CHARLES UNIVERSITY
FACULTY OF SOCIAL SCIENCES
Institute of Economic Studies

Stock Trading Using a Deep Reinforcement Learning and Text Analysis

Master's thesis

Author: Bc. Dominik Benk
Study program: Economics and Finance
Supervisor: doc. PhDr. Jozef Baruník Ph.D.
Year of defense: 2022

Declaration of Authorship

The author hereby declares that he or she compiled this thesis independently, using only the listed resources and literature, and the thesis has not been used to obtain any other academic title.

The author grants to Charles University permission to reproduce and to distribute copies of this thesis in whole or in part and agrees with the thesis being used for study and scientific purposes.

Prague, July 3, 2022

Dominik Benk

Abstract

The thesis focuses on exploiting imperfections on the stock market by utilizing state-of-the-art learning methods and applying them to algorithmic trading. The automated decisions are expected to have the capability of outperforming professional traders by considering much more information, reacting almost instantly and being unaffected by emotions. As an alternative to traditional supervised learning, the proposed model of reinforcement learning employs a principle of trial-and-error, which is essential for learning behaviours of all organisms. In the context of stocks, this allows to consider the involved uncertainty and therefore more precisely estimate the long-run returns. To collect the most relevant information for each trading decision, additionally to technical indicators the models build on investor's opinion - financial sentiment. This is derived from two textual sources, news and social media, and the main goal is to compare their relative contribution to trading. Models are applied to 11 different stocks and later combined into portfolio for greater robustness of results. The textual analysis proves to be important for the learning process, especially in case of stocks with good media coverage. The Twitter is found to provide more valuable information compared to news, but their combination shows even higher predictive potential. Nevertheless, proposed models have difficulties to reliably outperform passive strategies or the market.

JEL Classification	C02, C45, C54, C58, C61, C63
Keywords	Reinforcement Learning, Machine Learning, Stock Trading, Sentiment Analysis
Title	Stock Trading Using a Deep Reinforcement Learning and Text Analysis

Abstrakt

Práce se zaměřuje na využití nedokonalostí akciového trhu pomocí nejmodernějších metod učení a jejich aplikaci na algoritmické obchodování. Očekává se, že automatizovaná rozhodnutí budou schopna překonat profesionální obchodníky tím, že zohlední mnohem více informací, budou reagovat téměř okamžitě a nebudou ovlivněna emocemi. Jako alternativa k tradičnímu učení s takzvaným učitelem, využívá navrhovaný model zpětnovazebního učení principu pokus-omyl, který je nezbytný pro učení chování všech organismů. V kontextu akcií to umožňuje zohlednit zahrnutou nejistotu, a tedy přesněji odhadnout dlouhodobé výnosy. Pro shromáždění co nejrelevantnějších informací k jednotlivým obchodním rozhodnutím, staví modely kromě technických ukazatelů také na názoru investorů - finančním sentimentu. Ten je získáván ze dvou textových zdrojů, zpráv a sociálních médií, a hlavním cílem je porovnat jejich relativní přínos pro obchodování. Modely jsou aplikovány na 11 různých akcií a později spojeny do portfolia pro větší robustnost výsledků. Textová analýza se potvrdila jako důležitá při procesu učení, zejména v případě akcií s dobrým mediálním pokrytím. Ukazuje se, že Twitter poskytuje cennější informace ve srovnání se zprávami, ale jejich kombinace vykazuje ještě vyšší predikční potenciál. Nicméně i tak, navržené modely mají potíže polehlivě překonat pasivní strategie či trh.

Klasifikace JEL	C02, C45, C54, C58, C61, C63
Klíčová slova	Zpětnovazební Učení, Strojové Učení, Obchodování Akcií, Analýza Sentimentu
Název práce	Obchodování Akcií Pomocí Hlubokého Zpětnovazebního Učení a Analýzy Textu

Acknowledgments

The author would like to express his gratitude to doc. PhDr. Jozef Baruník Ph.D. for his guidance and valuable feedback. Special thanks to my family and friends for their continuous support and motivation.

This thesis is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 870245.

Typeset in LaTeXusing the IES Thesis Template.

Bibliographic Record

Benk, Dominik: *Stock Trading Using a Deep Reinforcement Learning and Text Analysis*. Master's thesis. Charles University, Faculty of Social Sciences, Institute of Economic Studies, Prague. 2022, pages 99. Advisor: doc. PhDr. Jozef Baruník Ph.D.

Contents

List of Tables

List of Figures

Acronyms

ABM Agent-based Modelling

AdaGrad Adaptive Gradient

Adam Adaptive Movement Estimation

AMH Adaptive Market Hypothesis

AI Artificial Intelligence

ARIMA Auto-Regressive Integrated Moving Average

ATR Average True Range

BERT Bidirectional Encoder Representations from Transformer

CNN Convolutional Neural Networks

DCA Dollar-cost Averaging

DDPG Deep Deterministic Policy Gradient

DDQ Double Deep Q-Learning

DDQN Double Deep Q-Network

DJIA Dow Jones Industrial Average

DL Deep Learning

DQN Deep Q-Network

DRL Deep Reinforcement Learning

EMH Efficient Market Hypothesis

ELMo Embeddings from Language Models

EMA Exponential Moving Average

FC Fully Connected

FinBERT Financial BERT

GARCH Generalized Auto-Regressive Conditional Heteroskedasticity

GDPG Gated Deterministic Policy Gradient

GDQN Gated Deep Q-learning

GRU Gated Recurrent Unit

LR logistic regression

LSTM Long Short-Term Memory

MACD Moving Average Convergence/Divergence

MCS Monte Carlo Simulations

MDP Markov Decision Process

MDRNN Multimodal Deep Recurrent Neural Networks

ML Machine Learning

MLP Multilayer Perceptron

NLP Natural Language Processing

NN Neural Networks

NNAR Neural Network Autoregressive

NYSE New York Stock Exchange

PMMDR Parallel Multi-Module Deep Reinforcement learning

POMDP Partially Observable Markov Decision Process

RL Reinforcement Learning

RMSProp Root Mean Squared Propagation

RNN Recurrent Neural Networks

RSI Relative Strength Index

SGD Stochastic Gradient Descent

STM Support Tensor Machine

SVI Search Volume Index

SVM Support Vector Machine

TD Temporal-difference

TRMI Thompson Reuters MarketPsych Index

VADER Valence Aware Dictionary and sEntiment Reasoner

VAR Vector Autoregressive

VIX Volatility Index

Master's Thesis Proposal

Author	Bc. Dominik Benk
Supervisor	doc. PhDr. Jozef Baruník Ph.D.
Proposed topic	Stock Trading Using a Deep Reinforcement Learning and Text Analysis

Motivation Algorithmic trading has attracted a lot of attention in recent years and as it has been shown many times, it is a very challenging task. Defining a proper strategy first requires to build some expectations about the future prices. For this, there exists two main approaches, namely technical and fundamental analyses. While the first focuses on evolution of past stock prices, the latter considers financial information about stock's intrinsic value. Nevertheless, they both fail to co-exists with Efficient market hypothesis, a theory which implies that the stock price reflects all available information and should therefore react to any changes almost immediately, leaving no space to "beat the market".

In spite of this, there is an alternative view, an Adaptive market hypothesis (Lo, 2019), that also introduced the irrational principles of behavioural finance. These can explain market anomalies via psychological processes, based on which there is some room for exploitation of the market imperfections. This is one of the reasons why models can yield positive returns, even though the prices are usually expected to follow a random walk.

However, there is much more to consider when trading, as there is no "labelled" data available, that would provide us with an information about what should have been done in the particular situation. Increase in price does not necessarily imply that the model should learn, that it was a good call to purchase it prior to knowing what will happen, nor to short it if the final direction was opposite. This opens space for reinforcement learning, a paradigm that allows to learn optimal actions via trial and error based on proper reward function. This approach has shown a lot of a success and is considered a state-of-the-art in many academic work (see for example Wu et al., 2020; Ma et al., 2021).

Additional to this, I also believe that using features only based on price history is just not sufficient. To accurately model the market's behaviour, one would have to include all relevant variables in the world, but this is clearly impossible. To proxy this and capture at least the most relevant information, I will perform a text based analysis of News articles as well as opinions shared on social media such as Twitter or Reddit. The nature of this information is subject to further research, but I will attempt to extract financial sentiment along with some descriptive metrics.

Hypotheses

Hypothesis #1: Text analysis plays an important role in stock price prediction and will improve trading profitability.

Hypothesis #2: Text analysis based on social media provides more valuable information for trading than news articles.

Hypothesis #3: Proposed model outperforms minimum variance portfolio as well as S&P500 in terms of returns.

Methodology

Model The reinforcement learning will be done via Q-learning algorithm, which however requires a deep learning extension. Generally, Q-learning agent maximizes the reward by learning so called policy, a set of rules of how to decide in a particular situation. It estimates action value for each state-action combination by exploring as well as exploiting, however, as there would be infinitely many states on the market, we have to approximate it. This will be done with a deep neural network and the whole algorithm is then called Deep Q-Network. The deep neural network will likely to be defined with recurrent cells such as LSTM or GRU and will incorporate combination of features from stock prices as well as other variables from text analysis.

Sentiment analysis is a process of extracting the opinion from a text, which is usually distinguished as positive, negative or neutral. This polarity can provide a valuable information about further expectations and should therefore be considered in predictions. In case of financial news, I will extract the stock specific sentiment from the news articles with a use of FinBERT (Araci, 2019). In case of Social media, it will not be as straightforward, especially because of short nature of the published content and its dependency on other context. Therefore, I will also consider sentiment based on emoji's as well as descriptive metrics, such as counts of articles, tweets or posts as additional features.

Data

Stock data: Since I will build a portfolio of publicly traded stocks, history of all their prices should be available via Yahoo finance API for example.

News data: I will attempt to collect articles from various sources, however, if I face some difficulties, I will utilize dataset from Kaggle.

Social media data: This will be quite tricky, but some API's are available. In case of Twitter, one can get all the public tweets for a specific keyword via Twint for example. In case of Reddit, or more specifically r/WallStreetBets, it should be possible to utilize official API. Again, it is also possible to work with data collected by someone else, but it might be problematic to find all the required datasets with overlapping periods.

Hypothesis testing I will start with defining a portfolio of a given number of stocks. The first hypothesis will be tested by including sentiment/other textual features versus not including them at all. In the second one, I will compare portfolio values based on model with news features versus social media features. And lastly, I will attempt to show that the best of all proposed strategies outperforms the traditional minimum variance portfolio and S&P500. The evaluation will be done in terms of final portfolio's value, together with a Sharpe ratio, representing the risk adjusted return.

Expected Contribution In my thesis, I will attempt to build a model via reinforcement learning, that will be capable of deciding whether to buy, hold or sell each particular stock from a defined portfolio. I will run several simulations to demonstrate its performance and to compare variety of features from text analysis, that are expected to play an important role in stock price predictions. The main contribution to the current literature lies in utilizing state of the art algorithms and providing a direct comparison of social media and financial news, in terms of their relevance as a stock trading signals.

Also, since recently there has been a lot of controversy about the manipulation of stock prices on social media (e.g. GameStop), I would like to attempt to incorporate such information into an algorithm and if validated, the model could be used as a real-time trading strategy that is aware of such practices.

Outline

1. Introduction
2. Literature Review
3. Data
   (a) Stock market
   (b) News
   (c) Social media
4. Methodology
   (a) Deep neural networks
   (b) Reinforcement learning
   (c) Sentiment analysis
5. Results
   (a) Training
   (b) Evaluating
   (c) Predicting
6. Conclusion

Core bibliography

Zhang, J. L., Härdle, W. K., Chen, C. Y., & Bommes, E. (2016). Distillation of news flow into analysis of stock reactions. Journal of Business & Economic Statistics, 34(4), 547-563.

Xiong, Z., Liu, X. Y., Zhong, S., Yang, H., & Walid, A. (2018). Practical deep reinforcement learning approach for stock trading. arXiv preprint arXiv:1811.07522.

Lo, A. W. (2019). The adaptive markets hypothesis (pp. 176-221). Princeton University Press.

Vanstone, B. J., Gepp, A., & Harris, G. (2019). Do news and sentiment play a role in stock price prediction?. Applied Intelligence, 49(11), 3815-3820.

Araci, D. (2019). Finbert: Financial sentiment analysis with pre-trained language models. arXiv preprint arXiv:1908.10063.

Nan, A., Perumal, A., & Zaiane, O. R. (2020). Sentiment and knowledge based algorithmic trading with deep reinforcement learning. arXiv preprint arXiv:2001.09403.

Wu, X., Chen, H., Wang, J., Troiano, L., Loia, V., & Fujita, H. (2020). Adaptive stock trading strategies with deep reinforcement learning methods. Information Sciences, 538, 142-158.

Ma, C., Zhang, J., Liu, J., Ji, L., & Gao, F. (2021). A parallel multi-module deep reinforcement learning algorithm for stock trading. Neurocomputing, 449, 290-302.

Carta, S., Corriga, A., Ferreira, A., Podda, A. S., & Recupero, D. R. (2021). A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning. Applied Intelligence, 51(2), 889-905.

Chapter 1

Introduction

On the stock market, nothing is without risk, but learning from past events can help to mitigate the uncertainty. Even though this is a strong statement that may seem to violate the market efficiency, the idea of exploitable behavioural anomalies has been rising. Investors do not necessarily act rationally by applying various mental shortcuts, providing investment opportunities, especially during periods of sudden shifts. The challenging task is, however, to reliably recognize them. Depending on the type of information, technical and fundamental analyses provide two distinct views - it is either the price history or the intrinsic value that should determine the decision. Nevertheless, these do not necessarily need to be separated, which will also be the case for this thesis.

Trading decisions are often very complex to be executed properly. There can be a lot of misleading information and it is important to filter out the noise. This results in plenty of room for human error. Algorithmic trading has the capability to save time and resources by automated execution of decisions that is also extremely scalable. If trained properly, it can understand the current situation better than humans and, most importantly, decide in a split of a second. The potential to have a significant financial impact is the reason for stock trading and prediction, in general, being one of the most researched topics in finance.

Difficulties arise when defining the process of learning from past experiences. Supervised learning has been prevalent in this domain, but has shown significant limitations. In most cases, it assumes that the historical price should have been predicted in the given circumstances. This ignores all of the uncertainty involved and often leads to perfectly modelling the history but failing to predict the future. On the other hand, a natural learning process of hu-

mans has always been a trial-and-error and one of the approaches that employ this concept is Reinforcement Learning (RL). The idea is to build an agent that interacts with environment and efficiently finds an optimal behaviour that maximizes a cumulative reward while also considering the risk. It has shown great success in many areas, however, in finance it is still an open problem and evolving field of study.

Furthermore, since it is impossible to consider everything that could possibly have some effect on a stock's price, it is necessary to focus only on the important factors. The most notable and discussed in the literature is the investor's attitude towards a particular stock, i.e. financial sentiment. This is where Natural Language Processing (NLP) has a lot to offer to finance, especially thanks to recent development in word embeddings and transformer architectures. A fully automated procedure to extract valuable information from textual sources is capable of monitoring a general opinion over time. Any significant changes then may possibly be a good indicator for trading decisions, but this is up for the agent to learn from the experiments.

The objective of the thesis will be to exploit the market imperfections and utilize state-of-the-art methods of learning in a domain which has not been thoroughly examined yet. The main contribution will be the focus on sentiment analysis and comparison of news and social media in the context of their effect on the learning process of trading agents. By introducing a large set of models, it is then expected to provide as robust results as possible, considering the limited computational resources. The selected period is also particularly challenging and will be interesting to evaluate as it experienced two significant black swan events that shook the markets - the Coronavirus pandemic and the Russian invasion of Ukraine.

The thesis will be organized as follows: Chapter 2 reviews the most relevant literature for building a trading algorithm in the context of sentiment analysis, and Chapter 3 describes the stock, news and social media data sources. Chapter 4 provides the financial context and builds a foundation for all of the methods that will be utilized and evaluated in Chapter 5. Last but not least, the results will be discussed in Chapter 6 and summarized in Chapter 7.

Chapter 2

Literature Review

In this chapter, an overview of research related to stock market trading in the context of textual analysis will be provided. It is also important to note, that even though textual analysis is a topic of the main focus of this thesis, the literature not always considers it. This is because its benefits are still unclear and the objective of hypothesis testing. The purpose is to prepare building blocks to later combine them into the final algorithm.

Artificial Intelligence (AI) has found much use in fields of finance and the first attempts can be found already back in the 1980s (Braun & Chandler 1987). Two outlying approaches can be distinguished and will be discussed separately in the following sections.

2.1 Supervised Learning

An important assumption of supervised learning is to know the golden label, which the algorithm is supposed to learn from the given data. In our terms, it is usually the price we want to predict, the direction of its movement or a buy/sell decision.

In this context, Tetlock (2007) considered articles from Wall Street Journal. Results from Vector Autoregressive (VAR) experiments suggested that pessimistic environment is likely to be followed by a short-term decrease in market price, while in long run there is a return to its fundamental value. Also, measuring different magnitudes of pessimism, it was observed that their extreme values (both low and high) increased the volume of traded stocks.

Similarly, Zhang *et al.* (2016) utilized multiple textual data sources on basis of financial news, consisting of "professional platforms, blog fora and stock mes-

sage boards". Using 3 sentiment lexica authors extracted average proportions of negative or positive words for each day, to examine their predictive power in terms of volatility, market volume and stock returns on the following day. In the panel regression, the stock reaction indicators were dependent on their lags as well as sentiment proportions. Estimation was done by VAR and the authors also experimented with Monte Carlo Simulations (MCS). A significant effect was observed especially in the case of negative sentiment, as asymmetric behaviour was confirmed. Additional to this, the attention of retail investors was also considered as measured by Search Volume Index (SVI). High and low attention companies were distinguished, while only in the first case, sentiment had a notable effect on future dynamics of the indicators.

In more recent study, Vanstone *et al.* (2019) suggested to centre the attention on "rational arbitrageurs", and therefore utilize sentiment scores published by Bloomberg. These are published daily for each company and consist of 6 different sentiment indicators, namely: News publication, News positive sentiment, News negative sentiment, Twitter publication, Twitter positive sentiment and Twitter negative sentiment. Authors evaluated their relative importance via regression analysis and chose only News and Twitter publications, leaving the positive and negative counts untouched for modelling. Such a decision might raise some concerns but it was argued, that investors are more affected by the volume of the published news rather than their sentiment nature. To evaluate the potential effect of sentiment, the authors built two Neural Network Autoregressive (NNAR) models, where the first utilized only the historical prices, while the second also includes the sentiment variables. It was concluded, that in terms of accuracy, there is indeed a positive impact on stock price predictions.

