the most widely used metric to measure risk-adjusted returns.

- The Deep RL models (DDPG policy and DQN policy) perform better than the SL on all metrics. In fact, the Sharpe ratio of the SL model is slightly better than the BAH portfolio (1.644 against 1.528). These results mean that, adjusted for risk, the SL portfolio, which *actively manages* a portfolio of long and short positions, has a performance on par with a naive BAH strategy which *passively buys and holds* all stocks.

- Even though the DDPG policy has the best absolute performance and the risk-adjusted Sharpe ratio, its MAR ratio is slightly worse than the SL (1.874 against 1.898). This is because the DDPG policy MDD (i.e. maximum consecutive loss from a peak to a trough) of -15.484% is the largest among the long and short portfolios and approx. 50% higher than the SL model. In other words, the best CAGR of the DDPG model cannot compensate for the drawdown risk. Note that if the same risk is assessed in terms of volatility, as in the Sharpe ratio, the DDPG still stands as the best model.

- The period under evaluation is marked by optimism. This can be seen in the BAH portfolio performance of 15.599% per year (CAGR). The S&P500 index, which is also a proxy of overall market performance, raised approx. 14.72% (CAGR) per year (not reported in the table), with gains similar to the BAH index. To put the gains into context, over the period from 2016 to early 2018 the market was driven by the U.S. campaign and election of Donald Trump, and its positive impact on tax cuts and pro-business policies.

Portfolio	Sharpe ratio	MAR ratio	CAGR	Mean	SD	Min	MDD
DDPG policy	**2.599**	1.874	**29.016**	**0.104**	**0.633**	**-2.405**	-15.484
DQN policy	2.328	**3.431**	26.647	0.096	0.657	-2.507	**-7.767**
SL†	1.644	1.898	17.797	0.067	0.651	-3.028	-9.375
BAH††	1.528	0.867	15.599	0.060	0.620	-4.315	-18.003

Table 5.3: **Portfolio evaluation**.

Overall, our results are in line with previous studies [204, 71, 50] where experimental results also indicate that the pair trading strategy generates consistency in profits.

Maximum Drawdown (MDD) – Additional Analysis

Before we analyse the drawdown results, it is worth understanding the different risk aspects measured by the Sharpe and MAR ratios. The Sharpe ratio is a risk-adjusted metric that contrasts the mean return of an investment strategy with its volatility. Although the Sharpe ratio is the most widely-used metric for portfolio performance evaluation, it does not capture all risks involved in an investment. This is because an investment strategy with very low volatility can present long periods of consecutive losses. This type of risk is precisely measured by the MAR ratio metric, which adjusts the yearly gain (CAGR) by the Maximum Drawdown (MDD).

In order to have a better picture of a portfolio's losing streak, it is common to make use of its *Equity curve*, which represents the cumulative gains for a given initial investment.

Figure 5.5a shows the Equity curve for \$1 invested in each of the long and short portfolios (DDPG policy, DQN policy and SL), along with the BAH (long only) portfolio. For each portfolio, the down triangle represents the starting point of the maximum drawdown (MDD), and the up triangle marks the period to recover to the previous peakThe triangles can be tracked using the dashed lines.

The MDD of the DDPG policy portfolio starts around Feb-2017, reaches a low in Jul-2017, and only recovers to its peak in Dec-2018 (precisely, 216 days $\sim$ 11 months to recover to its peak in 23-Jan-2017). Also, the MDD period of the DDPG policy is larger than the SL, which only takes 49 days to recover from the peak 25-Apr-2017, and also larger than the DDPG policy, which takes 47 days to recover.

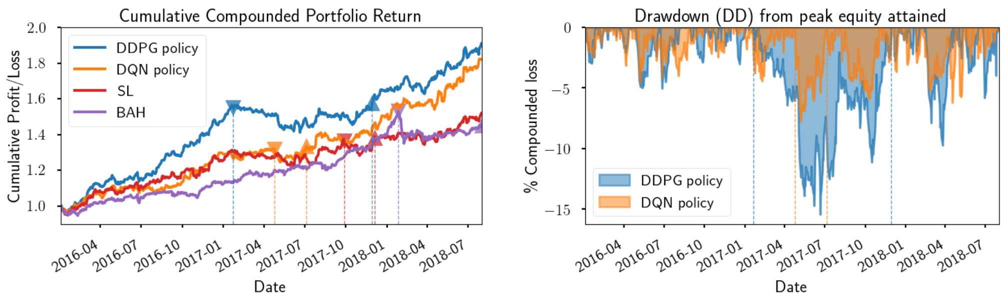
We can see that the BAH portfolio has the most abrupt drawdown ($\sim$ 18%), which has not recovered from its peak during the evaluation period and also stands as the worst portfolio in terms of the drawdown profile.

Figure 5.5b shows the complete drawdown (DD) history, where, to avoid cluttering, we only show the DD of the Deep RL portfolios. In this figure, each time a new high (peak) in the Equity curve is attained the DD value remains zero, and the DD from peak equity is computed w.r.t the most recent peak (as above, the dashed lines mark the MDD period). The DD time series reveals one important behaviour. We can see that the drawdowns have the opposite behaviours over some intervals. To make it clear, during the two months from Aug-2106 to Sept-2016 the DD of the DQN policy (portfolio) reached 4% DD, and this could be potentially offset by the positive performance of the DDPG policy during the same period. (Note that the DDPG values are zero, which means new highs). Similar DD behaviour can be also found from Jan-2018 onwards. Overall, this compensation is pronounced during the MDD period, apart from Sept-2017 to Dec-2017; even though the MDD of the DQN policy occurred within the MDD interval of the DDPG policy.

Because the DD are not in sync we could potentially ameliorate the DD profile of the DDPG policy, by combining both portfolios into an ensemble of portfolios. In fact, the correlation between the DDPG policy and DQN policy portfolios is 0.33, which goes in favour of a new ensemble model based on both predictions.

Null Portfolio – Random (uninformed) Policy

In this section, we analyse whether the performance of the long and short portfolios can be attributed to chance. In other words, does a *completely uninformed policy* lead to the same level of performance? To this end, we generated the *Null portfolio*, where the pair actions a_i^t were randomly sampled from the $[-100\%, +100\%]$ interval for each day of the test period. Importantly, once the pair allocations is obtained, we



(a) **Equity Curve** (b) **Drawdown (DD)**

Figure 5.5: **Equity curve and Drawdown**. *Left:* The portfolio cumulative compounded return for one dollar ($1) invested in each of the models under analysis. The dashed lines shows the period of maximum drawdown (MDD). *Right:* Drawdown (DD) series of Deep RL models (i.e. DDPG policy and DQN policy). This series is built in the following way: Each time a new high of cumulative gains is attained, according to the equity curve on the left side, the DD value is zero. All losses are in percentage points and computed w.r.t the most recent peak.

applied exactly the same steps of the Risk Management component described in . In other words, the Null portfolio has the *same number of constituent stocks* $N_s = 205$ as the long and short portfolios, and passes through the *same weight normalisation* (self-financing portfolio) and *volatility targeting* steps. Here, we use the Sharpe-ratio metric for comparison purposes, a choice grounded in its wide use for risk-adjusted portfolio evaluation.

Figure 5.6 shows the Sharpe ratio distribution (in blue) of 10,000 samples (i.e. trials) of the Null portfolio, which takes decisions (weights) randomly. The dotted line (blue) represents a Kernel-Density estimation (KDE) using Gaussian kernels, where we use the "rule of thumb" (Scott's rule) [162] to compute the estimator bandwidth. This Gaussian estimation is employed to calculate the 95% and 99% Confidence Intervals (CI) (vertical green dotted lines) of the Null (portfolio) hypothesis. More precisely, the integral over the intervals $[-1.24, 1.24]$ and $[-1.63, 1.63]$ sums to 0.95 and 0.99, respectively. The vertical dashed lines (red, orange, and blue) mark the Sharpe ratio results of the long and short portfolios, with values DDPG policy = 2.599, DQN policy = 2.328, SL = 1.644.

The Sharpe ratio of all long and short portfolios is greater than 1.63, and statistically significant ($p < 0.01$). Thus, *we reject the null hypothesis that the long and short portfolios are uninformed or generated by a random policy.* However, the result for the SL model endorses our previous findings. In other words, *compared to the (Null) portfolio, with random weights, the Sharpe ratio of SL model is relatively low.* On the other hand, the Deep RL models are more robust, i.e. have a lower probability of being attributed to chance.

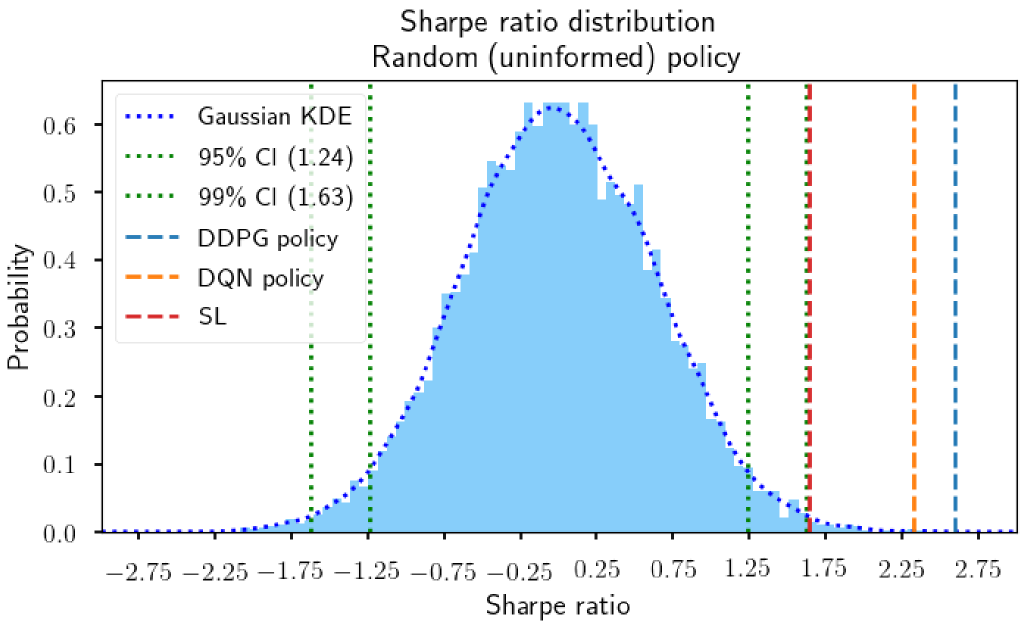


Figure 5.6: **Shape ratio distribution – Random policy**. The histogram of 10,000 allocations sampled from a uniform distribution in the $[-1.0, 1.0]$ interval. The dashed line is fitted using a Gaussian Kernel Density Estimation (KDE). For a given policy, the Sharpe ratio must be higher than 1.24 (1.63), orange (red) vertical line, in order to accept the statistical hypothesis that the policy is not random with 95% (99%) confidence.

Performance attribution analysis

In the last section, we saw that the performance of the long short portfolios under evaluation was significant, in the sense that the risk-adjusted Sharpe ratio has a very low probability of being attributed to chance. Here, we address another common question that emerges in portfolio evaluation, whether the portfolio performance can be attributed to *common risk factors*.

In a performance attribution study, the return of the investment strategy under analysis is regressed on the return of widely-known factors. Thus, the regression intercept α measures the portfolio return after controlling for risk factors, and the regression coefficient β measures exposure to the factors. In other words, if α is not statistically significant, the portfolio is of little value in terms of its active investment. This happens because the same portfolio return can be attained by having passive exposure to the factors.

We use the FF3 factor specification to contrast the performance of our portfolios against factors that are widely known to earn profits. These factors are explained in detail in Section 2.2.4. The first factor captures excess market return, i.e. the market return (R_M) minus the risk-free return rate (R_f), where the market return proxies as the value weighted return on all stocks on all NYSE, AMEX and NASDAQ stocks.[10] The return of the remaining two factors are computed using long and short portfolios, where, as in our study, the net investment is zero.

The High Minus Low (HML) factor, captures the returns of *value investing.* The portfolio is formed by buying undervalued stocks and selling-short overvalued stocks. Here, "value" is measure by the book-to-market ratio, which is calculated by dividing the company's accounting book value by the market capitalisation value. In other words, a stock with a low book-to-market ratio compared to other stocks has a market capitalisation that is not justified by its (accounting) book value, and as such, is overvalued. Similarly, a stock with high ratio compared to its peer's has the potential to appreciate based on the level of its book value.

Finally, the third *Small minus Big* (SMB) factor captures the returns of *size investing*, where stocks with a small market capitalisation are bought and and big capitalisation stocks are sold short.

Table 5.4 summarises the regression results of the three long and short portfolio returns (DDPG policy, DQN policy and SL), and of the BAH (long only) portfolio return on the three Fama-French factors, with p-values in parenthesis. The portfolios are ordered (descending) by α. As described above, the regression coefficients measure the exposure (or sensitivity) of each portfolio to the following factors:

1. Excess Market return ($R_M - r_f$), i.e. the market return discounted by the return of risk free interest rate
2. Value investing (HML)
3. Small-cap(italisation) investing.

As expected, the alpha (performance) of the (long only) BAH portfolio is not statistically significant and there is strong $R^2 = 0.874$ indications that portfolio return variability can be largely explained by the three factor returns. In particular, by its heavy exposure to the market ($\beta_{R_M - R_f} = 0.75$). Some exposures to risk, as measured by β, are statistically significant for the long and short portfolios, specifically, a higher exposure to value investing compared to the other two factors. However, after controlling for exposure to risk factors, the alpha (performance) is statistically significant for all long and short portfolios at the following levels: DDPG policy ($p < 0.001$), DQN policy ($p < 0.01$) and SL ($p < 0.05$). Moreover, the low R^2 value compared to the (long only) BAH portfolio, provides evidence that the variability in the long and short portfolio returns cannot be explained by any of the risk factors.

Overall, only a small portion of the excess return of pair trading can be attributed to their exposure to the three risk factors. More specifically, after controlling for risk exposure as measured by alpha, the best long and short portfolio (DDPG policy)

[10]NYSE, AMEX and NASDAQ are the main stock exchanges in the United States. Here, value-weight means that the weights are proportional to the market capitalisation value of each stock, which is computed for each stock by multiplying the number of outstanding shares by its price. Note that this portfolio assesses the risk of buying stocks, and because the portfolio is highly diversified its returns work as a proxy for stock market behaviour.

Portfolio	α	$\beta_{R_M-R_f}$	β_{SMB}	β_{HML}	R^2
DDPG policy	**0.0951** (0.000)	0.0722 (0.016)	0.1824 (0.000)	0.5004 (0.000)	0.203
DQN policy	0.0865 (0.001)	0.0758 (0.024)	0.0249 (0.624)	0.3128 (0.000)	0.073
SL†	0.0579 (0.016)	0.1124 (0.000)	0.1492 (0.002)	0.4318 (0.000)	0.158
BAH††	0.0083 (0.33)	0.7513 (0.000)	0.0275 (0.118)	0.1122 (0.000)	0.874

Table 5.4: **Portfolio exposure to common risk factors**. We regress the daily return of each portfolio on the return of FF3 factors [62, 63]. All computations are performed using the factors return data in Fama-French website. We have the regression intercept α (first column) followed by 3 regression coefficients β's and Coefficient of determination R^2 in the last column. The regression p-values are in parentheses. Ideally, a portfolio is expected to have a statistically significant α after controlling for underlying portfolio risk factors, as measured by β's. The three risk factors are: 1) Excess Market risk ($R_M - R_f$, where R_f is the risk free rate) 2) Small (market capitalisation) minus Big (SMB) and 3) High (book-to-market ratio) minus Low (HML).

has a daily return of 0.0951% or 26.21% annually. This performance is lower than the daily raw return of 0.0104% (as reported in Table 5.3) by only 2.2% per year. This is followed by the DQN policy and the SL with annual gains of 21.798% and 14.591%, respectively.

Compared with previous studies of the pair trading strategy, Gatev et al. [71] reports annualised profits after controlling for risk exposure of 11% per year, which are also statistically significant. This result is more in line with our SL portfolio performance of 14.59% per year. However, a direct comparison of profitability is not possible since his evaluation period dates back to 2006. Using a more recent evaluation period (from 2005 to 2012) Caldeira and Moura [33] reports annual profits of 16.38%, but for stocks traded in an emerging market (Brazil). Again, this makes the comparison difficult. Nontheless, as in our study, no correlation was found between the returns of the pair trading strategy and the the market. In particular, the study does not analyse the strategy returns in the light of the value (HML) and size (SML) investment returns.

Although we cannot directly contrast our results with previous studies, as mentioned they indicate a performance close to the SL portfolio, and according to our experiments, the performance of the pair trading strategy can be significantly improved by employing Deep RL techniques, compared to standard Supervised Learning (SL).

5.5.6 Sector Portfolio Results

In this section we evaluate the pair trading strategy for a hypothetical investor who has a preference for a given sector. Here, each *sector's* long and short portfolio has a different number of stocks (as in Table 5.1). For all sectors, we keep the same level of risk tolerance $\sigma_{\text{target}} = 10\%$ as our *global* long and short portfolio, which considers the full universe of $N_s = 205$ stocks, with results reported in Table 5.3. Moreover, each sector has net investment zero, i.e. the sum of all allocations among the sector constituent stocks is zero, i.e. we follow the same procedure used to report our global long and short portfolio results. Finally, for the analyses across sectors we consider the weights predicted by our best model, the DDPG policy.

Table 5.5 shows the performance of the sector long and short portfolios along with the mean return and volatility of the (long-only) sector BAH portfolio (in the last two columns). Here, to make comparisons easier, we annualised all mean and SD metrics. Similar to the global portfolio analysis, the BAH portfolio measures the performance of a strategy that buys the same amount of wealth for the sector constituent stocks and serves as a proxy for (passive) investment in the sector.[11].

Sector	Sharpe ratio	MAR ratio	CAGR	Mean	SD	Min	MDD	Mean ann. BAH	Vol. ann. BAH
Financial Services	1.68	1.08	19.05	18.05	10.72	-4.26	-17.59	22.47	17.53
Healthcare	1.42	0.77	15.46	14.95	10.53	-3.53	-20.09	11.75	13.53
Industrials	1.29	1.16	13.19	12.91	9.97	-2.25	-11.33	24.14	12.83
Basic Materials	1.25	1.86	13.27	13.02	10.41	-2.43	-7.13	27.26	16.59
Energy	0.96	0.75	9.56	9.65	10.02	-2.83	-12.76	18.58	25.82
Consumer Defensive	0.86	0.85	8.57	8.75	10.21	-2.89	-10.05	9.87	10.74
Technology	0.83	0.43	8.09	8.28	9.95	-4.22	-18.87	30.11	15.45
Real Estate	0.82	0.67	8.25	8.48	10.35	-3.38	-12.27	6.81	14.23
Communication Services	0.71	0.38	7.16	7.48	10.56	-3.26	-18.86	11.96	13.77
Utilities	0.54	0.40	5.35	5.81	10.80	-4.76	-13.50	14.00	13.60
Consumer Cyclical	0.34	0.25	3.06	3.57	10.54	-6.10	-12.17	14.80	13.45

Table 5.5: **Portfolio daily return statistics per sector**. All results are normalised by adjusting the expected portfolio volatility to 10%/year.

Based on the BAH portfolio metrics, we can see that our study covers sectors with different characteristics. On one hand, we have the energy sector (annual volatility of 25.82%), whose stock prices are driven by highly volatile commodity prices. On the other hand, we have the Consumer Defensive sector with the lowest annual volatility of all sectors (10.74%). The Consumer Defensive sector includes companies that manufacture food and personal care products, such as Colgate-Palmolive and Procter & Gamble. The sector's low volatility is attributed to the fact that these goods are essential, which makes the sector less sensitive to the ups and downs of the economy. We can also see that the sectors have different performances over the period, where the annual mean return of the top performing sector (Technology at 30.11%) is more than four times higher than the worst performer (Real state at 6.81%).

[11] As before, note that passive investment is in contrast with the active management of our long and short portfolios, where the portfolio weights are learnt using historical data and changes over time.

Our long and short portfolio results also present a large variability. In particular, we can see that the Sharpe ratio ranges from 1.68 (Financial services) to 0.34 (Consumer Cyclical). A natural question that emerges is whether this variability can be attributed to *specific* characteristics of each sector. In order to test this hypothesis we regressed the long and short portfolio metrics on the mean and volatility of the BAH portfolio. For the regression analysis we considered the following long and short metrics: The (risk-adjusted) Sharpe ratio and the (absolute) mean return.

Table 5.6 shows the regression results for the four possible relationships. We can see that the coefficient of determination R^2 is very low for all regressions. This indicates that the variability across sectors of our long and short portfolio *cannot* be attributed to the sector performance. Moreover, the sensitivity to each sector performance, as measured by the bs coefficient, is not statistically different from zero. This result evidences that our model generalises well across the different regimes of volatility and return experienced by each sector, i.e. the model performance is robust to these changes. We credit this high *generalisation* to our architecture. To make it clear, we designed an architecture where a single agent (spread-trader) learns from the experiences of *all* stock pairs via our multi-pair experience buffer. When proposing this architecture, our motivation was to encourage generalisation; we aimed to learn the "real" features that make short-term pair distortions converge in the long-term, which are common to all sectors. In other words, the fact that the results are not impacted by the specific characteristics of each sector indicates that the model effectively learns the real features that lead to profitability, and this happens regardless of the sector.

Regression of DDPG policy portfolio on BAH portfolio	a	b	R^2
Mean ann. (DDPG policy) $\sim a + b \cdot$ Mean ann. (BAH)	6.78 (0.07)	0.19 (0.30)	0.12
Mean ann. (DDPG policy) $\sim a + b \cdot$ Vol. ann. (BAH)	6.97 (0.24)	0.20 (0.57)	0.04
Sharpe ratio (DDPG policy) $\sim a + b \cdot$ Mean ann. (BAH)	0.64 (0.07)	0.02 (0.26)	0.14
Sharpe ratio (DDPG policy) $\sim a + b \cdot$ Vol. ann. (BAH)	0.66 (0.23)	0.02 (0.55)	0.04

Table 5.6: **Best long and short portfolio (DDPG policy) sensitivity to sector mean return and volatility**.

Global versus sector portfolios

As reported in Table 5.5, in what concerns investor's choices for a particular sector, we are still confronted with the lacklustre performance of our long and short portfolio for some sectors. However, we interpret this result as evidence that *the pair trading strategy demands a high level of diversification to achieve a good performance.* In particular, we showed in Table 5.3 that if *all* sectors are taken into consideration, as in the *global* long and short portfolio, an outstanding performance can be achieved, with the best portfolio reaching a Sharpe ratio of 2.60; this value being much higher than the best sector Sharpe ratio of 1.68.

In what follows, we show that the reduction in risk achieved by our long and short portfolio is substantially higher than the BAH portfolio. Thus, an investor can largely benefit by allocating to multiple sectors.

In terms of portfolio management, a substantial risk reduction (as measured by the portfolio volatility) cannot be attained by simply increasing the number of assets in the portfolio. It is also necessary that these assets do not move in tandem. More precisely, a portfolio with weights $\{w_i, \cdots, w_N\}$ has volatility denoted by $\sigma_p = \sqrt{\sigma_w^\intercal \Sigma \sigma_w}$, where Σ is the $N \times N$ correlation matrix of the asset's returns and σ_w the column vector representing each asset contribution to the volatility, i.e. $[\sigma_w]_i = w_i \cdot \sigma_i$. Thus, the lower the correlation between the assets the stronger the diversification effect in terms of reducing the portfolio volatility.

Intuitively, we can interpret the formulation above in the following way: the less in synchrony the assets move, as measured by the correlation matrix, the less probable it is that all assets will contribute simultaneously with the same return sign, and as a consequence, the portfolio volatility is smoothed out.

Figure 5.7 shows the correlation matrix of sector returns[12] for both the BAH strategy (on the right) and our long and short portfolio (on the left). As reported on the top of the table, the average correlation (μ) of the BAH strategy returns (0.48 ± 0.19) is much higher than the correlation of our model returns (0.03 ± 0.06). Moreover, not only is the average correlation lower, but the decreases is generally pairwise. To exemplify, the highest correlation of the BAH strategy is between the Basic Materials and Industrial sectors[13] with a value of 0.84. This same correlation decreases to 0.0049 for our long and short portfolio and similar behaviour can be found between other sectors. As mentioned above, this means that the diversification effect (i.e. the volatility reduction), which can be obtained by combining sectors, is much higher for our long and short strategy than the BAH strategy.

We showed above that an investor can largely benefit by allocating in more than one sector. The exact performance metrics will depend on the stocks selected by the investor. Nevertheless, we provide additional analysis considering our global long and short portfolio, with performance metrics reported in Table 5.3.

Figure 5.8 shows the evolution of the global long and short portfolio weights over time, where sectors are ordered by Sharpe ratio. For each day we compute the sector weight by summing the weights of all constituent stocks and the dashed line of each sector represents a zero sector weight. Thus, if the graph is above the dashed line

[12]Note that because this matrix is symmetrical and the diagonal values equal one, we only show its the lower triangular values.

[13]This high correlation value is as expected, since both sectors are strongly dependent on economic activity, which makes their stock returns move in tandem.

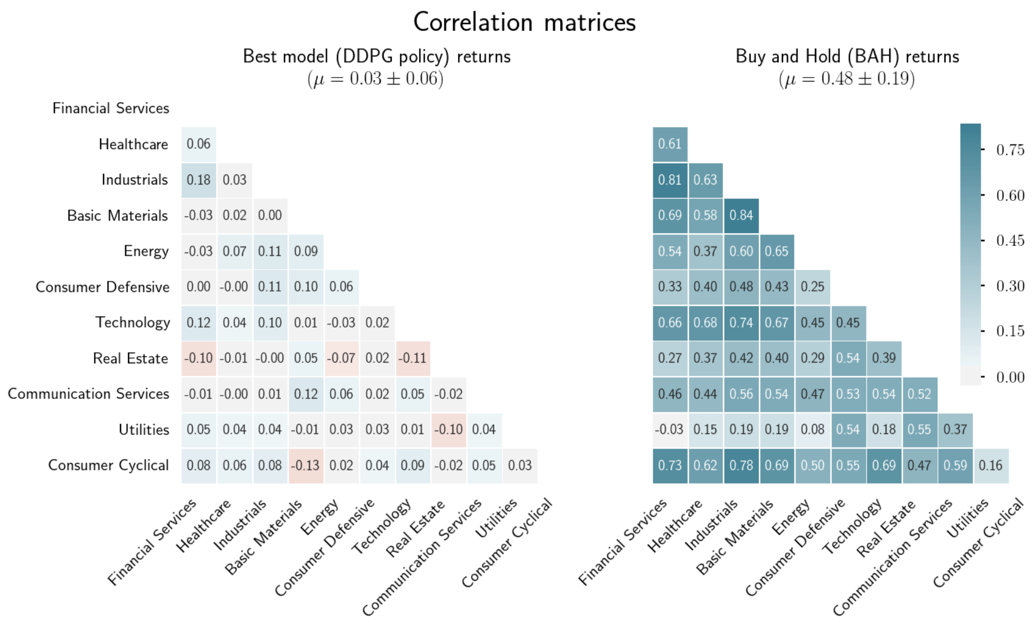


Figure 5.7: **Sector return pairwise correlation matrix: Comparison between our model and BAH**. The pairwise correlation matrix of sectorial portfolio returns. *Left:* long and short portfolio (DDPG policy). *Right:* BAH portfolio.

the sector contributes with long positions (buy) and below it with short positions (sell short).

Note that because the global portfolio is zero investment the sum of all weights is zero on a given day. Thus, the presence of short positions in one sector helps to protect the long positions in other sectors, and vice versa. This further corroborates the positive effect of diversifying to other sectors, as shown in the correlation matrix depicted in the left of

5.6 Conclusion

In this chapter, we investigate the pair trading strategy [71, 83]. This investment strategy relies on the idea of long-run equilibrium between a pair of stocks, and can be described by two independent steps: pair screening and decision-making. The concept of pair trading is relatively simple. Find two stocks whose prices tend to move together (the pair screening step). Then, in the decision-making step, if the stocks diverge from each other, a trading position is opened by selling short the stock that is overvalued and buying the stock that is undervalued. If history repeats itself, prices will converge to their long-run equilibrium and the investor profits from the short-term divergence.

Previous studies in pair trading [71, 50, 24, 51, 69, 31, 114, 33, 95, 99, 59, 199]) make use of a rule-based system in the decision-making step. More precisely, they use a fixed rule to trigger the pair positions based on the pair history (i.e. on the divergence from the long-term equilibrium).

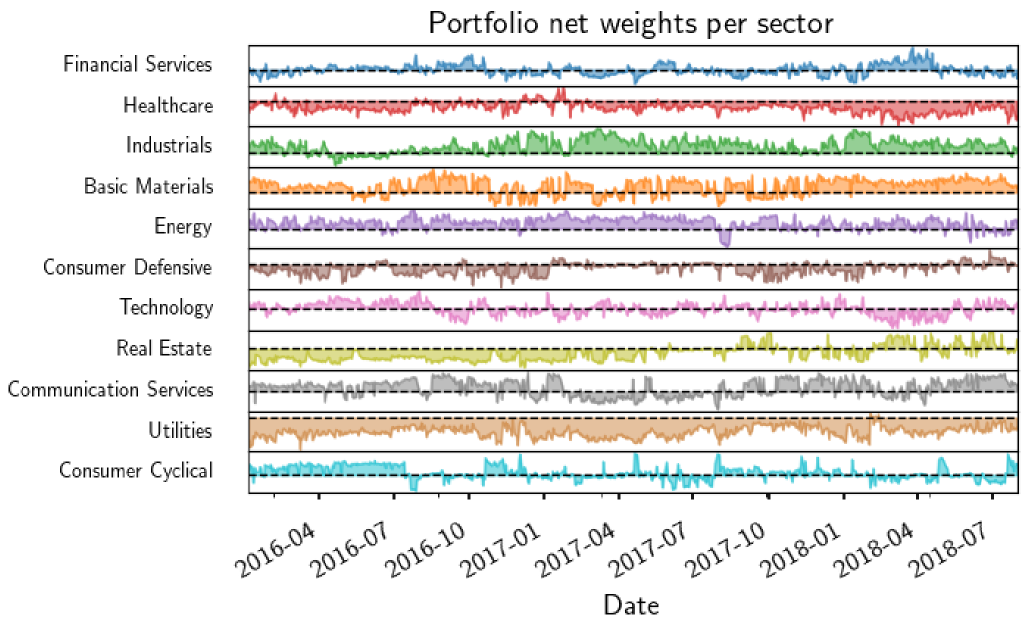


Figure 5.8: **Evolution of portfolio weights over time – Sector breakdown**. Plot showing how the weights allocated to each sector of long and short portfolio varies over time. The dashed line represents a zero allocation, where weight values above this line represent long positions, and below represent short positions. The portfolio is self-financing (or zero net investment), i.e. for each day the weights allocated to all sectors sum to zero.

We are the first to approach the pair trading strategy using model-free RL [173], where we learn a direct mapping from pair history to actions. Here, the actions represent the allocation, i.e. the percentage of wealth to invest in a given pair. In line with the studies mentioned above, we consider discrete actions, i.e. discrete allocations $\{\text{long} = 100\%, \text{neutral} = 0\%, \text{short} = -100\%\}$, but, we also extend to continuous actions, where the pair allocations is a value between minus and plus one. In order to cover both discrete and continuous actions, we employ Deep RL, which approximates the optimal actions using a neural network.

We introduce a novel framework, called *Investment Strategy with Investors' Preferences* (ISIP), which integrates the optimal allocations of our Deep RL models with a risk management component. This framework considers the individual preferences of each investor in terms of: (1) the level of volatility σ_{target} that he/she is comfortable with, i.e. the risk tolerance; and, (2) The N stocks he/she is interested in trading, i.e. the portfolio constituent stocks. At a high-level, given the preferences as input variables, our framework outputs the optimal vector of allocations $w^t = (w_1^t, \cdots, w_N^t)$. Moreover, the nature of the pair trading strategy implies that these allocations can be long (positive) or short (negative), but the portfolio is self-financing, i.e. the sum of all weights is zero.

We employ a large dataset for the evaluation of the pair trading strategy. This dataset consists of the daily price series of 205 stocks, and covers a broad range of

11 sectors. We use the cointegration method at the pairs screening step, where this leads to 233 pairs.

We evaluate the Deep RL portfolios (DQN policy and DDPG policy) against the Buy and Hold (BAH) and Supervised Learning (SL) baseline portfolios. The Buy and Hold is straightforward. It is considered a passive investment strategy that buys all stocks in the portfolio, but allocates equally to all stocks. As in the Deep RL portfolios, the SL portfolio also has long and short positions and uses the same LSTM encoder as the Deep RL architectures. In order to convert the SL predictions into allocations, we use a stardard history-prediction-action pipeline. First, for each pair, we predict the next day direction $[\text{long} = [1, 0, 0], \text{neutral} = [0, 1, 0], \text{short} = [0, 0, 1]]$, where we consider the same three classes of the DQN policy. Then, we apply the widely used allocation rule $[\text{long} = 100\%, \text{neutral} = 0\%, \text{short} = -100\%]$. In simple terms, the SL portfolio can be seen as an optimisation based on the immediate reward, i.e. the next day prediction, where we apply a rule to convert these predictions into pair allocations. On the other hand, the Deep RL portfolios are optimised based on future rewards and do not rely on intermediate predictions, as in the SL portfolio.

In what follows we start by summarising our findings for the global portfolio and then for the sector portfolios.

The efficacy of our framework in keeping the portfolio volatility at the level pre-set by the investor can be seen by comparing the (ex-ante) target volatility with (ex-post) realised portfolio volatity. More specifically, by dynamically adjusting the allocations to an ex-ante volatility $\sigma_{\text{target}} = 10\%$, experiments show a portfolio volatility of 10.163%. That is, we miss the target volatility by a very small margin of 0.163% per year.

Similar to other studies, we found that the pair trading strategy is profitable and its profit is uncorrelated with the market movement. Nonetheless, it does exhibit sensitivity to the following self-financing portfolios: Small (market capitalisation) minus Big (SMB) and High(book-to-market ratio) minus Low (HML). These sensitivities were also found in previous studies [95, 71]. In addition, both SMB and HML strategies have returns and a Sharpe ratio inferior to our Deep RL portfolios.

Significantly, we show that an investor can largely benefit from using Deep RL for pair trading. The Deep RL portfolios perform better than SL and BAH portfolios for all metrics under analysis. The best performing portfolio (DDPG policy), whose allocations are optimised using continuous actions, has a relevant absolute compounded annual return of $\sim 29\%$ and a relevant Sharpe ratio of $\sim 2.60\%$. In contrast, the Sharpe ratio of the SL portfolio is slightly better than the BAH portfolio (1.644 against 1.528). In other words, the SL portfolio, which *actively manages* a portfolio of long and short positions, has a performance on par with the BAH strategy that just *passively buys* all stocks in the portfolio.

The portfolio that was optimised using continuous actions, namely DQN policy, has a Sharpe ratio inferior to the portfolio optimised using discrete actions, namely

DDPG policy. However, because the DDPG policy has a large drawdown, its MAR ratio (CAGR/Max. Draw down) is worse than the DQN policy. A further investigation shows that the DDPG policy takes longer to recover from drawdowns compared to the DQN policy; however, the drawdown of both portfolios happens at different periods of time. In fact, the correlation between the return of the DDPG portfolio and the DQN portfolio is relatively low (0.33). This indicates that both the Deep RL portfolios learn different characteristics of pair history and potentially, a portfolio that combines both strategies (i.e. a portfolio of portfolios) would benefit from both the low correlation and most importantly, the fact that the drawdowns are scattered.

In order to investigate whether the performance of our portfolios can be attributed to chance, we consider a random portfolio, where allocations are taken uniformly from the $[-100\%, +100\%]$ interval. Using the same stocks as in the global portfolio, we compute the confidence interval of the Sharpe ratio of the Random portfolio. Based on this experiment, we reject the hypothesis that the Sharpe ratio of our portfolios could be due to chance: DDPG policy ($p < 0.001$), DQN policy ($p < 0.01$) and SL ($p < 0.05$).

We evaluate the performance of the pair trading strategy across 11 sectors using the allocations predicted by the DDPG policy. Results show a high degree of variability in performance. In particular, the Financial Services present the best performance (Sharpe ratio = 1.68) and the Consumer Cyclical the worst (Sharpe ratio = 0.34). However, we show that this performance is not correlated with sector volatility or return.

Based on this result, we argue that the pair trading strategy demands a high degree of diversification to attain attractive gains. We show that the advantages of diversification for pair trading are much higher than for the BAH portfolio, since the average pairwise correlation between the two sectors is very low 0.03 ± 0.06.

Overall, the advantages of using the pair trading strategy strongly depends on the stocks the investor is interested in trading and her/his appetite for risk. However, by using our framework, different scenarios involving different stocks and risk tolerances can be analysed without having to retrain the Deep RL models.

Part IV

Conclusion and Future Work

Chapter 6

Conclusion

The goal of the research described in this thesis was to push forward the performance of existing solutions to problems in Risk Management and Quantitative Trading. The way we approached different problems in the financial industry and the research questions we addressed are as discussed below.

Widely used econometric models for daily volatility only take into account the stock price time series. In Chapter 3, we focus on improving daily volatility prediction by using alternative sources of data in the form of news headlines. The method we used to account for the effects of news and prices is multimodal deep leaning. We propose a multimodal neural network, the MHAN, to learn from two modalities, stock price time series (price modality) and news articles (textual modality), to predict daily stock price volatility. Previous studies used the content of 10-K reports. These are only released on a yearly basis and focus on predicting quarterly or yearly volatility. By contrast, our study is the first aiming to predict daily volatility, rather than quarterly or yearly, using news headlines as opposed to 10-K reports. Importantly, we designed the MHAN architecture in order to address two main challenges in modelling news articles, news relevance and novelty. A considerable amount of news is released on any given day and much of it is distracting. This challenge was addressed in this thesis by using a component of our MHAN architecture called News Relevance Attention (NRA). The concept of novelty refers to accounting for the temporal information of past news, and is addressed using the News Temporal Context component. By using the ablation method, we investigated the importance of different components of MHAN. Experiments clearly show that the attention mechanism of our NRA, which weights different news items based on content, performs better than previously proposed architectures that simply take the average effect of all news. We also investigated different methods to represent text, namely sentence encoders. These include sentence encoders that use pre-trained word embedding and those transferred from different NLP tasks. These tasks are text categorisations of financial news (RCV1) and natural language inference (SNLI). Despite the fact that RCV1 is a financial domain corpus, experiments shows that the sentence encoders trained on SNLI have superior performance; this result being in line with previous studies. Therefore, we were able to address our first research question:

Question 1: Can we improve the one-day ahead volatility prediction by

adding textual data? Mainly, how does it compare to well-established econometric models that only use stock price data?

Yes. Experiments show that the use of textual data, in particular news headlines, improves the volatility prediction. Importantly, the daily volatility predictions of our MHAN model perform better than the prevalent GARCH model. In addition, the superior performance of MHAN is consistent, where experiments show that MHAN outperforms GARCH across different sectors of the stock market.

In Chapter 4, we investigate graph neural networks, a method that can inject prior knowledge into representation learning. We propose the novel Graph Transformer Network (GTN) which can be generally employed in any problem where the data is represented in the form of a graph, e.g. citation networks and knowledge bases. In the context of trading strategies, we propose a market graph where each stock in the portfolio is represented by a node and the edges represent the presence or absence of relationship between stocks. By using the GTN and a market graph, we then investigate the impact of adding a relational bias to the model of the range trading strategy. To extend the existing studies, we evaluate the range trading strategy in a setting where the portfolio positions are dynamically adjusted to target a constant risk level. By using this setting and taking into account the influence of other stocks via a graph we are able to answer the following research question:

Question 2: In terms of risk-adjusted profitability, is there any advantage in biasing the range prediction model via a graph?

To address this question we first propose a sectoral graph which uses the prior that only stocks within the same sector have a relationship. We then contrast models trained using the sectoral graph with two other graphs: (1) no relationship between stocks, and (2) all stocks related to each other (fully-connected graph). We use the Friedman test to rank all models according to the Sharpe ratio, and results show that the GTN with a sectoral graph achieves the best rank. Finally, by applying statistical methods for multiple pairwise comparisons, we show that the top results achieved by the GTN with a sectoral graph are statistically significant ($p < 0.04$) compared to all other models, including models with no relation between stocks and a fully-connected graph.

Another important conclusion of the pairwise comparison method is that the GTN outperforms the current state-of-the-art Graph Attention Network (GAT) [190] for the range trading strategy. Overall, experiments for a portfolio with a large number of stocks corroborate previous findings [127, 193, 77] showing that the range trading strategy is profitable.

In the stock market the relationship between companies changes over time. An important feature of our GTN architecture is that this dynamic relationship between companies can be captured by the GTN, since the attention mechanism depends on the stock price history. The analysis of the attention weight matrix after training the range trading strategy allowed us answer to the next research question:

Question 3: How can the analysis of the attention weight matrix be applied to provide useful information for the financial services industry?

A detailed analysis of the attention weight matrix contributes to the topic of model interpretation in machine learning. The analysis demonstrates that the weights can be formulated as a directed graph, where the influence of stock j on stock i is different from the influence of stock i on stock j. The attention weights are also used to quantify the sensitivity of a given stock to its peers. Considering its practical implication for financial markets, we name this measure the Self-connection Relative Strength Index. High values of the index suggest that the stock is characterised by a low level of integration in the graph; therefore, it is less prone to price shocks caused by the influence of other stocks. The analysis of the index values demonstrates that stocks from the Real Estate and Financial sectors are prone to systemic risks, where a default in one company from these sectors can trigger a collapse of the entire sector or even the economy.

In Chapter 5 we investigate Deep RL in the context of the pair trading strategy. The pair trading strategy is based on the idea that some stocks have a long-run equilibrium. The strategy consists of two separate steps: pairs screening and decision making. The pair screening step identifies two stocks whose prices historically tend to move together. The decision making step involves entering into a trading position if the stocks diverge from each other by short selling the overvalued stock and buying the undervalued stock. The assumption is that history repeats itself and prices will converge to the long-run equilibrium.

One of the main motivations for using Deep RL is to avoid relying on a *history-prediction-action* pipeline to convert predictions into trading actions, which is typical of the SL setting. To the best of our knowledge, this thesis presents the first results of approaching the pair trading strategy using RL in general and Deep RL in particular. We train the Deep RL model to map pair history directly to trading actions. The actions refer to the portion of wealth to be invested in a given pair. We consider both discrete actions and continuous actions and evaluate them in the context of the pair trading strategy.

This thesis proposes flexible Deep RL architectures that are capable of accommodating a varying number of stocks with no need to re-train the Deep RL model. Moreover, we introduce a novel framework ISIP (Investment Strategy with Investors' Preferences) to integrate the optimal allocations of the Deep RL models with the risk management component. The framework considers the individual preference of each investor with regards to the accepted level of volatility (risk tolerance) and the list of stocks of interest (portfolio constituent stocks). Given the preferences, the ISIP framework outputs the optimal allocation by normalising all positions, such that the portfolio is self-financing, i.e. sum of all allocations is equal to zero. By using this framework, we simulate the performance for a large number of stocks and address the following research question:

Question 4: In the specific case of pairs trading, where trades take a long time to mature, does Reinforcement Learning provide better performance than Supervised Learning (SL)?

Yes. Experimental results show that both portfolios trained using Deep RL, i.e. DQN policy and DDPG policy portfolio, perform better than the SL portfolio. In fact, the Sharpe ratio of the SL portfolio is not relevantly better than the baseline BAH portfolio, which represents a strategy which buys all stocks in equal proportions. We attribute the inferior performance of the SL portfolio as due to it being viewed as optimisation based on the immediate reward only, while the Deep RL portfolios are optimised taking the future into account. In addition, they do not rely on the history-prediction-action pipeline approach.

The experimental results confirm the profitability of the pairs trading strategy and its profit is uncorrelated with market performance. However, the profitability of the strategy is sensitive to two Fama-French factors, market capitalisation (Small minus Big) and book-to-market ratio (High minus Low).

6.1 Future Work

In the future, we plan to make use of intraday prices to better evaluate the performance of our volatility models. Additionally, we plan to further extend our analysis to other stock market sectors.

We plan to explore the potential of the proposed GTN architecture for other datasets and tasks using relational data, such as knowledge base completion and molecule bonds prediction. Even though our experimental results are robust and promising, they are limited to the financial domain. The efficacy of GTN in other tasks needs to be investigated. Furthermore, we expect that using more dynamic graph structures, such as correlation-based structures, can improve the results compared to the fixed sectoral graph studied in this work. The sensitivity of profitability to multiple graph structures will be explored in future work.

We plan to investigate the impact of news headlines (textual data) on the profitability of the trading strategies studies in Chapter 4 and Chapter 5

6.2 Limitations

One limitation of our work is the fact that it relies on daily OHLC data. The range trading strategy risk-adjusted profitability can be enhanced with intraday data. This is due to two main facts. First, we could trade the range more than once per day, i.e. for days with persistent swings. Note we do not know how many potentially recurring, high and low hits we have during the day relying only on OHLC data. Second, by knowing the order of the trade, i.e. if the high or the low was hit first, we would be able to accurately adjust the portfolio weights to the desired level of volatility, without having to rely on any assumptions. Finally, we observe that when accounting for both facts mentioned above, the profitability reported in our experimental results is conservative.

Part V

Appendices

Appendix A

Appendix A

A.1 Extending the *Loughran and McDonald Financial Sentiment Words List* from 10-K Corporate Fillings using Social Media Texts

A.1.1 Introduction

The sentiment classification of texts is a Natural Language Processing task that has increasingly attracted the attention of the research community in recent years. Broadly speaking, we can group sentiment classification into two approaches: those employing supervised [140, 139, 75, 169] or semi-supervised machine learning methods [42, 206, 146], and those using unsupervised learning [185].

Lexicon-based sentiment classification is performed by retrieving information from *sentiment word lists* or a *sentiment lexicon*, i.e. a database of words with positive and negative annotations. The main challenge of this approach is to compile the word list without any time-consuming human intervention. In other words, the goal is to learn the sentiment words lists rather than compiling them manually. The techniques developed to build the sentiment word lists can be arranged into three broad categories: *Dictionary-based*, *Corpus-based* and *Emoticon-based*. The first method starts with a seed of initial words that contains at least one positive and one negative word. The seed is then bootstrapped, e.g. using WordNet [130] *synsets* [93, 86, 151]. The *Corpus-based* technique is similar to the *Dictionary-based* one; however, it attempts to bootstrap the seed using a domain specific corpus. This method largely exploits *grammatical coherence*[1] of a given language (see, for example, the early studies in [87] and posterior advancements in [104]). One of the main drawbacks of this method is the limited occurrence of linguistic conventions in a given corpus. Finally, the *Emoticon-based* methods are grounded on the fact that emotion icons (*Emoticons*), such as `;-)`, `:)` and `:-(` have the advantage of summarising feelings. Therefore, they are useful to automatically assign a sentiment label to a given text. This method is employed in [76] and in [44].

[1]Grammatical coherence can be understood as linguistic conventions on connectives such as *and*, *or* and *neither nor*. To illustrate, from the text "*The service is good and staff are friendly*" we could infer that "friendly" and "good" have the same sentiment connotation even without knowing *a priori* the sentiment of each word individually.

A large number of publicly available language resources for sentiment classification, e.g. *Sentiment140* [76], *Bing Liu Sentiment Lexicon* [119], *MPQA Sentiment Lexicon* [197], *Harvard Dictionary* (the *General Inquirer*) [171] and *VADER* [97], were built using the three fundamental methods discussed above.

Although these resources are effective for sentiment classification in the general context of customer reviews, they are of limited use for financial domain corpora, such as US 10-K/10-Q corporate fillings, conference press releases or social media content related to stock markets. For instance, as stressed in [121]: "Almost three-fourths of the words identified as negative by the widely used *Harvard Dictionary* are words typically not considered negative in financial contexts."

This work focuses on building a sentiment lexicon specifically for texts from the financial domain. Three main contributions are made to the existing literature. Firstly, we propose a novel sentiment lexicon for words in financial contexts. This sentiment lexicon was learnt from user posts on the *Yahoo Message Board*, applying a supervised learning approach. In this regard, our work is helpful to extend the manually annotated *Loughran and McDonald Financial Sentiment Dictionary* [121]. Secondly, the method we propose to build a sentiment lexicon from a text sentiment classifier can be used as a general method for similar problems, regardless of the corpus domain. Thirdly, we have made the sentiment annotated dataset used to build the sentiment lexicon publicly available as an additional language resource.

A.1.2 Financial Domain Dataset

Description and Characteristics

Until *Yahoo*'s recent acquisition by Verizon, the company provided a financial message board service covering a broad range of individual stocks. When discussing a given stock, users could annotate their posts with one of the following fixed five sentiment labels: *Buy*, *Strong Buy*, *Sell*, *Strong Sell* and *Neutral*.

Aiming to make use of this sentiment annotation, we collected raw HTML content from each stock message board. Then, we parsed this content extracting tags that contain relevant information. Finally, we converted the parsed HTML content into open JSON (JavaScript Object Notation) format. This step converted the unstructured message board content (HTML) into structured data (JSON).

In total, we collected 4.9GB of Python serialized JSON objects by sending web requests through 8 parallel processes during two consecutive weeks.[2] Messages published in 2014 and 2015 were collected for a list of 492 stocks.[3] Below, we show two samples from the JSON dataset for *IBM* and *Exxon Mobil* stocks (the field `message_sentiment` describes the label):

[2]We relied on data parallelisation techniques, where each process/thread took care of one stock independently.

[3]The list of stocks was compiled based on all constituents of Standard & Poor's 500 Index (S&P500) 2017. Subsequently, stocks with no messages were disregarded, reducing the initial universe of 500 stocks.

```
{'is_reply': True,
 'message_sentiment': 'Buy',
 'message_title': 'IBM profit machine slows; layoffs
    planned',
 'timestamp': 1366340436.652},

{'is_reply': False,
 'message_sentiment': 'Strong Sell',
 'message_title': "Bloomberg: Crude Oil Erases Advance on
    OPEC's Reduced Demand Forecast",
 'timestamp': 1421371595.252},
```

Based on the distribution of **POS**, **NEG** and **NEUTRAL** labels in Table A.1 we see that our data has a strong bias towards messages with a positive tone. We perceive this bias as a behavioural manifestation of the overconfidence and excessive optimism investors see in the stock markets in general, as described in [166].

Label	**Number of Samples**	**Percentage**
POS	46,981	63
NEG	20,610	28
NEUTRAL	7,050	9
total	74,641	100

Table A.1: Proposed Sentiment Lexicon dataset: Labels distribution.

A closer look at some random samples reveals a certain degree of noise in the user annotated labels. For example, the text: "*All the cards on the table today!*" is labelled as **POS**. However, without the label most annotators would probably consider the message neutral. We presume that this "labelling mismatch" happens because some message board users tend to mistaken the message's true connotation for their own judgement about the future performance of the company. That being said, potentially, the user that posted this message was betting the market would go up and not exactly the fact that, from a linguistic viewpoint, "*all cards on the table*" is an utterance with neutral sentiment.

Pre-Processing and Wrangling

Our pre-processing phase started by filtering out all messages with the following characteristics: duplicate title, without any sentiment annotation and reply messages. After this phase, we ended up with $74,641$ messages which were dumped separately to a JSON file for each stock ticker.

We then performed the following additional pre-processing steps:

1. A naive lexical normalisation to convert Out-Of-Vocabulary (OOV) words to their canonical form. Normalisation treats the following cases: repeated words

(e.g. convert from "going up up up" to "going up"), repeated symbols (e.g. convert from "AMAZING!!!!!" to "AMAZING!"). This task was pipelined and executed before the Part-Of-Speech (POS) tagging task.

2. Spell checking using *GNU Aspell*. Any words that were still not recognised were filtered out.

3. We ignored terms that appeared in less than three message titles.

Finally, we aggregated the messages of each stock into three classes. In the **POS** (positive) class, we grouped all messages originally labelled as *Buy* or *Strong Buy*. The **NEG** (negative) class received all *Sell* and *Strong Sell* messages. Finally, all residual messages were assigned to the class **NEUTRAL**.

A.1.3 Methodology

Document Representation

We used a sparse vector space model to represent each message in our dataset. However, we expanded each message into a base of Semantic Orientation[4] (SO), rather than the standard "bag of words" model, according to which the message is represented by the set of words or *n-gram* occurrences (weighted or not). Our sentiment classifier is thus trained on what might be called "bag of Semantic Orientation (SO)".

The motivation to use the "bag of Semantic Orientation (SO)" representation comes from two facts:

1. The SO keywords are Part-of-Speech (POS) tag patterns that work as good indicators of explicit opinions. For instance, the tag pattern `JJ` (adjective) + `NNS` (noun) + `<any tag>` extracts "economic concerns" from the message "*Stocks tank on global economic concerns*". Since we represent each text of our corpora in this SO base, we can, at inference time, predict the sentiment of each SO keyword separately in order to build our sentiment lexicon.

2. The SO tag patterns are handy to disregard messages that do not convey any connotation in our context of binary sentiment classification, i.e. only **POS** or **NEG** classes. In other words, the SO tag patterns constitute a simple algorithmic way to filter out texts without explicit polarity.

Semantic Orientation Tag Patterns Extraction

To build the SO base representing each text described above, we extract the same Part-of-Speech (POS) tag patterns proposed in [185]. Table A.2 replicates these tag patterns, and the respective *TGrep2* [155] expressions we used in our code.

The Part-of-Speech (POS) tagging is performed using the tagger proposed in [181].

[4]Semantic Orientation is a measure of subjectivity and opinion of a given text, see the early works of [138] and a more recent review in [175].

Tgrep 2 expression	POS Tag pattern
`(JJ.(NN|NNS))`	JJ + NN or NS
`(RB.(JJ!.(NN|NNS)))`	RB + JJ +not NN, not NNS
`(RBR.(JJ!.(NN|NNS)))`	RBR + JJ +not NN, not NNS
`(RBS.(JJ!.(NN|NNS)))`	RBS + JJ +not NN, not NNS
`(JJ.(JJ!.(NN|NNS)))`	JJ + JJ +not NN, not NNS
`(NN.(JJ!.(NN|NNS)))`	NN + JJ +not NN, not NNS
`(NS.(JJ!.(NN|NNS)))`	NS + JJ +not NN, not NNS
`(RB.(VB|VBD|VBN|VBG))`	RB + VB or VBDor VBN or VBG
`(RBR.(VB|VBD|VBN|VBG))`	RBR + VB or VBDor VBN or VBG
`(RBS.(VB|VBD|VBN|VBG))`	RBS + VB or VBDor VBN or VBG

Table A.2: Extracted POS tag patterns using *TGrep2* expressions.

Sentiment Lexicon Learning and Compilation

Up to this point, we have not made use of any sentiment annotation of our dataset. That said, we could learn the polarity of each tag pattern using a totally unsupervised approach. One such approach is the *SO-PMI* method proposed in [185] and extensively discussed in [175]. This approach uses search engines hit counts to calculate the Pointwise Mutual Information (PMI) between a given "keyword" and two fixed strong opinion words, such as "good" and "bad", which are expected to have opposite sentiment polarity. The *SO-PMI* is the difference between the two PMI measures.[5] Sticking to our example, we would expect that the PMI between the word "bad" and "economic concerns" would be higher than the PMI between the word "good" and "economic concerns", which would make the text "*Stocks tank on global economic concerns*" more biased to a negative sentiment label than a positive one. Nonetheless, as pointed out in [174], search engines are living organisms, subject to a constant updating process, making the *SO-PMI* measure highly unstable over time. Additionally, the goal of this study is to learn a domain-specific lexicon, but typically, search engines do not segregate queries into texts from a specific domain, (in our case the financial markets domain).

As an alternative to unsupervised learning, we leverage the sentiment annotation of our dataset and train three supervised binary sentiment classifiers: Logistic Regression, Linear Support Vector Machine and Neural Network. All classifiers are trained to predict the probability of the positive sentiment label and except for the Neural Network classifier were implemented using the Scikit-learn library [142]. The Neural Network uses the Keras library [40] and is trained using an architecture with one hidden dense layer and one final dense layer with only one neuron.

Below, we provide a detailed explanation of all steps leading from the dataset messages to our proposed Sentiment Lexicon compilation:

[5]Note that the measure will be negative if the PMI ("distance") between a given keyword and "bad" is higher than the distance between the keyword "good". In simple terms, negative (positive) measures are associated with negative (positive) sentiments.