2.2 Reinforcement Learning

Explicitly telling the algorithm what the correct label is might end up with misleading results. For example, Nan *et al.* (2020) stressed the unavailability of reliable labelled data to predict future dynamics of the market as the biggest flaw of supervised learning algorithms. This led the authors to the idea of building a trading strategy with the use of reinforcement learning. The environment was defined as Partially Observable Markov Decision Processs (POMDPs) and solved by Deep Q-Network (DQN) of various reward signals. Authors considered sentiments of news headlines from Reuters, filtered by their relevance to

particular stock with a use of knowledge graphs. The model was trained on Microsoft, Amazon and Tesla stocks separately, while allowing only for a single stock purchase/sale per day. The agent with sentiment indicators showed higher profits over the observed period as well as higher Sharpe ratios, showing that the proposed trading strategy better-balanced risk and returns.

Alternatively, Xiong *et al.* (2018) employed a different reinforcement algorithm of a Deep Deterministic Policy Gradient (DDPG), and found the models to outperform baselines of Dow Jones Industrial Average (DJIA) as well as min-variance portfolio, which was also observed in terms of Sharpe ratio.

Wu *et al.* (2020) trained a Gated Recurrent Unit (GRU) for a feature extraction from the historical prices and technical indicators, in order to approximate the market environment. Policy for the trading strategy was taught via Gated Deep Q-learning (GDQN) and Gated Deterministic Policy Gradient (GDPG) and involved only a single stock. The authors also presented a unique volatility-adjusted reward function to ensure stability even during periods of high uncertainty. The proposed model was found superior to Turtle and Direct Reinforcement Learning trading strategies, regardless of the nature of the environment. Nevertheless, the authors also stressed the importance of portfolio management, which was omitted in their study.

Ma *et al.* (2021) also claimed that long-term trends should be considered and proposed a Parallel Multi-Module Deep Reinforcement learning (PMMDR). The first module estimated the current state of the stock market based on its price history, technical indicators (e.g. moving averages, relative strength, on balance volume) and a set of fundamental indicators (e.g. profits, market capitalization, turnover, operating revenues and costs, price to book ratio). The purpose of the second module was solely to identify historical market trends given the history of last 30 trading days. These were implemented by Fully Connected (FC) and Long Short-Term Memory (LSTM) layers respectively. The authors then built a model for each of 8 randomly selected stocks traded on the Chinese stock market. In terms of return and Sharpe ratios, models outperformed strategies such as Buy and Hold, Double Exponential Moving Average, Fuzzy deep Deep Reinforcement Learning (DRL), RL with price trailing or Gated DQN. The possibility of using this approach for a portfolio management strategy is the author's subject of further research and has not yet been published.

Unsatisfying results from single supervised classifier were pointed out by Carta *et al.* (2021), who proposed a novel approach to feature extraction relying on Gramian Angular Field images and Convolutional Neural Networks (CNN).

The final trading signal was then based on an ensemble of multiple agents trained by Double Deep Q-Learning (DDQ). According to results from tests on S&P 500, J.P. Morgan and Microsoft stocks, the model outperformed techniques such as Support Tensor Machine (STM) or CNN Technical Analysis.

Another RL approach was described by Yang *et al.* (2018), who introduced so-called Multimodal Deep Recurrent Neural Networks (MDRNN). The architecture consisted of two modules, an RNN feature extraction based on historical prices on one side, and an influence model evaluating the importance of sentiment overtime on the other. These were then combined via multimodal learning and used to describe a current state. Examined sentiments were of two kinds, first which consisted of news articles and second synthesized sentiment that also combined them with news titles. In conclusion, the authors pointed out different nature of sentiment behaviour for individual companies, showing distinct levels of "information disclosure". The greatest increase in profits was observed in the sector of information technology (e.g. APPL, GOOGL), for which many articles were available as they are trendy and often discussed.

Xiao & Chen (2018) focused solely on Twitter sentiment of two automotive sector companies, Tesla and Ford. Two types of data were collected, firstly filtering tweets by stock tickers and secondly, using the top twenty keywords from Google Trends related to the companies. The sentiment itself was then computed via Stanford coreNLP. Compared to logistic regression (LR) and Support Vector Machine (SVM), RL provided a more profitable trading policy especially for Tesla, likely to be explained by the high degree of dependence on expectations.

2.3 News versus Social Media Sentiments

Literature that directly compares sentiments of news and social media is very limited. Work of Mao *et al.* (2011) studied a wide range of data sources including Twitter ("bullish" and "bearish" keyword sentiment, the volume of stock tickers and other financial terms), news headlines (e.g. Wall Street Journal, Bloomberg, Forbes or Reuters), Investors sentiment surveys and Google search queries. Correlation and Granger causality analyses were performed on all sentiments in the context of trading volumes, DJIA and Volatility Index (VIX). The relationship was further examined via Multiple Linear Regressions and last but not least, they were also tested in terms of their predictive power. All sentiment indicators showed a significant correlation with log returns andVIX.

Furthermore, both Twitter sentiments turned out to be statistically significant in terms of forecasting returns, while in the case of news sentiment the significance level was much lower. Even when considering other types of sentiments, Twitter's predictions outperformed all of them.

One of the few other examples of comparison of news and social media was also provided by Jiao *et al.* (2020), who evaluated their predictive power on stock volatility and turnover. They introduced so-called "echo chambers", in which social media is an echo of news, while the investors still consider it as new information. The authors focused on two different sets of variables, namely precomputed sentiments and the "buzz" indicator (i.e. the degree of coverage), which were collected from the database of Thompson Reuters MarketPsych Index (TRMI). The authors described simple stock-level panel regression, followed panel VAR to test Granger causality between the news and social media coverage and the financial indicators. According to findings, high news media coverage is expected to predict a decrease in stock volatility and turnover, while in the case of social media buzz the effect is the opposite. This was also confirmed at the market level, i.e. the aggregated social media buzz is expected to predict increased volatility in the whole market, while news predicted the opposite. News media was also found to Granger cause social media, and even though this confirmed the "echo chambers" phenomena, authors argued that there are many other factors that could play an important role in this behaviour and that such conclusion might not be correct.

None of the papers above in this section, however, did consider stock trading or reinforcement learning. To the best of the author's knowledge, no study has been directly comparing social and news media effects on performance in trading via reinforcement learning, and will therefore be one of the main contributions of this thesis.

Chapter 3

Data

In this chapter, several different datasets will be described, each having its special purpose in defining the current state of the environment. First, to perform the technical analysis, a dataset of historical prices has to be considered. Additionally, there will be also 2 sources of textual data, namely News and Twitter, which will be then later used to extract the information about market sentiment.

The selection process of stocks was based on market capitalization, in order to include the 10 largest companies in the US market. Such a decision was mainly driven by the idea, that a great amount of money is expected to attract more attention, which then transfers into higher popularity and therefore also better textual data availability to estimate the sentiments. Additionally, one of the so-called "meme" stocks was added - i.e. *GameStop*. It has drawn a lot of attention during the last year, because of its controversy over practices such as the short squeezes. There has been a significant growth in its online community that has cult-like characteristics and thanks to coordination, it might have been able to significantly influence the prices. The idea is not to answer legal issues, but to evaluate the proposed model also on social media biased data.

The final list of tickers is available in Table 3.1 with their respective capitalizations. Also notice, that the majority of stocks are part of the Technology sector, which should be taken into account when applying the proposed model to other out-of-sample tickers, as it could suffer in terms of generalization.

Table 3.1: Selected Stocks

Ticker	Name	Sector	Market cap ($)
AAPL	Apple Inc.	Electronic Technology	2.808T
MSFT	Microsoft Corp.	Technology Services	2.431T
GOOGL	Alphabet Inc.	Technology Services	1.888T
AMZN	Amazon.com, Inc.	Retail Trade	1.724T
TSLA	Tesla, Inc.	Consumer Durables	936B
FB	Meta Platforms, Inc.	Technology Services	928B
NVDA	NVIDIA Corp.	Electronic Technology	695B
BRK-A	Berkshire Hathaway Inc.	Finance	657B
TSM	Taiwan Semiconductor Man.	Electronic Technology	564B
JPM	JPMorgan Chase & Co.	Finance	463B
GME	GameStop	Retail Trade	12B

Source: Author's calculations

To decide the length of examined period, textual data sources were inspected first, in order to ensure a sufficient amount of data for sentiment analysis. This resulted in collecting data from the beginning of 2015 up until the end of April 2022. Additionally, the period was then split into 3 subsets:

- **Training**: 1.1.2015 - 31.12.2019
- **Evaluation**: 1.1.2020 - 31.12.2020
- **Test**: 1.1.2021 - 30.4.2022

The reasoning behind this will be described in more detail in chapter Subsection 4.2.1.

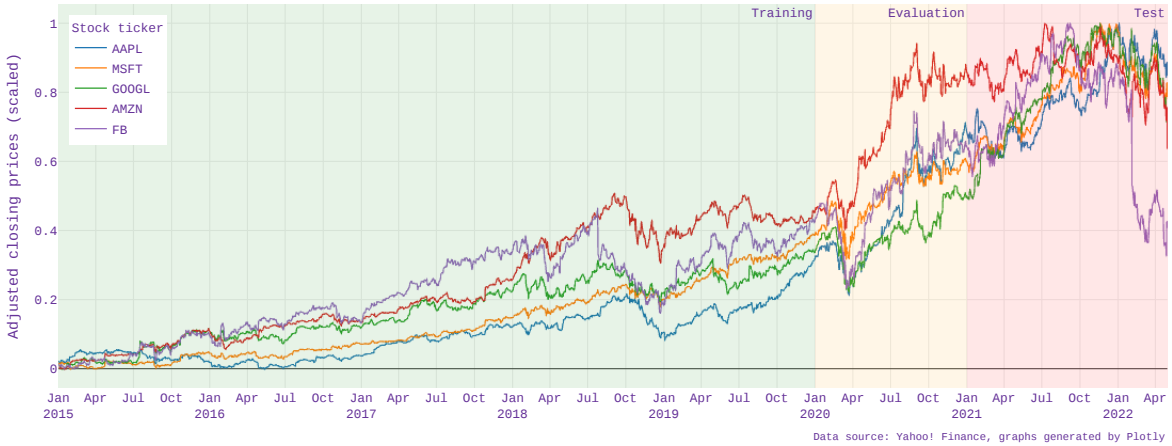
3.1 Stock Historical Prices

There is plenty of financial market data sources, but the choice should not be relevant as long as full history is available. For purpose of this thesis, Yahoo! Finance API was utilized, providing an access to data from New York Stock Exchange (NYSE). Even though closing price are usually enough to describe a trend in price, the intra-day trades can also be very volatile, which is possibly very informative for the model, that is making a prediction about the following day. Therefore, for each trading day, a set of 4 price variables was selected, including Open, High, Low and Close, together with traded Volume.

Figure 3.1 then depicts series of adjusted closing prices of the first set of stocks. For relative comparison, each time series was min-max scaled, and the

overall upward trend can be easily noticed. This might suggest that Buy-and-Hold strategy is indeed a viable option in longer-term in the case of any of the selected stocks and will therefore be a challenge for the trading model to outperform it. Also, an important event happened, namely the Corona crisis. The situation is very unclear and after the initial shock, markets were able to recover in a matter of months. The growth is still present and the situation around Covid is still very unclear at the point of writing the thesis. This, however, immediately raises concerns about whether such Training/Evaluation/Test split is representative enough so that the model is able to learn and generalize, but this will be the subject to further testing. The remaining stocks are available in the appendix (see Figure A.1)

Figure 3.1: Historical prices (part 1)



Many approaches aim to describe the behaviour of time series based on price lags (Auto-Regressive Integrated Moving Average (ARIMA) and Generalized Auto-Regressive Conditional Heteroskedasticity (GARCH) family models) or perform a pattern recognition with a use of neural network (LSTM models). In this thesis, however, the previously mentioned price variables will be used to derive 10 following features, that will hopefully best describe the current state. These include 5 return variables with different period lags and 5 financial indicators, including Stochastic, Ultimate Oscillator, Relative Strength Index (RSI), Moving Average Convergence/Divergence (MACD) and Average True Range (ATR). The choice of different time periods for returns was to account for a day, 2-days, a week, 2-weeks and a month trend. In the case of

indicators, it becomes more complicated and each will be described in more detail with appropriate formulas.

In the following subsections, it was convenient to denote i as an indicator of the time period. Variables $Open_i$, $High_i$, Low_i and $Close_i$ then stand for opening, the highest, the lowest and closing prices for a given day i. The implementation was done by TA-Lib library and formulas are based on provided documentation.[1]

3.1.1 Returns

First, a set of 1-day, 2-days, 5-days, 10-days and 21-days returns is defined to indicate recent trends in closing prices. They are defined as a simple percentage change, which can be rewritten as:

$$Return_{i,k} = 100 \times \frac{Close_i - Close_{i-k}}{Close_{i-k}}$$

where $k \in \{1, 2, 5, 10, 21\}$. This is expected to provide the most information for the proposed model, but to perform a proper technical analysis, also following variables were considered.

3.1.2 Momentum Indicators

- **Stochastic Oscillator** - Compares closing price to the maximum price range over the last 5 days. The final result is then obtained by smoothing this measure over a period of 3 days. Depending on whether the crossover of slow and fast oscillators happens from the bottom or top, it is then utilized as a trading signal.

$$StochOscFast_{i,k} = 100 \times \frac{Close_i - min(Low_{i-k:i})}{max(High_{i-k:i}) - min(Low_{i-k:i})}$$

$$StochOscSlow_i = \frac{\sum_{j=0}^{2} StochOscFast_{i-j,5}}{3}$$

- **Ultimate Oscillator** - First, Buying pressure (BP) and True range (TR) are defined, then summed and compared with each other to calculate a single period oscillator. The main advantage is, that it then corrects for bearish divergence as a weighted sum of three oscillators of different time frames is calculated (7-day, 14-day and 28-day).

[1] `https://www.fmlabs.com/reference`

$$BP_i = Close_i - min(Low_i, Close_{i-1})$$

$$TR_i = max(High_i, Close_{i-1}) - min(Low_i, Close_{i-1})$$

$$Average_{i,n} = \frac{\sum_{j=0}^{n} BP_{i-j}}{\sum_{j=0}^{n} TR_{i-j}}$$

$$UltOsc_i = 100 \times \frac{(4 \times Average_{i,7}) + (2 \times Average_{i,14}) + Average_{i,28}}{7}$$

- **Relative Strength Index** - Upward (Up) and downward (Down) price movements are measured, while also calculating their rolling averages. These are then used to derive the ratio of average upward movements to absolute average movements of closing prices during the last 14 days. The index itself may also help with indicating market reversals.

$$Up_i = \begin{cases} Close_i - Close_{i-1}, & \text{if } Close_i > Close_{i-1} \\ 0, & \text{otherwise} \end{cases}$$

$$Down_i = \begin{cases} Close_{i-1} - Close_i, & \text{if } Close_{i-1} > Close_i \\ 0, & \text{otherwise} \end{cases}$$

$$UpAvg_{i,n} = \frac{UpAvg_{i-1} \times (n-1) + Up_i}{n}$$

$$DownAvg_{i,n} = \frac{DownAvg_{i-1} \times (n-1) + Down_i}{n}$$

$$RSI_i = 100 \times \frac{UpAvg_{i,14}}{UpAvg_{i,14} + DownAvg_{i,14}}$$

- **Moving Average Convergence/Divergence** - Calculated as a difference between two Exponential Moving Averages (EMAs) with a multiplier of $\frac{2}{n+1}$ and periods set to 12 and 26 days. It is usually used to indicate beginning of a trend direction, and similarly to stochastic oscillator, the crossovers of the MACD and its 9-day EMA may also be interesting to derive trading signals.

$$EMA_{i,n} = Close_i \times \frac{2}{n+1} + EMA_{i-1,n} \times (1 - \frac{2}{n+1})$$

$$MACD_i = EMA_{i,12} - EMA_{i,26}$$

3.1.3 Volatility Indicators

- **Average True Range** - The only measure of volatility included. It is a simple moving average of daily true ranges of prices over the last 14 days. The indication is not in a form of direction and is usually applied as an exit method for trading strategy.

$$TR_i = max(High_i, Close_{i-1}) - min(Low_i, Close_{i-1})$$

$$ATR_i = \frac{13 \times TR_{i-1} + TR_i}{14}$$

3.2 Textual Data

Collecting textual data has shown to be very challenging. With the magnitude of textual sources that are available to everyone nowadays, it is nearly impossible to capture all of it. In many studies, two data sources, in particular, had proven great results, namely News and Social media. Additionally to the technical analysis described in Section 3.1, these are expected to track the development in fundamentals and provide valuable information via the market sentiment. Each of the sources will be described in more detail in the following subsections and the calculation of sentiment itself will be described in Chapter 4.

3.2.1 News Media

Every day, thousands of news media publish articles, that might be relevant to forming market sentiment. However, it might be very complicated to appropriately attribute these to particular stock and it is also common, that a single article can be related to multiple companies. The usual practice is to label them manually usually by author, however, one could also apply knowledge graphs in order to classify them[2].

For purpose of this thesis, the financial website `investing.com` was utilized, which provides many real-time financial data, charts, analyses and even an alert system. The main advantage is then that it also aggregates labelled financial news relevant for a particular stock, so a reader can better form his opinion about the current state of a company. Apart from their own published articles, it also gathers other news platforms such as *Reuters*, *Seeking Alpha*,

[2]See for example Nan *et al.* (2020)

Bloomberg, *The Motley Fool*, *CNBC*, *International Business Times*, *MarketWatch*, *Cryptovest*, *Business Insider*, *247wallst* or *Cointelegraph*. Additionally to this, so-called opinions and analyses are also provided, where the vast majority comes from *Zacks Investment Research*. Three datasets were derived, namely news articles, news headlines and opinions, and then used separately in order to derive the sentiment.

It is fortunate, that financial news and opinions archive web scraped from `investing.com` is publicly available on Kaggle[3]. However, the data covers only 12 years till the beginning of 2020, which is not sufficient for the thesis. Therefore, the remaining 2 years of data had to be web-scraped manually.

The first step was to collect the links along with other information for news articles and opinions. This was done by requesting the links[4], parsing the response and extracting the particular fields via *BeautifulSoup* package. These include *ticker label*, *article title*, *category*, *release date*, *provider*, *URL link* and *article id*.

The second step was to request each URL link to extract all paragraphs and concatenate them into article content. This was quite problematic as not all pages have identical structure, but some patterns were distinguished. If the scraper was unsuccessful even after all of the options, it would fail and not include the article.

Finally, there has been post-processing to clean the textual data and prepare it for sentiment analysis. This included lower casing and removal of URL links and white spaces. Other techniques such as stemming, lemmatization or stopwords removal were not necessary as FinBert is robust to them. Total counts of published news and opinions with respect to tickers are available in Table Table 3.2. It can be seen that Reuters and Zachs Investment Research will be the main source of news and opinions respectively. Also, it is clear that some of the tickers are poorly covered by these sources and therefore have to be taken with caution.

[3] `https://www.kaggle.com/gennadiyr/us-equities-news-data`

[4] For example *AAPL*:

`https://www.investing.com/equities/apple-computer-inc-news`

`https://www.investing.com/equities/apple-computer-inc-opinion`

Table 3.2: Published News and Opinions

	AAPL	MSFT	GOOGL	AMZN	FB	TSLA	BRK-A	TSM	NVDA	JPM	GME
News											
Reuters	6838	3489	5465	4871	11154	3825	19	6	313	2784	526
Investing.com	2860	1721	1418	1572	2394	1682	89	63	577	1035	420
Seeking Alpha	892	472	262	544	660	199	5	32	87	122	43
Bloomberg	394	154	163	237	351	279	0	5	17	740	53
Cointelegraph	174	157	404	153	464	330	0	0	43	203	86
Other	655	333	620	481	614	824	18	10	75	207	118
Opinions											
Zacks Investment	10874	5417	1463	3849	6594	1359	2	116	775	814	237
JJ Kinahan	272	124	78	135	157	138	2	5	71	55	17
Haris Anwar	188	126	86	176	156	108	5	26	40	38	14
Michael Kramer	214	84	40	190	144	124	0	15	96	39	5
Pinchas Cohen	210	73	67	101	174	110	4	8	16	50	10
Other	5231	1618	1428	1727	3099	1802	269	87	276	759	316

Source: Author's calculations

3.2.2 Social Media

Social networking services have seen an exponential increase in user base during the last decade. This has allowed anyone to express his opinion, while also being able to be heard and spread all over the world. There are many platforms available, each providing a bit different tools for communication and targetting various socio-demographic groups.

Almost every research published related to market sentiment and social media is based on *Twitter* data, which will also have the focus of this thesis. However, there are other options that might be worth considering, such as *Reddit*[5] or *Stocktwits*[6]. Originally, data from both platforms were supposed to be collected as well, but some difficulties occurred. In the case of *Reddit*, it was very complicated to navigate through the website in order to extract valuable information. This is because of the overall structure, as a single post is typically followed by thousands of comments, that can be further nested. Nevertheless, ignoring the comment section and focusing solely on posts would not provide a sufficient amount of data. In the case of *Stocktwits*, an equivalent of *Twitter* focused on finance and likely the most appropriate data source, its API was under review and not available at the time of writing the thesis. Therefore, both of the sources will be out of scope, but could be subject to further research.

Even though *Twitter* publicly provides an advanced search tool[7], there is no straightforward way of collecting the data points. Therefore, a scraping package

[5]Especially subreddit `https://www.reddit.com/r/wallstreetbets/`

[6]`https://stocktwits.com/`

[7]`https://twitter.com/search-advanced`

snscrape[8] was used, allowing to extract a great magnitude of tweets. It offers plenty of custom configuration, but most importantly the *time period*, *language*, *minimum number of likes* and last but not least the *search term* itself. The scraping was then performed by looping through all years and selecting only English-written tweets, that included the particular search term and satisfied other filter conditions. In this fashion, two distinct datasets were collected with different sets of keywords. In the first case, to attribute a tweet to a particular stock, it only had to contain the ticker symbol with a $ prefix, i.e. the cashtag. In the second more complex query, also other keywords were considered, such as stock names, products, services, subsidiaries and company founders or other key people. Their list with respect to each stock ticker is provided in Table 3.3.

Table 3.3: List of Tweet Keywords

Ticker	**Keywords**
AAPL	$aapl, apple, steve jobs, steve wozniak, ronald wayne, tim cook, macintosh, mac, ios, ipod, iphone, ipad, airpods, homepod, icloud, itunes
MSFT	$msft, microsoft, bill gates, paul allen, windows, office, skype, visual studio, xbox, azure, bing, linkedin, yammer, onedrive, outlook, github, game pass, sharepoint, visual studio
GOOGL	$googl, alphabet, google, larry page, sergey brin, sundar pichai, gmail, firebase, tensorflow, android, chrome, adwords, nexus, pixel, youtube, deepmind
AMZN	$amzn, amazon, jeff bezos, andy jassy, kindle, alexa, fire tv, fire tablet, aws, audible, goodreads, imdb, twitch
FB	$fb, meta, mark zuckerberg, zuckerberg, eduardo saverin, metaverse, facebook, messenger, instagram, whatsapp, oculus, mapillary
TSLA	$tsla, tesla, elon musk, elon, musk, model s, cybertruck, model x, model y, model 3, powerwall, powerpack, deepscale, solarcity
BRK-A	$brk.a, $brk.b, $brk, berkshire hathaway, oliver chace, warren buffett, buffett, altalink, kern river pipeline, northern natural gas, fruit of the loom, netjets, russell brands
TSM	$tsm, $tsmc, taiwan semiconductor, morris chang, mark liu, cybershuttle wafertech, ssmc
NVDA	$nvda, nvidia, jensen huang, geforce, mellanox, physx, deepmap
JPM	$jpm, jpmorgan, j.p. morgan, john pierpont morgan, jamie dimon, chase bank, one equity partners
GME	$gme, gamestop, ryan cohen, matt furlong

Source: Author's calculations

During post-processing, some cleaning was done to treat ambiguous tweets. In spite of that, there is still the possibility of misclassified tweets for keywords such as "apple", but they are expected to be consistent over time and therefore not significantly bias the data. Descriptive statistics are shown in Table 3.4. It is important to note, that to be able to collect the tweets in a reasonable

[8] `https://github.com/JustAnotherArchivist/snscrape`

time, one has to set some limits. Therefore, a minimum of 10 likes per tweet in *cashtag* and a minimum of 100 likes in *keyword* datasets were selected. Still, the runtime of the twitter scraping process was roughly two weeks.

Table 3.4: Descriptive Statistics of Posted Tweets

	AAPL	MSFT	GOOGL	AMZN	FB	TSLA	BRK-A	TSM	NVDA	JPM	GME
Cashtags only (min 10 likes per tweet)											
tweetCount	53,8K	20,9K	13,4K	44,0K	39,7K	299,0K	2,2K	1,8K	19,9K	6,0K	87,1K
replyCount	375,7K	144,4K	89,4K	317,7K	267,9K	2,7M	15,7K	12,7K	131,2K	33,4K	926,2K
retweetCount	816,3K	322,3K	241,0K	661,0K	553,7K	3,6M	33,1K	26,6K	219,5K	98,6K	1,9M
likeCount	4,0M	1,7M	1,1M	3,5M	2,8M	34,0M	157,0K	141,0K	1,5M	356,7K	13,7M
All keywords (min 100 likes per tweet)											
tweetCount	641,4K	913,5K	869,4K	381,8K	1,1M	196,2K	8,5K	0,5K	7,0K	4,6K	25,9K
replyCount	31,5M	60,5M	37,8M	18,4M	44,1M	15,6M	335,3K	20,5K	1,2M	284,8K	1,3M
retweetCount	224,7M	269,1M	282,8M	88,3M	335,3M	45,0M	2,3M	87,7K	2,3M	1,2M	4,8M
likeCount	864,4M	1179,7M	1261,1M	525,6M	1441,4M	336,2M	7,6M	454,9K	7,0M	4,8M	32,7M

Source: Author's calculations

It is interesting to see, that some of the tickers, see for example $MSFT, have up to 60-times more tweets in *keyword* dataset despite the more strict limitations. On the other hand, $TSLA and $GME tweets are in most cases cashtagged. These are the stocks that have experienced a controversy and a bullish market recently. Also, similarly to the case of the News dataset, some tickers have a problem of having very few textual sources to work with, but the problem is less severe.

Chapter 4

Methodology

The thesis combines multiple areas of research, most notably Finance, NLP and Deep Reinforcement Learning (DRL). This chapter starts with a basic financial background to build a foundation in economic theory for the proposed trading model. The remaining parts then focus on Machine Learning (ML) concepts and algorithms. Section 4.2 provides an introduction to ML, while Section 4.3 defines basic Neural Networks (NN) along with more complex neural architectures. Section 4.4 then describes the process of extracting valuable information from textual data sources and Section 4.5 defines an agent that is able to learn based on interactions with the environment. Finally, all of the methods described are put together in Section 4.6 and the reasoning behind the assumptions for a trading agent is explained.

4.1 Financial Background

Before diving deep into machine learning, it is important to answer questions about the validity of trading algorithms. From an economic point of view, there are many theories that might contradict it, but counterarguments will be presented. Additionally, the concepts from behavioural economics will be examined, explaining why there might be opportunities to exploit. This will build a foundation for further development of the sentiment-based trading agent.

4.1.1 Efficient Market Hypothesis

Efficient Market Hypothesis (EMH) has been a well-known theory for several decades and has been exposed to much research. It has been independently developed by Fama (1965) and Samuelson (1965), who took a different positions

even though the core concept was the same. While Fama argued that the fluctuations are explained by convergences to fundamental values, Samuelson described it by the competitive nature of investors without a strong link to fundaments. And despite the long history, it is still present in many market model assumptions.

Among other authors, Brealey *et al.* (2012) described markets to be efficient if one cannot earn higher returns than market returns when adjusted for risk. If investors are able to achieve higher returns, it is purely because of speculation with substantial risk. Similarly, Mishkin & Eakins (2006) argued that efficient markets project all available information into the stock prices.

In essence, EMH aims to answer the question of whether it is even possible to outperform the market. An efficient market would be expected to reflect the fundamental values directly into stock prices. In other words, the intrinsic value of a company should be exactly the same as its market value. As a consequence, investors should have no opportunities to purchase the stock undervalued or sell it overvalued.

Based on the extent of the known information, three forms of market efficiency are usually distinguished:

> **Weakly efficient** markets reflect all relevant information from the past such as the historical prices. This prohibits the possibility of excess profits and technical analysis should not be of any use.
>
> **Semi-strongly efficient** markets broaden it to all public information that is currently available. Because of this, even fundamental analysis can not assist investors.
>
> **Strongly efficient** markets then add assumption about the knowledge of private information, making insider trading impossible to be profitable. Even though it may sound unrealistic, it is still an illegal practice.

However, empirically it is very difficult to validate any form. There has also been research on market anomalies, such as January effect or Neglected firm effect, making it even harder to prove and up to this date, it remains controversial.

The theory is, of course, not only key for investors, but macroeconomists and policymakers are very interested in understanding the concept as well. Being able to precisely determine the current state of the economy would result in better decisions, that could possibly mitigate the effects of recessions or

appropriately boost the economy. Furthermore, company executives should also be involved in the understanding of what decisions are perceived positively and what drives the stock prices upwards in order to effectively maximize value for shareholders.

4.1.2 Adaptive Market Hypothesis

In recent years, the focus has moved from perfect rational agents to behavioural economics, providing useful concepts to explain market anomalies and inefficiencies in general. The research interconnects economics with psychology and focuses to explain irrationalities that seem to be part of human nature.

Behavioural theories are often based on field experiments, which is very uncommon for economics as it is usually the theory that precedes the empirics. There have also been many concepts of behavioural biases described in the literature which could be relevant in the context of EMH, often resulting in poor economic decisions:

Loss Aversion describes the risk-averse behaviour in relation to gains and risk-seeking to losses. As a result, investors tend to hold onto losing stocks for too long in a hope of mitigating it (Kahneman & Tversky 1979).

Psychological Accounting is an idea that contradicts money's property of fungibility, based on which people use multiple mental accounts, with different levels of willingness to withdraw the money (Tversky & Kahneman 1981).

Overreaction to new information was also observed as it is difficult for humans to evaluate unexpected events in a short period of time. This in turn can lead to markets driven by emotions (De Bondt & Thaler 1985).

Herding behaviour is also present in financial markets as investors often justify their decision by other's opinions. In their belief, they are more knowledgable, without realizing they can exploit them for their own benefits (Huberman & Regev 2001).

Taking this into account, an alternative theory, explaining why the markets are not always perfectly efficient, was published by Lo (2004). The author proposed an Adaptive Market Hypothesis (AMH) and attempted to find a middle ground between the perfect rationality of EMH and inefficiencies driven by human behaviour.

AMH can be perceived as EMH that also considers 3 evolutionary principles of competition, adaptation and natural selection, under which, the conflicting concepts can co-exist. This is because of parallels in evolutionary theory, according to which humans optimize their behaviour to achieve satisfactory levels of utility rather than maximal. The choices depend on past experiences acquired through positive or negative reinforcements, that are later shaped into heuristics. It is the trial and error as well as natural selection that defines the process of adaptation to market behaviour and all of the biases described above are consistent with it.

The author believes, that investors are rational in most situations, but sometimes, when the environment suddenly changes, all the experiences no longer yield good decisions and humans start to make mistakes. Sudden shifts in uncertainty on markets can cause overreaction, under which heuristics are not adequately adjusted and market participants become irrational. The author also argued that such behaviour should not be considered irrational but rather "sub-optimal" or "maladaptive".

As an implication, arbitrage opportunities are indeed possible during certain periods, usually characterized by increased volatility. In order to achieve higher expected returns, it is therefore important to keep track of changes in the market conditions through time. Based on this, author of this thesis believes that RL is indeed capable of describing this behaviour and is an appropriate method for stock trading.

4.1.3 Agent-based Modelling

Agent-based Modelling (ABM) is an approach that provides a great alternative to traditional financial market models. It is closely related to RL, however, ABM is more common in economic literature and has already helped to explain many economic phenomena, see for example Samanidou *et al.* (2007). Even though this thesis will focus solely on RL, it is still interesting to find parallels in these concepts.

In both cases, it relies on a substantial amount of simulations, allowing to address problems with no analytical solution or one that is too difficult to find. The central element is an agent that has some properties and is allowed to perform a given set of actions. It then interacts with other, most often heterogeneous, agents in a pre-defined environment. While the ABM focuses on describing the interactions themselves, RL's objective is to learn from past

experiences in order to find the best set of actions that will result in the highest cumulated reward.

The biggest benefit of ABM is the ability to capture the full complexity of interactions of many different entities, including institutions. Additionally, so-called emergent behaviour, in which actions of individual agents combine into a macro-level behaviour may show surprising results. This is in line with Adam Smith's invisible hand, which can be made partially visible in this manner and can provide many interesting insights and behavioural patterns.

This is why ABM is often used to model markets as a whole, rather than to optimize investors' decisions. Nevertheless, to be able to describe the behaviour, it usually requires a significant amount of simplifications, and one can examine different sets of scenarios based on altering model assumptions.

4.2 Machine Learning Introduction

Before proceeding to Deep Learning (DL), it is necessary to review the basic concepts of ML. The main idea behind this is to be able to learn from given data and later apply this knowledge to an unseen data. Usually, there is an outlying task that is the subject to learning and a performance measure[1] that gives feedback on how well is the algorithm currently performing. With additional data, this performance is expected to improve. The ultimate task of the proposed approach will be to train an agent that can trade stocks while maximizing returns. To achieve this, several sub-tasks need to be performed, such as estimation of market sentiment and extraction of features that can best describe the current state of the market. Most of the theory in this section will draw from Goodfellow *et al.* (2016) if not mentioned otherwise.

At least three different types of ML models can be distinguished:

Unsupervised Learning explores patterns and properties in data and tries to learn rules that would quantify them. The most common example is clustering[2] in which the data points are distributed into groups based on similarities. There are many other examples such as dimension reduction [3], density estimation or NN learning patterns, but none of them will be required for the thesis.

[1] Sometimes also referred to as an error

[2] Such as k-means, k-nearest neighbours or hierarchical clustering

[3] Such as principle component analysis, factor analysis, linear discriminant analysis

Supervised Learning on the other hand requires a golden label for each data point. This label can be both discrete or continuous, which distinguishes it between classification and regression. The goal is to learn to predict it based on input features while minimizing the error. Algorithms include for instance linear and logistic regression, decision trees, random forest or support vector machines. However, this thesis will utilize only deep NN for feature extraction as well as sentiment analysis.

Reinforcement Learning algorithms are a special case as they are able to generate data thanks to interactions with a defined environment and there is usually no need to label it. Based on this new experience, agent learns what decisions to make in particular situations in order to maximize/minimize cumulative reward/cost. Again, many types have been developed, for instance, policy gradient models, SARSA[4], DDPG and most importantly Q-learning, which will be described later.

4.2.1 Under-fitting and Over-fitting

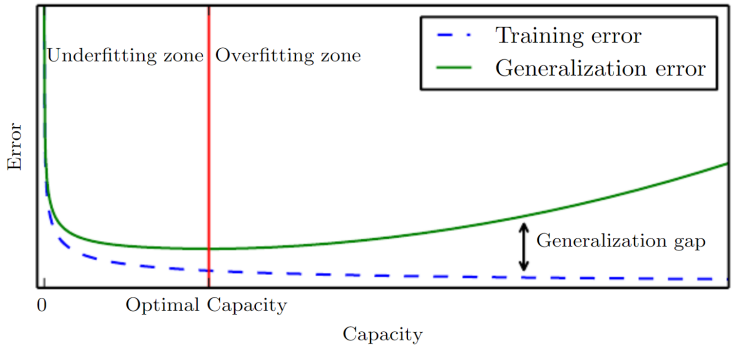
The primary target of machine learning algorithms is to be able to work with new data, that were not part of the training process. If this can be achieved, it is said that the algorithm is able to generalize. In the context of supervised learning, when the objective is to minimize some loss from predictions, the dataset needs to be split into two parts, namely training and test. Sometimes, a part of the training test is further split for evaluation, which is also this case. Evaluation and Test sets each usually have 10-20% of full data, the rest is distributed to training. This split is random, however, in time series the temporal structure is usually preserved.

Each of the 3 datasets is then connected with an error with a different purpose. Training error is minimized as an objective of the learning algorithm, while evaluation[5] error provides information on expected performance on unseen data. Test error, or sometimes generalization error, measures the final performance of the model and cannot be used to alter hyper-parameters, otherwise, it would no longer be out-of-sample. In order to generalize well, all of the datasets are assumed to be generated from approximately the same data generating process with a similar probability distribution.

[4] state–action–reward–state–action
[5] Sometimes also called validation.