1. *SO Tag Pattern Extraction*: After performing the pre-processing steps described in Appendix A.1.2, for each message we extracted all possible tag patterns (terms) described in Table A.2. We assigned the set of all the different tag patterns extracted from our dataset as the vocabulary set V. When performing this step we ended up with a vocabulary with $1,185$ entries, which constituted the dimension of our sparse vector space model.

2. *Instance Representation*: We represented each message in the base of terms V using three different weight schemes: Term-Frequency (TF), Term Frequency-Inverse Document Frequency[6] (TF-IDF) and One-hot, where the term representation is assigned to one if the term appears in the text and zero otherwise.

3. *Hyperparameter Selection*: We separated 15% of samples for testing. We then further split 85% of the data into training and validation sets. We used the grid search method to sweep all possible hyperparameters of the search space. We chose the model that had the best performance in the validation set, using 10-fold cross-validation. Table A.3 describes the hyperparameters space for each classifier. In total this step outputs 9 models, i.e. 3 (classifiers) times 3 text representations per classifier.

4. *Sentiment Lexicon Compilation*: At this stage, our models can be used to classify the binary sentiment of any text. However, in order to compile a sentiment lexicon as a handy language resource, we performed the following tasks:

   - First, at inference time, we predicted the **POS** label probability $\{p_i\}_{i=1}^{1,185}$ for all entries in our vocabulary V using the One-hot models. Technically, this step was implemented by passing though our classifiers $1,185$ vectors. Each of these vectors have zero elements for all dimensions expect for the ith dimension corresponding to the lexicon V_i, which has entry one.
   - Second, we introduced a cut-off probability for the decision boundary. The cut-off probability decides if a given term should be grouped in the positive or negative word list. Thus, all terms V_i with probability p_i greater (less) than 0.60 (0.40) are classified as positive (negative). The remaining terms are filtered out ($0.40 \leq p_i \leq 0.60$).

Our proposed "bag of Sentiment Orientations" representation addresses two main challenges. First, it filters out factual texts (neutral opinions). Second, it is our proposed solution to build a sentiment lexicon straight from a binary sentiment classifier. We have made the positive and negative word lists (sentiment lexicon) available as a language resource, which can be found in the files `stocksenti-word-list-pos` and `stocksenti-word-list-neg` for the positive and negative sentiments, respectively.

[6]The standard IDF weight scheme is employed in our work. This weight will lower the total TF weight of any term $V_i \in V$ that appears frequently in all instances of the dataset. For example, a term that appears in all instances (documents) will have a final TF-IDF weight equal to zero.

Classifier	**Hyperparameters**
Logistic Regression	`regularization_type` = {l1, l2} ,
	`C_regularization` = {0.10, 1, 10, 100}
Linear SVM	`C_regularization` = {0.10, 1, 10, 100}
Neural Network	`hidden_layer_n_neurons` = {64, 128}

Table A.3: Hyperparameters space.

A.1.4 Results

Table A.4 shows the Sentiment classifiers test set accuracy for the 15% of samples that were removed.

We can see the best classifier uses a Neural Network. In addition, the One-Hot representation outperforms all other representations, namely TF and TF-IDF. Based on this result, we choose the Neural Network model with One-hot representation to compile the Sentiment Lexicon.

Classifier	**TF**	**TF-IDF**	**One-Hot**
Logistic Regression	82.9	83.7	82.4
Linear SVM	82.8	83.0	82.7
Neural Network	91.3	90.8	91.4

Table A.4: Test set accuracy for different classifiers (rows) and instance representations (columns).

To evaluate our learnt sentiment lexicon, here named *StockSenti*, we used two different strategies. To begin with, we assessed how far our word lists were able to capture the sentiment of terms commonly used in financial parlance, taking into account formal and informal language variations. In addition, we evaluated the effectiveness of our sentiment lexicon as a potential tool to extend the manually compiled *Loughran and McDonald Financial Sentiment Dictionary* [121], which was built using exclusively corporate disclosures.

We provide many examples where our Sentiment Lexicon is able to learn terms related to positive and negative polarity for stock market texts. To exemplify, the term *next resistance* and *strong support* have high positive probability (0.82). In contrast, *next support* is highly negative (with probability equal to 0.21). Furthermore, the keywords *short squeeze*, *short covering* and *too cheap* are positive, whereas *shorting opportunity*, *great short*, *too high*, *high price* and *buy puts* are negative.[7]

Interestingly, our dictionary is able to capture some relationships between the economic environment and stock markets. For instance, *cheap oil* is classified as positive, in agreement with the average negative correlation between inflation and stock prices. Even phrases like *bad weather* (0.2 probability) that are less obvious

[7]The reader not familiar with the words “long”, “short”, “support”, “resistance”, “covering” and “put”/“call” derivatives instruments is encourage to consult introductory capital markets books to gain specific domain knowledge.

to grasp[8] were correctly classified. In particular, all the examples cited above are misclassified by all publicly available dictionaries built using general corpora (*Sentiment140* [76], *Bing Liu Sentiment Lexicon* [119], *MPQA Sentiment Lexicon* [197], *Harvard Dictionary* (the *General Inquirer*) [171] and *VADER* [97]).

Regarding the effectiveness in extending the *Loughran and McDonald Financial Sentiment Dictionary* [121], in Table A.5 we group a few examples of words that are not present in this financial domain dictionary, and thus, are potential candidates to extend the same.

Positive	**Negative**
solid quarter, extremely undervalued, green day, buyback program, strong cash, outperform recommendation, solid company, major upgrade, legislative inaction	strong sell-off, insider trading, unprofitable allocation, expensive debt, litigious fraud, profitless resources

Table A.5: Proposed sentiment lexicon: Examples.

A.1.5 Conclusion

This work makes a novel text corpora and sentiment lexicon for financial text mining available to the research community. Indisputably, both language resources are valuable to the studies of corporate disclosures, e.g. corporate press releases, annual reports and so forth.

The sentiment classifier built on top of the sentiment labelled *Yahoo Message Board* service covers a broad range of stocks and is effective in classifying the sentiment of terms common to the stock markets parlance. We extensively assessed different classifiers and text representations and the best combination of text representation weights and classifier model achieves 91.4% accuracy in the test set.

Additionally, we propose a method to build a sentiment lexicon from a sentiment classifier by representing each dataset instance (message) in a base of terms with high polarity, what we named "bag of Semantic Orientation".

We assessed the potential of our learnt sentiment lexicon to be used to extend manually annotated sentiment lexicons (crafted using 10-K or 10-Q financial documents). Not only is our sentiment lexicon effective in extending financial sentiment dictionaries, but it is also able to capture the sentiment of terms from the formal and informal language of the financial stock markets.

[8]A closer look at the dataset reveals that the *bad weather* phrase was extracted from oil companies. In this case, the negative hint is a consequence of damages caused by hurricane seasons.

Appendix B

Appendix B

B.1 MHAN hyperparameters

Table B.1 reports the best hyperparameter (first column) found using grid search, where the search space is reported in the second column. Each table split refers to a block of the full MHAN architecture proposed in Section 3.4.6.

B.2 TL Training setting and results

Both tasks are trained for a maximum of 50 epochs using mini-batch SGD with Adam optimiser [105]. Moreover, at the end of each epoch, we evaluate the validation metrics, which are accuracy (Stanfor SNLI dataset) and F_1 (RCV1 dataset), and save the weights with the best values. Aiming to seeped up training, we implement *early stopping* with patience set to 8 epochs. That is, if the validation scores do not improve for more than 10 epochs we halt the training. Finally, we use Glove pre-trained word embedding [143] as fixed features.

Table B.2 compares our test scores with previous works. We match the results reported for the SNLI task. Regarding RCV1 dataset, we observe that our models are trained using headlines only, while the refereed works use both headline and message body. The reason for training headlines only is that both tasks are learnt with the sole purpose of transferring the sentence encoders to our target volatility prediction task, whose textual mode is restricted to headlines. Thus, the RCV1 results are not directly comparable.

B.3 Additional Analysis – Tests of significance

In this section, we present additional analysis on the significance between the different mean values reported in Tables 3.4 to 3.6. Note all mean values reported in these tables were computed for the a with $n = 50$ stocks under analysis, which constitutes the sample size.

To this end, we test the following Null Hypothesis (H_0): The mean reported for our best model (i.e. + **News (BiLSTM Att) + NRA)**) is equal to the mean reported for any other model. That is,

$$H_0 : \mu_{\text{best model}} = \mu_{\text{another model}} \quad \textit{(population means are equal)}$$
$$H_a : \mu_{\text{best model}} > \mu_{\text{another model}} \quad \textit{(alternative: difference is significant)}$$

More specifically, we use the Aspin-Welch t-test[1] that compares two independent population means with unknown and potentially unequal population standard deviations. This test is based on the t-statistic:

$$t = \frac{\bar{x}_{\text{best model}} - \bar{x}_{\text{another model}}}{\sqrt{\frac{s^2_{\text{best model}}}{n} + \frac{s^2_{\text{another model}}}{n}}},$$

where the numerator is the difference between two sample means and s denotes sample standard deviation.

Table B.3 shows the experimental results of the hypothesis test for all ablations in Table 3.4. At the 1% level of significance, there is sufficient evidence to conclude that our full-fledged model (top row in bold), which use News (as an additional data source) and our proposed NRA component, has superior performance as compared to other models. That is, we reject the null hypothesis that the reported results are equal ($p < 0.01$). However, after controlling for these features (i.e. News + NRA), at 1% level of significance, there is no evidence to support additional improvements when the sentence encoder that uses an Attention mechanism (Att), as reported in the top row in bold, is replaced with a more straightforward Max Pooling (MP) (last row).

In other words, this result corroborates the fact that the main improvements in volatility prediction come from the use of additional source of data (i.e. News) and from being able to “distill” the most relevant news released released on a given day (NRA component).

Table B.4 extends the test of significance to comparisons involving the increase of stock embedding dimensionality, as previously reported in Table 3.5. At the 1% level of significance, there is enough evidence to reject the hypothesis that the reported results are all equal ($p < 0.01$) and accept the alternative hypothesis that the our full-fledged model with stock embedding dimension optimised using hyperparameter tuning (top row in bold) performs better than models with different stock embedding dimensions.

Importantly, as reported in the first row ($d_E = 0$), we degrade the performance of our global model by completely removing the stock embedding. Thus, if the stock embedding dimension is properly tuned using hyperparameter search, the increase in the global model dimensionality by using a stock embedding is compensated by the difference in performance ($p < 0.01$).

Table B.5 extends the test of significance to comparisons involving our full-fledged model with GARCH and using different proxies of daily volatility, as previously

[1]In terms of implementation, we use the STAC platform described in [154]

reported in Table 3.6. At the 1% level of significance, there is enough evidence to reject the hypothesis that the reported results are all equal (p $<$ 0.01) and accept the alternative hypothesis that the reported R^2, MSE and MAE of our full-fledged model (top row in bold) are better than the widely-known GARCH model (first row), as well as, our model using only price as data source (second row).

2. **News Encoder**

Best Hyperparameter	Search Space
$d_S = 512$	$\{256, 512, 1024, 2048\}$

3. **Daily News Relevance Attention (NRA)**

Best Hyperparameter	Search Space
$W_R = 1024 \times 512$	
$v_R = 1024$	$\{512, 1024, 2048\}$
$b_R = 1024$	

4. **News Temporal Attention (NTA)**

Best Hyperparameter	Search Space
$d_{MN} = 512$	$\{256, 512, 1024, 2048\}$
$W_T = 1024 \times 512$	
$v_T = 1024$	$\{512, 1024, 2048\}$
$b_T = 1024$	

5. **Price Encoder**

Best Hyperparameter	Search Space
$d_{MP} = 128$	$\{32, 64, 128, 512\}$

7. **Stock Encoder**

Best Hyperparameter	Search Space
$d_E = 16$	$\{8, 16, 32, 64\}$

9. **Joint Representation Encoder**

Best Hyperparameter	Search Space
$d_{JR} = 1024$	$\{512, 1024, 2048\}$

Table B.1: **MHAN – best hyperparameters and respective search space**. Hyperparameters for each block of the full MHAN described in Section 3.4.6: 2. **News Encoder** d_S is the number of units of the BiLSTM that outputs an encoded sentence S and operates on top of Glove word embedding with dimension equal to 300; 3. **Daily News Relevance Attention (NRA)** hyperparameters of Equation (3.20) and also depicted in Figure 3.6. The attention is employed to learn the most relevant news released on a given day and outputs a single Daily News (DN) vector (green component in Figure 3.7); 4. **News Temporal Attention (NTA)** is an attention mechanism similar to above, but designed to capture news novelty. The NTA output is a single Market News (MN) vector (purple component in Figure 3.7); 5. **Price Encoder** d_{MP} is the number of units of the LSTM that outputs a single Market Price (MP) vector (see purple component in Figure 3.7); 7. **Stock Encoder** d_E is the number of units of the FC layer that outputs a single Stock Embedding (E) vector; 9. **Joint Representation Encoder** d_{JR} is the number of units of the FC layer that receives a concatenation of the MN, MP and E vectors, with respective dimensions 512, 128 and 16.

Dataset	Sentence Encoder	Score
SNLI	LSTM original paper ([29])	0.806
	BiLSTM Mean Pooling ([120])	0.833
	BiLSTM Att with multiple views and factored fusion layer ([116])	0.844
	BiLSTM MP with sentence embedding size 4096 ([41])	0.845
	Our BiLSTM Att with sentence embedding size 2048	<u>0.838</u>
	Our BiLSTM MP with sentence embedding size 2048	<u>0.841</u>
RCV1	k-NN† ([111])	0.765
	Best Support Vector Machine (SVM)† ([111])	0.816
	bow-CNN† ([102])	0.840
	Our BiLSTM Att with sentence embedding size 2048 (headlines only)	<u>0.809</u>
	Our BiLSTM MP with sentence embedding size 2048 (headlines only)	<u>0.811</u>

Table B.2: **TL auxiliary tasks – Comparison with previous works.** Test scores are accuracy (SNLI dataset) and $F1$ (RCV1 dataset). BiLSTM Att and BiLSTM MP are the attention and max-polling sentence encoders detailed in Section 3.3.1 and Section 3.3.1, respectively. All training is performed with the goal of transferring the sentence encoders to the target volatility task. † indicates model trained with both headlines and body content and using the original 103 classes of the RCV1 dataset, rather than our models that are trained using headlines only and a total of 55 classes (see Section 3.4.2 for a complete description).

Comparison with best model:	MSE			MAE		
+ News (BiLSTM Att) + NRA	H_0	t-stat	p-value	H_0	t-stat	p-value
- News (price only unimodal)	rejected	10.43	0.000	rejected	17.69	0.000
+ News (BiLSTM Att) - NRA	rejected	7.91	0.000	rejected	13.25	0.000
+ News (BiLSTM MP) - NRA	rejected	7.64	0.000	rejected	14.75	0.000
+ News (TL Reuters RCV1 BiLSTM MP) + NRA	rejected	6.87	0.000	rejected	11.70	0.000
+ News (TL Reuters RCV1 BiLSTM Att) + NRA	rejected	4.90	0.000	rejected	14.13	0.000
+ News (W-L Att) + NRA	rejected	4.59	0.000	rejected	7.68	0.000
+ News (TL SNLI BiLSTM Att) + NRA	rejected	4.22	0.000	rejected	6.99	0.000
+ News (TL SNLI BiLSTM MP) + NRA	rejected	3.56	0.000	rejected	6.22	0.000
+ News (BiLSTM MP) + NRA	not rejected	0.26	0.793	not rejected	1.99	0.047

Table B.3: **Test of significance for ablations – Two population means with unknown standard deviations.** All models under analysis are described in details in Section 3.5.2 and the statistical tests refer to the ablations reported in Table 3.4. In addition, for all models the mean MSE and MAE were computed considering our universe of stocks and, as a consequence, have all the same sample size of $n = 50$ stocks. Null hypothesis (H_0): The population mean of our best model (top in bold) is equal to the population mean reported for another model. We conclude that the difference between the means (both for MSE and MAE) are significant (p < 0.01) for all models expect for the model reported in the last row. This result corroborates the advantages of using News (as an additional data source) and the NRA component (unique to our proposed neural network architecture). We also conclude that, at 1% level of significance, there is no clear advantage in replacing a sentence encoder that uses attention (Att), as reported in the first row in bold, with a more straightforward Max Pooling (MP), as reported in the last row.

Comparison with our full-fledged model using	MSE			MAE		
optimal stock embedding dimension† ($d_E = 16$)	H_0	t-stat	p-value	H_0	t-stat	p-value
no stock embedding†† ($d_E = 0$)	rejected	3.62	0.000	rejected	7.34	0.000
moderate increase in stock embedding dimension ($d_E = 32$)	rejected	2.62	0.000	rejected	2.85	0.000
extreme increase in stock embedding dimension ($d_E = 1024$)	rejected	5.70	0.000	rejected	19.22	0.000

Table B.4: **Test of significance for changes in embedding dimensionality – Two population means with unknown standard deviations.** All models under analysis are described in details in Section 3.5.2 and the statistical tests refer to the impact of increasing the stock embedding dimension reported in Table 3.5. In addition, for all models the mean MSE and MAE were computed considering our universe of stocks and, therefore, have all the same sample size of $n = 50$ stocks. Null hypothesis (H_0): The population mean of our best model (top in bold) is equal to the population mean reported for another model. At the 1% level of confidence, there is enough evidence to conclude that the differences in mean performance (for both MSE and MAE) between the model reported in the top row in bold and another model (remaining rows) are significant. † indicates the optimal stock embedding dimension ($d_E = 16$) is obtained by performing hyperparameter search on the validation set. †† indicates that we completely remove from our full-fledged model any increase in dimensionality attributed to the stock embedding.

Comparison with **_Our Model: Price + News_**	Vol	R^2			MSE			MAE		
		H_0	t-stat	p-value	H_0	t-stat	p-value	H_0	t-stat	p-value
GARCH	$\widehat{\sigma_{GK}}$	reject	4.14	0.000	reject	25.25	0.000	reject	19.70	0.000
	$\widehat{\sigma_{PK}}$	reject	3.32	0.000	reject	17.77	0.000	reject	11.44	0.000
Our Model: Price (Unimodal)	$\widehat{\sigma_{GK}}$	reject	3.39	0.000	reject	10.35	0.000	reject	17.96	0.000
	$\widehat{\sigma_{PK}}$	reject	2.65	0.007	reject	11.21	0.000	reject	18.22	0.000

Table B.5: **Test of significance for GARCH – Two population means with unknown standard deviations.** The three models under analysis are described in details in Section 3.5.2 and the statistical tests refer to the results reported in Table 3.6. Null hypothesis (H_0): The population mean of our best model (top in bold) is equal to the population mean reported for GARCH or a model that does not take news into account, namely *Price (unimodal)*. The results show that the difference in the mean performance is statistically significant ($p < 0.01$). This being true for all three metrics (i.e. R^2, MSE and MAE) and also taking into account different proxies for daily volatility, i.e. Garman-Klass ($\widehat{\sigma_{GK}}$) and Parkinson ($\widehat{\sigma_{PK}}$).

Appendix C

Appendix C

C.1 Multi-task Graph Neural Network hyperparameters

Table C.1 reports the best hyperparameter (first column) found using grid search, where the search space is reported in the second column. Each table split refers to the hyperparameters of a block of the full Multi-task Graph Neural Network in Figure 4.5. More specifically, we have the following blocks: **Sequence Encoder** (light green), **Graph Transformer Network (GTN)** (purple) and **Task-specific layers** (yellow).

C.2 Return statistics for all stocks

Table C.2 reports experimental results of return statistics for all stocks of our universe.

Table C.2: **Strategy daily return statistics**.

Stocks	Sectors	Sharpe Ratio	MAR Ratio	CAGR	Mean	Std	Min	Max. DD	Acc.	Pct Trades
ALB	Basic Materials	0.010	0.016	-0.374	0.000	0.654	-2.940	-23.154	51.740	85.686
APD	Basic Materials	2.247	3.168	20.196	0.093	0.654	-3.073	-6.376	59.360	80.716
ECL	Basic Materials	0.579	0.426	3.817	0.024	0.654	-2.432	-8.955	51.744	68.390
EMN	Basic Materials	2.963	2.705	30.574	0.122	0.654	-1.532	-11.301	56.306	88.270
FCX	Basic Materials	-0.080	0.069	-1.264	-0.003	0.654	-4.197	-18.279	56.317	92.843
FMC	Basic Materials	0.485	0.398	4.039	0.020	0.654	-5.744	-10.154	54.628	88.072
IFF	Basic Materials	0.883	0.488	7.582	0.036	0.654	-3.039	-15.544	54.460	84.692
MAS	Basic Materials	0.042	0.005	-0.088	0.002	0.654	-3.587	-17.855	50.682	87.475
MLM	Basic Materials	0.162	0.038	1.012	0.007	0.654	-2.461	-26.830	52.477	88.270
MOS	Basic Materials	-1.114	0.496	-10.317	-0.046	0.654	-3.008	-20.799	48.451	89.861
NEM	Basic Materials	-0.200	0.168	-2.273	-0.008	0.654	-3.068	-13.498	54.751	87.873
NUE	Basic Materials	-0.301	0.120	-3.194	-0.012	0.654	-2.879	-26.673	50.112	88.469
PPG	Basic Materials	0.825	0.794	6.559	0.034	0.654	-4.036	-8.264	54.774	79.125
PX	Basic Materials	0.750	0.491	5.707	0.031	0.654	-2.717	-11.615	52.468	76.541
SHW	Basic Materials	0.236	0.094	1.597	0.010	0.654	-3.321	-17.042	50.120	82.903
VMC	Basic Materials	-0.404	0.166	-4.089	-0.017	0.654	-2.894	-24.578	51.467	88.072
WY	Basic Materials	-1.688	0.449	-14.194	-0.070	0.654	-3.518	-31.634	48.122	84.692

Continued on next page

Table C.2 – continued from previous page

Stocks	Sectors	Sharpe Ratio	MAR Ratio	CAGR	Mean	Std	Min	Max. DD	Acc.	Pct Trades
ANDV	Energy	0.250	0.202	1.841	0.010	0.654	-4.645	-9.120	50.112	88.867
APA	Energy	0.651	0.428	5.662	0.027	0.654	-2.553	-13.221	52.809	88.469
APC	Energy	0.980	1.134	9.191	0.040	0.654	-2.965	-8.101	55.338	91.252
COG	Energy	-0.115	0.076	-1.572	-0.005	0.654	-3.113	-20.584	50.980	91.252
COP	Energy	0.026	0.018	-0.231	0.001	0.654	-3.094	-12.876	50.816	85.288
CVX	Energy	2.019	1.112	16.493	0.083	0.654	-2.820	-14.834	56.383	74.751
DVN	Energy	0.725	0.766	6.645	0.030	0.654	-3.079	-8.679	52.268	92.048
EOG	Energy	1.402	1.224	12.830	0.058	0.654	-2.393	-10.483	54.965	86.083
EQT	Energy	-0.058	0.086	-1.022	-0.002	0.654	-3.102	-11.863	53.982	89.861
HAL	Energy	1.578	1.493	14.144	0.065	0.654	-2.454	-9.474	53.810	83.499
HES	Energy	0.063	0.010	0.110	0.003	0.654	-2.555	-10.892	48.485	91.849
HFC	Energy	-0.021	0.060	-0.695	-0.001	0.654	-4.130	-11.582	51.627	91.650
HP	Energy	0.904	0.494	8.277	0.037	0.654	-3.122	-16.759	54.425	89.861
MRO	Energy	-0.114	0.162	-1.593	-0.005	0.654	-3.215	-9.814	50.641	93.042
NBL	Energy	0.621	0.309	5.428	0.026	0.654	-2.422	-17.555	50.000	89.463
NFX	Energy	0.209	0.056	1.533	0.009	0.654	-2.420	-27.301	51.812	93.241
NOV	Energy	-0.040	0.075	-0.833	-0.002	0.654	-2.848	-11.126	51.131	87.873
OKE	Energy	0.612	0.439	5.062	0.025	0.654	-3.794	-11.539	53.162	84.891
OXY	Energy	0.883	0.657	7.219	0.036	0.654	-2.818	-10.988	51.970	80.716
PXD	Energy	1.310	0.910	12.017	0.054	0.654	-2.676	-13.207	52.403	86.879
SLB	Energy	2.518	1.408	23.901	0.104	0.654	-2.719	-16.979	57.482	83.698
VLO	Energy	1.249	1.413	11.350	0.051	0.654	-3.244	-8.031	54.713	86.481
WMB	Energy	-0.654	0.323	-6.181	-0.027	0.654	-6.442	-19.166	48.513	86.879
XOM	Energy	3.044	2.382	23.671	0.125	0.654	-2.995	-9.939	55.814	68.390
ADP	Industrials	0.049	0.002	-0.036	0.002	0.654	-7.660	-22.314	50.732	81.511
AME	Industrials	1.277	0.925	11.655	0.053	0.654	-3.487	-12.600	57.569	86.680
AOS	Industrials	2.278	3.659	22.798	0.094	0.654	-3.135	-6.230	57.047	88.867
ARNC	Industrials	-1.026	0.404	-9.834	-0.042	0.654	-5.018	-24.336	50.538	92.445
AVY	Industrials	1.647	2.294	15.327	0.068	0.654	-2.192	-6.682	53.580	86.083
BA	Industrials	1.619	1.787	15.824	0.067	0.654	-3.254	-8.857	55.507	90.258
CAT	Industrials	1.191	1.210	11.394	0.049	0.654	-2.749	-9.417	53.595	91.252
CHRW	Industrials	2.353	1.274	23.128	0.097	0.654	-2.526	-18.159	55.936	87.078
CMI	Industrials	1.189	1.382	11.140	0.049	0.654	-3.848	-8.058	55.778	89.463
CSX	Industrials	0.340	0.119	2.783	0.014	0.654	-2.544	-23.464	49.458	91.650
CTAS	Industrials	0.686	0.458	5.596	0.028	0.654	-4.111	-12.214	54.087	82.704
DE	Industrials	0.723	0.544	6.104	0.030	0.654	-6.346	-11.217	53.972	85.089
DOV	Industrials	0.700	0.505	6.192	0.029	0.654	-2.728	-12.263	53.229	89.264
EFX	Industrials	0.279	0.161	2.070	0.012	0.654	-8.028	-12.887	54.566	87.078
EMR	Industrials	-0.384	0.220	-3.884	-0.016	0.654	-3.912	-17.695	49.773	87.475
ETN	Industrials	1.771	0.775	17.023	0.073	0.654	-3.475	-21.953	54.402	88.072
EXPD	Industrials	3.949	4.135	41.871	0.163	0.654	-3.707	-10.125	67.126	86.481
FAST	Industrials	0.840	0.430	7.573	0.035	0.654	-4.135	-17.613	54.343	89.264
FDX	Industrials	0.078	0.008	0.235	0.003	0.654	-2.577	-28.732	52.727	87.475
FISV	Industrials	1.655	0.916	15.060	0.068	0.654	-2.271	-16.436	55.189	84.294
FLS	Industrials	-0.178	0.084	-2.140	-0.007	0.654	-4.732	-25.399	52.412	90.656
GD	Industrials	0.685	0.324	5.778	0.028	0.654	-3.894	-17.851	50.233	85.487
GE	Industrials	-0.000	0.017	-0.422	-0.000	0.654	-4.612	-24.545	54.220	77.734

Continued on next page

Table C.2 – continued from previous page

Stocks	Sectors	Sharpe Ratio	MAR Ratio	CAGR	Mean	Std	Min	Max. DD	Acc.	Pct Trades
GWW	Industrials	-0.077	0.053	-1.194	-0.003	0.654	-5.139	-22.395	53.659	89.662
HON	Industrials	1.946	1.046	15.565	0.080	0.654	-2.419	-14.881	54.054	73.559
IR	Industrials	0.150	0.049	0.909	0.006	0.654	-2.918	-18.399	50.112	88.469
IRM	Industrials	-0.336	0.159	-3.561	-0.014	0.654	-3.246	-22.423	52.539	90.060
ITW	Industrials	0.309	0.097	2.161	0.013	0.654	-2.803	-22.227	51.861	80.119
JBHT	Industrials	1.127	0.750	10.720	0.046	0.654	-2.019	-14.299	54.902	91.252
JCI	Industrials	0.676	0.403	5.918	0.028	0.654	-2.754	-14.684	51.794	88.668
JEC	Industrials	1.386	1.311	13.380	0.057	0.654	-2.385	-10.203	55.702	90.656
KSU	Industrials	1.909	1.244	19.324	0.079	0.654	-3.024	-15.532	54.664	91.650
LLL	Industrials	1.318	0.926	12.274	0.054	0.654	-3.962	-13.256	54.402	88.072
LMT	Industrials	0.637	0.475	4.849	0.026	0.654	-2.802	-10.199	51.276	77.932
LUV	Industrials	1.189	1.442	11.349	0.049	0.654	-2.538	-7.870	57.642	91.054
MMM	Industrials	0.528	0.225	3.695	0.022	0.654	-2.485	-16.424	50.407	73.360
NOC	Industrials	0.274	0.146	1.876	0.011	0.654	-3.183	-12.826	49.877	80.517
NSC	Industrials	-0.052	0.050	-0.964	-0.002	0.654	-3.459	-19.391	52.212	89.861
PAYX	Industrials	0.923	0.862	7.922	0.038	0.654	-2.832	-9.193	53.066	84.294
PCAR	Industrials	1.097	0.648	10.077	0.045	0.654	-4.021	-15.541	54.831	88.469
PH	Industrials	0.345	0.110	2.647	0.014	0.654	-2.963	-24.096	51.620	85.885
PNR	Industrials	0.572	0.325	4.882	0.024	0.654	-2.224	-15.024	55.631	88.270
PWR	Industrials	0.276	0.221	2.125	0.011	0.654	-3.609	-9.632	51.982	90.258
ROK	Industrials	1.050	0.474	9.440	0.043	0.654	-3.087	-19.933	53.653	87.078
ROP	Industrials	1.199	0.816	10.775	0.049	0.654	-2.776	-13.198	53.935	85.885
RSG	Industrials	-0.317	0.212	-2.769	-0.013	0.654	-2.376	-13.063	50.407	73.360
RTN	Industrials	-0.296	0.221	-2.860	-0.012	0.654	-3.280	-12.964	49.505	80.318
SNA	Industrials	0.266	0.147	2.006	0.011	0.654	-3.677	-13.685	51.448	89.264
SRCL	Industrials	0.533	0.225	4.561	0.022	0.654	-3.777	-20.237	53.007	89.264
SWK	Industrials	1.404	0.845	12.439	0.058	0.654	-2.889	-14.721	54.286	83.499
TXT	Industrials	1.392	0.830	13.198	0.057	0.654	-3.349	-15.893	55.134	89.066
UNP	Industrials	1.538	1.432	14.909	0.063	0.654	-3.661	-10.411	52.980	90.060
UPS	Industrials	-0.415	0.121	-3.754	-0.017	0.654	-2.787	-31.053	50.378	78.926
URI	Industrials	-0.045	0.039	-0.958	-0.002	0.654	-3.204	-24.536	52.588	96.024
UTX	Industrials	0.702	0.353	5.412	0.029	0.654	-3.466	-15.316	55.725	78.131
WM	Industrials	0.239	0.120	1.368	0.010	0.654	-3.319	-11.391	52.557	69.980
AMZN	Consumer Cyclical	-1.678	0.473	-14.333	-0.069	0.654	-3.662	-30.313	47.344	86.083
AZO	Consumer Cyclical	-0.390	0.218	-4.059	-0.016	0.654	-2.339	-18.601	49.780	90.258
BBY	Consumer Cyclical	-0.271	0.130	-3.017	-0.011	0.654	-3.171	-23.225	49.673	91.252
BKNG	Consumer Cyclical	0.806	0.626	6.823	0.033	0.654	-3.370	-10.897	54.245	84.294
BLL	Consumer Cyclical	0.983	0.587	8.366	0.041	0.654	-3.504	-14.256	56.459	83.101
BWA	Consumer Cyclical	0.218	0.130	1.523	0.009	0.654	-2.749	-11.700	52.045	87.475
CCL	Consumer Cyclical	0.102	0.029	0.432	0.004	0.654	-2.567	-14.707	48.798	82.704
CPRT	Consumer Cyclical	3.038	1.947	31.556	0.125	0.654	-3.897	-16.209	60.449	88.469
DHI	Consumer Cyclical	2.355	3.635	24.266	0.097	0.654	-2.055	-6.675	56.018	90.855
DIS	Consumer Cyclical	0.957	0.300	7.841	0.039	0.654	-3.701	-26.129	52.723	80.318
DRI	Consumer Cyclical	0.546	0.230	4.566	0.022	0.654	-3.262	-19.889	54.338	87.078
EBAY	Consumer Cyclical	0.474	0.378	3.937	0.020	0.654	-2.047	-10.423	53.950	88.072
F	Consumer Cyclical	0.696	0.548	5.965	0.029	0.654	-2.996	-10.888	53.211	86.680
FL	Consumer Cyclical	0.626	0.622	5.590	0.026	0.654	-2.664	-8.991	55.120	91.252

Continued on next page

Table C.2 – continued from previous page

Stocks	Sectors	Sharpe Ratio	MAR Ratio	CAGR	Mean	Std	Min	Max. DD	Acc.	Pct Trades
FOX	Consumer Cyclical	-0.236	0.128	-2.658	-0.010	0.654	-4.408	-20.781	51.435	90.060
FOXA	Consumer Cyclical	0.059	0.004	0.061	0.002	0.654	-4.618	-16.156	55.835	86.879
GPC	Consumer Cyclical	-0.478	0.219	-4.657	-0.020	0.654	-2.087	-21.231	49.771	86.680
GPS	Consumer Cyclical	-0.728	0.275	-7.290	-0.030	0.654	-3.339	-26.525	51.064	93.439
GT	Consumer Cyclical	2.381	4.149	24.265	0.098	0.654	-2.654	-5.848	58.628	89.861
HAS	Consumer Cyclical	0.383	0.175	2.986	0.016	0.654	-3.375	-17.113	51.860	85.487
HD	Consumer Cyclical	-0.126	0.103	-1.532	-0.005	0.654	-3.799	-14.810	50.238	83.499
HOG	Consumer Cyclical	0.926	0.795	8.406	0.038	0.654	-8.448	-10.574	55.357	89.066
HRB	Consumer Cyclical	-1.093	0.303	-10.014	-0.045	0.654	-3.004	-33.092	48.430	88.668
IP	Consumer Cyclical	0.386	0.328	3.072	0.016	0.654	-4.052	-9.380	54.897	87.276
IPG	Consumer Cyclical	1.158	1.051	10.797	0.048	0.654	-3.287	-10.273	52.784	89.264
JWN	Consumer Cyclical	-0.784	0.260	-7.714	-0.032	0.654	-3.967	-29.726	49.247	92.445
KMX	Consumer Cyclical	-0.071	0.059	-1.142	-0.003	0.654	-3.836	-19.451	53.097	89.861
KSS	Consumer Cyclical	0.120	0.027	0.646	0.005	0.654	-2.710	-24.280	52.193	90.656
LB	Consumer Cyclical	-0.407	0.303	-4.302	-0.017	0.654	-2.731	-14.193	51.293	92.247
LEG	Consumer Cyclical	0.007	0.023	-0.395	0.000	0.654	-2.976	-17.485	51.515	85.288
LEN	Consumer Cyclical	1.252	2.078	11.875	0.052	0.654	-2.184	-5.714	54.525	90.060
LOW	Consumer Cyclical	-0.342	0.220	-3.500	-0.014	0.654	-3.247	-15.940	53.196	87.078
M	Consumer Cyclical	0.408	0.222	3.507	0.017	0.654	-3.059	-15.805	51.599	93.241
MAR	Consumer Cyclical	0.316	0.240	2.405	0.013	0.654	-3.381	-10.020	54.358	86.680
MAT	Consumer Cyclical	-1.067	0.300	-9.986	-0.044	0.654	-4.785	-33.249	49.231	90.457
MCD	Consumer Cyclical	-0.044	0.052	-0.732	-0.002	0.654	-3.680	-14.063	53.639	73.757
MGM	Consumer Cyclical	1.361	1.146	12.989	0.056	0.654	-2.619	-11.335	53.982	89.861
MHK	Consumer Cyclical	2.768	2.821	28.621	0.114	0.654	-3.837	-10.147	58.575	89.264
NKE	Consumer Cyclical	-0.024	0.058	-0.703	-0.001	0.654	-2.781	-12.220	49.889	89.662
OMC	Consumer Cyclical	1.655	1.465	14.985	0.068	0.654	-2.757	-10.228	54.739	83.897
ORLY	Consumer Cyclical	0.798	0.744	6.978	0.033	0.654	-5.236	-9.383	56.393	87.078
PHM	Consumer Cyclical	1.668	2.233	16.468	0.069	0.654	-2.354	-7.376	53.173	90.855
PKG	Consumer Cyclical	0.483	0.379	4.034	0.020	0.654	-2.994	-10.636	55.631	88.270
PVH	Consumer Cyclical	1.673	0.670	16.794	0.069	0.654	-3.300	-25.054	58.405	92.247
RCL	Consumer Cyclical	-0.282	0.181	-3.143	-0.012	0.654	-4.181	-17.412	52.268	92.048
RL	Consumer Cyclical	0.550	0.234	4.831	0.023	0.654	-5.380	-20.661	54.031	91.252
ROST	Consumer Cyclical	1.395	1.111	13.137	0.058	0.654	-2.846	-11.821	54.607	88.469
SBUX	Consumer Cyclical	-0.576	0.342	-5.352	-0.024	0.654	-4.462	-15.632	50.000	84.294
SEE	Consumer Cyclical	0.657	0.631	5.672	0.027	0.654	-4.207	-8.986	54.525	87.873
TIF	Consumer Cyclical	1.049	0.717	9.817	0.043	0.654	-2.154	-13.690	53.187	90.457
TJX	Consumer Cyclical	0.237	0.101	1.715	0.010	0.654	-2.963	-16.964	51.802	88.270
TSCO	Consumer Cyclical	0.449	0.186	3.796	0.019	0.654	-3.223	-20.455	54.185	90.258
VFC	Consumer Cyclical	0.181	0.123	1.183	0.007	0.654	-2.601	-9.610	53.864	87.475
WHR	Consumer Cyclical	1.583	1.663	15.436	0.065	0.654	-2.359	-9.282	56.608	90.258
YUM	Consumer Cyclical	1.782	2.193	15.436	0.073	0.654	-3.098	-7.040	55.224	79.920
ADM	Consumer Defensive	0.524	0.410	4.583	0.022	0.654	-2.354	-11.170	48.478	91.451
BF-B	Consumer Defensive	1.741	1.406	17.350	0.072	0.654	-2.311	-12.344	57.516	91.252
CAG	Consumer Defensive	0.734	0.461	6.576	0.030	0.654	-2.144	-14.268	51.770	89.861
CHD	Consumer Defensive	0.571	0.344	4.915	0.024	0.654	-4.102	-14.307	53.571	89.066
CL	Consumer Defensive	0.483	0.254	3.764	0.020	0.654	-5.071	-14.827	55.663	82.505
CLX	Consumer Defensive	0.301	0.128	2.271	0.012	0.654	-3.833	-17.758	51.259	86.879

Continued on next page

Table C.2 – continued from previous page

Stocks	Sectors	Sharpe Ratio	MAR Ratio	CAGR	Mean	Std	Min	Max. DD	Acc.	Pct Trades
COST	Consumer Defensive	-0.021	0.044	-0.652	-0.001	0.654	-4.156	-14.650	50.694	85.885
CPB	Consumer Defensive	0.215	0.076	1.495	0.009	0.654	-3.381	-19.697	51.357	87.873
DLTR	Consumer Defensive	1.329	0.911	13.076	0.055	0.654	-2.988	-14.352	54.936	92.644
EL	Consumer Defensive	1.967	1.573	18.767	0.081	0.654	-2.872	-11.934	55.632	86.481
GIS	Consumer Defensive	-0.375	0.189	-3.747	-0.015	0.654	-3.578	-19.871	50.577	86.083
HRL	Consumer Defensive	-0.285	0.146	-3.161	-0.012	0.654	-4.878	-21.603	49.351	91.849
HSY	Consumer Defensive	1.198	0.983	11.084	0.049	0.654	-9.623	-11.277	55.730	88.469
K	Consumer Defensive	-0.472	0.199	-4.659	-0.019	0.654	-3.931	-23.401	51.247	87.674
KMB	Consumer Defensive	1.474	1.578	13.860	0.061	0.654	-2.681	-8.782	54.525	87.873
KO	Consumer Defensive	0.563	0.263	3.829	0.023	0.654	-2.475	-14.538	50.843	70.775
KR	Consumer Defensive	-0.884	0.394	-8.690	-0.036	0.654	-8.106	-22.044	48.511	93.439
MKC	Consumer Defensive	0.157	0.059	0.967	0.006	0.654	-3.558	-16.517	54.730	88.270
MNST	Consumer Defensive	0.756	0.422	6.998	0.031	0.654	-2.794	-16.602	52.903	92.445
MO	Consumer Defensive	-0.043	0.044	-0.852	-0.002	0.654	-7.257	-19.247	51.501	86.083
NWL	Consumer Defensive	1.322	1.512	12.850	0.054	0.654	-3.702	-8.499	56.182	91.650
PEP	Consumer Defensive	0.256	0.149	1.586	0.011	0.654	-3.113	-10.681	52.139	74.354
PG	Consumer Defensive	1.219	1.084	9.058	0.050	0.654	-2.354	-8.353	55.000	71.571
SJM	Consumer Defensive	-0.221	0.119	-2.487	-0.009	0.654	-3.245	-20.948	54.484	88.668
STZ	Consumer Defensive	0.275	0.096	2.062	0.011	0.654	-4.017	-21.579	52.144	88.072
SYY	Consumer Defensive	-0.417	0.243	-3.902	-0.017	0.654	-3.405	-16.072	53.285	81.710
TAP	Consumer Defensive	0.103	0.025	0.486	0.004	0.654	-3.136	-19.253	50.756	92.048
TGT	Consumer Defensive	-0.039	0.033	-0.853	-0.002	0.654	-6.418	-26.026	53.187	90.457
TSN	Consumer Defensive	0.078	0.014	0.253	0.003	0.654	-3.208	-17.680	52.473	92.445
WBA	Consumer Defensive	-0.717	0.321	-6.794	-0.030	0.654	-3.156	-21.168	49.887	88.072
WMT	Consumer Defensive	-0.085	0.082	-1.236	-0.004	0.654	-3.653	-14.988	51.818	87.475
AMT	Communication Services	-0.642	0.401	-5.907	-0.026	0.654	-3.032	-14.730	47.059	84.493
CMCSA	Communication Services	-0.076	0.048	-1.202	-0.003	0.654	-4.284	-24.989	50.000	90.656
CTL	Communication Services	-0.808	0.331	-7.571	-0.033	0.654	-2.810	-22.872	50.339	88.072
DISH	Communication Services	0.219	0.113	1.573	0.009	0.654	-2.560	-13.932	54.525	90.060
SBAC	Communication Services	1.884	2.323	18.691	0.078	0.654	-2.379	-8.045	58.278	90.060
T	Communication Services	-0.006	0.029	-0.504	-0.000	0.654	-3.421	-17.105	50.236	84.294
VZ	Communication Services	0.375	0.274	2.832	0.015	0.654	-2.047	-10.347	52.267	83.300
AFL	Financial Services	2.678	2.931	22.675	0.110	0.654	-3.037	-7.736	58.621	74.950
AIG	Financial Services	0.981	0.397	8.181	0.040	0.654	-3.081	-20.614	56.098	81.511
AJG	Financial Services	0.719	0.544	5.638	0.030	0.654	-2.955	-10.369	51.759	79.125
ALL	Financial Services	0.792	0.542	5.465	0.033	0.654	-3.280	-10.075	51.149	69.185
AMG	Financial Services	1.634	1.510	16.706	0.067	0.654	-2.608	-11.061	58.562	94.036
AON	Financial Services	-0.663	0.447	-5.385	-0.027	0.654	-3.372	-12.033	50.400	74.553
AXP	Financial Services	1.225	1.127	10.355	0.050	0.654	-4.269	-9.191	55.037	80.915
BAC	Financial Services	0.288	0.164	2.239	0.012	0.654	-3.317	-13.646	51.101	90.258
BBT	Financial Services	0.831	0.682	6.955	0.034	0.654	-3.673	-10.193	54.785	83.101
BEN	Financial Services	0.841	0.672	7.742	0.035	0.654	-6.055	-11.522	55.240	91.054
BK	Financial Services	0.555	0.593	4.601	0.023	0.654	-2.727	-7.756	55.196	86.083
BLK	Financial Services	1.802	1.437	16.343	0.074	0.654	-2.123	-11.371	53.938	83.300
BRK-B	Financial Services	0.521	0.337	3.417	0.021	0.654	-2.332	-10.130	50.720	68.986
C	Financial Services	1.454	1.203	14.507	0.060	0.654	-2.875	-12.059	57.265	93.042
CB	Financial Services	0.252	0.152	1.589	0.010	0.654	-3.600	-10.457	52.756	75.746

Continued on next page

Table C.2 – continued from previous page

Stocks	Sectors	Sharpe Ratio	MAR Ratio	CAGR	Mean	Std	Min	Max. DD	Acc.	Pct Trades
CINF	Financial Services	0.785	1.032	6.562	0.032	0.654	-2.108	-6.361	52.143	83.499
CMA	Financial Services	1.333	1.129	13.268	0.055	0.654	-3.253	-11.754	56.051	93.638
COF	Financial Services	0.547	0.502	4.737	0.023	0.654	-3.426	-9.444	52.318	90.060
ETFC	Financial Services	2.687	2.627	29.267	0.111	0.654	-2.985	-11.142	58.475	93.837
FITB	Financial Services	1.366	1.492	13.353	0.056	0.654	-2.883	-8.947	55.195	91.849
GS	Financial Services	1.116	0.804	10.395	0.046	0.654	-2.547	-12.935	56.000	89.463
HBAN	Financial Services	1.508	1.068	14.969	0.062	0.654	-3.022	-14.021	57.974	92.247
HIG	Financial Services	0.396	0.343	2.996	0.016	0.654	-3.223	-8.740	51.923	82.704
IVZ	Financial Services	0.491	0.370	4.205	0.020	0.654	-3.347	-11.377	52.863	90.258
JEF	Financial Services	-0.441	0.210	-4.504	-0.018	0.654	-2.679	-21.405	50.773	90.060
JPM	Financial Services	1.036	0.744	8.842	0.043	0.654	-2.545	-11.891	54.436	82.903
KEY	Financial Services	1.948	1.701	20.380	0.080	0.654	-3.778	-11.981	58.439	94.235
L	Financial Services	0.897	0.611	6.962	0.037	0.654	-4.546	-11.391	57.254	76.740
LNC	Financial Services	2.234	3.012	23.353	0.092	0.654	-3.158	-7.753	57.296	92.644
MCO	Financial Services	0.670	0.584	5.435	0.028	0.654	-3.727	-9.301	53.253	82.505
MMC	Financial Services	0.016	0.024	-0.282	0.001	0.654	-3.682	-11.635	52.255	74.950
MS	Financial Services	1.132	1.203	10.878	0.047	0.654	-3.397	-9.043	53.780	92.048
MTB	Financial Services	1.862	1.503	18.004	0.077	0.654	-2.925	-11.980	57.336	88.072
NTRS	Financial Services	3.607	5.350	39.316	0.149	0.654	-2.258	-7.349	63.053	89.861
PBCT	Financial Services	2.075	1.530	21.284	0.086	0.654	-3.093	-13.910	56.061	91.849
PNC	Financial Services	1.516	1.905	14.185	0.062	0.654	-2.883	-7.448	55.353	87.276
RE	Financial Services	0.611	0.280	5.135	0.025	0.654	-4.142	-18.353	51.613	86.282
RF	Financial Services	1.946	1.832	20.308	0.080	0.654	-2.727	-11.082	56.660	94.036
RJF	Financial Services	0.763	0.529	6.923	0.031	0.654	-3.040	-13.094	54.386	90.656
SCHW	Financial Services	1.215	1.393	11.760	0.050	0.654	-2.518	-8.443	54.212	92.048
SIVB	Financial Services	2.137	2.632	22.892	0.088	0.654	-3.027	-8.699	60.752	95.229
STI	Financial Services	1.070	1.045	10.202	0.044	0.654	-3.334	-9.764	54.329	91.849
STT	Financial Services	1.620	2.304	15.720	0.067	0.654	-3.350	-6.822	55.876	89.662
TMK	Financial Services	3.798	4.075	35.808	0.157	0.654	-2.387	-8.787	60.859	78.728
TROW	Financial Services	2.477	2.237	24.448	0.102	0.654	-4.710	-10.930	58.124	86.879
TRV	Financial Services	0.597	0.304	4.461	0.025	0.654	-3.403	-14.674	55.928	77.137
TSS	Financial Services	0.868	0.749	7.764	0.036	0.654	-2.569	-10.361	50.676	88.270
UNM	Financial Services	1.384	1.567	12.924	0.057	0.654	-2.864	-8.246	54.751	87.873
USB	Financial Services	1.865	1.480	16.719	0.077	0.654	-3.553	-11.301	56.174	82.107
WFC	Financial Services	0.935	0.883	8.406	0.039	0.654	-3.656	-9.518	53.725	88.072
XL	Financial Services	-0.256	0.150	-2.793	-0.011	0.654	-4.032	-18.605	48.764	88.469
ZION	Financial Services	2.184	1.605	23.354	0.090	0.654	-2.845	-14.548	56.604	94.831
A	Healthcare	-0.334	0.226	-3.331	-0.014	0.654	-2.542	-14.723	51.059	84.493
ABC	Healthcare	1.294	1.189	11.998	0.053	0.654	-4.650	-10.093	54.977	87.873
ABMD	Healthcare	-0.136	0.198	-1.706	-0.006	0.654	-4.033	-8.593	51.810	87.873
ABT	Healthcare	-0.291	0.250	-2.936	-0.012	0.654	-3.215	-11.742	50.831	83.698
AET	Healthcare	-0.350	0.273	-3.581	-0.014	0.654	-6.482	-13.120	51.936	87.276
AGN	Healthcare	0.130	0.042	0.740	0.005	0.654	-2.078	-17.536	53.829	90.855
ALXN	Healthcare	-0.402	0.194	-4.155	-0.017	0.654	-3.288	-21.460	50.110	90.060
AMGN	Healthcare	1.484	1.212	14.365	0.061	0.654	-2.533	-11.856	57.269	90.258
BAX	Healthcare	0.078	0.012	0.231	0.003	0.654	-2.816	-18.881	54.861	85.885
BDX	Healthcare	-0.150	0.105	-1.736	-0.006	0.654	-3.258	-16.459	51.190	83.499

Continued on next page

Table C.2 – continued from previous page

Stocks	Sectors	Sharpe Ratio	MAR Ratio	CAGR	Mean	Std	Min	Max. DD	Acc.	Pct Trades
BIIB	Healthcare	-0.002	0.036	-0.500	-0.000	0.654	-2.556	-13.694	51.678	88.867
BMY	Healthcare	1.031	1.019	9.125	0.042	0.654	-2.988	-8.959	55.093	85.885
BSX	Healthcare	0.957	1.069	8.505	0.039	0.654	-4.379	-7.955	54.233	86.879
CAH	Healthcare	1.024	0.750	9.275	0.042	0.654	-3.253	-12.362	54.525	87.873
CELG	Healthcare	-0.124	0.093	-1.623	-0.005	0.654	-2.593	-17.434	50.776	89.662
CI	Healthcare	0.850	0.687	7.587	0.035	0.654	-3.268	-11.042	51.577	88.270
COO	Healthcare	0.549	0.276	4.643	0.023	0.654	-3.068	-16.794	54.299	87.873
CVS	Healthcare	0.119	0.048	0.600	0.005	0.654	-2.838	-12.400	52.791	85.487
DGX	Healthcare	2.603	1.647	24.748	0.107	0.654	-3.258	-15.028	60.238	83.499
DHR	Healthcare	0.008	0.020	-0.335	0.000	0.654	-3.199	-17.135	53.476	74.354
DVA	Healthcare	-0.607	0.214	-5.671	-0.025	0.654	-3.414	-26.545	52.214	85.288
ESRX	Healthcare	0.573	0.278	4.759	0.024	0.654	-3.821	-17.135	56.019	85.885
GILD	Healthcare	-0.238	0.158	-2.686	-0.010	0.654	-4.250	-16.990	50.220	90.258
HOLX	Healthcare	0.840	0.860	7.493	0.035	0.654	-3.138	-8.708	53.604	88.270
HSIC	Healthcare	1.333	0.596	12.216	0.055	0.654	-3.381	-20.487	55.275	86.680
HUM	Healthcare	-0.021	0.042	-0.655	-0.001	0.654	-5.529	-15.728	57.373	86.282
IDXX	Healthcare	0.949	0.360	8.731	0.039	0.654	-5.884	-24.227	55.531	89.861
INCY	Healthcare	1.120	0.669	10.649	0.046	0.654	-3.038	-15.927	53.159	91.252
JNJ	Healthcare	0.779	0.501	5.587	0.032	0.654	-3.694	-11.162	54.420	71.968
LH	Healthcare	1.772	0.977	16.706	0.073	0.654	-3.712	-17.104	58.391	86.481
LLY	Healthcare	0.786	0.753	6.769	0.032	0.654	-3.771	-8.988	54.861	85.885
MCK	Healthcare	0.093	0.025	0.374	0.004	0.654	-2.703	-14.727	52.294	86.680
MDT	Healthcare	-0.747	0.284	-6.476	-0.031	0.654	-3.067	-22.825	50.493	80.716
MRK	Healthcare	2.475	2.309	23.189	0.102	0.654	-2.663	-10.042	59.712	82.903
MTD	Healthcare	2.884	3.069	29.476	0.119	0.654	-2.287	-9.603	57.692	87.873
MYL	Healthcare	0.241	0.133	1.824	0.010	0.654	-2.868	-13.733	53.247	91.849
NKTR	Healthcare	0.460	0.244	3.938	0.019	0.654	-3.924	-16.135	55.773	91.252
PFE	Healthcare	1.842	1.660	16.442	0.076	0.654	-3.544	-9.908	57.039	81.909
PKI	Healthcare	0.378	0.327	2.901	0.016	0.654	-2.831	-8.877	52.941	84.493
PRGO	Healthcare	-1.260	0.379	-11.362	-0.052	0.654	-4.791	-29.962	53.483	88.469
REGN	Healthcare	-0.236	0.120	-2.659	-0.010	0.654	-3.164	-22.083	46.256	90.258
RMD	Healthcare	0.046	0.003	-0.056	0.002	0.654	-4.681	-18.769	51.364	87.475
SYK	Healthcare	-0.509	0.231	-4.515	-0.021	0.654	-2.477	-19.526	47.368	79.324
TMO	Healthcare	-0.274	0.160	-2.759	-0.011	0.654	-3.374	-17.287	49.760	82.704
UHS	Healthcare	0.948	1.061	8.679	0.039	0.654	-2.650	-8.181	53.556	89.463
UNH	Healthcare	1.766	2.507	16.726	0.073	0.654	-2.738	-6.671	53.776	86.879
VAR	Healthcare	0.624	0.332	5.095	0.026	0.654	-2.919	-15.331	53.207	83.698
VRTX	Healthcare	1.709	1.353	16.759	0.070	0.654	-2.617	-12.387	53.863	90.060
WAT	Healthcare	1.911	1.799	18.481	0.079	0.654	-3.946	-10.276	59.502	87.873
XRAY	Healthcare	1.896	1.056	18.002	0.078	0.654	-4.300	-17.051	56.782	86.481
AIV	Real Estate	1.881	1.875	18.342	0.078	0.654	-2.647	-9.780	56.278	88.668
ARE	Real Estate	1.430	0.906	13.365	0.059	0.654	-3.271	-14.750	53.968	87.674
AVB	Real Estate	2.378	3.567	23.699	0.098	0.654	-2.141	-6.644	56.208	88.072
BXP	Real Estate	0.883	1.017	7.896	0.036	0.654	-2.696	-7.766	55.079	88.072
CCI	Real Estate	0.675	0.288	5.687	0.028	0.654	-3.375	-19.735	53.953	85.487
DRE	Real Estate	1.270	1.165	12.127	0.052	0.654	-3.328	-10.408	54.066	90.457
EQR	Real Estate	1.784	2.234	17.245	0.074	0.654	-2.248	-7.721	56.180	88.469