During the optimization, only training error is minimized, while generalization error is being estimated on the validation set. Hyper-parameters[6] can be then iteratively altered in order to minimize the generalization error. If the training error is too low compared to the evaluation, the model is over-fitting. On the other hand, if the model can not achieve reasonably low error on training, it is said to be under-fitting. This can be adjusted with the model's capacity and is depicted in Figure 4.1. Higher-capacity models are expected to handle more diverse data during the training while being more prone to over-fitting. Lower capacity can prevent the model from learning important relationships and therefore under-fit.

Figure 4.1: Under-fitting versus Over-fitting



Source: `https://www.deeplearningbook.org/`

The adjustment of capacity can be done in various fashions. Apart from model choice itself, each has so-called representational capacity. This defines the "family of functions" the model is able to fit. For instance, in the case of NN, this could be a number of neurons, layers or overall architecture. At the same time, there is an effective capacity, which limits the representational and is achieved by the introduction of optimization imperfections. The training error is essentially increased in belief to reduce the generalization error. Typical examples are L1 and L2 regularizations.

4.2.2 Gradient Descent

Now, the optimization of ML models will be inspected. The target is usually to minimize a loss function. To achieve this, partial derivatives with respect to all inputs are calculated first. The model's weights are then updated by subtracting a fraction ϵ of the gradient. The idea is to shift the weights in the direction

[6] Any parameter that has to be set a priori as it controls the algorithm's behaviour

where the loss is decreasing. Gradient descent is an iterative application of this algorithm and is the most common technique used for optimization.

Parameter ϵ is also called learning rate as it represents the length of each step. In order to ensure convergence of loss function to a local minimum, it must satisfy:

$$\forall i : \epsilon_i > 0, \quad \sum_i \epsilon_i = \infty, \quad \sum_i \epsilon_i^2 < \infty$$

In other words, it has to be positive, sufficiently high but not too much to over-step. If the loss function is non-convex, it cannot be ensured that the local optimal translates to a global one. Usually, it is sufficient to stop on reasonably low loss values, as finding a global optimum is nearly impossible in multidimensional space and is considered at least NP–hard.

If null derivatives are reached, i.e. every element of the gradient is equal to zero, one of three critical points is found: local minimum, local maximum or saddle point. In practice, for algorithm convergence, it is sufficient if the derivatives are very close to zero. Problems could arise once ending up exactly on the local maximum or saddle point during minimization, but this is very unlikely to happen as the algorithm has a tendency to move away from them.

In standard gradient descent, all training data are used at the same time to calculate the Jacobian[7]. This is in most cases very computationally demanding and requires a great amount of memory. It could also be computed from a single data point, which is called Stochastic Gradient Descent (SGD), but this would produce very noisy estimates. Therefore, it is more convenient to draw mini-batches of training data points and average the Jacobian.

Mini-batch SGD is an example of a first-order optimization algorithm. However, calculating the Hessian[8] has proved to be useful for optimization as well. The idea is to measure the curvature of the loss function and use this information for faster convergence. This usually allows algorithms such as AdaGrad, RMSProp or their successor Adam to outperform vanilla SGD.

4.3 Neural Networks

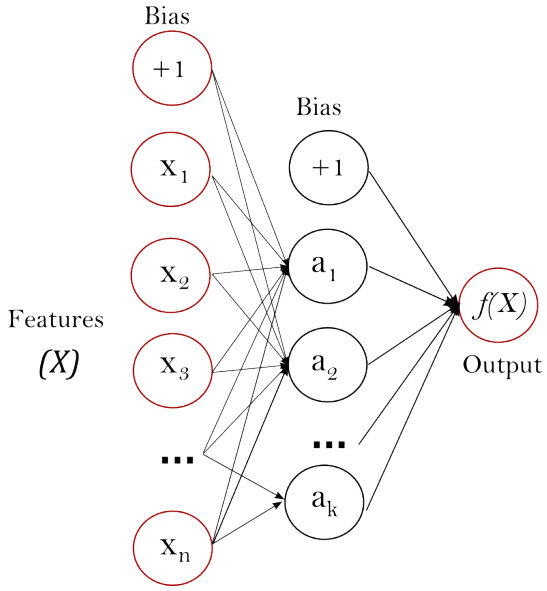
In this section, the knowledge required in Section 4.3.2 and Subsection 4.5.3 will be built. NNs are considered to be a backbone of most DL algorithms and similarly, this section also draws from Goodfellow *et al.* (2016). It will start

[7]Matrix of first-order partial derivatives
[8]Matrix of second-order partial derivatives

by defining a general architecture, which will then be further develop in the context of DL.

The basic NN architecture, sometimes also called Multilayer Perceptron (MLP) or Feed-forward NN, is depicted in Figure 4.2. As the name suggests, computations during the model's inference are only fed forward in the same direction[9]. The reference can also be seen in neurobiology, as each node represented by a single weight (neuron) sends a signal to multiple other weights via predefined connections (axons). In this fashion, NN tries to define a mapping for an underlying function, which structure can be often very abstract. The idea is not to perfectly explain the inner behaviour of the model (brain), but rather to approximate it and apply such a system for generalization. This could be a prediction of time series or a measurement of sentiment.

Figure 4.2: Multilayer Perceptron



Source: `https://scikit-learn.org/stable/modules/neural_networks_supervised.html`

Each NN layer is represented by vector-to-scalar or vector-to-vector mapping function and three types can be distinguished:

Input layer only provides input features and does not perform any calculation itself. The dimensionality is exactly the same as input features.

Hidden layer does perform calculations, but its output is very abstract and not controlled during training. However, it can be used as an efficient

[9]This will however be violated in Section 4.3.2.

feature extraction technique as NN tends to learn rules that provide better feature representations than input features.

Output layer is similar as it also has weights and activation function, but the output has meaning. During training, it is adjusted to best fit the golden labels. The activation functions are dependent on particular task, the most common are Linear[10], Sigmoid[11] and Softmax[12].

Both input and output layers are necessary for every NN, while a number of hidden layers determines the model's depth and is the reason why more complex models are called "deep". The number of weights in each layer then describes the width of the model.

In the simple case, using the same notation as in Figure 4.2, the calculations are done in the following order. First, the hidden layer is evaluated by computing a weighted linear sum of input features and hidden weights and typically non-linear activation is applied. Finally, the activated dot product of output from the hidden layer and output weights is calculated in a similar fashion to provide a single continuous output[13]. This process can be rewritten as:

$$a_i = A^h(\sum_{j=1}^{n} w_{i,j}^h x_j + b^h)$$

$$f(X) = A^o(\sum_{j=1}^{n} w_{i,j}^o a_j + b^o)$$

where X is vector of input features $\{x_1, x_2, ..., x_n\}$, activation function A, weights $w_{i,j}$ and bias b. Superscripts h and o distinguish between hidden and output layer. Formally, it is a higher-order function and a simple forward pass evaluates the model's output. Nevertheless, this explains only a fraction of how NNs work as there needs to be a mechanism that will adjust the weights to fit the data and learn to generalize well.

Training a NN follows the same concept of gradient descent learning as described in Subsection 4.2.2. However, the analytical calculation of gradient is computationally very demanding. This is when back-propagation comes to the rescue. Gradients are calculated efficiently thanks to the recursive application of a chain rule. In this fashion, gradients of all nodes with respect to any prior

[10]Regression
[11]Binary classification
[12]Multi-class classification
[13]Just an example, a number of outputs and activation function depends on the task.

node can be calculated. For SGD, the most important are gradients of the loss function with respect to weights and biases in each layer. Denoting L as loss, the chain rule implies:

$$\frac{\partial L}{\partial w_i^o} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial w_i^o} \quad \text{and} \quad \frac{\partial L}{\partial b^o} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial b^o}$$

$$\frac{\partial L}{\partial w_i^h} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial a_i}\frac{\partial a_i}{\partial w_i^h} \quad \text{and} \quad \frac{\partial L}{\partial b^h} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial a_i}\frac{\partial a_i}{\partial b^h}$$

To sum up, output from forward-propagation is used to calculate loss, while back-propagation provides information about how to change weights to decrease it. By applying this process iteratively during minibatch SGD, the model is learning to better approximate the underlying function.

4.3.1 Neural Network Configuration

Setting up a NN can be a complex procedure as there are many parameters to consider. Most of them have a significant impact on the model's behaviour during training and inference and have to be set prior to the experiment. They are referred to as hyper-parameters. To find an optimal setup, there is no universal strategy. In most cases, it is needed to iterate through the space of all hypothesised combinations and pick the one that performs best on the evaluation set. This is a technique of grid search. Alternatively, one could also perform cross-validation on each combination, but it is usually not employed in the case of deep learning as it requires a lot of computational resources.

In the following subsections, the main groups of hyper-parameters will be briefly described. A complete list will not be provided, but it suffices the needs for empirical analysis in Chapter 5

Loss Function

Optimization requires a loss function as an objective of the minimization problem, providing it with a measure of error. It is usually convenient to derive it via the maximum likelihood framework, which as in many use cases in econometrics or machine learning in general, allows finding parameter estimates that best describe the training data and therefore maximize its likelihood function. It can be also interpreted in terms of cross-entropy from information theory since the error is minimized between two probability distributions - empirical

and model. Consistency property then implies a better performance as the number of training examples increase.

The decision of which loss function to choose is directly connected with activation function of the output layer. In the case of binary and multi-class classification, cross-entropy is directly applied, while regression often leads to a Mean Squared Error. There is also so-called metric[14], which is usually correlated with loss function but used to evaluate the performance during validation. Depending on the task, one could use the metric for both purposes. However, it does not necessarily have the desirable properties and can be saturating, meaning that the gradient does not provide enough information for the algorithm to learn.

Regularization

As discussed in 4.2.1, the desired outcome is a model that perfectly generalizes on unseen data. To control for this and restrict the over-fitting, there are many regularization techniques depending on a particular task. It is also one of the most creative areas of deep learning, where one can find an edge to outperform other similar models.

The loss function itself often comes with some kind of regularization. The idea is to add an extra component that will penalize large weights as measured by their norm, which is either L1 or L2. It can be interpreted as a preference for less complicated models over more complex ones, which hopefully decorrelates the NN. The strength of this penalty can be then controlled directly by a single hyper-parameter.

Another example is label smoothing, a procedure that introduces noise to the output label by subtracting a small margin. In other words, it is sufficient if the model's predictions are reasonably close to the target and not needed to fit perfectly, which is often a sign of over-fitting. Therefore, the model has more space to focus on examples at which it was not yet very successful. Such an approach is, however, used exclusively for classification.

A different widely used technique is a dropout. As the name suggests, some information is dropped in order to avoid learning over-fitted rules. This is achieved by setting a probability threshold of dropping each individual node during training. This results in a simpler structure, that is likely to be unique on each iteration. It addresses cases of high dependence on particular nodes

[14]Such as Accuracy, F1 score, Mean absolute error, Area under curve

and aims to distribute the information among all nodes. There are also more sophisticated versions[15], respecting structures of deeper NN. On the other hand, dropping a single cell might not always be enough, which is the reason some authors experimented with dropping whole blocks of neurons, but more on that later in Subsection 4.3.2.

To some degree, one could implicitly regularize via batch normalization, even though it is not its main purpose. It normalizes inputs to ensure each feature has the same range in terms of the selected mini-batch. This in turn reduces learning time and improves stability, and the model is then expected to be more robust to outliers. Nevertheless, the connection with regularization is more on an empirical level rather than based on theory.

In some cases, it is also useful to stop the training process early, in a spite of the potential to further minimize the training loss. If the validation error starts to increase for a certain period, the learning process is immediately stopped and use the weights in their current state. This is essentially a selection of a perfect number of training epochs that results in the best performance on the validation set.

Last but not least, there are also approaches, that aim to make the best use of available data, commonly called data augmentation. A limited amount of data is a significant concern in most ML problems. Either the resources to collect it are not sufficient or data does not exist at all. No matter the reason, in essence, it is possible to generate new artificial examples by introducing some kind of noise. For instance, in the case of image processing, simple flipping, rotating, cropping, shrinking or colour modifications provide a unique set of inputs without much additional effort. One could also cut-out random regions or generate new examples fully by Generative Adversarial Networks to further reduce over-fitting. It is very dependent on a particular task, but sometimes possibilities may be endless.

Activation Function

By simply multiplying weights from each layer and passing it to the following one, a model would behave very similarly to the simple linear regression. To allow for non-linearities and capture more complex relationships in data, it is desirable to add a so-called activation function. It usually has a form of straightforward function, that in combination with other layers is able to cap-

[15] e.g. Variational Dropout, Gaussian Dropout, Inverted Dropout, Attention Dropout

ture very complex structures, which is one of the main benefits of using NN. Many activations can be distinguished depending on purpose as well as the type of layer. In most cases, the choice is based on the following list:

Output Layer Activation

- Linear: $f(x) = x$
- Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
- Softmax: $f(x) = \frac{e^x}{\sum e^x}$

Hidden Layer Activation

- ReLU: $f(x) = max(0, x)$
- Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
- Tanh: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

The choice of hidden activation has a significant effect on the model's ability to learn, and nowadays, it is common to use ReLU because of its convenient properties. On the other hand, the output activation determines the type of predictions. Therefore, a very limited set of options is available and it is usually a matter of regression[16] versus categorization[17].

Weight Initialization

Before any NN model can start learning, it is necessary to initialize all node weights that will be later optimized by gradient descent. There are many arguments why setting all parameters to the same values is not a good idea. For example, it could result in an inevitable convergence to local minima or saddle point. Also, if all nodes are updated by the same gradient, they end up having identical behaviour, which is the opposite of what is desirable to achieve. Biases, on the other hand, are usually initialized with zeros as there is no such issue.

Therefore, small values are randomly picked from a given distribution, usually uniform or Gaussian. There have been many experiments on scaling these random variables, most notably performed by Glorot & Bengio (2010). The author proposed to uniformly draw values from a range of $(-\frac{1}{\sqrt{N}}), \frac{1}{\sqrt{N}})$, where

[16] Linear
[17] Sigmoid - binary, Softmax - multi-class

N stands for the number of layer inputs. In a spite of the other variants, this is the rule-of-thumb in most NN setups.

Such randomness, however, has the potential to bring instability, since each initialization, as well as an optimization problem, is unique. Nevertheless, if the model is well defined and has reasonable weights in the beginning, it was shown not to have any significant effect on convergence.

Architecture

Term architecture in the context of NN is very broad and these are the simplest variants:

Perceptron is an example of a fully connected two-layer[18] architecture, which simply applies a linear combination on each of the inputs and activates it. Apart from setting up the number of neurons, there is not much to adjust.

Multi-layer Perceptron then allows to connect additional hidden layers and capture more complex structures. It is usually sufficient to add 2 or 3 hidden layers, but there is no limitation. More layers and neurons increase capacity as well as the danger of over-fitting, which was discussed in Subsection 4.2.1.

In both of these examples, the output from each layer is fed forward without any backward loops[19]. This is the reason why they are a subclass of so-called Feed-forward NN, which do not need to be fully connected. More advanced architectures that violate the forward rule will be introduced in the following Subsection 4.3.2.

4.3.2 Deep Neural Networks

Completely altering NN architecture is expected to better represent even more complicated functions. In general, this is the main merit of DL and has been a crucial area of focus since the early 90s. For instance, the first convolutions were described by LeCun *et al.* (1989), allowing to efficiently process image data, or the LSTM cell that was introduced by Hochreiter & Schmidhuber (1997), which is nowadays an inseparable part of recurrent networks. The

[18] Input and output layers.
[19] Not considering the back-propagation algorithm.

fastest growth could be seen in the last 10 years as there have been many milestones, such as very efficient convolutions in AlexNet by Krizhevsky *et al.* (2012) or the AlphaGo beating a world champion.

Many researchers have been successful in finding ways to make the models train quicker, perform better and utilize big data thanks to great development in the computing power of CPUs[20] and GPUs[21]. But there has been no architecture that would fit all kinds of data and each has its specific use cases. Images are likely to utilize convolutions, while time-series benefit most from recurrent cells and transformers. There are of course many other structures, but these three will be described in the following subsections as they are most relevant for building the trading agent.

Convolutional Neural Networks

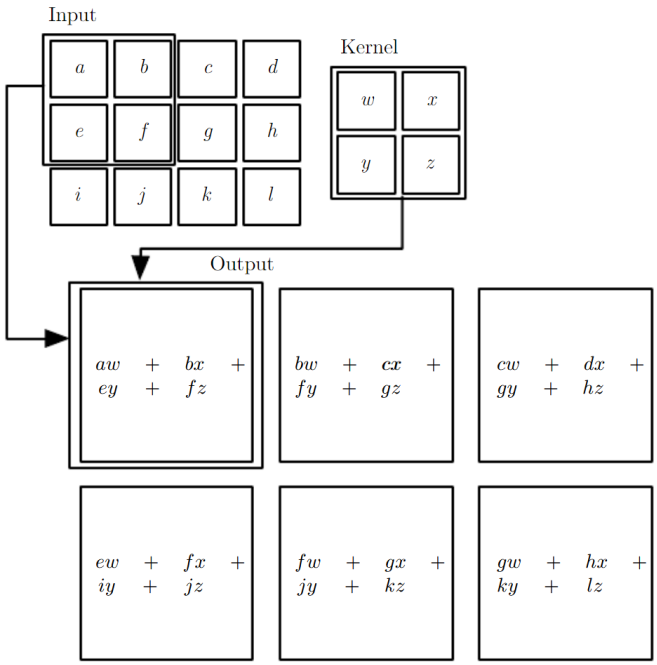
Convolutions excel in preserving data structures, especially in capturing the local interactions and being invariant to shifts. This is very useful for images, as it means that pixels that are closer are more relevant to each other and there is no difference whether the object is in left or right corner. Nevertheless, spatial and temporal dependencies can be important in all other kinds of applications, not only images. For example, CNN have found its use also in time series modelling, where they can successfully capture cyclical and trend components.

A simple convolutional operation is depicted in Figure 4.3. As you can see, cells are no longer fully connected and there is a special pattern for connecting them. This is achieved by so-called kernel, which shares parameters for all outputs. By stacking multiple convolutions on top of each other, it transfers from low-level details, to high-level. This allows to extract features on different aggregation levels and at the same time combine them. Apart from the size of the kernel, setting up convolutions also involves the decision of what paddings and strides to use, possibly making the computation more efficient by skipping some of the inputs.

In most cases, pooling layers are added in between convolutions, which have a similar purpose of dimensional reduction. But in this case, no multiplication is performed and only dominant features are preserved. The operation is either max pooling or average pooling, and as the name suggests, only maximum/average of all values of the weight matrix covered by the kernel is extracted. The final layer in CNN is usually fully connected as was in MLP.