Continued on next page

Table C.2 – continued from previous page

Stocks	Sectors	Sharpe Ratio	MAR Ratio	CAGR	Mean	Std	Min	Max. DD	Acc.	Pct Trades
ESS	Real Estate	2.544	3.486	26.827	0.105	0.654	-3.722	-7.696	58.009	91.849
FRT	Real Estate	1.110	0.655	10.205	0.046	0.654	-3.327	-15.571	52.135	88.469
GGP	Real Estate	-0.100	0.075	-1.434	-0.004	0.654	-7.480	-19.242	52.620	91.054
HCP	Real Estate	-0.181	0.222	-2.202	-0.007	0.654	-5.714	-9.924	52.484	92.048
HST	Real Estate	1.306	1.482	13.027	0.054	0.654	-3.620	-8.790	57.717	94.036
MAA	Real Estate	2.798	3.044	29.562	0.115	0.654	-2.215	-9.710	59.300	90.855
MAC	Real Estate	0.603	0.473	5.181	0.025	0.654	-4.185	-10.954	54.054	88.270
O	Real Estate	0.061	0.006	0.088	0.003	0.654	-3.587	-13.832	52.516	90.855
PLD	Real Estate	-0.880	0.426	-8.262	-0.036	0.654	-2.429	-19.389	45.982	89.066
PSA	Real Estate	0.998	0.588	9.448	0.041	0.654	-2.254	-16.058	51.948	91.849
REG	Real Estate	1.060	0.805	10.090	0.044	0.654	-2.839	-12.529	53.896	91.849
SLG	Real Estate	-0.058	0.065	-1.031	-0.002	0.654	-6.001	-15.872	53.829	90.855
SPG	Real Estate	0.936	0.877	8.802	0.039	0.654	-3.029	-10.038	51.515	91.849
UDR	Real Estate	0.536	0.502	4.554	0.022	0.654	-2.762	-9.076	51.121	88.668
VNO	Real Estate	1.461	1.261	13.953	0.060	0.654	-2.573	-11.067	54.566	89.264
VTR	Real Estate	-0.535	0.275	-5.352	-0.022	0.654	-2.886	-19.442	51.542	90.258
AAPL	Technology	-0.089	0.110	-1.174	-0.004	0.654	-2.845	-10.627	49.877	80.517
ADBE	Technology	1.209	0.904	10.980	0.050	0.654	-4.107	-12.147	57.798	86.680
ADI	Technology	0.820	0.463	7.393	0.034	0.654	-3.623	-15.960	55.556	89.463
ADSK	Technology	1.857	1.202	18.657	0.077	0.654	-3.173	-15.516	57.081	91.252
AKAM	Technology	0.988	0.561	9.215	0.041	0.654	-4.434	-16.426	53.509	90.656
AMAT	Technology	-0.002	0.031	-0.517	-0.000	0.654	-3.348	-16.448	53.333	92.445
AMD	Technology	-1.234	0.435	-11.764	-0.051	0.654	-5.561	-27.069	50.955	93.638
ANSS	Technology	2.474	2.811	25.099	0.102	0.654	-3.639	-8.928	58.482	89.066
APH	Technology	2.684	1.996	24.947	0.111	0.654	-2.395	-12.497	57.561	81.511
ATVI	Technology	-0.241	0.245	-2.801	-0.010	0.654	-3.025	-11.422	52.340	93.439
CA	Technology	2.099	1.214	19.672	0.087	0.654	-3.391	-16.201	54.588	84.493
CDNS	Technology	0.076	0.019	0.212	0.003	0.654	-5.286	-10.978	54.042	86.083
CERN	Technology	0.187	0.098	1.245	0.008	0.654	-2.664	-12.679	53.483	88.469
CSCO	Technology	2.753	2.259	27.377	0.113	0.654	-2.296	-12.120	56.221	86.282
CTSH	Technology	0.480	0.357	3.894	0.020	0.654	-4.028	-10.908	54.398	85.885
CTXS	Technology	1.827	2.381	17.921	0.075	0.654	-3.328	-7.527	56.444	89.463
EA	Technology	0.763	0.725	6.912	0.031	0.654	-2.293	-9.530	52.967	90.457
FFIV	Technology	2.723	2.144	27.871	0.112	0.654	-2.576	-13.001	58.296	88.668
FLIR	Technology	1.418	1.389	12.987	0.058	0.654	-4.010	-9.349	54.734	86.083
GLW	Technology	1.501	1.587	13.999	0.062	0.654	-4.529	-8.823	57.306	87.078
HPQ	Technology	0.417	0.363	3.429	0.017	0.654	-3.367	-9.453	53.468	88.867
HRS	Technology	1.368	1.431	12.395	0.056	0.654	-4.605	-8.664	56.512	85.487
IBM	Technology	1.450	1.669	12.469	0.060	0.654	-2.848	-7.472	55.283	80.915
INTC	Technology	2.522	2.312	25.784	0.104	0.654	-2.179	-11.151	56.889	89.463
INTU	Technology	1.436	2.450	12.847	0.059	0.654	-3.132	-5.245	53.901	84.095
IT	Technology	0.740	0.332	6.374	0.031	0.654	-5.614	-19.210	54.943	86.481
JNPR	Technology	0.230	0.101	1.615	0.009	0.654	-2.747	-15.935	54.128	86.680
KLAC	Technology	-0.534	0.312	-5.143	-0.022	0.654	-5.025	-16.479	52.064	86.680
LRCX	Technology	-0.667	0.356	-6.537	-0.028	0.654	-4.131	-18.363	49.451	90.457
MCHP	Technology	0.698	0.384	6.219	0.029	0.654	-4.474	-16.184	52.434	89.861
MSFT	Technology	1.078	0.896	9.633	0.044	0.654	-2.954	-10.756	54.147	86.282

Continued on next page

Table C.2 – continued from previous page

Stocks	Sectors	Sharpe Ratio	MAR Ratio	CAGR	Mean	Std	Min	Max. DD	Acc.	Pct Trades
MSI	Technology	1.338	1.428	12.484	0.055	0.654	-3.843	-8.741	54.853	88.072
MU	Technology	-0.486	0.403	-5.154	-0.020	0.654	-2.713	-12.796	48.950	94.632
NTAP	Technology	0.949	0.911	8.695	0.039	0.654	-4.313	-9.547	54.222	89.463
NVDA	Technology	-1.641	0.446	-14.740	-0.068	0.654	-3.927	-33.050	48.904	90.656
ORCL	Technology	1.171	1.029	9.988	0.048	0.654	-3.110	-9.703	53.398	81.909
QCOM	Technology	-0.368	0.233	-3.774	-0.015	0.654	-3.454	-16.205	54.505	88.270
RHT	Technology	0.793	0.467	7.196	0.033	0.654	-3.592	-15.403	55.286	90.258
SNPS	Technology	3.723	4.657	38.087	0.153	0.654	-2.530	-8.178	58.451	84.692
SWKS	Technology	0.515	0.316	4.481	0.021	0.654	-2.752	-14.166	51.747	91.054
SYMC	Technology	1.254	1.234	11.645	0.052	0.654	-3.208	-9.438	56.532	88.270
TTWO	Technology	0.000	0.040	-0.488	0.000	0.654	-4.495	-12.315	52.735	90.855
TXN	Technology	0.675	0.515	5.711	0.028	0.654	-3.326	-11.091	52.083	85.885
VRSN	Technology	2.752	1.707	28.711	0.113	0.654	-4.181	-16.822	59.161	90.060
WDC	Technology	0.304	0.197	2.440	0.013	0.654	-3.563	-12.400	53.348	92.048
XLNX	Technology	1.109	1.030	10.243	0.046	0.654	-2.960	-9.942	57.494	88.867
XRX	Technology	-0.938	0.392	-8.621	-0.039	0.654	-4.863	-21.975	48.526	87.674
AEE	Utilities	0.126	0.051	0.670	0.005	0.654	-2.207	-13.161	50.342	87.276
AEP	Utilities	-0.078	0.086	-1.172	-0.003	0.654	-2.699	-13.599	47.836	87.276
AES	Utilities	0.652	0.257	6.220	0.027	0.654	-2.405	-24.182	52.567	96.819
CMS	Utilities	0.787	0.689	6.780	0.032	0.654	-3.156	-9.843	52.083	85.885
CNP	Utilities	-0.066	0.060	-1.108	-0.003	0.654	-2.647	-18.413	49.673	91.252
D	Utilities	0.853	0.634	7.386	0.035	0.654	-2.451	-11.649	52.900	85.686
DTE	Utilities	0.047	0.002	-0.048	0.002	0.654	-2.473	-21.400	51.395	85.487
DUK	Utilities	-0.369	0.196	-3.702	-0.015	0.654	-2.141	-18.916	51.152	86.282
ED	Utilities	0.273	0.172	1.981	0.011	0.654	-2.516	-11.489	49.417	85.288
EIX	Utilities	-0.617	0.249	-6.157	-0.025	0.654	-8.420	-24.725	52.505	91.252
ES	Utilities	0.141	0.053	0.821	0.006	0.654	-2.045	-15.382	53.258	88.469
ETR	Utilities	1.030	0.676	9.694	0.042	0.654	-2.869	-14.330	53.712	91.054
EXC	Utilities	-0.344	0.147	-3.673	-0.014	0.654	-2.911	-24.944	50.655	91.054
FE	Utilities	-0.804	0.405	-7.944	-0.033	0.654	-2.438	-19.613	49.145	93.042
LNT	Utilities	0.250	0.093	1.819	0.010	0.654	-2.660	-19.646	50.567	87.674
NEE	Utilities	0.872	0.684	7.787	0.036	0.654	-2.627	-11.387	52.144	88.072
NI	Utilities	0.403	0.251	3.408	0.017	0.654	-2.542	-13.558	52.597	91.849
PCG	Utilities	-0.188	0.075	-2.180	-0.008	0.654	-6.094	-28.878	53.273	88.072
PEG	Utilities	0.427	0.188	3.558	0.018	0.654	-2.752	-18.957	51.991	89.861
PNW	Utilities	1.337	0.890	12.740	0.055	0.654	-2.437	-14.311	54.646	89.861
PPL	Utilities	0.236	0.093	1.689	0.010	0.654	-2.871	-18.165	51.247	87.674
SCG	Utilities	-0.992	0.279	-9.520	-0.041	0.654	-4.564	-34.085	50.431	92.247
SO	Utilities	0.810	0.457	7.029	0.033	0.654	-2.050	-15.380	54.378	86.282
SRE	Utilities	-0.404	0.194	-4.122	-0.017	0.654	-3.682	-21.291	51.678	88.867
WEC	Utilities	0.234	0.067	1.662	0.010	0.654	-2.246	-24.716	50.685	87.078
XEL	Utilities	-0.242	0.107	-2.582	-0.010	0.654	-3.299	-24.223	49.420	85.686

C.3 Data for statistical model comparison

Table C.3 describes the data used for the Non-parametric Friedman test.

Table C.3: **Non-parametric Friedman test**. The table consist of the Sharpe ratio, which is the metric chosen to rank five ($k = 5$) models (columns). According to the Friedman test [68], the stocks (rows) play the role of “datasets” or “trails” ($n = 388$) and the test input variable is represented by a matrix $X \in \mathbb{R}^{n\times k}$.

Stocks	GTN A_s	GAT A_s	GAT A_J	GTN A_J	LSTM
ALB	0.010	0.075	0.136	-0.087	-0.124
APD	2.247	1.636	2.180	2.175	2.186
ECL	0.579	0.736	0.745	1.403	0.331
EMN	2.963	2.523	2.706	2.528	2.675
FCX	-0.080	-0.487	-0.559	0.406	0.161
FMC	0.485	0.663	0.035	-0.001	0.271
IFF	0.883	1.074	1.087	1.043	1.084
MAS	0.042	-0.318	0.336	0.156	0.289
MLM	0.162	0.322	0.353	0.242	0.265
MOS	-1.114	-0.545	-0.613	-0.679	-0.563
NEM	-0.200	-0.244	-0.239	-0.631	-0.364
NUE	-0.301	0.094	-0.326	-0.044	-0.685
PPG	0.825	0.489	0.416	0.349	0.189
PX	0.750	1.039	0.696	0.861	0.850
SHW	0.236	0.225	0.539	0.703	-0.093
VMC	-0.404	-0.573	-0.385	-0.435	-0.732
WY	-1.688	-1.839	-1.755	-2.016	-1.639
ANDV	0.250	0.443	-0.225	-0.257	-0.136
APA	0.651	0.351	0.365	0.203	0.420
APC	0.980	0.623	1.018	1.256	0.841
COG	-0.115	-0.437	-0.189	-0.348	-0.610
COP	0.026	0.372	0.362	0.429	0.473
CVX	2.019	1.361	1.638	0.883	1.485
DVN	0.725	0.601	0.369	0.221	0.173
EOG	1.402	1.746	1.614	0.912	1.616
EQT	-0.058	-0.045	0.232	-0.121	0.374
HAL	1.578	1.454	1.114	1.093	1.539
HES	0.063	0.171	0.174	0.338	0.326
HFC	-0.021	-0.275	-0.395	-0.660	-0.255
HP	0.904	0.769	0.643	0.458	0.604
MRO	-0.114	0.316	0.372	0.160	0.081
NBL	0.621	0.467	0.670	0.759	0.623

Continued on next page

Table C.3 – continued from previous page

Stocks	GTN A_s	GAT A_s	GAT A_J	GTN A_J	LSTM
NFX	0.209	0.390	0.489	0.195	0.628
NOV	-0.040	0.388	-0.079	-0.155	0.370
OKE	0.612	0.772	0.672	0.252	0.553
OXY	0.883	0.883	0.952	0.730	0.502
PXD	1.310	1.453	1.800	1.354	1.444
SLB	2.518	2.601	1.856	1.903	1.829
VLO	1.249	1.167	1.030	1.097	1.029
WMB	-0.654	-0.915	-0.617	-0.466	-0.881
XOM	3.044	2.517	1.866	1.806	1.953
ADP	0.049	-0.096	-0.172	-0.220	-0.259
AME	1.277	1.555	1.571	1.510	1.200
AOS	2.278	1.987	2.450	2.394	2.391
ARNC	-1.026	-1.323	-1.475	-1.454	-1.476
AVY	1.647	1.169	1.437	1.737	1.232
BA	1.619	1.431	1.008	1.009	1.326
CAT	1.191	0.969	1.226	1.076	1.071
CHRW	2.353	2.326	2.464	2.329	2.627
CMI	1.189	1.365	1.633	1.397	1.737
CSX	0.340	-0.435	-0.472	-0.359	-0.274
CTAS	0.686	1.192	1.153	1.620	0.711
DE	0.723	0.675	0.962	0.970	0.695
DOV	0.700	0.791	0.994	0.913	0.896
EFX	0.279	-0.535	0.016	-0.290	-0.111
EMR	-0.384	0.078	-0.169	-0.412	-0.542
ETN	1.771	1.916	1.690	1.649	1.353
EXPD	3.949	3.736	4.319	3.666	4.162
FAST	0.840	0.927	0.954	0.898	0.786
FDX	0.078	0.593	0.223	-0.137	0.159
FISV	1.655	1.983	1.734	1.493	1.153
FLS	-0.178	-0.068	-0.269	-0.152	-0.564
GD	0.685	0.510	0.537	0.216	0.659
GE	-0.000	0.103	0.174	-0.080	0.106
GWW	-0.077	-0.064	-0.177	0.212	0.196
HON	1.946	2.569	2.969	2.233	2.019
IR	0.150	0.419	0.178	-0.152	0.184
IRM	-0.336	-0.431	-0.206	-0.354	-0.326
ITW	0.309	0.483	0.432	0.725	0.472
JBHT	1.127	1.381	0.964	1.064	1.311

Continued on next page

Table C.3 – continued from previous page

Stocks	GTN A_s	GAT A_s	GAT A_J	GTN A_J	LSTM
JCI	0.676	0.699	0.930	0.860	0.637
JEC	1.386	1.330	1.384	1.403	0.989
KSU	1.909	1.953	1.745	1.707	1.661
LLL	1.318	0.902	1.338	1.518	1.391
LMT	0.637	0.289	0.344	0.868	0.426
LUV	1.189	1.501	1.416	1.231	1.320
MMM	0.528	0.312	0.807	1.392	0.960
NOC	0.274	0.075	0.356	0.454	0.363
NSC	-0.052	0.369	0.105	-0.086	-0.541
PAYX	0.923	0.911	0.974	1.147	1.189
PCAR	1.097	1.133	0.775	0.787	0.606
PH	0.345	0.332	0.748	0.430	0.802
PNR	0.572	0.217	0.346	0.050	0.621
PWR	0.276	0.486	0.453	0.510	0.236
ROK	1.050	1.098	0.836	0.935	1.155
ROP	1.199	0.714	1.116	1.025	0.871
RSG	-0.317	0.460	0.186	0.447	-0.111
RTN	-0.296	0.146	-0.021	0.389	0.460
SNA	0.266	0.403	0.185	-0.114	0.100
SRCL	0.533	1.042	0.619	0.477	0.472
SWK	1.404	1.815	1.510	1.503	1.311
TXT	1.392	1.492	1.477	0.936	1.478
UNP	1.538	0.927	0.935	1.375	1.107
UPS	-0.415	-0.085	0.421	0.482	-0.425
URI	-0.045	0.962	0.575	0.304	0.609
UTX	0.702	0.951	0.672	1.077	0.408
WM	0.239	0.315	0.056	1.047	0.297
AMZN	-1.678	-1.451	-1.453	-1.352	-1.535
AZO	-0.390	-0.622	-0.927	-0.707	-1.196
BBY	-0.271	-1.062	-0.749	-1.091	-0.967
BKNG	0.806	0.573	0.616	0.829	0.934
BLL	0.983	0.858	1.280	0.937	1.188
BWA	0.218	0.241	0.231	-0.109	-0.086
CCL	0.102	-0.141	0.066	-0.181	0.042
CPRT	3.038	2.652	2.752	2.511	2.811
DHI	2.355	1.896	1.903	1.688	1.813
DIS	0.957	1.092	1.149	1.076	0.732
DRI	0.546	0.292	0.192	0.077	0.173

Continued on next page

Table C.3 – continued from previous page

Stocks	GTN A_s	GAT A_s	GAT A_J	GTN A_J	LSTM
EBAY	0.474	0.324	0.130	0.178	0.130
F	0.696	0.798	0.860	-0.193	0.435
FL	0.626	0.457	1.313	1.124	1.073
FOX	-0.236	-0.376	-0.138	-0.164	-0.112
FOXA	0.059	0.117	0.103	0.168	0.129
GPC	-0.478	-0.542	-0.608	-0.586	-0.693
GPS	-0.728	-0.516	-1.065	-0.349	-0.425
GT	2.381	1.939	2.303	1.998	1.894
HAS	0.383	0.387	0.449	0.603	0.122
HD	-0.126	-0.289	-0.490	-0.485	-0.620
HOG	0.926	0.904	0.638	0.761	0.993
HRB	-1.093	-1.665	-1.465	-1.381	-1.554
IP	0.386	0.102	0.225	-0.018	0.306
IPG	1.158	1.056	1.068	0.451	0.913
JWN	-0.784	-0.906	-0.755	-0.642	-0.825
KMX	-0.071	-0.161	0.015	-0.143	-0.323
KSS	0.120	0.138	0.474	-0.097	0.183
LB	-0.407	0.097	-0.202	-0.571	-0.362
LEG	0.007	-0.351	-0.077	-0.199	0.120
LEN	1.252	1.564	1.581	1.414	1.328
LOW	-0.342	-0.326	-0.539	-0.092	-0.071
M	0.408	-0.423	-0.000	-0.246	-0.025
MAR	0.316	-0.301	0.339	-0.173	0.207
MAT	-1.067	-1.276	-0.836	-0.854	-1.271
MCD	-0.044	-0.207	-0.322	0.489	-0.089
MGM	1.361	1.750	1.854	1.679	1.687
MHK	2.768	2.701	2.561	2.478	2.582
NKE	-0.024	-0.098	0.443	-0.437	-0.274
OMC	1.655	1.383	1.373	1.300	1.557
ORLY	0.798	0.395	0.085	0.140	0.368
PHM	1.668	2.312	1.994	2.089	1.923
PKG	0.483	0.570	0.740	0.361	0.368
PVH	1.673	1.308	1.546	1.494	1.525
RCL	-0.282	-0.667	-0.810	-0.616	-0.858
RL	0.550	0.069	0.314	0.545	0.376
ROST	1.395	1.617	1.415	1.951	1.686
SBUX	-0.576	-0.458	-0.512	-0.460	-0.985
SEE	0.657	0.272	0.203	-0.151	0.094

Continued on next page

Table C.3 – continued from previous page

Stocks	GTN A_s	GAT A_s	GAT A_J	GTN A_J	LSTM
TIF	1.049	0.706	1.323	0.837	1.243
TJX	0.237	-0.311	0.251	-0.061	-0.022
TSCO	0.449	0.605	0.420	0.282	0.162
VFC	0.181	0.351	0.129	-0.552	0.079
WHR	1.583	1.191	1.376	1.306	1.631
YUM	1.782	1.779	1.866	1.589	2.071
ADM	0.524	0.145	-0.007	0.244	0.421
BF-B	1.741	1.423	1.400	0.726	1.280
CAG	0.734	0.956	0.920	0.707	0.892
CHD	0.571	0.214	0.324	0.545	-0.094
CL	0.483	0.345	0.635	0.298	0.372
CLX	0.301	0.445	0.653	0.951	0.274
COST	-0.021	-0.052	-0.115	0.161	-0.193
CPB	0.215	-0.353	-0.384	0.051	-0.086
DLTR	1.329	0.941	1.197	1.187	0.946
EL	1.967	1.856	2.134	2.416	2.220
GIS	-0.375	-0.486	-0.621	-0.307	-0.701
HRL	-0.285	-0.413	-0.438	-0.276	-0.804
HSY	1.198	0.900	0.891	1.048	1.002
K	-0.472	-0.761	-0.775	-0.938	-0.841
KMB	1.474	1.354	1.208	1.322	1.210
KO	0.563	1.027	0.491	0.924	0.807
KR	-0.884	-1.225	-1.171	-1.295	-1.174
MKC	0.157	-0.093	0.294	0.185	0.075
MNST	0.756	1.147	0.959	0.880	0.610
MO	-0.043	0.040	0.041	-0.013	0.149
NWL	1.322	1.084	0.598	1.007	0.521
PEP	0.256	0.545	-0.213	0.248	0.169
PG	1.219	1.668	1.249	1.215	0.997
SJM	-0.221	-0.088	-0.656	0.039	-0.095
STZ	0.275	-0.052	0.214	0.165	0.230
SYY	-0.417	0.049	-0.143	0.195	-0.164
TAP	0.103	-0.617	-0.286	-0.300	-0.511
TGT	-0.039	-0.568	-0.534	-0.158	-0.514
TSN	0.078	0.333	-0.017	0.234	0.453
WBA	-0.717	-0.342	-0.279	-0.350	-0.168
WMT	-0.085	0.103	0.150	0.206	-0.199
AMT	-0.642	-0.693	-0.836	-0.444	-0.697

Continued on next page

Table C.3 – continued from previous page

Stocks	GTN A_s	GAT A_s	GAT A_J	GTN A_J	LSTM
CMCSA	-0.076	-0.146	-0.262	0.085	0.038
CTL	-0.808	-1.004	-1.191	-1.379	-1.198
DISH	0.219	-0.236	0.092	-0.346	0.087
SBAC	1.884	1.309	1.492	1.145	1.672
T	-0.006	-0.632	-0.381	-0.627	-0.412
VZ	0.375	-0.143	0.042	0.067	-0.360
AFL	2.678	2.004	2.921	2.832	2.682
AIG	0.981	1.075	1.098	0.851	0.865
AJG	0.719	1.205	0.912	0.922	0.327
ALL	0.792	0.858	1.596	1.659	1.487
AMG	1.634	0.988	0.832	0.896	0.809
AON	-0.663	0.454	0.382	-0.141	0.098
AXP	1.225	0.954	0.997	1.058	0.685
BAC	0.288	0.705	0.930	0.740	0.840
BBT	0.831	0.461	0.252	0.408	0.609
BEN	0.841	0.951	1.067	0.502	0.633
BK	0.555	0.402	0.834	0.705	0.384
BLK	1.802	2.192	1.671	1.750	2.128
BRK-B	0.521	0.062	0.370	0.356	0.126
C	1.454	1.839	1.729	1.063	1.700
CB	0.252	-0.032	0.346	0.057	0.205
CINF	0.785	0.454	0.555	0.230	0.532
CMA	1.333	1.269	1.366	0.938	1.645
COF	0.547	0.358	0.873	0.370	1.055
ETFC	2.687	2.672	2.332	2.424	2.553
FITB	1.366	1.030	0.980	1.015	1.168
GS	1.116	1.306	1.017	0.985	1.085
HBAN	1.508	1.486	1.281	1.525	1.440
HIG	0.396	0.404	0.609	0.522	0.435
IVZ	0.491	0.676	0.479	0.579	0.491
JEF	-0.441	-0.414	-0.632	-0.709	-0.490
JPM	1.036	1.266	1.921	1.905	1.644
KEY	1.948	1.917	1.907	1.974	2.177
L	0.897	0.816	0.979	0.825	0.208
LNC	2.234	1.947	1.958	1.500	2.169
MCO	0.670	0.505	0.862	0.746	0.895
MMC	0.016	-0.167	0.269	0.168	0.274
MS	1.132	0.925	1.343	1.301	1.614

Continued on next page

Table C.3 – continued from previous page

Stocks	GTN A_s	GAT A_s	GAT A_J	GTN A_J	LSTM
MTB	1.862	1.618	1.855	1.564	1.787
NTRS	3.607	3.697	3.154	3.211	3.004
PBCT	2.075	2.328	1.857	1.941	1.813
PNC	1.516	1.832	1.304	1.089	1.764
RE	0.611	0.427	1.131	1.161	1.365
RF	1.946	1.866	1.492	1.833	1.600
RJF	0.763	-0.033	0.393	0.288	0.481
SCHW	1.215	0.596	0.793	0.631	0.724
SIVB	2.137	1.862	1.794	1.793	2.330
STI	1.070	1.462	1.244	1.776	1.249
STT	1.620	1.512	1.262	1.257	1.207
TMK	3.798	4.406	3.891	3.812	3.503
TROW	2.477	1.802	2.172	1.943	1.788
TRV	0.597	0.368	0.467	0.682	0.415
TSS	0.868	0.383	0.713	0.156	0.368
UNM	1.384	1.454	1.682	1.573	1.678
USB	1.865	1.745	1.553	1.262	1.622
WFC	0.935	0.192	0.317	0.427	0.005
XL	-0.256	0.078	0.017	0.195	0.139
ZION	2.184	2.335	2.079	1.914	1.841
A	-0.334	-0.308	-0.719	-0.394	-0.781
ABC	1.294	0.914	1.207	0.966	1.166
ABMD	-0.136	-0.292	-0.092	-0.269	-0.170
ABT	-0.291	-0.268	-0.306	-0.260	-0.323
AET	-0.350	-0.407	-0.394	-0.762	-0.792
AGN	0.130	-0.405	-0.516	-0.095	0.030
ALXN	-0.402	-0.038	0.013	-0.433	-0.360
AMGN	1.484	1.357	1.442	1.335	1.491
BAX	0.078	0.281	0.143	-0.191	0.170
BDX	-0.150	0.115	-0.192	-0.398	-0.448
BIIB	-0.002	0.324	-0.022	-0.049	-0.259
BMY	1.031	0.820	0.859	0.975	1.155
BSX	0.957	0.591	0.752	0.587	0.848
CAH	1.024	1.042	0.778	0.304	0.673
CELG	-0.124	-0.268	-0.424	-0.421	-0.047
CI	0.850	1.120	0.811	0.877	0.504
COO	0.549	0.479	0.363	0.729	0.597
CVS	0.119	-0.369	0.102	0.380	0.212

Continued on next page

Table C.3 – continued from previous page

Stocks	GTN A_s	GAT A_s	GAT A_J	GTN A_J	LSTM
DGX	2.603	2.987	3.360	2.834	2.533
DHR	0.008	-0.303	0.006	0.081	-0.144
DVA	-0.607	-0.823	-0.967	-0.796	-0.827
ESRX	0.573	0.816	0.641	0.562	0.598
GILD	-0.238	-0.456	-0.530	0.166	-0.279
HOLX	0.840	0.794	0.800	0.766	0.732
HSIC	1.333	1.019	0.431	1.013	0.792
HUM	-0.021	-0.483	-0.080	-0.061	-0.151
IDXX	0.949	0.871	0.724	1.087	1.180
INCY	1.120	0.853	0.972	1.050	0.709
JNJ	0.779	1.428	1.004	0.860	1.051
LH	1.772	1.607	1.966	1.748	1.761
LLY	0.786	0.701	0.194	0.393	0.066
MCK	0.093	-0.211	-0.172	-0.271	-0.308
MDT	-0.747	-0.725	-0.950	-0.831	-0.519
MRK	2.475	2.575	2.011	2.491	2.196
MTD	2.884	2.983	2.639	2.482	2.855
MYL	0.241	-0.054	-0.311	-0.175	-0.198
NKTR	0.460	0.092	0.223	-0.012	0.091
PFE	1.842	1.813	1.564	1.658	1.763
PKI	0.378	-0.141	0.393	-0.246	0.233
PRGO	-1.260	-1.643	-1.045	-1.286	-1.151
REGN	-0.236	-0.258	-0.076	-0.267	-0.267
RMD	0.046	-0.471	-0.060	-0.309	-0.725
SYK	-0.509	0.041	-0.531	0.240	-0.258
TMO	-0.274	-0.272	-0.505	-0.430	-0.169
UHS	0.948	0.959	0.896	0.584	0.641
UNH	1.766	1.014	1.392	1.541	1.412
VAR	0.624	0.229	0.757	0.454	0.237
VRTX	1.709	1.605	1.796	1.598	1.359
WAT	1.911	1.742	1.980	2.255	1.700
XRAY	1.896	2.066	1.822	1.772	1.692
AIV	1.881	1.801	2.170	1.850	1.907
ARE	1.430	1.430	1.412	1.839	1.410
AVB	2.378	2.545	2.436	2.506	2.416
BXP	0.883	1.043	1.014	0.894	1.640
CCI	0.675	0.369	-0.040	0.558	0.467
DRE	1.270	1.450	1.382	1.564	1.222

Continued on next page

Table C.3 – continued from previous page

Stocks	GTN A_s	GAT A_s	GAT A_J	GTN A_J	LSTM
EQR	1.784	2.199	1.988	1.471	1.802
ESS	2.544	2.643	2.989	3.182	3.069
FRT	1.110	0.354	0.594	0.682	0.676
GGP	-0.100	0.116	0.175	0.217	-0.023
HCP	-0.181	-0.402	-0.457	-0.212	-0.455
HST	1.306	1.197	1.311	1.512	1.434
MAA	2.798	2.609	2.069	2.996	2.618
MAC	0.603	0.565	0.331	0.442	0.543
O	0.061	-0.208	-0.580	-0.089	-0.474
PLD	-0.880	-0.631	-0.861	-0.817	-1.077
PSA	0.998	0.612	0.374	0.941	0.633
REG	1.060	0.899	0.506	0.986	1.119
SLG	-0.058	-0.274	-0.470	-0.274	-0.049
SPG	0.936	1.184	0.739	1.192	1.026
UDR	0.536	0.701	0.429	0.878	0.822
VNO	1.461	1.420	1.359	1.110	0.751
VTR	-0.535	-1.059	-1.263	-0.511	-1.130
AAPL	-0.089	-0.180	0.040	0.088	-0.553
ADBE	1.209	1.032	0.975	0.726	1.122
ADI	0.820	0.845	1.076	1.249	0.448
ADSK	1.857	1.480	0.886	0.202	0.788
AKAM	0.988	1.198	1.429	1.235	0.944
AMAT	-0.002	-0.057	-0.140	-0.253	-0.280
AMD	-1.234	-1.129	-1.349	-1.641	-1.700
ANSS	2.474	3.252	2.794	2.847	2.633
APH	2.684	1.728	1.811	1.551	1.540
ATVI	-0.241	-0.466	-0.483	-0.369	-1.080
CA	2.099	2.357	2.032	2.399	1.555
CDNS	0.076	0.309	0.721	-0.014	0.326
CERN	0.187	0.700	0.451	0.962	0.512
CSCO	2.753	2.889	2.656	2.421	2.511
CTSH	0.480	0.875	0.219	0.126	0.678
CTXS	1.827	1.598	2.100	1.813	2.244
EA	0.763	0.229	0.591	0.663	0.427
FFIV	2.723	2.212	2.143	2.787	1.677
FLIR	1.418	1.945	1.967	1.874	1.818
GLW	1.501	1.094	1.306	0.905	1.377
HPQ	0.417	0.448	-0.010	0.037	-0.144

Continued on next page

Table C.3 – continued from previous page

Stocks	GTN A_s	GAT A_s	GAT A_J	GTN A_J	LSTM
HRS	1.368	0.961	0.918	1.302	1.133
IBM	1.450	1.263	0.904	0.881	1.083
INTC	2.522	2.529	2.807	2.153	2.657
INTU	1.436	1.730	1.666	1.939	0.986
IT	0.740	1.137	0.851	0.730	0.663
JNPR	0.230	0.227	0.136	-0.312	0.145
KLAC	-0.534	-0.451	-0.976	-0.984	-0.559
LRCX	-0.667	-1.164	-0.930	-0.790	-1.208
MCHP	0.698	0.684	1.167	1.037	0.885
MSFT	1.078	1.229	0.862	0.720	0.840
MSI	1.338	1.417	1.481	1.501	1.227
MU	-0.486	-0.285	-0.313	-0.455	0.040
NTAP	0.949	1.436	1.351	0.796	0.941
NVDA	-1.641	-1.341	-1.263	-1.465	-1.525
ORCL	1.171	1.435	1.857	1.967	1.087
QCOM	-0.368	-0.576	-0.747	-0.885	-0.582
RHT	0.793	0.479	0.868	0.425	0.879
SNPS	3.723	3.328	3.034	2.860	2.583
SWKS	0.515	0.383	0.504	0.638	0.542
SYMC	1.254	1.320	1.905	1.706	1.598
TTWO	0.000	-0.336	-0.263	-0.183	-0.223
TXN	0.675	0.852	0.925	0.147	0.473
VRSN	2.752	2.517	3.043	2.226	2.395
WDC	0.304	0.220	0.404	0.493	0.629
XLNX	1.109	0.759	1.070	0.632	0.390
XRX	-0.938	-0.821	-0.959	-0.595	-0.904
AEE	0.126	0.045	0.331	0.500	0.557
AEP	-0.078	0.148	0.395	0.298	0.411
AES	0.652	0.520	0.234	0.160	-0.018
CMS	0.787	1.113	0.959	1.560	1.042
CNP	-0.066	0.309	0.499	0.048	0.003
D	0.853	0.617	0.812	0.773	1.021
DTE	0.047	0.046	0.235	0.087	0.590
DUK	-0.369	-0.378	-0.517	-0.130	0.061
ED	0.273	0.139	-0.284	0.334	0.045
EIX	-0.617	-0.392	-0.704	-0.518	-0.212
ES	0.141	0.385	-0.059	0.702	0.235
ETR	1.030	0.946	1.209	0.897	1.200

Continued on next page

Table C.3 – continued from previous page

Stocks	GTN A_s	GAT A_s	GAT A_J	GTN A_J	LSTM
EXC	-0.344	-0.047	-0.023	-0.295	-0.290
FE	-0.804	-0.748	-0.488	-0.530	-0.718
LNT	0.250	0.493	0.389	0.230	-0.009
NEE	0.872	0.926	0.626	0.739	0.736
NI	0.403	0.868	1.103	0.768	0.551
PCG	-0.188	0.011	-0.034	0.127	-0.185
PEG	0.427	0.203	0.140	0.770	0.507
PNW	1.337	1.447	1.277	1.589	1.628
PPL	0.236	0.561	0.831	0.621	0.441
SCG	-0.992	-0.496	-0.806	-0.609	-0.832
SO	0.810	0.603	0.942	0.606	1.081
SRE	-0.404	0.092	0.106	-0.210	-0.176
WEC	0.234	0.305	0.065	0.535	0.646
XEL	-0.242	0.220	-0.023	-0.013	0.144

Sequence Encoder

Best Hyperparameter	Search Space
$d = 256$	$d = \{128, 256, 512, 1024\}$

Graph Transformer Network (GTN)

Best Hyperparameter	Search Space
$e_{\text{sf}} = d = 256$	–
$e_{\text{ns}} = d = 256$	–
$W^Q_{h=3} = 256 \times d_k = 512$	$d_k = \{256, 512, 1024\}$
$W^K_{h=3} = 256 \times d_k = 512$	
$W = 256 \times d' = 512$	$d' = \{256, 512, 1024\}$
$W^O = 3 * 512 \times d' = 512$	–

Task-specific layers

Best Hyperparameter	Search Space
$d_{\text{high}} = 1024$	$d_{\text{high}} = d_{\text{low}} = \{512, 1024, 2048\}$
$d_{\text{low}} = 1024$	

Table C.1: **Multi-task Graph Neural Network – best hyperparameters and respective search space**. Hyperparameters for each block of the full Multi-task Graph Neural Network proposed in Figure 4.5. In all our experiments we consider the number of heads $h = 3$. **Sequence Encoder**: d is the number of units of the LSTM that encodes the price history and is shared among all stocks. **Graph Transformer Network (GTN)**: the trainable embeddings (e_{sf} and e_{ns}) have dimensions that conform with d. The intrinsic dimension of the Transformer attention d_k is as in Equations (4.17c) and (4.17d). Finally, d' is the dimension associated with the graph weight matrix W and the output projection matrix W^O, as in Equations (4.17a) and (4.17b), respectively. **Task-specific layers**: d_{high} and d_{low} are the dimensions of the FC layers for the high and low predictions – specific to this layer, when searching the hyperparameter space, we restrict both layer weights to have the same dimension.

Appendix D

D.1 Deep RL networks and SL network hyperparameters

Table D.1 reports the best hyperparameter (first column) found using grid search, where the search space is reported in the second column. The first two table splits refer to the DQN and DDPG architectures in Figures 5.2 and 5.3, respectively. The last table split refers to the SL network architecture in Figure 5.4.

D.2 Pairs Screening

Table D.2 reports the result of the pairs screening process for each sector, as detailed in Section 5.4.1.

DQN

Best Hyperparameter	Search Space
$d = 512$	$\{128, 256, 512, 1024\}$
$d' = 32$	$\{16, 32, 128\}$
$d_{\text{ReLU}} = 1024$	$\{128, 256, 512, 1024, 2048\}$

DDPG

Best Hyperparameter	Search Space
$d = 256$ (actor)	$\{128, 256, 512\}$
$d' = 32$ (actor)	$\{16, 32, 128\}$
$d_{\text{ReLU}} = 512$ (actor)	$\{256, 512, 1024\}$
$d = 512$ (critic)	$\{128, 256, 512\}$
$d' = 32$ (critic)	$\{16, 32, 128\}$
$d_{\text{ReLU}} = 512$ (critic)	$\{256, 512, 1024\}$

SL

Best Hyperparameter	Search Space
$d = 512$	$\{128, 256, 512, 1024\}$
$d' = 32$	$\{16, 32, 128\}$
$d_{\text{ReLU}} = 1024$	$\{128, 256, 512, 1024, 2048\}$

Table D.1: **Deep RL networks and SL network – best hyperparameters and respective search space**. The Deep RL architectures (first and second table splits) are depicted in Figure 5.2 (**DQN**) and Figure 5.3 (**DDPG**). The **SL** network architecture (last table split) is depicted in Figure 5.4. For all architectures d is number of units of the LSTM layer (green blocks) that encode the stock pair history. d' the dimension of the embedding layer (blue blocks). Finally, d_{ReLU} is the number of units of the FC layer (yellow blocks). As in all hyperparameter tunning reported in this thesis, the best parameter is selected based on the validation set performance.

Table D.2: **Pairs Screening – sectors breakdown**

Sector	Cointegrated Pairs
Basic Materials	Ecolab Inc. (ECL) x International Flavors & Fragrances Inc. (IFF), International Flavors & Fragrances Inc. (IFF) x PPG Industries, Inc. (PPG), International Flavors & Fragrances Inc. (IFF) x Praxair, Inc. (PX), Albemarle Corporation (ALB) x The Sherwin-Williams Company (SHW), Ecolab Inc. (ECL) x The Sherwin-Williams Company (SHW), Ecolab Inc. (ECL) x PPG Industries, Inc. (PPG)
Communication Services	AT&T Inc. (T) x Verizon Communications Inc. (VZ), CenturyLink, Inc. (CTL) x AT&T Inc. (T), American Tower Corporation (REIT) (AMT) x SBA Communications Corporation (SBAC)
Consumer Cyclical	Leggett & Platt, Incorporated (LEG) x Whirlpool Corporation (WHR), Mohawk Industries, Inc. (MHK) x Whirlpool Corporation (WHR), Leggett & Platt, Incorporated (LEG) x Mohawk Industries, Inc. (MHK), Hasbro, Inc. (HAS) x Royal Caribbean Cruises Ltd. (RCL), Booking Holdings Inc. (BKNG) x Hasbro, Inc. (HAS), International Paper Company (IP) x Sealed Air Corporation (SEE), Ball Corporation (BLL) x Packaging Corporation of America (PKG), Darden Restaurants, Inc. (DRI) x YUM! Brands, Inc. (YUM), Darden Restaurants, Inc. (DRI) x Starbucks Corporation (SBUX), eBay Inc. (EBAY) x Genuine Parts Company (GPC), AutoZone, Inc. (AZO) x eBay Inc. (EBAY), Best Buy Co., Inc. (BBY) x eBay Inc. (EBAY), eBay Inc. (EBAY) x O'Reilly Automotive, Inc. (ORLY), Amazon.com, Inc. (AMZN) x eBay Inc. (EBAY), Advance Auto Parts, Inc. (AAP) x Genuine Parts Company (GPC), Advance Auto Parts, Inc. (AAP) x AutoZone, Inc. (AZO), Amazon.com, Inc. (AMZN) x O'Reilly Automotive, Inc. (ORLY), Amazon.com, Inc. (AMZN) x Genuine Parts Company (GPC)
Consumer Defensive	Colgate-Palmolive Company (CL) x The Clorox Company (CLX), The Clorox Company (CLX) x The Procter & Gamble Company (PG), McCormick & Company, Incorporated (MKC) x The J. M. Smucker Company (SJM), General Mills, Inc. (GIS) x The J. M. Smucker Company (SJM), Conagra Brands, Inc. (CAG) x Kellogg Company (K)

Continued on next page

Table D.2 – continued from previous page

Sector	Cointegrated Pairs
Energy	Anadarko Petroleum Corporation (APC) x Hess Corporation (HES), EQT Corporation (EQT) x Pioneer Natural Resources Company (PXD), Marathon Oil Corporation (MRO) x Noble Energy, Inc. (NBL), EOG Resources, Inc. (EOG) x Pioneer Natural Resources Company (PXD), ConocoPhillips (COP) x Pioneer Natural Resources Company (PXD), Devon Energy Corporation (DVN) x Newfield Exploration Company (NFX), Devon Energy Corporation (DVN) x Hess Corporation (HES), Apache Corporation (APA) x Devon Energy Corporation (DVN), Occidental Petroleum Corporation (OXY) x Pioneer Natural Resources Company (PXD), Anadarko Petroleum Corporation (APC) x Devon Energy Corporation (DVN), Halliburton Company (HAL) x Schlumberger Limited (SLB), Halliburton Company (HAL) x National Oilwell Varco, Inc. (NOV)

Continued on next page

Table D.2 – continued from previous page

Sector	Cointegrated Pairs
Financial Services	Northern Trust Corporation (NTRS) x State Street Corporation (STT), BlackRock, Inc. (BLK) x T. Rowe Price Group, Inc. (TROW), The Bank of New York Mellon Corporation (BK) x Northern Trust Corporation (NTRS), The Bank of New York Mellon Corporation (BK) x State Street Corporation (STT), The Bank of New York Mellon Corporation (BK) x BlackRock, Inc. (BLK), The Bank of New York Mellon Corporation (BK) x Invesco Ltd. (IVZ), BlackRock, Inc. (BLK) x Northern Trust Corporation (NTRS), Franklin Resources, Inc. (BEN) x Jefferies Financial Group Inc. (JEF), Affiliated Managers Group, Inc. (AMG) x Franklin Resources, Inc. (BEN), Huntington Bancshares Incorporated (HBAN) x KeyCorp (KEY), KeyCorp (KEY) x Regions Financial Corporation (RF), Fifth Third Bancorp (FITB) x KeyCorp (KEY), SunTrust Banks, Inc. (STI) x Zions Bancorporation (ZION), Huntington Bancshares Incorporated (HBAN) x Regions Financial Corporation (RF), BB&T Corporation (BBT) x M&T Bank Corporation (MTB), SVB Financial Group (SIVB) x U.S. Bancorp (USB), KeyCorp (KEY) x Zions Bancorporation (ZION), The PNC Financial Services Group, Inc. (PNC) x U.S. Bancorp (USB), BB&T Corporation (BBT) x The PNC Financial Services Group, Inc. (PNC), M&T Bank Corporation (MTB) x U.S. Bancorp (USB), Fifth Third Bancorp (FITB) x Zions Bancorporation (ZION), KeyCorp (KEY) x SunTrust Banks, Inc. (STI), BB&T Corporation (BBT) x U.S. Bancorp (USB), Comerica Incorporated (CMA) x U.S. Bancorp (USB), Huntington Bancshares Incorporated (HBAN) x Zions Bancorporation (ZION), SunTrust Banks, Inc. (STI) x U.S. Bancorp (USB), M&T Bank Corporation (MTB) x The PNC Financial Services Group, Inc. (PNC), Fifth Third Bancorp (FITB) x SunTrust Banks, Inc. (STI), Fifth Third Bancorp (FITB) x Regions Financial Corporation (RF), The Charles Schwab Corporation (SCHW) x S&P Global Inc. (SPGI), Moody's Corporation (MCO) x S&P Global Inc. (SPGI), Moody's Corporation (MCO) x Raymond James Financial, Inc. (RJF), Raymond James Financial, Inc. (RJF) x S&P Global Inc. (SPGI), Lincoln National Corporation (LNC) x Principal Financial Group, Inc. (PFG), Aflac Incorporated (AFL) x Prudential Financial, Inc. (PRU), Prudential Financial, Inc. (PRU) x Unum Group (UNM), MetLife, Inc. (MET) x Prudential Financial, Inc. (PRU), Lincoln National Corporation (LNC) x Prudential Financial, Inc. (PRU), Aflac Incorporated (AFL) x Torchmark Corporation (TMK), Lincoln National Corporation (LNC) x MetLife, Inc. (MET)

Continued on next page

Table D.2 – continued from previous page

Sector	Cointegrated Pairs
Healthcare	Amgen Inc. (AMGN) x Biogen Inc. (BIIB), Biogen Inc. (BIIB) x Vertex Pharmaceuticals Incorporated (VRTX), Celgene Corporation (CELG) x Gilead Sciences, Inc. (GILD), Amgen Inc. (AMGN) x Vertex Pharmaceuticals Incorporated (VRTX), Biogen Inc. (BIIB) x Regeneron Pharmaceuticals, Inc. (REGN), Mettler-Toledo International Inc. (MTD) x Thermo Fisher Scientific Inc. (TMO), Agilent Technologies, Inc. (A) x Quest Diagnostics Incorporated (DGX), Agilent Technologies, Inc. (A) x Laboratory Corporation of America Holdings (LH), Agilent Technologies, Inc. (A) x Waters Corporation (WAT), Agilent Technologies, Inc. (A) x PerkinElmer, Inc. (PKI), Quest Diagnostics Incorporated (DGX) x Laboratory Corporation of America Holdings (LH), PerkinElmer, Inc. (PKI) x Waters Corporation (WAT), Agilent Technologies, Inc. (A) x Danaher Corporation (DHR), Agilent Technologies, Inc. (A) x Mettler-Toledo International Inc. (MTD), Danaher Corporation (DHR) x Laboratory Corporation of America Holdings (LH), Thermo Fisher Scientific Inc. (TMO) x Waters Corporation (WAT), Laboratory Corporation of America Holdings (LH) x Waters Corporation (WAT), Danaher Corporation (DHR) x Mettler-Toledo International Inc. (MTD), Agilent Technologies, Inc. (A) x IDEXX Laboratories, Inc. (IDXX), Agilent Technologies, Inc. (A) x Thermo Fisher Scientific Inc. (TMO), Quest Diagnostics Incorporated (DGX) x Thermo Fisher Scientific Inc. (TMO), Mettler-Toledo International Inc. (MTD) x Waters Corporation (WAT), Laboratory Corporation of America Holdings (LH) x PerkinElmer, Inc. (PKI), Eli Lilly and Company (LLY) x Merck & Co., Inc. (MRK), Eli Lilly and Company (LLY) x Pfizer Inc. (PFE), Merck & Co., Inc. (MRK) x Pfizer Inc. (PFE), Bristol-Myers Squibb Company (BMY) x Merck & Co., Inc. (MRK), Aetna Inc. (AET) x Humana Inc. (HUM), Cigna Corporation (CI) x CVS Health Corporation (CVS), Anthem, Inc. (ANTM) x Cigna Corporation (CI), Cigna Corporation (CI) x Centene Corporation (CNC), Medtronic plc (MDT) x Zimmer Biomet Holdings, Inc. (ZBH), Align Technology, Inc. (ALGN) x Edwards Lifesciences Corporation (EW), ABIOMED, Inc. (ABMD) x Align Technology, Inc. (ALGN)

Continued on next page

Table D.2 – continued from previous page

Sector	Cointegrated Pairs
Industrials	Lockheed Martin Corporation (LMT) x Northrop Grumman Corporation (NOC), L3 Technologies, Inc. (LLL) x Lockheed Martin Corporation (LMT), General Dynamics Corporation (GD) x L3 Technologies, Inc. (LLL), The Boeing Company (BA) x L3 Technologies, Inc. (LLL), L3 Technologies, Inc. (LLL) x United Technologies Corporation (UTX), L3 Technologies, Inc. (LLL) x Raytheon Company (RTN), L3 Technologies, Inc. (LLL) x Northrop Grumman Corporation (NOC), General Dynamics Corporation (GD) x United Technologies Corporation (UTX), General Dynamics Corporation (GD) x Lockheed Martin Corporation (LMT), The Boeing Company (BA) x Raytheon Company (RTN), The Boeing Company (BA) x Northrop Grumman Corporation (NOC), L3 Technologies, Inc. (LLL) x Textron Inc. (TXT), Lockheed Martin Corporation (LMT) x United Technologies Corporation (UTX), The Boeing Company (BA) x Lockheed Martin Corporation (LMT), Rockwell Collins, Inc. (COL) x United Technologies Corporation (UTX), Rockwell Collins, Inc. (COL) x General Dynamics Corporation (GD), Automatic Data Processing, Inc. (ADP) x Paychex, Inc. (PAYX), Automatic Data Processing, Inc. (ADP) x Global Payments Inc. (GPN), Cintas Corporation (CTAS) x Global Payments Inc. (GPN), Cintas Corporation (CTAS) x Paychex, Inc. (PAYX), Fiserv, Inc. (FISV) x Global Payments Inc. (GPN), Equifax Inc. (EFX) x Iron Mountain Incorporated (IRM), Fidelity National Information Services, Inc. (FIS) x Fiserv, Inc. (FISV), Fidelity National Information Services, Inc. (FIS) x Iron Mountain Incorporated (IRM), Equifax Inc. (EFX) x Paychex, Inc. (PAYX), Equifax Inc. (EFX) x Fidelity National Information Services, Inc. (FIS), Automatic Data Processing, Inc. (ADP) x Fidelity National Information Services, Inc. (FIS), Global Payments Inc. (GPN) x Paychex, Inc. (PAYX), Automatic Data Processing, Inc. (ADP) x Iron Mountain Incorporated (IRM), Fidelity National Information Services, Inc. (FIS) x Global Payments Inc. (GPN), Automatic Data Processing, Inc. (ADP) x Equifax Inc. (EFX), Eaton Corporation plc (ETN) x Parker-Hannifin Corporation (PH), Eaton Corporation plc (ETN) x Pentair plc (PNR), Ingersoll-Rand Plc (IR) x Illinois Tool Works Inc. (ITW), Eaton Corporation plc (ETN) x Ingersoll-Rand Plc (IR), Parker-Hannifin Corporation (PH) x Pentair plc (PNR), Cummins Inc. (CMI) x Flowserve Corporation (FLS), Rockwell Automation Inc. (ROK) x Roper Technologies, Inc. (ROP), Ingersoll-Rand Plc (IR) x Parker-Hannifin Corporation (PH), Dover Corporation (DOV) x Honeywell International Inc. (HON), Illinois Tool Works Inc. (ITW) x 3M Company (MMM)

Continued on next page

Table D.2 – continued from previous page

Sector	Cointegrated Pairs
Industrials *continued*	Ingersoll-Rand Plc (IR) x Pentair plc (PNR), Dover Corporation (DOV) x Illinois Tool Works Inc. (ITW), Dover Corporation (DOV) x 3M Company (MMM), Eaton Corporation plc (ETN) x Rockwell Automation Inc. (ROK), Dover Corporation (DOV) x Emerson Electric Co. (EMR), Pentair plc (PNR) x Rockwell Automation Inc. (ROK), Dover Corporation (DOV) x Ingersoll-Rand Plc (IR), 3M Company (MMM) x Pentair plc (PNR), Ingersoll-Rand Plc (IR) x 3M Company (MMM), Honeywell International Inc. (HON) x Illinois Tool Works Inc. (ITW), Illinois Tool Works Inc. (ITW) x Parker-Hannifin Corporation (PH), Fluor Corporation (FLR) x Quanta Services, Inc. (PWR), Fluor Corporation (FLR) x Johnson Controls International plc (JCI), Johnson Controls International plc (JCI) x Quanta Services, Inc. (PWR), Fluor Corporation (FLR) x Jacobs Engineering Group Inc. (JEC), C.H. Robinson Worldwide, Inc. (CHRW) x United Parcel Service, Inc. (UPS), Expeditors International of Washington, Inc. (EXPD) x FedEx Corporation (FDX), FedEx Corporation (FDX) x United Parcel Service, Inc. (UPS), Expeditors International of Washington, Inc. (EXPD) x United Parcel Service, Inc. (UPS), C.H. Robinson Worldwide, Inc. (CHRW) x Expeditors International of Washington, Inc. (EXPD)
Real Estate	Equity Residential (EQR) x Mid-America Apartment Communities, Inc. (MAA), Equity Residential (EQR) x Essex Property Trust, Inc. (ESS), Macerich Company (MAC) x Regency Centers Corporation (REG), Federal Realty Investment Trust (FRT) x Macerich Company (MAC)

Continued on next page

Table D.2 – continued from previous page

Sector	Cointegrated Pairs
Technology	Cognizant Technology Solutions Corporation (CTSH) x Gartner, Inc. (IT), Cognizant Technology Solutions Corporation (CTSH) x Xerox Corporation (XRX), Accenture plc (ACN) x Gartner, Inc. (IT), Applied Materials, Inc. (AMAT) x KLA-Tencor Corporation (KLAC), Applied Materials, Inc. (AMAT) x Lam Research Corporation (LRCX), Analog Devices, Inc. (ADI) x Xilinx, Inc. (XLNX), Texas Instruments Incorporated (TXN) x Xilinx, Inc. (XLNX), Analog Devices, Inc. (ADI) x Texas Instruments Incorporated (TXN), Intel Corporation (INTC) x Xilinx, Inc. (XLNX), Microchip Technology Incorporated (MCHP) x QUALCOMM Incorporated (QCOM), Analog Devices, Inc. (ADI) x Intel Corporation (INTC), Akamai Technologies, Inc. (AKAM) x ANSYS, Inc. (ANSS), Autodesk, Inc. (ADSK) x Akamai Technologies, Inc. (AKAM), Autodesk, Inc. (ADSK) x Red Hat, Inc. (RHT), Adobe Systems Incorporated (ADBE) x Intuit Inc. (INTU), ANSYS, Inc. (ANSS) x Red Hat, Inc. (RHT), Adobe Systems Incorporated (ADBE) x Red Hat, Inc. (RHT), Akamai Technologies, Inc. (AKAM) x Symantec Corporation (SYMC), Intuit Inc. (INTU) x Red Hat, Inc. (RHT), Adobe Systems Incorporated (ADBE) x Akamai Technologies, Inc. (AKAM), Akamai Technologies, Inc. (AKAM) x Intuit Inc. (INTU), Red Hat, Inc. (RHT) x Symantec Corporation (SYMC), Adobe Systems Incorporated (ADBE) x Autodesk, Inc. (ADSK), Autodesk, Inc. (ADSK) x Intuit Inc. (INTU), Citrix Systems, Inc. (CTXS) x Oracle Corporation (ORCL), Citrix Systems, Inc. (CTXS) x Microsoft Corporation (MSFT), Citrix Systems, Inc. (CTXS) x F5 Networks, Inc. (FFIV), CA, Inc. (CA) x Citrix Systems, Inc. (CTXS)