[20] Central processing unit
[21] Graphics processing unit

Figure 4.3: Convolution



Source: `https://www.deeplearningbook.org/contents/convnets.html`

Recurrent Neural Networks

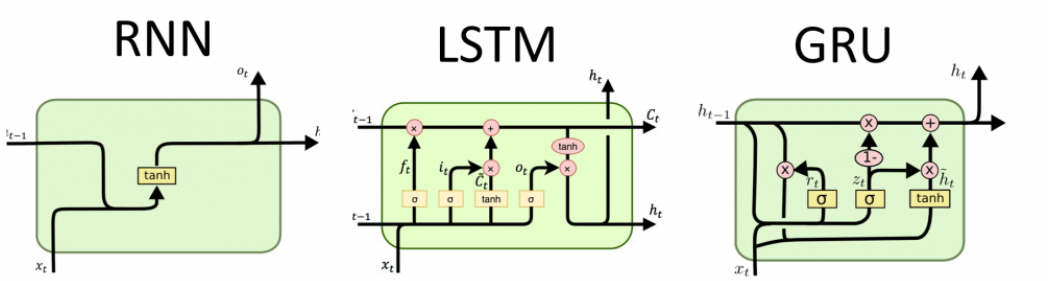
MLPs and CNNs are, however, not very efficient for time series or sequential data in general. The dependency on other data points from the same series has a vital role in determining the future outcome. This is where RNNs excel and proved to outperform other architectures.

Recurrent in this context means, that for each element of time series input, there is the same operation performed in a sequence, that also utilizes output from the previous one. In essence, it is a special module that memorizes past information to reflect the temporal structure and even though it is usually abstract, it can represent various meanings. For example, in NLP domain, the memory stands for previous words in a speech, or similarly in time series forecasting, its the evolution of stock's market value, its trend or seasonality. While CNNs share the weights in filters, RNNs share them across sequences.

Even though the idea is simple, there are many challenges to deal with. Vanishing gradients are one of them and are crucial to address, as otherwise, the NN will fail to learn long-term dependencies. This is because gradients passed through RNN layers have a tendency to exponentially diminish with the increasing length of the series. Therefore in practice, a simple RNN is rarely

used and requires some modifications. Most commonly used are GRU and LSTM and the idea is very similar but executed differently. While in GRU, there are 2 types of gates, namely reset and update, the LSTM has 3 gates - input, output and forget. These are shown in Figure 4.4.

Figure 4.4: RNN cells



Source: `http://dprogrammer.org/rnn-lstm-gru`

The input gate learns what information from the previous state should be saved in long-term memory and forget gate then decides which part of it should be kept. The output gate combines short-term and long-term memory and determines the information sent to the next state. On the other hand, GRU has no longer a separate memory unit and the forgetting and saving is done simultaneously by the update gate, which makes GRU a bit faster to compute. The reset gate then determines the amount of information to be used from the previous state. Such a setup allows memorizing the long-term dependencies while keeping the gradients reasonable for learning. Exploding gradients can also be an issue, but these are usually solved by simple gradient clipping.

Apart from the cell types, RNNs can be distinguished based on the length of input and output series. These are *One-to-one*, *One-to-many*, *Many-to-one*, *Many-to-many* with the same input and output lengths and *Many-to-many* with different lengths. Additionally, to incorporate both backward and forward information, 2 RNN layers can be stacked on top of each other with reversed sequence order in the latter one. This produces bidirectional layer, commonly used in the majority of RNN architectures.

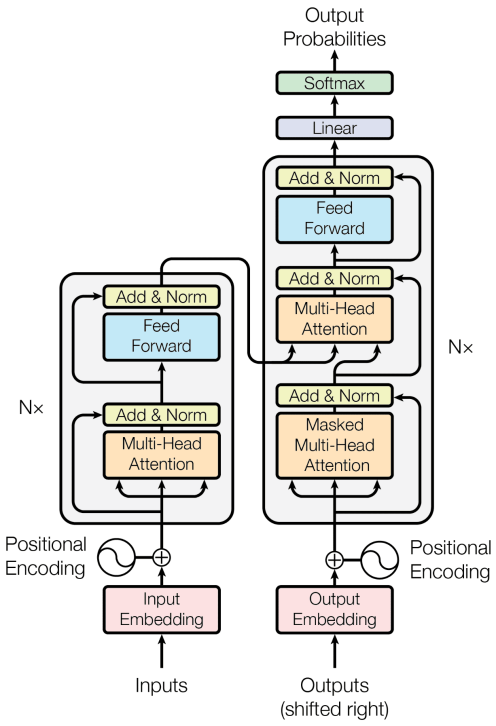
Transformers

Nevertheless, even GRUs are too slow in most big data tasks, because of the sequential dependency. The need to process data in a given order prevents making computation efficient by parallelization. Also, if input sequences are

too long, even the GRU nor LSTM will not be able to remember long-term relationships. This is why the popularity of Transformers has been rising and in many tasks, they are the state-of-the-art approach for sequential data.

Transformers were first described by Vaswani *et al.* (2017), and the proposed architecture is shown in Figure 4.5. It has two important components, i.e. encoder and decoder.

Figure 4.5: Transformer



Source: `https://arxiv.org/abs/1706.03762`

The encoder starts with embedding of the input sequence and representing it by a vector. Embedding has been a widely used technique in how machines can easily understand the meanings of data as similar sequences are expected to be closer to each other in the n-dimensional space. This is further combined with positional encoding that also captures the context of an element depending on its position in a given sequence. A crucial concept in transformers is attention. In essence, its purpose is to learn what inputs should the network focus on and reflect on this by multiplying it with learned importance weights. Self-attention is then the relevance of element in the context series.

The decoder then takes the all golden outputs right-shifted by one, referred to as teacher forcing. Similarly as in an encoder, it is embedded with positional encodings and attention is computed. But now, there is additional attention

that combines both encoder and decoder attentions. Other than that, simple MLPs are added which finalizes the building blocks that can be used N-times in succession.

The positional embeddings and the teacher forcing are the main reason why the encoder and decoder can be trained in parallel, while also learning about the context. During inference, however, the process is autoregressive as outputs need to be passed to the decoder one by one, which results in slower performance. It is a very useful architecture that will be further examined in Subsection 4.4.1 and utilized as a market sentiment model.

4.4 Market Sentiment Analysis

Breaking news have shown a significant impact on stock prices, reflecting the change in the attitude of investors. Some authors also refer to it as a crowd psychology that can result in abrupt price changes. Therefore, it might be desirable to learn from past events and use this knowledge to our advantage in future decisions. However, keeping track of all events that are happening around the globe is time-consuming and the reaction has to be quick. This is why there have been many efforts to automate the collection process of relevant information and consequently evaluate its meaning.

Monitoring textual sources is not that difficult task to address nowadays, even in the context of higher volumes of data. Online services often provide APIs[22] to connect all your applications or one could simply scrape the websites, even though the latter is not always legal and has to be approached with caution. A more complex problem is to evaluate the collected data and make an assessment of its relevance and polarity.

Some of approaches to tackle this are simple sentiment models, that start with bag-of-words representations without trying to understand the moods and meanings. A famous rule-based sentiment system is the Valence Aware Dictionary and sEntiment Reasoner (VADER) described by Hutto & Gilbert (2014). It is built with a lexicon of weighted words and emoticons based on their semantics. Combining weights from all words in a given article then provides an estimate of overall polarity. Similarly, Loughran & McDonald (2016) also created a financial dictionary based on which they were able to assign “positive” or “uncertain” sentiments.

[22] Application Programming Interface

Nevertheless, ML approaches in the domain of NLP have been found superior for such tasks. One of the first authors who applied DL in this domain were Kraus & Feuerriegel (2017). Their model estimated sentiments by LSTM architecture trained on company disclosures and showed a great performance in terms of stock price movement prediction. There have been many other approaches developed since then, but the state-of-the-art language models are often from a family of Bidirectional Encoder Representations from Transformers (BERTs), which will be discussed later in this chapter. Instead of word counting, the meaning is represented by a vector of embeddings allowing the computer to better understand it and predict sentiment estimates. This results in an ability to extract various semantics such as opinions, polarities, feelings, moods, topics and other. The sentiment itself can have many forms, but in most cases, 3 types are recognized, i.e. positive, negative and neutral.

There are many factors that make it complicated to estimate the sentiment. For example, it might be difficult for a machine to learn about sarcasm, irony or word ambiguities. On the other hand, this could be problematic even for humans. It is also very challenging because of the lack of labelled data, but once an appropriate method for a particular domain is chosen, it has been shown that computers can approach human-level understanding and in a few domains even overcome it.

4.4.1 FinBERT

This section is closely connected with Section 4.3.2 and its direct continuation of Transformer architectures. But before proceeding to Financial BERT (FinBERT), its predecessor should be examined first.

BERT-like models have gained in popularity as a go-to approach for pre-training word embeddings in all kinds of domains. It consists of several sets of Transformer encoders which are also bidirectional. The main strength lies in its training procedure which focuses on 2 distinct objectives. In the first one, "language masking", a fraction of input words is masked and the model tries to predict the missing gaps based on context. In the latter one, "next sentence prediction", it then predicts whether the second sentence follows the first one. In this fashion, the model is able to efficiently understand language by creating the embedding vectors that can be further used in any NLP task. Also, thanks to a special token representing whole sentences, it is able to handle text classification problems.

In general, financial sentiments are, however, very different from general ones as they are closely connected with financial markets and their behaviour. The FinBERT was published by Araci (2019) and driven by the importance of using specialized financial textual sources. The authors mentioned the lack of labelled data for the financial domain, which was the reason why it was inevitable to use a general pre-trained language model first. To improve the performance, it was then fine-tuned on financial news articles from two sentiment datasets of Reuter and Financial PhraseBank. Authors provided both text classification and regression variants, but for purposes of sentiment analysis, only the former is required. Furthermore, to avoid so-called "catastrophic forgetting" of understanding from the pre-trained model, several techniques of adjusting the learning rate were proposed. The model's potential in using sentiments for predicting stock market direction and volatility as crucial information for decision-making was also pointed out.

FinBERT achieved substantial improvement in accuracy compared to its counterparts and was very efficient even with small training sets. Sentiment models require a great amount of labelled data, but unfortunately, to our knowledge there has not been a labelled financial sentiment dataset that would be expected to improve the performance of this model. Therefore, it was found superior for the purposes of this thesis and decided to work with the pre-trained FinBERT model without any need for weight adjustments.

4.4.2 TweetEval

In spite of news being expected to be more objective and informative, in sentiment analysis research there has been a tendency to focus more on social media. It is also one of the main research questions of the thesis to compare news and social media. Therefore, appropriate methods need to be employed for both of them and even though FinBERT has outstanding performance in the financial article domain, its application to tweets is unclear.

Concerns about its reliability are based on the fact that Araci (2019) did not explore this area at all. Financial news often use unique words and vague expression, are lengthy and have to be formal with engaging content to avoid losing readers. Meanwhile, user contributions on social media are noisy, much more concise and informal and full of emojis and abbreviations. Most often trying to promote further conversation by being controversial, as anyone is free to react and share their feelings on the topic as well.

Based on this it was decided to add a secondary measurement of sentiment for Twitter. A useful inspiration can be found in Barbieri *et al.* (2020), whose authors focused solely on the dataset of tweets. The evaluation was done on seven distinct classification tasks, which names are self-explanatory. These include Emoji Recognition, Emoji Prediction, Irony Detection, Hate Speech Detection, Offensive Language Identification, Stance Detection and most importantly Sentiment Analysis. Again, the authors stressed the importance of using a general pre-trained BERT language model prior to applying it to tweets.

Despite the limited contextual information that is very common for social media, the authors of TweetEval were able to achieve satisfactory results. For this reason and the fact that no better-labelled tweet dataset was available to us, similarly as before no fine-tuning will be performed in this work.

4.4.3 Sentiment Features

Using data sources described in Section 3.2 and by utilizing BERT and TweetEval, it is finally possible to provide the model with features of sentiment. The main objective is to compare news and media features and their contribution to trading. Therefore, 6 sets of features for news domain and 4 sets for Twitter are introduced, all of which are depicted in Table 4.1.

First of all, simple descriptive statistics for each domain will be examined to see if the sentiment is indeed useful or if it is just the volume of information that has some predictive power on the prices. These are *news-set-1* and *twitter-set-1*.

Furthermore, the language models are applied to extract sentiment predictions for each textual source and since the trading bot will operate on a daily basis, aggregation is required. This starts by centring sentiment scores from each of the predictions as follows:

$$sent_i^{centered} = 1 \times sent_i^{pos} - 1 \times sent_i^{neg} + 0 \times sent_i^{neu}$$

In this manner, 3 scores are reduced to a single value from a range of $(-1, 1)$ that still retains information about pseudo-certainty of the prediction. To perform a daily aggregation, all individual predictions are averaged:

$$sent^{daily} = \frac{1}{n} \sum_{i=1}^{n} sent_i^{centered}$$

Table 4.1: Sets of Sentiment Features

Domain	Set name	Description	Sentiment Features
None	core-set		no sentiment features
News	news-set-1	volumes	news volume opinion volume
	news-set-2	news titles	1-day average sentiment of news titles 5-days average sentiment of news titles 21-days average sentiment of news titles
	news-set-3	opinion titles	1-day average sentiment of opinion titles 5-days average sentiment of opinion titles 21-days average sentiment of opinion titles
	news-set-4	news articles	1-day average sentiment of news articles 5-days average sentiment of news articles 21-days average sentiment of news articles
	news-set-5	opinion articles	1-day average sentiment of opinion articles 5-days average sentiment of opinion articles 21-days average sentiment of opinion articles
	news-set-6	full set	combines all news sets
Twitter	twitter-set-1	volumes	cashtag tweet volume cashtag retweet volume keyword tweet volume keyword retweet volume
	twitter-set-2	cashtag tweets	1-day weighted average sentiment of cashtag tweets 5-days weighted average sentiment of cashtag tweets 21-days weighted average sentiment of cashtag tweets
	twitter-set-3	keyword tweets	1-day weighted average sentiment of keyword tweets 5-days weighted average sentiment of keyword tweets 21-days weighted average sentiment of keyword tweets
	twitter-set-4	full set	combines all Twitter sets
News + Twitter	full-set	full set	combines all news and Twitter sets

Source: Author's calculations

Additionally, to reflect the audience reach of each tweet, daily sentiment is weighted by the number of retweets:

$$sent^{daily\,weighted} = \frac{\sum_{i=1}^{n}(n_i^{retweets}+1) \times sent_i^{centered}}{\sum_{i=1}^{n}(n_i^{retweets}+1)}$$

Inspired by Altuner & Kilimci (2021), 1-day, 5-days and 21-days averages of daily sentiments are added, which will be examined by *news-set-{2,3,4,5}* and *twitter-set-{2,3}*. All of the features above with respect to their domain will be combined in *news-set-6* and *twitter-set-4*. Last but not least, the news and Twitter domains are interconnected in *full-set*, which might carry the ultimate information for trading decisions, yet at high risk of over-fitting.

4.5 Reinforcement Learning

RL is one of the crucial areas of research of ML, that has found use cases in many different fields. These include self-driving cars, advertisement recommendation systems, chatbots, beating human players in games or, most relevant for this thesis, stock trading. The inspiration for most of these approaches can be found in the human and animal world, in which trial-and-error is the most important learning principle. A similar concept is also applied in ABM.

RL provide a compelling alternative to supervised and unsupervised learning. In supervised learning, the scientist has to teach the model what is the desired output. In practice, this is what golden labels are used for. Unsupervised learning is about finding patterns in data such as clusters or associations without any target. Meanwhile, RL has its greatest power in generating new data points itself, i.e. experience. Because of this, not only the cumbersome process of labelling data is no longer necessary, but there is also a potential to achieve a better-than-human level of performance.

The major challenge is to set up these algorithms to be able to learn. There are many techniques that address this issue differently, but they share the same property of a great number of hyper-parameters. A slight change can often result in the difference between learning and not learning at all, making it computationally demanding to find a feasible setup. One could apply rule-of-thumb or replicate other research, but in the end, they are very domain and dataset-specific and one has to explore at least some of the combinations, which might be the biggest weakness of RL.

The vast majority of RL algorithms focuses on finding optimal policies, that

will result in the largest cumulated reward. The reward can be a goal position when navigating through a maze, or profits in the context of stock trading. The challenge is how to estimate the expected cumulated reward as one can only observe the immediate ones by interacting with the environment. It is computationally impossible to evaluate all possibilities even in finite space, not even considering continuous spaces with an infinite number of states. Therefore, RL algorithms mainly focus on efficient exploration of these possible scenarios in a way, that it is not necessary to find the policy yielding the highest reward, but on average, it should be very close to that. This could also be seen as some kind of a heuristic in the context of behavioural economics - a solution that is satisfactory rather than maximizing.

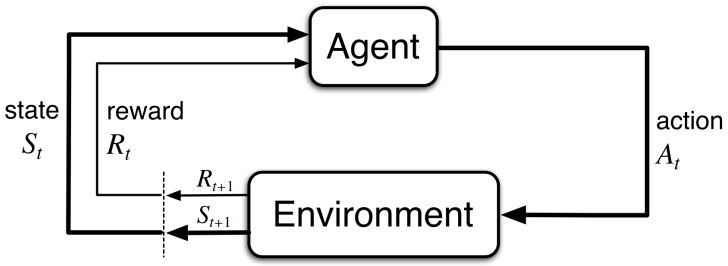
A detailed overview of RL algorithms is provided by Sutton & Barto (2018), which will be the main literature source for this chapter. The following sections will start by describing basic concepts of RL in finite domain. Later, more complex approaches based on NN that fit the domain of infinite-state spaces, such as the stock markets, will be presented. The focus will be given to the methodology that is the most relevant for trading agent implementation.

4.5.1 Markov Decision Process

Markov Decision Process (MDP) is a basic framework that can solve many RL tasks. It is based on so-called *Markov property*, which assumes the conditional probability of future states to be dependent only on the present state, without any dependency on past states. In other words, the past has no effect on the future, and the current state fully describes what can happen next. It is a strong premise, but it is required for the formulation of discounted expected reward.

The idea of agent's learning via interactions is shown in Figure 4.6. At each time step t, the agent occurs in state S_t and chooses action A_t, which in turn has an effect on the environment and the agent ends up in new state S_{t+1}. As a result of performing this action, the agent receives an immediate reward R_{t+1}, which will help to evaluate the choice of A_t under S_t. Sets of all possible states and actions will be denoted as $\mathcal{S}$ and $\mathcal{A}$ respectively, both of them are assumed to be finite at this point.