Continued on next page

Table D.2 – continued from previous page

Sector	Cointegrated Pairs
Utilities	Exelon Corporation (EXC) x FirstEnergy Corp. (FE), Public Service Enterprise Group Incorporated (PEG) x Sempra Energy (SRE), Dominion Energy, Inc. (D) x Sempra Energy (SRE), Entergy Corporation (ETR) x FirstEnergy Corp. (FE), Eversource Energy (ES) x NextEra Energy, Inc. (NEE), Eversource Energy (ES) x Xcel Energy Inc. (XEL), DTE Energy Company (DTE) x Eversource Energy (ES), Eversource Energy (ES) x Alliant Energy Corporation (LNT), American Electric Power Company, Inc. (AEP) x Pinnacle West Capital Corporation (PNW), Eversource Energy (ES) x PPL Corporation (PPL), DTE Energy Company (DTE) x Xcel Energy Inc. (XEL), Edison International (EIX) x PG&E Corporation (PCG), PG&E Corporation (PCG) x SCANA Corporation (SCG), Eversource Energy (ES) x The Southern Company (SO), Eversource Energy (ES) x Pinnacle West Capital Corporation (PNW), Pinnacle West Capital Corporation (PNW) x Xcel Energy Inc. (XEL), Consolidated Edison, Inc. (ED) x Pinnacle West Capital Corporation (PNW), DTE Energy Company (DTE) x Consolidated Edison, Inc. (ED), Edison International (EIX) x Eversource Energy (ES), DTE Energy Company (DTE) x Pinnacle West Capital Corporation (PNW), Consolidated Edison, Inc. (ED) x Eversource Energy (ES), Eversource Energy (ES) x WEC Energy Group, Inc. (WEC)

List of Abbreviations

OHLC	**O**pen, **H**igh, **L**ow and **C**lose prices
TL	**T**ransfer **L**earning
FF3	**F**ama-**F**rench three factors
EMH	**E**fficient **M**arket **H**ypothesis
MHAN	**M**ultimodal **H**ierarchical **A**ttention **N**etwork
NRA	**N**ews **R**elevance **A**ttention
NTA	**N**ews **T**emporal **A**ttention
RCV1	**R**euters **C**orpus **V**olume I dataset
SNLI	**S**tanford **N**atural **L**anguage **I**nference dataset
SOTA	**S**tate **O**f **T**he **A**rt
GARCH	**G**eneralized **A**uto**R**egressive **C**onditional **H**eteroskedasticity model
ARCH	**A**uto**R**egressive **C**onditional **H**eteroskedasticity model
ML	**M**achine **L**earning
NLP	**N**atural **L**anguage **P**reprocessing
RL	**R**einforcement **L**earning
GNN	**G**raph **N**eural **N**etworks
GTN	**G**raph **T**ansformer **N**etworks
GCN	**G**raph **C**onvolutional **N**etwork
GAT	**G**raph **N**attention **N**etworks
MLP	**M**ulti**L**ayer **P**erceptron
LSTM	**L**ong **S**hort-**T**erm **M**emory neural network
MDD	**M**aximum **D**draw**D**own
RNN	**R**ecurrent **N**eural **N**etwork
MTL	**M**ulti-**T**ask **L**earning
MDD	**M**aximum **D**raw**D**own
TP	**T**rue **P**ositives
TN	**T**rue **N**egatives
BAH	**B**uy **A**and **H**old portfolio
KDE	**K**ernel **D**ensity **E**stimation
ISIP	**I**nvestment **S**trategy with **I**nvestors' **P**references
SL	**S**upervised **L**earning
DQN	**D**epp **Q**-**N**etwork
DDPG	**D**eep **D**terministic **P**olicy **G**radient

Bibliography

[1] Revisions to the Basel II market risk framework. Technical report, Bank for International Settlements (BIS), 2011. URL `https://www.bis.org/publ/bcbs193.htm`.

[2] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, dec 1974. ISSN 0018-9286. doi: 10.1109/TAC.1974.1100705. URL `http://ieeexplore.ieee.org/document/1100705/`.

[3] J. Alberg and Z. C. Lipton. Improving Factor-Based Quantitative Investing by Forecasting Company Fundamentals. In *31st Conference on Neural Information Processing Systems (NIPS)*, nov 2017. URL `http://arxiv.org/abs/1711.04837`.

[4] I. Aldridge. *High-frequency trading : a practical guide to algorithmic strategies and trading systems.* 2013. ISBN 1118343506. URL `https://books.google.co.uk/books?hl=en{&}lr={&}id=6l0DDQAAQBAJ{&}oi=fnd{&}pg=PP11{&}dq={%}22high+frequency+trading:+a+practical{%}22{&}ots=E9qUEzP-Ae{&}sig=i5s3EA9qibNoEU91twlq3BquXsw{&}redir{_}esc=y{#}v=onepage{&}q={%}22highfrequencytrading{%}3Aapractical{%}22{&}f=false`.

[5] G. J. Alexander. On Back-Testing "Zero-Investment" Strategies. *The Journal of Business*, 73(2):255–278, apr 2000. doi: 10.1086/209642. URL `https://www.jstor.org/stable/10.1086/209642`.

[6] S. Almahdi and S. Y. Yang. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, 87:267–279, nov 2017. doi: 10.1016/J.ESWA.2017.06.023. URL `https://www.sciencedirect.com/science/article/pii/S0957417417304402`.

[7] T. G. Andersen and T. Bollerslev. Answering the Skeptics: Yes, Standard Volatility Models do Provide Accurate Forecasts. *International Economic Review*, 39(4):885, nov 1998. doi: 10.2307/2527343. URL `https://www.jstor.org/stable/2527343?origin=crossref`.

[8] C. Antoniou, J. A. Doukas, and A. Subrahmanyam. Cognitive Dissonance, Sentiment, and Momentum. *Journal of Financial and Quantitative Analysis*,

48(01):245–275, feb 2013. doi: 10.1017/S0022109012000592. URL `http://www.journals.cambridge.org/abstract{_}S0022109012000592`.

[9] W. Antweiler and M. Z. Frank. Is All That Talk Just Noise? The Information Content of Internet Stock Message Boards. *The Journal of Finance*, 59(3): 1259–1294, nov 2004. URL `http://www.jstor.org/stable/info/3694736`.

[10] C. S. Asness. The Interaction of Value and Momentum Strategies. *Financial Analysts Journal*, 53(2):29–36, mar 1997. doi: 10.2469/faj.v53.n2.2069. URL `https://www.tandfonline.com/doi/full/10.2469/faj.v53.n2.2069`.

[11] G. S. Atsalakis and K. P. Valavanis. Surveying stock market forecasting techniques – Part II: Soft computing methods. *Expert Systems with Applications*, 36(3):5932–5941, apr 2009. ISSN 09574174. doi: 10.1016/j.eswa.2008.07.006. URL `http://www.sciencedirect.com/science/article/pii/S0957417408004417`.

[12] G. S. Atsalakis, E. E. Protopapadakis, and K. P. Valavanis. Stock trend forecasting in turbulent market periods using neuro-fuzzy systems. *Operational Research*, 16(2):245–269, jul 2016. doi: 10.1007/s12351-015-0197-6. URL `http://link.springer.com/10.1007/s12351-015-0197-6`.

[13] D. Bagnell and A. Ng. On Local Rewards and Scaling Distributed Reinforcement Learning. *Nips*, 18:91, 2006. ISSN 10495258. URL `https://papers.nips.cc/paper/2951-on-local-rewards-and-scaling-distributed-reinforcement-learninghttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.75.1532{&}rep=rep1{&}type=pdf`.

[14] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *ICLR*, sep 2015. URL `http://arxiv.org/abs/1409.0473`.

[15] T. Baltrušaitis, C. Ahuja, and L.-P. Morency. Multimodal Machine Learning: A Survey and Taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443, may 2019. URL `http://arxiv.org/abs/1705.09406`.

[16] R. W. Banz. The relationship between return and market value of common stocks. *Journal of Financial Economics*, 9(1):3–18, mar 1981. ISSN 0304-405X. doi: 10.1016/0304-405X(81)90018-0. URL `https://www.sciencedirect.com/science/article/pii/0304405X81900180`.

[17] L. Barras, O. Scaillet, and R. Wermers. False Discoveries in Mutual Fund Performance: Measuring Luck in Estimated Alphas. *The Journal of Finance*, 65(1):179–216, feb 2010. doi: 10.1111/j.1540-6261.2009.01527.x. URL `http://doi.wiley.com/10.1111/j.1540-6261.2009.01527.x`.

[18] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks. *ArXiv*, 2018. URL `https://arxiv.org/pdf/1806.01261.pdf`.

[19] S. Battiston, J. D. Farmer, A. Flache, D. Garlaschelli, A. G. Haldane, H. Heesterbeek, C. Hommes, C. Jaeger, R. May, and M. Scheffer. Complexity theory and financial regulation. *Science*, 351(6275), 2016.

[20] J. Baxter. A Model of Inductive Bias Learning. *Journal Of Artificial Intelligence Research*, 12:149–198, 2000. doi: 10.1613/jair.731. URL `http://arxiv.org/abs/1106.0245`.

[21] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research*, 7(Nov):2399–2434, 2006. URL `http://www.jmlr.org/papers/v7/belkin06a.html`.

[22] Y. Bengio, A. Courville, and P. Vincent. Representation learning: a review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–828, aug 2013. ISSN 1939-3539. doi: 10.1109/TPAMI.2013.50. URL `http://www.ncbi.nlm.nih.gov/pubmed/23787338`.

[23] F. Bertoluzzo and M. Corazza. Making Financial Trading by Recurrent Reinforcement Learning. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 619–626. Springer, Berlin, Heidelberg, 2007. doi: 10.1007/978-3-540-74827-4_78. URL `http://link.springer.com/10.1007/978-3-540-74827-4{_}78`.

[24] T. Bogomolov. Pairs Trading in the Land Down Under. In *Finance and Corporate Governance Conference*, nov 2011. doi: 10.2139/ssrn.1717295. URL `http://www.ssrn.com/abstract=1717295`.

[25] J. Bollen, H. Mao, and X.-J. Zeng. Twitter Mood Predicts the Stock Market. *Journal of Computational Science*, 2(1):1–8, 2011. URL `http://www.sciencedirect.com/science/article/pii/S187775031100007X`.

[26] T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, apr 1986. doi: 10.1016/0304-4076(86)90063-1. URL `https://www.sciencedirect.com/science/article/pii/0304407686900631`.

[27] L. Bottou. On-Line Learning and Stochastic Approximations. In D. Saad, editor, *On-Line Learning in Neural Networks*, chapter 2,

page 398. Cambridge University Press, 1998. ISBN 0521117917. URL https://www.cambridge.org/core/books/online-learning-in-neural-networks/7D901F98C2A4F1CF69648FDAEF6877CD.

[28] J. Boudoukh, R. Feldman, S. Kogan, and M. Richardson. Which News Moves Stock Prices? A Textual Analysis. *NBER Working Paper*, jan 2013. URL http://www.nber.org/papers/w18725.

[29] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Stroudsburg, PA, USA, 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL http://aclweb.org/anthology/D15-1075.

[30] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *Arxiv*, jun 2016. URL http://arxiv.org/abs/1606.01540.

[31] J. P. Broussard and M. Vaihekoski. Profitability of pairs trading strategy in an illiquid market with multiple share classes. *Journal of International Financial Markets, Institutions and Money*, 22(5):1188–1201, dec 2012. doi: 10.1016/J.INTFIN.2012.06.002. URL https://www.sciencedirect.com/science/article/pii/S1042443112000583.

[32] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*, dec 2013. URL http://arxiv.org/abs/1312.6203.

[33] J. Caldeira and G. V. Moura. Selection of a Portfolio of Pairs Based on Cointegration: A Statistical Arbitrage Strategy. *SSRN Electronic Journal*, jan 2013. ISSN 1556-5068. doi: 10.2139/ssrn.2196391. URL http://www.ssrn.com/abstract=2196391.

[34] H. R. Campbell, E. Hoyle, R. Korgaonkar, S. Rattray, M. Sargaison, and O. V. Hemert. The Impact of Volatility Targeting. *The Journal of Portfolio Management*, 45(1):14–33, nov 2017. doi: 10.3905/jpm.2018.45.1.014. URL http://jpm.pm-research.com/lookup/doi/10.3905/jpm.2018.45.1.014.

[35] M. M. Carhart. On Persistence in Mutual Fund Performance. *The Journal of Finance*, 52(1):57, mar 1997. ISSN 00221082. doi: 10.2307/2329556. URL http://www.jstor.org/stable/info/2329556.

[36] R. A. Caruana. Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *ICML*, pages 41–48. Elsevier, 1993. doi: 10.1016/B978-1-55860-307-3.50012-5. URL https://linkinghub.elsevier.com/retrieve/pii/B9781558603073500125.

[37] 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*, 55:194–211, aug 2016. doi: 10.1016/J.ESWA.2016.02.006. URL `https://www.sciencedirect.com/science/article/pii/S095741741630029X`.

[38] W. S. Chan. Stock price reaction to news and no-news: drift and reversal after headlines. *Journal of Financial Economics*, 70(2):223–260, nov 2003. ISSN 0304405X. doi: 10.1016/S0304-405X(03)00146-6. URL `http://www.sciencedirect.com/science/article/pii/S0304405X03001466`.

[39] Y. Chen, Z. Wei, and X. Huang. Incorporating Corporation Relationship via Graph Convolutional Neural Networks for Stock Price Prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management - CIKM '18*, pages 1655–1658, New York, New York, USA, 2018. ACM Press. ISBN 9781450360142. doi: 10.1145/3269206.3269269. URL `http://dl.acm.org/citation.cfm?doid=3269206.3269269`.

[40] F. Chollet. Keras. https://github.com/fchollet/keras, 2015.

[41] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *EMNLP*, pages 670–680, may 2017. ISBN 978-1-109-24088-7. doi: 10.1.1.156.2685. URL `http://arxiv.org/abs/1705.02364`.

[42] S. Dasgupta and V. Ng. Mine the easy, classify the hard: a semi-supervised approach to automatic sentiment classification. In *ACL '09 Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 701–709. Association for Computational Linguistics, aug 2009. ISBN 978-1-932432-46-6. URL `http://dl.acm.org/citation.cfm?id=1690219.1690244`.

[43] J. (David) Li. A two-step rejection procedure for testing multiple hypotheses. *Journal of Statistical Planning and Inference*, 138(6):1521–1527, jul 2008. doi: 10.1016/J.JSPI.2007.04.032. URL `https://www.sciencedirect.com/science/article/abs/pii/S0378375807002960`.

[44] A. Davies and Z. Ghahramani. Language-independent Bayesian sentiment mining of Twitter. In *5th SNA-KDD Workshop 11 (SNA-KDD 11)*, 2011. URL `http://users.cis.fiu.edu/{~}lzhen001/activities/KDD2011Program/workshops/SNAKDD2011/SNAKDD2011-Proceedings.pdf{#}page=99`.

[45] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016. URL `https://papers.nips.cc/paper/6081-convolutional-neural-networks-on-graphs-with-fast-localized-spectral-filtering.pdf`.

[46] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai. Deep Direct Reinforcement Learning for Financial Signal Representation and Trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):653–664, mar 2017. ISSN 2162-237X. doi: 10.1109/TNNLS.2016.2522401. URL `http://ieeexplore.ieee.org/document/7407387/`.

[47] S. Devlin and D. Kudenko. Theoretical Considerations of Potential-Based Reward Shaping for Multi-Agent Systems. In *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 225–232, 2011. URL `www.ifaamas.org`.

[48] S. Devlin, L. Yliniemi, D. Kudenko, and K. Tumer. Potential-Based Difference Rewards for Multiagent Reinforcement Learning. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems (AAMAS)*, pages 165–172. International Foundation for Autonomous Agents and Multiagent Systems, 2014. ISBN 9781450327381. URL `https://dl.acm.org/citation.cfm?id=2615731.2615761`.

[49] X. Ding, Y. Zhang, T. Liu, and J. Duan. Deep learning for event-driven stock prediction. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (ICJAI 15)*, pages 2327–2333, 2015. URL `https://www.ijcai.org/Proceedings/15/Papers/329.pdf`.

[50] B. Do and R. Faff. Does Simple Pairs Trading Still Work? *Financial Analysts Journal*, 66(4):83–95, jan 2010. URL `http://www.jstor.org/stable/info/25741293`.

[51] B. Do and R. Faff. Are pairs trading profits robust to trading costs. *Journal of Financial Research*, 35(2):261–287, jun 2012. doi: 10.1111/j.1475-6803.2012.01317.x. URL `http://doi.wiley.com/10.1111/j.1475-6803.2012.01317.x`.

[52] C. Dunis, P. W. M. Middleton, K. Theofilatos, and A. Karathanasopoulos. *Artificial intelligence in financial markets : cutting edge applications for risk management, portfolio optimization and economics*. Palgrave Macmillan, 2016. ISBN 1137488808.

[53] F. Durante, E. Foscolo, R. Pappadà, and H. Wang. A portfolio diversification strategy via tail dependence measures. In *Soft Methods for Data Science*, chapter 63, pages 511–518. Springer-Verlag, 2015. ISBN 978-88-8303-720-7. URL `https://www.springer.com/gp/book/9783319429717`.

[54] A. Eckner. A Framework for the Analysis of Unevenly Spaced Time Series Data. Technical report, 2014. URL `www.nyxdata.com/Data-Products/Historical-Data`.

[55] J. E. Engelberg, A. V. Reed, and M. C. Ringgenberg. Short-Selling Risk. *The Journal of Finance*, 73(2):755–786, apr 2018. doi: 10.1111/jofi.12601. URL `http://doi.wiley.com/10.1111/jofi.12601`.

[56] R. F. Engle. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50(4):987, jul 1982. doi: 10.2307/1912773. URL `https://www.jstor.org/stable/1912773?origin=crossref`.

[57] R. F. Engle and C. W. J. Granger. Co-Integration and Error Correction: Representation, Estimation, and Testing. *Econometrica*, 55(2):251–276, mar 1987. ISSN 00129682. doi: 10.2307/1913236. URL `https://www.jstor.org/stable/1913236?origin=crossref`.

[58] Eugene F. Fama and Kenneth R. French. The Cross-Section of Expected Stock Returns. *The Journal of Finance*, 47(2):427–465, jun 1992. doi: 10.1111/j.1540-6261.1992.tb04398.x. URL `http://doi.wiley.com/10.1111/j.1540-6261.1992.tb04398.x`.

[59] S. Fallahpour, H. Hakimian, K. Taheri, and E. Ramezanifar. Pairs trading strategy optimization using the reinforcement learning method: a cointegration approach. *Soft Computing*, 20(12):5051–5066, dec 2016. ISSN 1432-7643. doi: 10.1007/s00500-016-2298-4. URL `http://link.springer.com/10.1007/s00500-016-2298-4`.

[60] E. F. Fama. Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, 25(2):383–417, 1970. URL `http://onlinelibrary.wiley.com/doi/10.1111/j.1540-6261.1991.tb04636.x/full`.

[61] E. F. Fama. Efficient Capital Markets: II. *The Journal of Finance*, 46 (5):1575–1617, 1991. URL `http://onlinelibrary.wiley.com/doi/10.1111/j.1540-6261.1991.tb04636.x/full`.

[62] E. F. Fama and K. R. French. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1):3–56, feb 1993. ISSN 0304405X. doi: 10.1016/0304-405X(93)90023-5. URL `http://www.sciencedirect.com/science/article/pii/0304405X93900235`.

[63] E. F. Fama and K. R. French. Multifactor Explanations of Asset Pricing Anomalies. *The Journal of Finance*, 51(1):55–84, mar 1996. ISSN 00221082. doi: 10.1111/j.1540-6261.1996.tb05202.x. URL `http://doi.wiley.com/10.1111/j.1540-6261.1996.tb05202.x`.

[64] E. F. Fama and K. R. French. Luck versus Skill in the Cross-Section of Mutual Fund Returns. *The Journal of Finance*, 65(5):1915–1947, oct 2010. ISSN 00221082. doi: 10.1111/j.1540-6261.2010.01598.x. URL `http://doi.wiley.com/10.1111/j.1540-6261.2010.01598.x`.

[65] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, and T.-S. Chua. Temporal Relational Ranking for Stock Prediction. *ACM Transactions on Information Systems (TOIS)*, 37(2):1–30, mar 2019. doi: 10.1145/3309547. URL `http://dl.acm.org/citation.cfm?doid=3306215.3309547`.

[66] P. Fiedor. Networks in financial markets based on the mutual information rate. *Physical Review E*, 89(5), may 2014. doi: 10.1103/PhysRevE.89.052801. URL `https://link.aps.org/doi/10.1103/PhysRevE.89.052801`.

[67] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual Multi-Agent Policy Gradients. *Thirty-Second AAAI Conference on Artificial Intelligence*, apr 2018. URL `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewPaper/17193`.

[68] M. Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32(200):675–701, dec 1937. doi: 10.1080/01621459.1937.10503522. URL `http://www.tandfonline.com/doi/abs/10.1080/01621459.1937.10503522`.

[69] A. Galenko, E. Popova, and I. Popova. Trading in the Presence of Cointegration. *The Journal of Alternative Investments*, 15(1):85–97, jun 2012. doi: 10.3905/jai.2012.15.1.085. URL `http://jai.pm-research.com/lookup/doi/10.3905/jai.2012.15.1.085`.

[70] M. B. Garman and M. J. Klass. On the Estimation of Security Price Volatilities from Historical Data. *The Journal of Business*, 53:67–78, 1980. doi: 10.2307/2352358. URL `https://www.jstor.org/stable/2352358`.

[71] E. Gatev, W. N. Goetzmann, and K. G. Rouwenhorst. Pairs Trading: Performance of a Relative-Value Arbitrage Rule. *Review of Financial Studies*, 19(3):797–827, feb 2006. ISSN 0893-9454. doi: 10.1093/rfs/hhj020. URL `http://rfs.oxfordjournals.org/content/19/3/797.short`.

[72] E. A. Gerlein, M. McGinnity, A. Belatreche, and S. Coleman. Evaluating machine learning classification for financial trading: An empirical approach. *Expert Systems with Applications*, 54:193–207, jul 2016. ISSN 0957-4174. doi: 10.1016/J.ESWA.2016.01.018. URL `https://www.sciencedirect.com/science/article/pii/S0957417416000282`.

[73] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10):2451–2471, oct 2000. doi: 10.1162/089976600300015015. URL `http://www.mitpressjournals.org/doi/10.1162/089976600300015015`.

[74] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining Explanations: An Overview of Interpretability of Machine Learning. *Arxiv*, may 2018. URL `http://arxiv.org/abs/1806.00069`.

[75] X. Glorot, A. Bordes, and Y. Bengio. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 513–520, New York, NY, USA, jun 2011. ACM. ISBN 978-1-4503-0619-5. URL `http://machinelearning.wustl.edu/mlpapers/paper{_}files/ICML2011Glorot{_}342.pdf`.

[76] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. Technical report, Stanford University (CS224N Project Report), 2009. URL `http://cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf`.

[77] M. Gorenc Novak and D. Velušček. Prediction of stock price movement based on daily high prices. *Quantitative Finance*, 16(5):793–826, may 2016. ISSN 1469-7688. doi: 10.1080/14697688.2015.1070960. URL `http://www.tandfonline.com/doi/full/10.1080/14697688.2015.1070960`.

[78] C. Granger and P. Newbold. Spurious regressions in econometrics. *Journal of Econometrics*, 2(2):111–120, jul 1974. ISSN 0304-4076. doi: 10.1016/0304-4076(74)90034-7. URL `https://www.sciencedirect.com/science/article/pii/0304407674900347`.

[79] C. W. J. Granger. Time series analysis, cointegration, and applications. *American Economic Review*, 94(3):421–425, 2004.

[80] J. K. Gupta, M. Egorov, and M. Kochenderfer. Cooperative Multi-Agent Control Using Deep Reinforcement Learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 66–83, 2017. URL `http://ala2017.it.nuigalway.ie/papers/ALA2017{_}Gupta.pdf`.

[81] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017. URL `https://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.pdf`.

[82] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30 (2):129–150, 2011. URL `https://www.sciencedirect.com/science/article/pii/S1063520310000552`.

[83] S. Hansell. Inside Morgan Stanley's Black Box. *Institutional Investor*, 1989.

[84] P. R. Hansen and A. Lunde. A forecast comparison of volatility models: does anything beat a GARCH(1,1)? *Journal of Applied Econometrics*, 20(7):873–889, dec 2005. doi: 10.1002/jae.800. URL `http://doi.wiley.com/10.1002/jae.800`.

[85] M. Harris and A. Raviv. Differences of Opinion Make a Horse Race. *Review of Financial Studies*, 6(3):473–506, jul 1993. ISSN 0893-9454. doi: 10.1093/rfs/5.3.473. URL http://rfs.oxfordjournals.org/content/6/3/473.abstract.

[86] A. Hassan and D. Radev. Identifying text polarity using random walks. In *Proceeding ACL '10 Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 395–403. Association for Computational Linguistics, jul 2010. URL http://dl.acm.org/citation.cfm?id=1858681.1858722.

[87] V. Hatzivassiloglou and K. R. McKeown. Predicting the Semantic Orientation of Adjectives. In *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*, EACL '97, pages 174–181, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics. doi: 10.3115/979617.979640. URL http://dx.doi.org/10.3115/979617.979640.

[88] M. J. Hausknecht. *Cooperation and communication in multiagent deep reinforcement learning.* PhD thesis, University of Texas at Austin, 2016. URL https://repositories.lib.utexas.edu/handle/2152/45681.

[89] B. M. Henrique, V. A. Sobreiro, and H. Kimura. Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, 124:226–251, jun 2019. ISSN 0957-4174. doi: 10.1016/J.ESWA.2019.01.012. URL https://www.sciencedirect.com/science/article/pii/S095741741930017X.

[90] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735.

[91] H. Hong and J. C. Stein. A Unified Theory of Underreaction, Momentum Trading, and Overreaction in Asset Markets. *The Journal of Finance*, 54(6):2143–2184, dec 1999. ISSN 0022-1082. doi: 10.1111/0022-1082.00184. URL http://doi.wiley.com/10.1111/0022-1082.00184.

[92] J. Howard and S. Ruder. Universal Language Model Fine-tuning for Text Classification. In *ACL*, pages 328–339, jan 2018. URL http://arxiv.org/abs/1801.06146.

[93] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*, page 168, New York, New York, USA, aug 2004. ACM Press. ISBN 1581138889. doi: 10.1145/1014052.1014073. URL http://dl.acm.org/citation.cfm?id=1014052.1014073.

[94] W.-Q. Huang, X.-T. Zhuang, S. Yao, and S. Uryasev. A financial network perspective of financial institutions' systemic risk contributions. *Physica A: Statistical Mechanics and its Applications*, 456:183–196, aug 2016. doi: 10.1016/J.PHYSA.2016.03.034. URL `https://www.sciencedirect.com/science/article/pii/S0378437116300322`.

[95] N. Huck and K. Afawubo. Pairs trading and selection methods: is cointegration superior? *Applied Economics*, 47(6):599–613, feb 2015. doi: 10.1080/00036846.2014.975417. URL `http://www.tandfonline.com/doi/abs/10.1080/00036846.2014.975417`.

[96] B. Hurst, Y. H. Ooi, and L. H. Pedersen. A Century of Evidence on Trend-Following Investing. *The Journal of Portfolio Management*, 44(1): 15–29, oct 2017. ISSN 0095-4918. doi: 10.3905/jpm.2017.44.1.015. URL `http://jpm.iijournals.com/lookup/doi/10.3905/jpm.2017.44.1.015`.

[97] C. Hutto and E. Gilbert. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. In *Procceddings of 8th International AAAI Conference on Weblogs and Social Media*, pages 216–225, 2014. URL `http://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8109`.