Figure 4.6: Markov Decision Process



Source: Sutton & Barto (2018)

Therefore, to precisely describe MDP, one has to define the following:

$\mathcal{S}$ - Set of all possible states s

$\mathcal{A}(s)$ - Set of available actions based on state s

$p\left(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\right)$ - Probability of ending up in state s' with immediate reward r after performing action a in state s

$\gamma \in [0, 1]$ - Discount factor for expected returns

During the process of training, the agent should learn to maximize the long-run cumulative reward instead of the immediate one. This is expressed by the following formula:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$$

and the goal is $max\left(\mathbb{E}\left[G_0\right]\right)$. The discount factor γ is a parameter that allows adjusting of the weights of the future rewards. The closer to one, the more important the future is and once it is set to 1, all future rewards are considered with the same weight. Usually, the rule-of-thumb choices are close to 1.

MDPs can also be distinguished in episodic and continuous tasks. In the former, the training episode is finished after reaching a finite number of steps or given time, and the following one starts by resetting the environment. In the latter, there is no terminal reward and the task never ends.

The learned behaviour of an agent is called policy and will be denoted π. It can also be interpreted as a distribution of actions in a particular state, i.e. $\pi(a \mid s)$. To evaluate the quality of the state under a given policy, the value function $v_\pi(s)$ is used:

$$v_\pi(s) = \mathbb{E}_\pi\left[G_t \mid S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

Quality in this context is expressed in terms of expected discounted return, which assumes the agent to start in state s and follow the policy π for the whole episode. It can be rewritten in recursive form, in which case it is referred to as *Bellman equation*:

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s',r} p\left(s', r \mid s, a\right) \left[r + \gamma v_\pi\left(s'\right)\right]$$

This allows expressing the value of a state in terms of the value of all following states weighted by the transition probabilities. In some cases, it might also be more appropriate to work with state-action pairs. For this purpose, the action-value function is defined as:

$$q_\pi(s,a) = \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

which represents discounted expected return of choosing action a in state s.

The task of RL is to find an optimal policy π_* that will maximize expected reward. Let the optimal value and optimal action-value functions be defined as:

$$v_*(s) = \max_\pi v_\pi(s) \quad \text{and} \quad q_*(s,a) = \max_\pi q_\pi(s,a)$$

The policy $\pi_*(s)$ is then optimal if $v_{\pi_*} = v_*$, which can be rewritten as:

$$\pi_*(s) = \arg\max_a q_*(s,a) = \arg\max_a \mathbb{E}\left[R_{t+1} + \gamma v_*\left(S_{t+1}\right) \mid S_t = s, A_t = a\right]$$

There might also be multiple actions maximizing $q_*(s,a)$, and in such case, π_* can choose randomly.

Similarly, it is also useful to compute the recursive form of the optimal value function, which is known as *Bellman optimality equation*:

$$v_*(s) = \max_a \sum_{s',r} p\left(s', r \mid s, a\right) \left[r + \gamma v_*\left(s'\right)\right]$$

Recursion holds because of the Markov assumption and is crucial as it allows $v_\pi(s)$ to be estimated efficiently via dynamic programming.

4.5.2 Temporal-difference Learning

There are many methods of solving MDPs and their appropriateness is usually task-dependent. For purposes of this thesis, Temporal-difference (TD) methods

were selected and will be described in this section. TDs are built on action-value estimates of return from a single iteration of the *Bellman equation*. The main advantage of this, compared to its biggest competitor Monte Carlo Method, is no need to finish the whole episode as the TDs are computed incrementally:

$$v\left(S_t\right) \leftarrow v\left(S_t\right) + \alpha\left[R_{t+1} + \gamma v\left(S_{t+1}\right) - v\left(S_t\right)\right]$$

where α stands for learning rate and $R_{t+1} + \gamma v\left(S_{t+1}\right)$ is an approximation of G_t. Thanks to this immediate learning, update of the value function is calculated from the existing estimate and immediate reward, which can be best utilized in environments characterized by very long-lasting or even never-ending episodes. The disadvantage of such an approach is, however, greater noise and instability.

Additionally, TDs are model-free, meaning there is no need to define a model of environment for inference. In other words, the policy is evaluated by trial-and-error approach, without planning what would happen if action was taken.

Q-learning

Q-learning is an example of a TD method, which was first described by Watkins (1989). The algorithm itself is shown in Figure 4.7.

Figure 4.7: Q-learning algorithm

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Source: Sutton & Barto (2018)

It starts by choosing a learning rate and initializing all action-value pairs with some value except the terminal state, which has to be 0 as it makes no sense to continue. For every episode, after an initial state is set, the looping through each step begins. This means that policy determines which action A to be selected based on current state S. If the algorithm would always

choose an action with the highest estimated Q, it would get stuck around options it has already tried. Therefore, a policy that balances the exploration vs exploitation has to be incorporated. In most cases, the ϵ-greedy method is used, which simply chooses a greedy action with a probability of $(1 - \epsilon)$ and with a probability of ϵ a non-greedy action. As more of the action-values get explored, the ϵ typically decays to refine the approximations. Finally, after taking the action, a new state and reward are observed and used to update the action-value estimates. This repeats until the episode ends.

The algorithm belongs to a family of off-policy methods. This is because the policy is used only for generating episodes and not for a target of the value function. Some authors refer to these as *behaviour* and *target* policies, and the main advantage of Q-learning is, that it allows the former to behave exploratory, while the latter can still follow the learned optimal strategy. As a result, off-policy methods tend to be slower compared to on-policy.

4.5.3 Deep Reinforcement Learning

Until now, the environment was characterized by a finite number of states and actions. But handling large state spaces may be problematic even in the context of great computational power and high memory. This obviously produces many limitations when applied in practice. It is also inefficient that action-state pairs cannot share information with each other and it is needed to visit all of them at least once.

This is where DL helps RL significantly. Action-value and value functions can be approximated by a deep NN, allowing to capture non-linear relationships and even extract features from states that were not part of training data. This is thanks to NN generalization that provides a vector of latent variables defining the state based on a given set of input features. Vectors that are close to each other in the n-dimensional space are then expected to represent a similar state. It is analogous to embeddings in NLP that represent sentences, but now, it will be the abstraction of the state that is expected to help build a trading agent by using the representation to estimate the action-value.

Deep Q-Learning

DQN is a more sophisticated version of the Q-network presented in Section 4.5.2. Similarly, it is based on TD methods, but the action-value space is now approximated by NN. Even though it might sound straightforward, many challenges

arise. The most notable one is, that the learning is very unstable by default and for a very long time, nobody was able to tackle this issue.

Mnih *et al.* (2013) was a group of researchers who showed a lot of success by applying DQN to Atari games. In their task, RL agent learned to play a set of 2600 games, without any prior game-specific knowledge, but only by seeing a sequence of images and a final reward. Thanks to a few adjustments, DQN was then able to reach human-level results and even overcome them in several games.

The first very important adjustment that significantly increased stability during training was *experienced replay*. The idea is to keep already generated episodes in a buffer, a short memory, that includes the quadruples of state, action, reward and new state. It has a limited size and during training, it is used to uniformly sample batches of experiences from it. This completely reduces the correlation of samples as it is improbable that all will come from the same sequence and makes sure the data is independent. Additionally, since the buffer size is greater than the batch size, it allows past experiences to be sampled multiple times. If then, a very different episode is added to a buffer, there is a much lower chance the NN will forget all of past relationships and experience a catastrophic forgetting.

The second improvement is the *target network freezing*. For this, one NN is used to estimate the value function and another is the target NN. Weights of the former are updated during the training and copied once in a while to the latter one, freezing it for a certain period. This avoids the danger of changing the value of the very next state that is required to update the current state. Training on a batch of experiences with frozen target NN has shown to be more stable and robust and is again connected with the catastrophic forgetting.

As a third one, the authors also employed *reward clipping*, which is, however, not very relevant as it mainly addressed the great differences in the scale of rewards among games.

The DQN algorithm is described in a detail in Figure 4.8. It starts with initialization of the buffer for *experienced replay* and two networks for the *target network freezing*. In each episode, the initial state is set and ϵ-greedy is used as a behaviour policy. If greedy action is to be selected, the action-value NN for all possible actions is evaluated. The best action is then executed and the new state and reward are observed and stored in the memory. To update the weights of action-value NN, a mini-batch is sampled from the buffer and by combining it with the target network, the gradient descent is performed as

described in the figure. Finally, if a given number of iterations have already passed, the weights from action-value NN are directly transferred to the target NN and if the state was not terminal, it proceeds to the next step.

Figure 4.8: Deep Q-learning Algorithm

Algorithm 1: deep Q-learning with experience replay.
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

Source: Mnih *et al.* (2015)

Double Deep Q-Learning

DQN inspired many other authors, such as Hessel *et al.* (2018) who described a whole set of extensions. One of them was the Double Deep Q-Network (DDQN), which focuses on the issue of action-value being over-estimated in the early stages of training.

This concept was known already prior to incorporating NN into Q-learning and originally there had to be two action-value tables. During each step of training, it was decided randomly which one of them will be updated, in a way that action it considers to be optimal is taken but the evaluation is based on the other Q table. This independency then mitigates the problem of wrong actions having high action-values.

In DDQN, however, there is no need to introduce a new NN as there are two already, i.e. action-value function Q and target action-value $\hat{Q}$. Using these, the y_j in the loss computation is modified as follows:

$$y_j = \begin{cases} r_j & \text{if episode terminates at step j} + 1 \\ r_j + \gamma \hat{Q}\left(\phi_{j+1}, \underset{a'}{\text{argmax}} Q\left(\phi_{j+1}, a', \theta\right), \theta^-\right) & \text{otherwise} \end{cases}$$

This will be the final RL algorithm to be used for the trading agent. It might be reasonable to explore other variants of DQN as well, but author of this thesis considers potential improvements to be very marginal.

4.6 Stock Trading

Term *stock trading* is usually used in the context of short-term while *investing* is more longer-term. Ideally, in the case of the trading agent, this should intersect and the bot should be able to learn both at the same time. This is why DDQN is perfect for such tasks as the algorithm can learn both of these relationships thanks to the bellman equation. In other words, the evaluation is based on discounted accumulated reward (long-term), but at the same time, the algorithm can immediately react to changes in states (short-term).

There are, however, still a few details that need to be addressed before finalizing the trading bot. Also, some benchmark strategies need to be introduced for evaluation. Both of these will be discussed in the following sections.

4.6.1 Trading Agent Setup

To train the agent via DDQN, one has to start by defining what actions can the agent take. This will be in line with the vast majority of literature that employs 3 possibilities: buy, sell or hold. Combining this with the current position, the agent then transfers to a new position by making n number of transactions and paying transaction costs for each of them. This is described in Table 4.2. Also, based on similar research, it was found reasonable to set the costs to 0.1% of the stock's price.

The interpretation of the learned behaviour is straightforward. *Flat* position suggests that the algorithm considers it to be too risky or is unsure to make any decision, therefore it is best to hold cash. On *long* position, the agent expects

appreciation in future, while being *short* is interpreted as pessimistic and value is expected to drop.

Table 4.2: Effect of Actions on Positions

Current position	**&**	**Action**	**→**	**New position**	**&**	**Number of trades**
Long	&	Buy	→	No change	&	0
Long	&	Sell	→	Short	&	2
Long	&	Hold	→	Flat	&	1
Short	&	Buy	→	Long	&	2
Short	&	Sell	→	No change	&	0
Short	&	Hold	→	Flat	&	1
Flat	&	Buy	→	Long	&	1
Flat	&	Sell	→	Short	&	1
Flat	&	Hold	→	No change	&	0

The trade action will be decided on every trading day right before midnight so that the agent can collect as much relevant information as possible. It might be more appropriate to collect data from the textual sources up until the opening hour, but historically, the publish times of news are very difficult to extract.

States will be approximated from a given set of features, representing the information available for making a decision. These will include 10 core features as described in Section 3.1, and other sentiment-based features depending on the particular setup (see Table 4.1). The main objective is then to compare different setups to see which features are the most useful.

Last but not least, the reward will be in a form of market returns and the importance of future rewards, i.e. the γ parameter, will be empirically tested along with other hyper-parameters.

4.6.2 Evaluating Strategies

There are many strategies that could be used as a benchmark for the trading agent. They are usually based on technical indicators, however, it is difficult to robustly outperform Buy-and-Hold or Dollar-cost Averaging (DCA) strategies (see for example Lohpetch & Corne (2009)). Therefore, only approaches that do not require any adjusting or training will be used as a comparison for the proposed model, even though they are closer to investing rather than trading. Furthermore, for evaluation purposes, it is important to combine the individual stocks into a portfolio. The reasoning and other description will be given in the following sections.

Metrics

To evaluate the performance, two alternative metrics will be used, i.e. Net Asset Value and Sharpe Ratio. The first is a simple indicator of accumulated return adjusted for transaction costs and will be computed for each day. The second will represent the portfolio's risk-adjusted value over the whole trading period.

Returns

$$Returns_i = \begin{cases} \frac{Close_i - Close_{i-1}}{Close_{i-1}} & \text{if long} \\ \frac{Close_{i-1} - Close_i}{Close_i} & \text{if short} \\ 0 & \text{if flat} \end{cases}$$

Net Asset Value

$$NAV_i = NAV_{i-1} \times (1 + Returns_i - Costs_i)$$

Sharpe Ratio

$$Sharpe = \frac{Returns^{portfolio} - Returns^{risk-free}}{\sigma^{portfolio}}$$

$$Sharpe^{annualized} = \sqrt{252} \times Sharpe$$

Portfolio Initialization

The agent will be able to predict an action for each stock. But since there are 11 stocks in total, it would be very complicated and noisy to evaluate the hypotheses by inspecting all of the results. The solution is to average them by building a portfolio and comparing it with a suitable benchmark. Therefore, there will be no active portfolio weight optimization, and will only happen at the beginning.

Equally Weighted Portfolio will consider all of the stocks used in training and weigh them equally. This should essentially be equal to average of Buy-and-Hold for individual stocks.

Minimum Variance Portfolio is similar to the previous one, but the weights are now selected such that the overall variance of the portfolio is minimized.

Portfolio Management

Stocks in a portfolio will be either actively or passively managed. In the former, DDQN agent will trade individual stocks, while in the latter, only initial or equally distributed contributions will be given. The strategies used are described in more detail below.

Buy-and-Hold is a simple yet powerful strategy in which stocks are purchased and held over the whole period. It is a vital comparison for every trading algorithm.

DCA is about distributing the risk over time. This is done by splitting the initial investment equally over the whole period into regular, usually monthly, contributions.

Daily Trading will be performed solely by the DDQN agent as described in Subsection 4.6.1.

Timed DCA is an original idea of potentially outperforming a standard DCA. Instead of investing in the first day of the month, the day will be selected based on DDQN agent's first decision to go long in each month.

This finalizes the theory behind the process of setting up and evaluating the trading algorithm. More detailed specifications will be provided in the following Chapter 5, when inspecting the results.

Chapter 5

Results

This chapter starts with a brief exploratory analysis of the estimated sentiments and other features defining the state of the stock market. This will then be crucial for training, evaluation and prediction of the trading agent. Also, the strategies will be evaluated in the context of portfolio and the proposed hypotheses will be discussed.

5.1 Sentiment Predictions

Despite the great amount of data from social media, it was not very time consuming to apply the already pre-trained NLP models for inference and thanks to GPU, this took no longer than a day of run time. It is also important to note, that if a particular ticker had no textual source on a given day, the sentiment was assumed to be 0 to avoid losing a lot of incomplete observations.

There is no point in inspecting the day-by-day dynamics of the sentiments as they fluctuate significantly and the graphs are confusing. It is, however, interesting to compare the correlation of sentiment estimates, which are shown in Table 5.1. TweetEval and FinBERT applied on tweets appear to be correlated both in *cashtag* and *keyword* datasets. It is therefore not expected to be useful to add both of them and since TweetEval is more appropriate, it will also be the only one used further.

Table 5.1: Sentiment Correlations

	Keywords (TweetEval)	Keywords (FinBERT)	Cashtags (TweetEval)	Cashtags (FinBERT)	News (content)	News (titles)	Opinion (content)
Keywords (FinBERT)	**0.610**						
Cashtags (TweetEval)	0.120	0.071					
Cashtags (FinBERT)	0.052	0.092	**0.536**				
News (content)	0.063	0.080	0.111	0.124			
News (titles)	0.057	0.076	0.074	0.096	**0.555**		
Opinion (content)	-0.023	0.004	0.097	0.101	0.133	0.081	
Opinion (titles)	-0.022	0.008	0.048	0.052	0.084	0.077	**0.359**

All values correspond to Pearson's correlation.
Source: Author's calculations.

Another finding is, that the *News titles* and *News content* have also quite a high correlation. This is also the case of *Opinion titles* and *Opinion content*, suggesting FinBERT's relevance in the context of long and short textual sources. It is, however, questionable whether it is worth including both of them and will be interesting to see which one is more helpful. On the other hand, it is hard to find any other correlations. This means that all datasets, namely *Keywords*, *Cashtags*, *News* and *Opinions*, have no significant mutual correlation and all are expected to provide unique information for the agent decision.

The correlation of sentiments and other features with respect to future returns is depicted in Table B.1. It is a very simplistic interpretation without any lagged variables but can provide some insights. Most notably, it is very difficult to find any correlation at all. This is in line with EMH or AMH as there is no significant predictor of future returns, not even among the sentiments. Additionally, $\$GME$ appears to have the highest positive correlation with tweets and negative with news, but the significance of this is questionable.

5.2 DDQN Trading Agent

This section describes the process of training a trading agent along with evaluation in the context of single stocks as well as portfolios. The implementation of the DDQN algorithm was done in Python with inspiration from a useful *ML for Trading* repository[1]. The framework provided a good building foundation for the trading environment, however, it was simplified and required a great number of adjustments to suit the needs of this thesis.

5.2.1 Hyper-parameter Setup

As was already discussed, hyper-parameters need to be set prior to training. To find an optimal set, one has to iteratively explore the space of possible options. In this thesis, a group of 13 parameters was examined, which are described in Table 5.2.

Table 5.2: DDQN Hyper-parameters

Hyper-parameter	Value	Description
max_episodes	1000	Maximum number of episodes after which training is terminated
gamma	0.99	Long-run discount rate from Bellman equation
architecture	MLP(512, 256)	MLP architecture of Q-network
learning_rate	0.0001	Initial learning rate for Adam
l2_reg	1.00e-6	L2 regularization strength
replay_capacity	1000000	Capacity for experience replay
tau	1000	Frequency for target network freezing
batch_size	500	Number of experiences taken in each update
epsilon_start	1	Initial value of exploration parameter
epsilon_end	0.1	Final value of exploration parameter
epsilon_decay_steps	200	Number of steps to reach the final exploration parameter
train_trading_days	252	Number of trading days randomly sampled during each episode

The initial choice was based on related literature, most notably Bajpai (2021) or Kim *et al.* (2022). However, from several experiments, it was clear that the choice is very sample dependent and even the same setups can result in different strategies based on randomness. This was further complicated by the fact, that there are 132 models[2] in total to be trained, making it inappropriate to evaluate them on each set of hyper-parameters. Therefore, only *core-set* was selected for the search, and during training, the exact same setup that was found to be best-performing was then used in all feature sets.

The rule of thumb was to make significant adjustments to parameters that are believed to have a great influence on the learning process. At the same

[1] `https://github.com/stefan-jansen/machine-learning-for-trading`
[2] 12 feature sets for each of 11 tickers

time, it was important to make the learning framework as robust as possible and resilient to changes in the random seed. It is also important to keep in mind, that even 13 parameters can lead to enormously large parameter spaces. For instance, if we would assume 10 possibilities for each of them and each setup to be computed in 1s, this would result in more than 300, 000 years of run time. Therefore, it was crucial to efficiently find a good-enough configuration rather than the best one.

Starting with the *max_episodes*, the idea was not to train for too long. After this period the model usually over-fitted a lot and it was unlikely to be able to generalize better if the training was extended. Another important hyper-parameter *train_trading_days*, had a great effect on generalization since each episode considered only a random 252-day subset of training data. There were also some experiments with shorter periods, but this appeared to be the most stable. The *gamma* was explored only partially as values close to 1 are likely to be the best for modelling long-term investments.

Author of this thesis also examined different Q-network *architectures*, but most of them had significant learning issues. Namely, the GRU that considered very similar sets as described in Table 4.1 with an addition of 5 lags of each variable, was much slower than a simple MLP and the agent usually learned to repeat the same action under any scenario. Therefore, a simple 2-hidden layer MLP with a proper number of neurons was selected.

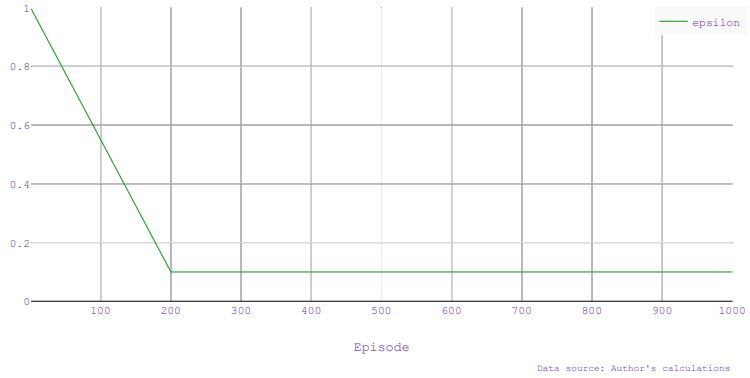
The *learning_rate* was set to default since Adam behaves reliably in most settings, the only addition was a cosine decay of the rate over the whole learning period to mitigate the possibility of diverging from already found optimums. The *capacity* and the *tau* were also explored, but it was decided that it is a good idea to remember all of the experience and to update roughly once every 4 episodes. It is quite similar to the original DDQN paper[3]. Exploration rate *epsilon* was set high in the early stages and remained reasonably high for the rest of the training, this is depicted in Figure 5.1. Last but not least, the selection of *batch_size* was run by intuition not to over-fit too quickly but still learn fast enough.

5.2.2 Training Process

Once hyper-parameters were set, the training of all setups was performed. The stock market has proved to be very easy to over-fit and the hardest task to tackle

[3]Mnih *et al.* (2013)

Figure 5.1: Exploration Parameter



was the balance between sufficient fit on training data and generalization on development. To automate this process, a special strategy of early stopping was employed. The rule was to train on a maximum of 1000 episodes and terminate the process once the model showed sufficient generalization capability on the development set. In essence, this meant that the agent outperformed Buy-and-Hold by at least 10%. Additionally, since the calculations are relatively slow, the evaluation was done once every 25 episodes.

The training algorithm as described in Section 4.5.3 was applied on all 11 tickers each with 12 feature sets, resulting in 132 models in total. To demonstrate the training process in more detail, a single model from *core-set* was randomly selected. The Q-network's training loss for NVDA is depicted on Figure 5.2. There is no need for the loss to diminish because the Q-value is abstract and unsupervised and its only purpose is to relatively compare different actions. It is usually sufficient if the loss converges to some value, nevertheless, a gradual decrease was observed in most of the trained networks.

Since each episode considered a random subset of 252 trading days, there is a significant variance in returns in every training episode. This is shown in Figure 5.3, which compares the absolute difference between Buy-and-Hold and Daily Trading returns. The overall trend is upward sloping, suggesting the model ends up fitting, potentially over-fitting, on training data. The process appears to be quite slow and if early stopping was applied, it may never reach such a state at all. Nevertheless, it is not necessary for generalization as was already seen in experiments with fewer training episodes.

The performance of all models is shown in Table B.2 along with the number of training episodes in parentheses. The training of each model took 630

Figure 5.2: Training Loss Example

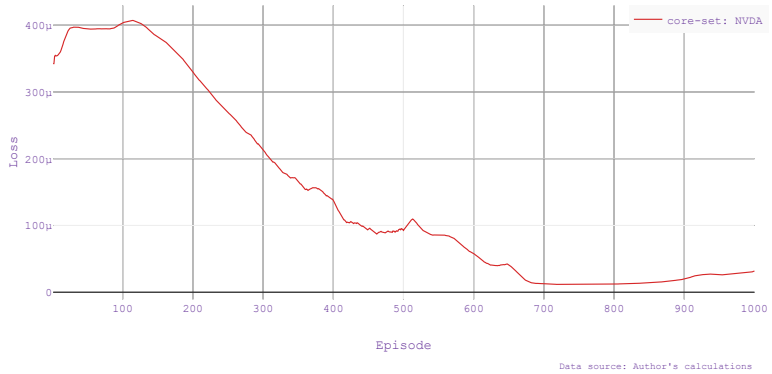
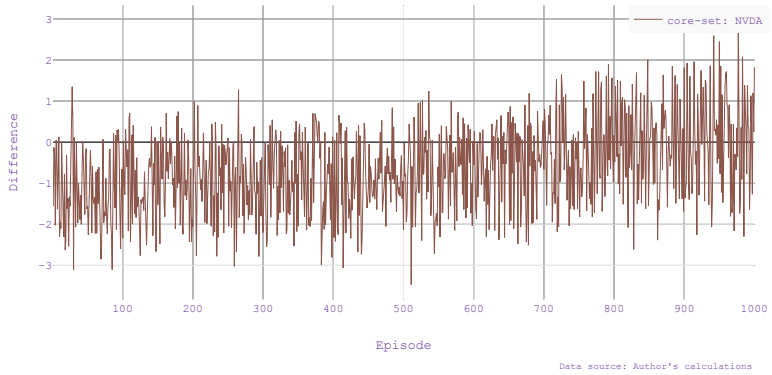


Figure 5.3: Training Example: Buy-and-Hold versus Daily Trading



episodes on average and the total runtime of roughly 18 days was distributed to 3 devices[4]. As expected, models trained on a full number of episodes tend to over-fit, which can be seen especially in cases of TSLA and GME. On the other hand, the great amount of features in *full-set* appears to make it very difficult to do the same. Nevertheless, this is far from actual performance on unseen data and will be interesting to see if these over-trained models will perform significantly worse on the test set.

The list of all final net asset values on the development set is available in Table 5.3. It is interesting that trading of GOOGL, BRK-A and JPM was able to outperform Buy-and-Hold in the majority of setups, while AMZN had only 3 that were terminated before reaching 1000 episodes. It is hard to make any conclusions from this but will be interesting to compare the correlation of these results with test data.

[4]2x Google Colab Pro, 1x Desktop with GeForce RTX 3060 Ti and AMD Ryzen 5 3600

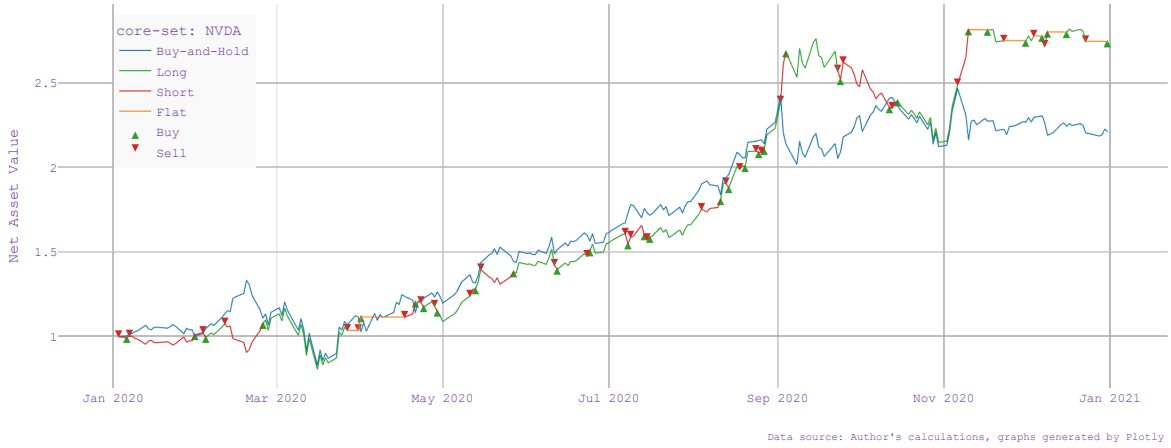
Table 5.3: Development Data: Stock's Final Net Asset Value

	AAPL	MSFT	GOOGL	AMZN	FB	TSLA	BRK-A	TSM	NVDA	JPM	GME
Buy-and-Hold	1.784	1.402	1.287	1.737	1.309	7.964	1.026	1.878	2.212	0.919	3.204
DDQN Agent											
core-set	0.568	1.402	**1.524**	1.115	**1.520**	**8.637**	**1.483**	1.212	**2.744**	**1.031**	0.047
news-set-1	1.003	0.550	**1.465**	1.621	0.764	4.491	**1.427**	**2.183**	**3.304**	**1.073**	0.294
news-set-2	**2.031**	**1.905**	**1.743**	**2.063**	0.836	1.107	**1.185**	**2.089**	**2.624**	**1.085**	**7.143**
news-set-3	**2.463**	0.750	**1.466**	**1.928**	**1.506**	1.790	**1.421**	**2.189**	**2.578**	**1.012**	**5.415**
news-set-4	0.870	0.407	**1.503**	1.429	**1.486**	**11.146**	**1.282**	0.432	2.123	**1.013**	**3.913**
news-set-5	0.782	**1.602**	**1.480**	1.425	**1.540**	1.959	**1.177**	1.767	0.745	**1.345**	0.804
news-set-6	1.784	**1.813**	**1.680**	**2.349**	**1.551**	0.761	**1.195**	**2.127**	**3.547**	**1.012**	0.430
twitter-set-1	**2.168**	**1.656**	**1.751**	0.986	**1.675**	0.055	0.580	1.101	0.322	**1.077**	0.118
twitter-set-2	0.462	0.816	**1.439**	1.525	1.309	**10.240**	**1.172**	1.878	**2.665**	**1.085**	**3.850**
twitter-set-3	**2.071**	0.864	0.688	0.666	1.207	7.964	**1.267**	1.878	**2.567**	**1.106**	**3.808**
twitter-set-4	**2.007**	**1.545**	**1.736**	0.520	**1.441**	**11.142**	**1.123**	1.000	**2.698**	**1.489**	**3.707**
full-set	0.781	**1.603**	**1.638**	1.228	0.594	0.227	**1.129**	1.558	0.758	**1.088**	**4.960**

Final NAV of agent's trading strategy when applied on development data. Bold values indicate outperformance of Buy-and-Hold.
Source: Author's calculations.

Even though the final net asset value represents a good measure of performance, it only tells a part of the story. In the background, there is a whole trading process that is simulated and may be evaluated in the context of events that occurred. It is, however, confusing to analyse 132 graphs at the same time. Therefore, for better understanding, an example of a sequence of trading decisions is presented in Figure A.2 (train data) and in Figure 5.4 (development data).

Figure 5.4: Development Data: Trading Example



The value of Buy-and-Hold strategy may be compared to daily trading at any given point in time. The green-red-orange line corresponds to a current position, based on the Buy or Sell actions that are taken. In the example on development data, the agent was able to utilize two significant drops and outperform the passive strategy. On training data, the agent decided to make significantly more decisions, which is a typical scenario in any of the models. Surprisingly, higher costs had no effect on reducing the number of transactions and the agent would end up not learning anything at all. The higher *gamma* was also expected to mitigate it, but it did not show any significant changes. It may be questionable whether it is desirable to make the decisions so frequently, but this was not as much of an issue out of the sample.

5.2.3 Out-of-sample Performance

The inference is straightforward, the already trained models are directly applied to data which were not seen during the training nor used to terminate the learning process early. Final net asset values for all stocks and feature sets are provided in Table 5.4[5].

Table 5.4: Test Data: Stock's Final Net Asset Value

	AAPL	MSFT	GOOGL	AMZN	FB	TSLA	BRK-A	TSM	NVDA	JPM	GME
Buy-and-Hold	1.203	1.274	1.312	0.772	0.740	1.185	1.417	0.824	1.384	0.950	7.200
DDQN Agent											
core-set	0.757	1.274	1.174	0.433	0.496	**1.453**	1.294	**0.827**	0.504	0.770	0.000
news-set-1	1.000	0.595	1.129	**0.841**	0.514	0.950	0.716	0.756	0.756	0.647	0.000
news-set-2	1.203	0.787	**1.726**	**0.791**	0.576	**1.797**	0.717	**0.844**	0.322	**0.960**	0.000
news-set-3	0.802	0.843	1.151	0.732	0.647	1.093	1.122	**0.853**	1.384	0.776	0.818
news-set-4	0.818	0.990	**1.313**	0.645	**0.760**	1.140	0.835	**0.870**	1.384	**1.014**	-0.000
news-set-5	0.882	1.135	0.807	0.742	0.535	0.826	1.379	0.731	1.060	**1.104**	0.000
news-set-6	1.203	1.037	1.154	**0.804**	**1.038**	**1.603**	0.685	0.807	1.254	**1.017**	-0.005
twitter-set-1	1.199	0.985	**1.405**	0.619	0.526	0.536	1.023	**0.841**	0.498	0.765	-0.014
twitter-set-2	0.756	0.874	**1.333**	0.546	0.740	1.185	1.185	0.824	0.632	**0.999**	1.079
twitter-set-3	0.832	1.024	**1.328**	0.592	**0.741**	1.185	0.915	0.824	1.298	0.910	0.322
twitter-set-4	0.944	0.857	**1.376**	**0.881**	**0.817**	0.997	1.295	**1.000**	0.821	0.633	7.200
full-set	0.777	1.137	0.894	**0.864**	**0.984**	**2.265**	0.908	0.789	1.053	**1.647**	**8.518**

Final NAV of agent's trading strategy when applied on test data. Bold values indicate outperformance of Buy-and-Hold. Agent's strategies to hold for whole period are underlined.
Source: Author's calculations.

DDQN agent had a difficult time beating a simple passive strategy, especially when trading AAPL, MSTF, BRK-A, NVDA and GME and in most of these cases it ended up failing. Even though in the development set there were

[5]An example of trading behaviour on test set is depicted on Figure A.3

just 2 cases in which agent imitated the Buy-and-Hold strategy, in the test this was much more frequent. These cases were underlined in the table. Trading of GOOGL, AMZN, FB, TSM and JPM turned out to have the most outperforming setups, indicating some form of robustness in generalization. Nevertheless, the increase was in most cases rather marginal. The GME is especially interesting since the algorithm struggled not to lose all of the initial investment in most of the setups apart from *twitter-set-4* and *full-set*, suggesting some degree of predictable dependence on sentiments.

Feature sets that were able to outperform Buy-and-Hold in trading of 4 or more stocks, include *news-set-2*, *news-set-4*, *news-set-6*, *twitter-set-4* and *full-set*. At this point, it is difficult to determine which has the highest potential of added value for trading, but these sets will be of the main focus when evaluating agent's performance in terms of different portfolios.

To further analyse the degree of generalization that was achieved, a very naive comparison of different correlations is provided in Table 5.5. The coefficient is calculated for each ticker separately and is based on different feature sets. The idea is to quantify whether a development set is a good indicator of performance on unseen data, in spite of being used for early stopping. The only reasonable correlation can be seen in AAPL, but it was still not enough to outperform the passive strategy. At the same time, TSM has a negative coefficient which is quite counter-intuitive. This can be seen as another form of evidence of the market's unpredictability and the difficulty to generalize.

Table 5.5: Correlation of Final Net Asset Values

	AAPL	MSFT	GOOGL	AMZN	FB	TSLA	BRK-A	TSM	NVDA	JPM	GME
$corr(dev, test)$	0.526	0.362	0.087	0.224	-0.132	-0.217	-0.020	-0.496	0.065	-0.145	0.338

A simple Pearson's correlation coefficient calculated on different feature sets.
Source: Author's calculations.

5.3 Portfolio Strategies

As was already discussed, the focus is given to trading, not portfolio optimization. The reason why portfolios are considered is to average the individual trading strategies with appropriate weights and provide a more robust evaluation framework for the hypotheses.

The first step was to calculate the weights that were used to split the initial investment. These are shown in Table 5.6. Equal weights are straightforward to

calculate, but minimum variance weights are more complex. These were based solely on the training dataset with an additional lower constraint of 0 to avoid starting in a short position. From the results can be seen, that NVDA turned out to be the only ticker that had no place in the portfolio, while BRK-A's contributed by more than a half.

Table 5.6: Portfolio Weights

Weights	AAPL	MSFT	GOOGL	AMZN	FB	TSLA	BRK-A	TSM	NVDA	JPM	GME
Equal	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091	0.091
Min-Var	0.030	0.000	0.074	0.005	0.073	0.005	0.611	0.139	0.000	0.036	0.026

Minimum variance weights calculated via *PyPortfolioOpt* package.
Source: Author's calculations.