[98] B. I. Jacobs and K. N. Levy. *Market neutral strategies.* John Wiley & Sons, 2005. ISBN 9780471268680. URL `https://www.wiley.com/en-us/Market+Neutral+Strategies-p-9780471268680`.

[99] H. Jacobs and M. Weber. On the determinants of pairs trading profitability. *Journal of Financial Markets*, 23:75–97, mar 2015. doi: 10.1016/J.FINMAR.2014.12.001. URL `https://www.sciencedirect.com/science/article/abs/pii/S1386418114000809`.

[100] Z. Jiang, D. Xu, and J. Liang. A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem. *Arxiv*, jun 2017. URL `http://arxiv.org/abs/1706.10059`.

[101] H. Joe. *Multivariate models and dependence concepts.* Chapman & Hall, 1997. ISBN 9780412073311. URL `https://www.crcpress.com/Multivariate-Models-and-Multivariate-Dependence-Concepts/Joe/p/book/9780412073311`.

[102] R. Johnson and T. Zhang. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. In *NAACL*, pages 103–112, 2015. URL `http://www.anthology.aclweb.org/N/N15/N15-1011.pdf`.

[103] P. Jorion. *Value at risk : the new benchmark for managing financial risk (3rd. edition).* McGraw-Hill, 2006. ISBN 9780071464956.

[104] H. Kanayama and T. Nasukawa. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *EMNLP '06 Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 355–363. Association for Computational Linguistics, jul 2006. ISBN 1-932432-73-6. URL `http://dl.acm.org/citation.cfm?id=1610075.1610125`.

[105] D. P. Kingma and J. Lei Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. URL `https://arxiv.org/pdf/1412.6980.pdf`.

[106] T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017. URL `https://www.semanticscholar.org/paper/Semi-Supervised-Classification-with-Graph-Networks-Kipf-Welling/1762baa638866a13dcc6d146fd5a49b36cbd9c30`.

[107] S. Kogan, D. Levin, B. R. Routledge, J. S. Sagi, and N. A. Smith. Predicting Risk from Financial Reports with Regression. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 272–280, 2009. URL `http://www.aclweb.org/anthology/N09-1031`.

[108] C. Krauss. Statistical arbitrage pairs trading strategies: Review and outlook. *FAU Discussion Papers in Economics, Friedrich-Alexander-Universität Erlangen-Nürnberg*, 2015. URL `https://www.econstor.eu/handle/10419/116783`.

[109] S. Lai, L. Xu, K. Liu, J. Z. AAAI, and U. 2015. Recurrent Convolutional Neural Networks for Text Classification. In *AAAI*, pages 2267–2273, 2015. URL `http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/download/9745/9552`.

[110] G. J. Laurent, L. Matignon, and N. L. Fort-Piat. The World of Independent Learners is Not Markovian. *Int. J. Know.-Based Intell. Eng. Syst.*, 15(1):55–64, jan 2011. ISSN 1327-2314. URL `http://dl.acm.org/citation.cfm?id=1971886.1971887`.

[111] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. *The Journal of Machine Learning Research*, 5:361–397, dec 2004. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1005332.1005345`.

[112] P. Li, W. Li, Z. He, X. Wang, Y. Cao, J. Zhou, and W. Xu. Dataset and Neural Recurrent Sequence Labeling Model for Open-Domain Factoid Question Answering. *ArXiv*, 2016. URL `https://arxiv.org/pdf/1607.06275.pdf`.

[113] Z. Liang, K. Jiang, H. Chen, J. Zhu, and Y. Li. Deep Reinforcement Learning in Portfolio Management. *Arxiv*, aug 2018. URL `http://arxiv.org/abs/1808.09940`.

[114] R. Q. Liew and Y. Wu. Pairs trading: A copula approach. *Journal of Derivatives & Hedge Funds*, 19(1):12–30, feb 2013. ISSN 1753-965X. doi: 10.1057/jdhf.2013.1. URL `http://link.springer.com/10.1057/jdhf.2013.1`.

[115] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *ArXiv*, sep 2015. URL `http://arxiv.org/abs/1509.02971`.

[116] Z. Lin, M. Feng, C. Nogueira, D. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A Structured Self-Attentive Sentence Embedding. In *ICLR*, 2017. URL `https://arxiv.org/pdf/1703.03130.pdf`.

[117] Z. C. Lipton, D. Kale, and R. Wetzel. Directly Modeling Missing Data in Sequences with RNNs: Improved Classification of Clinical Time Series. In *Proceedings of the 1st Machine Learning for Healthcare Conferenc*, pages 253–270, dec 2016. URL `http://proceedings.mlr.press/v56/Lipton16.html`.

[118] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel. Learning to Diagnose with LSTM Recurrent Neural Networks. In *ICLR*, nov 2016. URL `http://arxiv.org/abs/1511.03677`.

[119] B. Liu. *Sentiment analysis and opinion mining.* 2012. ISBN 9781608458844. URL `http://www.morganclaypool.com/doi/abs/10.2200/S00416ED1V01Y201204HLT016`.

[120] Y. Liu, C. Sun, L. Lin, and X. Wang. Learning Natural Language Inference using Bidirectional LSTM model and Inner-Attention. *ArXiv*, 2016. URL `https://arxiv.org/pdf/1605.09090.pdf`.

[121] T. Loughran and B. Mcdonald. When is a Liability not a Liability? Textual Analysis , Dictionaries , and 10-Ks. *The Journal of Finance*, 66(1):35–65, 2011. URL `http://bit.ly/15GhT7K`.

[122] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013. URL `https://ai.stanford.edu/{~}amaas/papers/relu{_}hybrid{_}icml2013{_}final.pdf`.

[123] J. G. MacKinnon. Critical values for cointegration tests. Technical report, Queen's Economics Department Working Paper, 2010. URL `https://ideas.repec.org/p/qed/wpaper/1227.html`.

[124] R. Mantegna. Hierarchical structure in financial markets. *The European Physical Journal B*, 11(1):193–197, sep 1999. doi: 10.1007/s100510050929. URL `http://link.springer.com/10.1007/s100510050929`.

[125] G. Marcus. Deep Learning: A Critical Appraisal. *Arxiv*, jan 2018. URL `http://arxiv.org/abs/1801.00631`.

[126] Mart\'\in~Abadi, Ashish~Agarwal, Paul~Barham, Eugene~Brevdo, Zhifeng~Chen, Craig~Citro, Greg~S.~Corrado, Andy~Davis, Jeffrey~Dean, Matthieu~Devin, Sanjay~Ghemawat, Ian~Goodfellow, Andrew~Harp, Geoffrey~Irving, Michael~Isard, Y. Jia, Rafal~Jozefowicz, Lukasz~Kaiser, Manjunath~Kudlur, Josh~Levenberg, Dandelion~Mané, Rajat~Monga, Sherry~Moore, Derek~Murray, Chris~Olah, Mike~Schuster, Jonathon~Shlens, Benoit~Steiner, Ilya~Sutskever, Kunal~Talwar, Paul~Tucker, Vincent~Vanhoucke, Vijay~Vasudevan, Fernanda~Viégas, Oriol~Vinyals, Pete~Warden, Martin~Wattenberg, Martin~Wicke, Yuan~Yu, and Xiaoqiang~Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL `https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf`.

[127] L. C. Martinez, D. N. da Hora, J. R. de M. Palotti, W. Meira, and G. L. Pappa. From an artificial neural network to a stock market day-trading system: A case study on the BM&F BOVESPA. In *2009 International Joint Conference on Neural Networks*, pages 2006–2013. IEEE, jun 2009. ISBN 978-1-4244-3548-7. doi: 10.1109/IJCNN.2009.5179050. URL `http://ieeexplore.ieee.org/document/5179050/`.

[128] J. V. Messias, M. Spaan, and P. U. Lima. Efficient Offline Communication Policies for Factored Multiagent POMDPs. *NIPS*, pages 1917–1925, 2011. URL `https://papers.nips.cc/paper/4385-efficient-offline-communication-policies-for-factored-multiagent-pomdps`.

[129] P. Milgrom and N. Stokey. Information, trade and common knowledge. *Journal of Economic Theory*, 1982. URL `http://www.sciencedirect.com/science/article/pii/0022053182900461`.

[130] G. A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, nov 1995. ISSN 00010782. doi: 10.1145/219717.219748. URL `http://dl.acm.org/citation.cfm?id=219717.219748`.

[131] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, feb 2015. doi: 10.1038/nature14236. URL `http://www.nature.com/articles/nature14236`.

[132] P. Molnár and P. Molnar. Properties of range-based volatility estimators. *International Review of Financial Analysis*, 23:20–29, jun 2012. ISSN 1057-5219. doi: 10.1016/J.IRFA.2011.06.012. URL `https://www.sciencedirect.com/science/article/pii/S1057521911000731`.

[133] J. Moody, L. Wu, Y. Liao, and M. Saffell. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, 17(5-6):441–470, 1998. URL `https://onlinelibrary-wiley-com.libproxy.york.ac.uk/doi/abs/10.1002/{%}28SICI{%}291099-131X{%}281998090{%}2917{%}3A5/6{%}3C441{%}3A{%}3AAID-FOR707{%}3E3.0.CO{%}3B2-{%}23`.

[134] NA. Consumer Confidence Survey – technical note. Technical report, The Conference Board, 2011. URL `https://www.conference-board.org/pdf{_}free/press/TechnicalPDF{_}4134{_}1298367128.pdf`.

[135] A. Y. Ng, M. I. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2002. URL `http://papers.nips.cc/paper/2092-on-spectral-clustering-analysis-and-an-algorithm.pdf`.

[136] T. H. Nguyen and K. Shirai. Topic Modeling based Sentiment Analysis on Social Media for Stock Market Prediction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1354–1364, 2015. URL `http://www.aclweb.org/anthology/P15-1131`.

[137] C. Nopp, T. Wien, and A. Hanbury. Detecting Risks in the Banking System by Sentiment Analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,*, pages 591–600, 2015. URL `http://www.emnlp2015.org/proceedings/EMNLP/pdf/EMNLP071.pdf`.

[138] C. E. Osgood. The nature and measurement of meaning. *Psychological Bulletin*, 49(3):197–237, 1952. doi: http://dx.doi.org/10.1037/h0055737.

[139] B. Pang and L. Lee. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135, jan 2008. ISSN 1554-0669. doi: 10.1561/1500000011. URL `http://dl.acm.org/citation.cfm?id=1454711.1454712http://www.nowpublishers.com/article/Details/INR-011`.

[140] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02*, volume 10, pages 79–86, Morristown, NJ, USA, jul 2002. Association for Computational Linguistics. doi: 10.3115/1118693.1118704. URL `http://dl.acm.org/citation.cfm?id=1118693.1118704`.

[141] M. Parkinson. The Extreme Value Method for Estimating the Variance of the Rate of Return. *The Journal of Business*, 53:61–65, 1980. doi: 10.2307/2352357. URL `https://www.jstor.org/stable/2352357`.

[142] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[143] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global Vectors for Word Representation. In *EMNLP*, pages 1532–1543, 2014. URL `https://nlp.stanford.edu/pubs/glove.pdf`.

[144] B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: Online Learning of Social Representations. *ArXiv*, mar 2014. doi: 10.1145/2623330.2623732. URL `http://arxiv.org/abs/1403.6652http://dx.doi.org/10.1145/2623330.2623732`.

[145] L. d. S. Pinheiro and M. Dras. Stock Market Prediction with Deep Learning: A Character-based Neural Language Model for Event-based Trading. In *Proceedings of the Australasian Language Technology Association Workshop 2017*, pages 6–15, 2017. URL `https://aclanthology.coli.uni-saarland.de/papers/U17-1001/u17-1001`.

[146] N. Ponomareva and M. Thelwall. Semi-supervised vs. Cross-domain Graphs for Sentiment Analysis. In G. Angelova, K. Bontcheva, and R. Mitkov, editors, *Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP'13)*, pages 571–578, Hissar, Bulgaria, 2013. URL `http://www.aclweb.org/anthology/R13-1075`.

[147] S. Proper and K. Tumer. Modeling difference rewards for multiagent learning. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1397–1398. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[148] H. Puspaningrum, Y.-X. Lin, and C. M. Gulati. Finding the Optimal Pre-Set Boundaries for Pairs Trading Strategy Based on Cointegration Technique. *Journal of Statistical Theory and Practice*, 4(3):391–419, sep 2010. doi: 10.1080/15598608.2010.10411994. URL `http://www.tandfonline.com/doi/abs/10.1080/15598608.2010.10411994`.

[149] D. V. Pynadath and M. Tambe. The Communicative Multiagent Team Decision Problem: Analyzing Teamwork Theories and Models. *Journal of Artificial Intelligence Research*, 16:389–423, jun 2002. doi: 10.1613/jair.1024. URL `https://jair.org/index.php/jair/article/view/10304`.

[150] H. Rad, R. K. Y. Low, and R. Faff. The profitability of pairs trading strategies: distance, cointegration and copula methods. *Quantitative Finance*, 16(10):1541–1558, oct 2016. ISSN 1469-7688. doi: 10.1080/14697688.2016.1164337. URL `https://www.tandfonline.com/doi/full/10.1080/14697688.2016.1164337`.

[151] D. Rao and D. Ravichandran. Semi-supervised polarity lexicon induction. In *EACL '09 Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 675–682. Association for Computational Linguistics, mar 2009. URL `http://dl.acm.org/citation.cfm?id=1609067.1609142`.

[152] N. Rekabsaz, M. Lupu, A. Baklanov, A. Hanbury, A. Ur, L. Anderson, and T. Wien. Volatility Prediction using Financial Disclosures Sentiments with Word Embedding-based IR Models. In *55th Annual Meeting of the Association for Computational Linguistics*, pages 1712–1721, 2017. doi: 10.18653/v1/P17-1157. URL `https://doi.org/10.18653/v1/P17-1157`.

[153] M. Research. Morningstar Global Equity Classification Structure. *Morningstar Research*, page 46, 2010. URL `http://gladmainnew.morningstar.com/clientcomm/MorningstarGlobalEquityClassificationStructureJan2010.pdf`.

[154] I. Rodriguez-Fdez, A. Canosa, M. Mucientes, and A. Bugarin. STAC: A web platform for the comparison of algorithms using statistical tests. In *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, aug 2015. ISBN 978-1-4673-7428-6. doi: 10.1109/FUZZ-IEEE.2015.7337889. URL `http://ieeexplore.ieee.org/document/7337889/`.

[155] D. L. T. Rohde. TGrep2 User Manual. 2001. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.7702`.

[156] S. Ruder. An Overview of Multi-Task Learning in Deep Neural Networks. *ArXiv*, jun 2017. URL `http://arxiv.org/abs/1706.05098`.

[157] W. Samek, T. Wiegand, and K.-R. Müller. Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models. *Arxiv*, aug 2017. URL `http://arxiv.org/abs/1708.08296`.

[158] A. M. Santomero and D. F. Babbel. Financial Risk Management by Insurers: An Analysis of the Process. *The Journal of Risk and Insurance*, 64(2): 231, jun 1997. doi: 10.2307/253730. URL `https://www.jstor.org/stable/253730?origin=crossref`.

[159] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, jan 2009. ISSN 1045-9227. doi: 10.1109/TNN.2008.2005605. URL `http://ieeexplore.ieee.org/document/4700287/`.

[160] R. P. Schumaker and H. Chen. Textual Analysis of Stock Market Prediction Using Breaking Financial News: The AZFin Text System. *ACM Trans. Inf. Syst.*, 27(2):12:1—-12:19, mar 2009. ISSN 1046-8188. doi: 10.1145/1462198.1462204. URL `http://doi.acm.org/10.1145/1462198.1462204`.

[161] M. Schuster and K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. ISSN 1053587X. doi: 10.1109/78.650093. URL `http://ieeexplore.ieee.org/document/650093/`.

[162] D. W. Scott. *Multivariate Density Estimation : Theory, Practice, and Visualization*. Wiley, 2nd editio edition, 2015. ISBN 9780471697558. URL `https://onlinelibrary.wiley.com/doi/book/10.1002/9780470316849`.

[163] W. F. Sharpe. The Sharpe Ratio. *The Journal of Portfolio Management*, 21(1):49–58, oct 1994. doi: 10.3905/jpm.1994.409501. URL `http://jpm.pm-research.com/lookup/doi/10.3905/jpm.1994.409501`.

[164] D. J. Sheskin. *Handbook of parametric and nonparametric statistical procedures*. Chapman and Hall/CRC, 2003. URL `https://books.google.co.uk/books/about/Handbook{_}of{_}Parametric{_}and{_}Nonparametric.html?id=YDd2cgAACAAJ{&}source=kp{_}book{_}description{&}redir{_}esc=y`.

[165] R. J. Shiller. Do Stock Prices Move Too Much to be Justified by Subsequent Changes in Dividends? *The American Economic Review*, 71:421–436, 1981. doi: 10.2307/1802789. URL `https://www.jstor.org/stable/1802789`.

[166] R. J. Shiller. Measuring Bubble Expectations and Investor Confidence. *Journal of Psychology and Financial Markets*, 1(1):49–60, mar 2000. ISSN 1520-8834. doi: 10.1207/S15327760JPFM0101_05. URL `http://www.tandfonline.com/doi/abs/10.1207/S15327760JPFM0101{_}05`.

[167] F. N. Silva, C. H. Comin, T. K. D. Peron, F. A. Rodrigues, C. Ye, R. C. Wilson, E. Hancock, and L. d. F. Costa. Modular Dynamics of Financial Market Networks. *ArXiv*, jan 2015. URL `http://arxiv.org/abs/1501.05040`.

[168] D. Silver, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic Policy Gradient Algorithms. In *31st International Conference on Machine Learning*, pages 387–395, 2014. URL `http://proceedings.mlr.press/v32/silver14.pdf`.

[169] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *Nips*, pages 926–934, 2013. URL `www.socher.org.http://papers.nips.cc/paper/5028-reasoning-with-neural-tensor-networks-for-knowledge-base-completion`.

[170] T. O. Sprenger, P. G. Sandner, A. Tumasjan, and I. M. Welpe. News or Noise? Using Twitter to Identify and Understand Company-specific News Flow. *Journal of Business Finance & Accounting*, 41(7-8):791–830, sep 2014. ISSN 0306686X. doi: 10.1111/jbfa.12086. URL `http://doi.wiley.com/10.1111/jbfa.12086`.

[171] P. J. Stone, D. C. Dunphy, and M. S. Smith. The General Inquirer: A Computer Approach to Content Analysis. 1966.

[172] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel. Value-Decomposition Networks For Cooperative Multi-Agent Learning. *ArXiv*, jun 2017. URL `http://arxiv.org/abs/1706.05296`.

[173] R. S. Sutton and A. G. Barto. *Reinforcement learning : an introduction.* MIT Press; second edition edition (20 Nov. 2018), 2018. ISBN 9780262039246. URL `https://mitpress.mit.edu/books/reinforcement-learning-second-edition`.

[174] M. Taboada, C. Anthony, and K. Voll. Methods for Creating Semantic Orientation Databases. In *Proceeding of LREC-06, the 5th International Conference on Language Resources and Evaluation*, pages 427–432, 2006. URL `http://www.sfu.ca/{%}7B{~}{%}7Dmtaboada/docs/Taboada{%}7B{_}{%}7Det{%}7B{_}{%}7Dal{%}7B{_}{%}7DLREC{%}7B{_}{%}7D2006.pdfhttp://www.sfu.ca/{~}mtaboada/docs/Taboada{_}et{_}al{_}LREC{_}2006.pdf`.

[175] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-Based Methods for Sentiment Analysis. *Computational Linguistics*, 37(2):267–307, jun 2011. ISSN 0891-2017. doi: 10.1162/COLI_a_00049. URL `http://dl.acm.org/citation.cfm?id=2000518`.

[176] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[177] S. S. J. Taylor. *Asset price dynamics, volatility, and prediction.* Princeton University Press, 2007. ISBN 9780691134796. URL `https://press.princeton.edu/books/paperback/9780691134796/asset-price-dynamics-volatility-and-prediction`.

[178] P. C. Tetlock. Giving Content to Investor Sentiment: The Role of Media in the Stock Market. *The Journal of Finance*, 62(3):1139–1168, jun 2007. ISSN 0022-1082. doi: 10.1111/j.1540-6261.2007.01232.x. URL `http://doi.wiley.com/10.1111/j.1540-6261.2007.01232.x`.

[179] A. Timmermann. Excess Volatility and Predictability of Stock Prices in Autoregressive Dividend Models with Learning. *The Review of Economic Studies*, 63(4):523–557, oct 1996. doi: 10.2307/2297792. URL `https://academic.oup.com/restud/article-lookup/doi/10.2307/2297792`.

[180] A. Tourin and R. Yan. Dynamic pairs trading using the stochastic control approach. *Journal of Economic Dynamics and Control*, 37(10):

1972–1981, oct 2013. doi: 10.1016/J.JEDC.2013.05.010. URL `https://www.sciencedirect.com/science/article/pii/S0165188913001164`.

[181] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL '03*, volume 1, pages 173–180, Morristown, NJ, USA, may 2003. Association for Computational Linguistics. doi: 10.3115/1073445.1073478. URL `http://dl.acm.org/citation.cfm?id=1073445.1073478`.

[182] P. Treleaven, M. Galas, and V. Lalchand. Algorithmic trading review. *Communications of the ACM*, 56(11):76–85, nov 2013. doi: 10.1145/2500117. URL `http://dl.acm.org/citation.cfm?doid=2524713.2500117`.

[183] M. F. Tsai and C. J. Wang. Financial keyword expansion via continuous word vector representations. In *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1453–1458, Stroudsburg, PA, USA, 2014. Association for Computational Linguistics. ISBN 9781937284961. doi: 10.3115/v1/d14-1152. URL `http://aclweb.org/anthology/D14-1152`.

[184] C. K. Tse, J. Liu, and F. C. Lau. A network perspective of the stock market. *Journal of Empirical Finance*, 17(4):659–667, sep 2010. ISSN 0927-5398. doi: 10.1016/J.JEMPFIN.2010.04.008. URL `https://www.sciencedirect.com/science/article/pii/S0927539810000368`.

[185] P. D. Turney. Thumbs up or thumbs down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, pages 417–424, Stroudsburg, PA, USA, jul 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073153. URL `http://dx.doi.org/10.3115/1073083.1073153`.

[186] K. Tuyls and G. Weiss. Multiagent Learning: Basics, Challenges, and Prospects. *AI Magazine (AAAI)*, 33(3):41–52, sep 2012. doi: 10.1609/aimag.v33i3.2426. URL `https://aaai.org/ojs/index.php/aimagazine/article/view/2426`.

[187] G. E. Uhlenbeck and L. S. Ornstein. On the Theory of the Brownian Motion. *Physical Review*, 36(5):823–841, sep 1930. ISSN 0031-899X. doi: 10.1103/PhysRev.36.823. URL `https://link.aps.org/doi/10.1103/PhysRev.36.823`.

[188] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. *NIPS*, pages 5998–6008, jun 2017. URL `http://arxiv.org/abs/1706.03762`.

[189] D. Vayanos and P. Woolley. An Institutional Theory of Momentum and Reversal. *Review of Financial Studies*, 26(5):1087–1145, may 2013. doi: 10.1093/rfs/hht014. URL `https://academic.oup.com/rfs/article-lookup/doi/10.1093/rfs/hht014`.

[190] P. Veličkovi, G. Cucurull, A. Casanova, A. Romero, P. Li, and Y. Bengio. Graph Attention Networks. In *ICLR*, pages 1–12, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

[191] G. Vidyamurthy. *Pairs Trading: Quantitative Methods and Analysis.* John Wiley & Sons, 2004. ISBN 0471684570. URL `http://books.google.com/books?id=yK2mvtpcumcC{&}pgis=1http://books.google.com/books?id=yK2mvtpcumcC{%}7B{&}{%}7Dpgis=1`.

[192] U. von Luxburg. A tutorial on spectral clustering. *arXiv*, 17(4): 395–416, nov 2007. ISSN 0960-3174. doi: 10.1007/s11222-007-9033-z. URL `http://link.springer.com/10.1007/s11222-007-9033-zhttp://arxiv.org/abs/0711.0189v1`.

[193] H.-J. von Mettenheim and M. H. Breitner. Forecasting and Trading the High-Low Range of Stocks and ETFs with Neural Networks. In *International Conference on Engineering Applications of Neural Networks*, pages 423–432. Springer, Berlin, Heidelberg, 2012. doi: 10.1007/978-3-642-32909-8_43. URL `http://link.springer.com/10.1007/978-3-642-32909-8{_}43`.

[194] C.-J. Wang, M.-F. Tsai, T. Liu, and C.-T. Chang. Financial Sentiment Analysis for Risk Prediction. In *International Joint Conference on Natural Language Processing*, pages 802–808, 2013. URL `http://www.aclweb.org/anthology/I13-1097`.

[195] C. J. Watkins and P. Dayan. Technical Note: Q-Learning. *Machine Learning*, 8(3/4):279–292, 1992. doi: 10.1023/A:1022676722315. URL `http://link.springer.com/10.1023/A:1022676722315`.

[196] J. Weston, F. Ratle, H. Mobahi, and R. Collobert. *Deep Learning via Semi-supervised Embedding*, pages 639–655. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-35289-8. doi: 10.1007/978-3-642-35289-8_34. URL `https://doi.org/10.1007/978-3-642-35289-8{_}34`.

[197] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05*, pages 347–354, Morristown, NJ, USA, oct 2005. Association for Computational Linguistics. doi: 10.3115/1220575.1220619. URL `http://dl.acm.org/citation.cfm?id=1220575.1220619`.

[198] R. A. Wood, T. H. Mcinish, and J. K. Ord. An Investigation of Transactions Data for NYSE Stocks. *The Journal of Finance*, 40(3):723–739, jul 1985. doi: 10.1111/j.1540-6261.1985.tb04996.x. URL http://doi.wiley.com/10.1111/j.1540-6261.1985.tb04996.x.

[199] W. Xie, R. Q. Liew, Y. Wu, and X. Zou. Pairs Trading with Copulas. *The Journal of Trading*, 11(3):41–52, jun 2016. doi: 10.3905/jot.2016.11.3.041. URL http://jot.pm-research.com/lookup/doi/10.3905/jot.2016.11.3.041.

[200] F. Z. Xing, E. Cambria, and R. E. Welsch. Natural language based financial forecasting: a survey. *Artificial Intelligence Review*, 50(1):49–73, oct 2018. ISSN 0269-2821. doi: 10.1007/s10462-017-9588-9. URL http://link.springer.com/10.1007/s10462-017-9588-9.

[201] Z. Xiong, X.-Y. Liu, S. Zhong, H. Yang, and A. Walid. Practical Deep Reinforcement Learning Approach for Stock Trading. *NIPS Workshop on Challenges and Opportunities for AI in Financial Services*, nov 2018. URL http://arxiv.org/abs/1811.07522.

[202] X. Zhang, B. Podobnik, D. Y. Kenett, and H. Eugene Stanley. Systemic risk and causality dynamics of the world international shipping market. *Physica A: Statistical Mechanics and its Applications*, 415:43–53, dec 2014. doi: 10.1016/J.PHYSA.2014.07.068. URL https://www.sciencedirect.com/science/article/pii/S0378437114006542.

[203] Y. Zhang and Q. Yang. A Survey on Multi-Task Learning. *ArXiv*, jul 2017. URL http://arxiv.org/abs/1707.08114.

[204] Z. Zhao and D. P. Palomar. Mean-Reverting Portfolio With Budget Constraint. *IEEE Transactions on Signal Processing*, 66(9):2342–2357, may 2018. ISSN 1053-587X. doi: 10.1109/TSP.2018.2799193. URL https://ieeexplore.ieee.org/document/8272418/.

[205] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems (NIPS)*, pages 321–328, 2004. URL http://papers.nips.cc/paper/2506-learning-with-local-and-global-consistency.pdf.

[206] S. Zhou, Q. Chen, and X. Wang. Active deep learning method for semi-supervised sentiment classification. *Neurocomputing*, 120:536–546, nov 2013. ISSN 09252312. doi: 10.1016/j.neucom.2013.04.017. URL http://www.sciencedirect.com/science/article/pii/S0925231213004888.

[207] E. Zivot. Practical Issues in the Analysis of Univariate GARCH Models. In *Handbook of Financial Time Series*, pages 113–155. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. doi: 10.1007/978-3-540-71297-8_5. URL http://link.springer.com/10.1007/978-3-540-71297-8{_}5.