Further split is based on the strategy itself. In total, one passive, one active and three conservative variants of DCA were considered. To be more precise, these are briefly described as follows:

B&H invests all at the start and holds till the end

Trade follows agent's daily decisions of buying and selling

DCA-F splits the initial investment into monthly contributions at the start of each month

DCA-L splits the initial investment into monthly contributions at the end of each month

DCA-T splits the initial investment into monthly contributions on the first day of the month with agent's long position

Results from train and development sets are not particularly interesting as they played no role during the training process. Nevertheless, for completeness, they are available in Table B.3 and Table B.4. Substantially more interesting are the results from the test set shown in Table 5.7. The weighting made it very difficult for the agent to outperform *Buy-and-Hold* and only *full-set* was successful in this task with equal weights. The best social media set turned out to be the *twitter-set-4* as it got very close to the passive strategy in both cases. News media sets were less fortunate, out of which *news-set-5* and *news-set-6* performed the best.

Table 5.7: Test Data: Portfolio's Final Net Asset Value

	Equal weights					Minimum Variance weights				
Feature set	B&H	Trade	DCA-F	DCA-L	DCA-T	B&H	Trade	DCA-F	DCA-L	DCA-T
core-set	1.658	0.817	0.945	0.909	0.909	1.396	1.090	1.035	1.017	1.028
news-set-1	1.658	0.719	0.945	0.909	0.906	1.396	0.727	1.035	1.017	1.016
news-set-2	1.658	0.884	0.945	0.909	0.940	1.396	0.810	1.035	1.017	1.025
news-set-3	1.658	0.929	0.945	0.909	0.910	1.396	1.020	1.035	1.017	1.022
news-set-4	1.658	0.888	0.945	0.909	0.917	1.396	0.855	1.035	1.017	1.017
news-set-5	1.658	0.836	0.945	0.909	0.943	1.396	1.118	1.035	1.017	1.034
news-set-6	1.658	0.963	0.945	0.909	0.928	1.396	0.778	1.035	1.017	1.027
twitter-set-1	1.658	0.762	0.945	0.909	0.939	1.396	0.955	1.035	1.017	1.026
twitter-set-2	1.658	0.923	0.945	0.909	0.947	1.396	1.088	1.035	1.017	1.034
twitter-set-3	1.658	0.906	0.945	0.909	0.942	1.396	0.902	1.035	1.017	1.026
twitter-set-4	1.658	1.529	0.945	0.909	0.945	1.396	1.337	1.035	1.017	1.037
full-set	1.658	1.803	0.945	0.909	0.925	1.396	1.120	1.035	1.017	1.024

Final NAV of agent's portfolio with respect to strategy and initialization when applied on test data.
Source: Author's calculations.

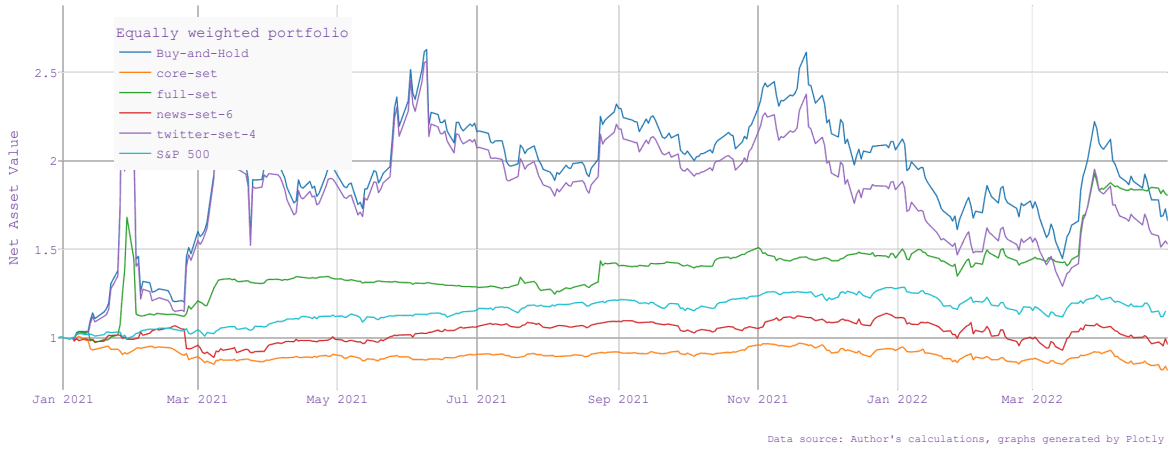
To better understand the dynamics, portfolio values over time from the testing period are shown in Figure 5.5 and Figure A.8[6]. The selection was based on the highest performing set from each group and an index S&P 500 was added as another form of benchmark. By comparing the different initializations, it can be seen that it paid off to distribute investment based on more risky equal weights, which is mainly because of the large share of underperforming BRK-A. Also, it appears, that *twitter-set-4* quite closely imitates the *Buy-and-Hold*, while *full-set* was able to catch up only during the last month.

From DCA results it is clear that the earlier the investment is executed, the higher the profits. This generally holds in the train, development as well as test sets. Timing the decision has been successful only for *twitter-set-2* and *twitter-set-4*, but the improvement is very marginal. For illustration, a similar graph as before is also provided in Figure A.9, however, it is not very informative as the strategies are quite similar.

To adjust the return for a risk, an alternative metric is provided in Table 5.8, in which the risk-free rate was assumed to be 0% for simplicity. The rule of thumb is, that Sharpe Ratio above 1 is usually acceptable, but this of course depends on investor's risk-aversion. It is quite counter-intuitive since minimum variance does not always have a higher ratio compared to equal weights, but this is likely caused by the significantly different nature of train and test sets.

[6]Train and development figures are available in Figure A.4, Figure A.5, Figure A.6 and Figure A.7

Figure 5.5: Test Data: Equally Weighted Portfolio's Value



The only setup that reached the acceptable value was *core-set*, and similarly as before also *twitter-set-4* got close to *Buy-and-Hold*. In other words, there was no substantial difference in risk that the trading has experienced compared to *Buy-and-Hold*.

Table 5.8: Test Data: Annualized Sharpe Ratio

	Equal Weights	Minimum Variance Weights
Buy-and-Hold	0.852	0.823
DDQN Agent		
core-set	-0.865	0.642
news-set-1	-1.517	-2.608
news-set-2	-0.272	-1.227
news-set-3	-0.196	0.185
news-set-4	-0.476	-1.288
news-set-5	-0.736	0.708
news-set-6	-0.018	-1.442
twitter-set-1	-0.364	-0.169
twitter-set-2	0.034	0.469
twitter-set-3	0.274	-0.060
twitter-set-4	0.790	0.738
full-set	1.286	0.589

Sharpe ratio calculated from test data.
Source: Author's calculations.

5.4 Evaluating Hypotheses

The purpose of this section is to summarize what evidence was found in favour or against the hypotheses from the thesis proposal. Most of it was already discussed in Section 5.2 and Section 5.3.

Hypothesis #1: Text analysis plays an important role in stock price prediction and will improve trading profitability.

Referring to test set results in Table 5.7, only *news-set-1* and *twitter-set-1* were not able to surpass the *core-set* in the equally weighted portfolio. In minimum variance this is not so clear, but it appears that the greater number of sentiment features is correlated with stronger generalization. A very similar pattern can also be seen in the case of timed DCA. Furthermore, Table 5.4 shows that this does not necessarily apply to all stocks. The stocks which were the most affected by sentiments include AAPL, AMZN, FB and GME. On the other hand, stocks that had very poor news and social media coverage, such as TSM and BRK-A, did not experience any significant improvements. All of this represents reasonable arguments in favour of not rejecting the hypothesis.

Hypothesis #2: Text analysis based on social media provides more valuable information for trading than news articles.

This is closely connected to the previous hypothesis. Once the improvement can be at least partially confirmed, it is interesting to see which sentiments are generally more important. From the results can be seen, that the *twitter-set-4* managed to outperform news in both of the weighting initializations. Similarly, average portfolio performance of news versus twitter feature sets also suggests slight superiority of social media. When inspecting the results of individual stocks, the highest average improvement was observed in GOOGL and GME. The opposite is quite surprising since TSLA appears to be better explained by news rather than social media. Taking into account all of the above, author of this thesis considers enough evidence to be in favour of the hypothesis. Nevertheless, the combination of both of these sentiments in *full-set* appears to have even higher predictive potential.

Hypothesis #3: Proposed model outperforms minimum variance portfolio as well as S&P500 in terms of returns.

For the final hypothesis, the weakest evidence was found. In terms of trading a minimum variance portfolio as shown in Figure A.8, none of the feature sets were able to surpass the *Buy-and-Hold*. Even though the *twitter-set-4* appeared to be quite promising by outperforming S&P500 as well as showing success in the case of timed DCA, there is a notable problem. It was not clear prior to inference on the test set, that this set should have been selected. While development set with equal weights suggested it, minimum variance had other more appealing candidates. Therefore, it cannot be concluded, that the proposed approach of sentiment-based stock trading in min-var portfolio is robust and will consistently outperform *Buy-and-Hold* or S&P500. In the case of equally weighted portfolio, *full-set* model was able to outperform these strategies, but this is not relevant to the examined hypothesis.

Chapter 6

Discussion

6.1 Limitations

The proposed approach was definitely not perfect. It was difficult to train a model that will robustly generalize as suggested by the low correlation of performance on development and test. Stock markets also proved to be heavily prone to over-fitting and it was challenging to find the optimal learning procedure. Furthermore, the process was often unstable and sometimes even dependent on random initialization. Therefore, finding a reliable and replicable configuration was crucial, but in a lot of cases effortful to achieve.

The low trading costs were also quite questionable. During many experiments, when approaching a more realistic scenario of 1%, the ability of DDQN agent to learn significantly diminished. Author of this thesis believes that the agent's ability to approximate the state does not necessarily imply that the agent also understands in which position it currently is. For example, a short position with buy action would result in $2 \times 1\%$ of transaction costs, while a long position in the same state would have none of these. As a result, this highly complicates the convergence, but no solution that would address this issue was found.

Regardless of many attempts, the present writer also did not manage to efficiently parallelize the training process. The neural networks were part of the problem, but the collection of experiences also played an important role. Succeeding in this area would significantly increase the number of explored hyper-parameter setups during tuning. Providing a much higher volume of alternative models to choose from and increasing the chances of finding a superior configuration.

6.2 Further Improvements

In spite of not being able to find a robust trading strategy that would reliably outperform the mentioned benchmarks, there is a lot more to be explored.

Approximating the Q-function, which should be able to determine the most optimal trading action in all possible states of the market, has proved to be incredibly difficult. This is mainly the result of the unpredictable nature of financial markets. Therefore, the highest potential comes from adjusting the neural network to be able to efficiently generate features and understand the given state. Even though author of this thesis did not further explore the possibilities of RNNs or the more popular Transformers, they are expected to be superior in this domain. The problem is, that there is too many variants to examine and one has to either adopt an architecture from a different field or develop a new one for the stock domain as there has still not been anything as groundbreaking as for example seen in NLP.

It might be also very beneficial to have a pre-training procedure for the Q-network prior to DDQN. A procedure, that based on some supervised metric such as upward or downward future price movement, would generate a latent set of useful features. This is in essence very similar to the pre-trained word embeddings. Consequently, this could significantly quicken the convergence as the Q-network would mainly focus on training the final action classification layer based on these features.

Another interesting improvement would be intra-day trading. In this case, the agent would be able to react to important events even quicker. It is possible to manage the scrapped dataset in almost real-time, allowing to make a decision for instance once an hour. It might be interesting to see if such a short time window provides enough information for a proper decision.

Last but not least, the thesis focused solely on individual stock trading, which composed a portfolio based on different initial weights. But another approach could be active portfolio management. In other words, learning the most optimal allocation of resources in a given state to maximize profits at reasonable level of risk. The biggest disadvantage would be the continuous action space of weight rebalancing, but even for this RL can offer various methods.

Chapter 7

Conclusion

Stock trading is a competitive game, in which it is important to keep up with the current events. Any new information might be vital for further development and it is necessary to filter out the noise. This is where NLP provides a useful tool for aggregating investors' opinions and this process is referred to as sentiment analysis. Even though there are generally two textual sources to draw from, namely social and news media, no general consensus of which one is more valuable has been reached. Additionally, learning from the past is a complex process, that in the case of stock markets often fails to perform under new (unseen) circumstances. DRL offers a unique framework of learning via trial-and-error, which is capable of better reflecting the uncertainty involved in each action. The combination of these state-of-the-art machine learning techniques was expected to mimic a real-world decision process of investors, that could possibly be more consistent than a human-driven one.

The thesis started by questioning the existence of exploitable opportunities in financial markets with the use of two alternative theories - EMH and AMH. It then proceeded to a brief introduction of machine learning concepts, describing the main objective and general idea behind algorithm's learning from past experiences. A special focus was then given to NN architectures since they represent the building blocks both for sentiment analysis and DRL. Furthermore, methods that have been recently developed for the extraction of financial and Twitter sentiments were presented along with basic intuition. The DRL section gave an overview of the evolution of the implemented DDQN algorithm, which has been found suitable for the purposes of stock trading. And finally, the trading environment was defined along with the evaluation framework of the agent-based strategies in comparison to their passive counterparts.

Even though it is not a novel idea to apply the DRL to the stock domain, it is difficult to find a robust trading strategy in the current literature as the results are often inconclusive and very dependent on the selected period. It is hard to distinguish viable methods from the noise, which is likely to be explained by the fact that their publication is quite counter-productive. While a perfect agent outperforming humans in games is very difficult to monetize, a trading bot once trained and proved to be reliable, is straightforward to employ for own profits. And even if the method was proved to be robust on historical data, the employment is likely to result in market adaptation and the possibilities to exploit the imperfections would no longer be valid.

For these reasons, rather than attempting to find an ultimate trading strategy, the main contribution of the thesis was the comparison of the effects of news and social media on the learning process of optimal trading strategy. According to the results, it was found that there is significant evidence for sentiment analysis having a positive effect on learning in both of the cases and the highest improvements were seen in stocks with rich textual sources. Additionally, tweets appeared to be the most useful and more relevant than content or title sentiments extracted from the news. Yet still, the combination of all of the sentiments had the highest performance on the given test set in terms of returns and Sharpe ratio and was able to beat the passive Buy-and-Hold strategy. Nevertheless, there was no clear indication that the given feature setup and portfolio weights should have been selected based on development results and it is, therefore, questionable whether it is a reliable strategy.

To conclude, the DRL research is still in very early stages, especially in the context of stock trading, and there is a lot more to further develop. The highest potential is believed to be hidden in the structure of the NN approximating the value of each decision. It might be reasonable to build a more sophisticated sequential architecture based on recurrent networks, convolutions or transformers, but since there are so many possibilities, it would need to be done very efficiently. It complicates the already unstable learning process of DRL, but techniques such as pre-training on labelled data could potentially be crucial. Nevertheless, compared to other tasks, it always needs to be taken into account that the financial markets are very stochastic in nature and the methods of finding the optimal behaviour have to be fully adapted to that.

Bibliography

Altuner, A. B. & Z. H. Kilimci (2021): "A novel deep reinforcement learning based stock direction prediction using knowledge graph and community aware sentiments." *arXiv preprint arXiv:2107.00931* .

Araci, D. (2019): "Finbert: Financial sentiment analysis with pre-trained language models." *arXiv preprint arXiv:1908.10063* .

Bajpai, S. (2021): "Application of deep reinforcement learning for indian stock trading automation." *arXiv preprint arXiv:2106.16088* .

Barbieri, F., J. Camacho-Collados, L. Neves, & L. Espinosa-Anke (2020): "Tweeteval: Unified benchmark and comparative evaluation for tweet classification." *arXiv preprint arXiv:2010.12421* .

Braun, H. & J. S. Chandler (1987): "Predicting stock market behavior through rule induction: an application of the learning-from-example approach." *Decision Sciences* **18(3)**: pp. 415–429.

Brealey, R. A., S. C. Myers, F. Allen, & P. Mohanty (2012): *Principles of corporate finance*. Tata McGraw-Hill Education.

Carta, S., A. Corriga, A. Ferreira, A. S. Podda, & D. R. Recupero (2021): "A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning." *Applied Intelligence* **51(2)**: pp. 889–905.

De Bondt, W. F. & R. Thaler (1985): "Does the stock market overreact?" *The Journal of finance* **40(3)**: pp. 793–805.

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

Glorot, X. & Y. Bengio (2010): "Understanding the difficulty of training deep feedforward neural networks." In "Proceedings of the thirteenth in-

ternational conference on artificial intelligence and statistics," pp. 249–256. JMLR Workshop and Conference Proceedings.

Goodfellow, I., Y. Bengio, & A. Courville (2016): *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Hessel, M., J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, & D. Silver (2018): "Rainbow: Combining improvements in deep reinforcement learning." In "Thirty-second AAAI conference on artificial intelligence," .

Hochreiter, S. & J. Schmidhuber (1997): "Long short-term memory." *Neural computation* **9(8)**: pp. 1735–1780.

Huberman, G. & T. Regev (2001): "Contagious speculation and a cure for cancer: A nonevent that made stock prices soar." *The Journal of Finance* **56(1)**: pp. 387–396.

Hutto, C. & E. Gilbert (2014): "Vader: A parsimonious rule-based model for sentiment analysis of social media text." In "Proceedings of the international AAAI conference on web and social media," volume 8, pp. 216–225.

Jiao, P., A. Veiga, & A. Walther (2020): "Social media, news media and the stock market." *Journal of Economic Behavior & Organization* **176**: pp. 63–90.

Kahneman, D. & A. Tversky (1979): "Prospect theory: An analysis of decision under risk." *Econometrica* **47(2)**: pp. 263–292.

Kim, S.-H., D.-Y. Park, & K.-H. Lee (2022): "Hybrid deep reinforcement learning for pairs trading." *Applied Sciences* **12(3)**: p. 944.

Kraus, M. & S. Feuerriegel (2017): "Decision support from financial disclosures with deep neural networks and transfer learning." *Decision Support Systems* **104**: pp. 38–48.

Krizhevsky, A., I. Sutskever, & G. E. Hinton (2012): "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* **25**.

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, & L. D. Jackel (1989): "Backpropagation applied to handwritten zip code recognition." *Neural computation* **1(4)**: pp. 541–551.

Lo, A. W. (2004): "The adaptive markets hypothesis." *The Journal of Portfolio Management* **30(5)**: pp. 15–29.

Lohpetch, D. & D. Corne (2009): "Discovering effective technical trading rules with genetic programming: Towards robustly outperforming buy-and-hold." In "2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)," pp. 439–444. IEEE.

Loughran, T. & B. McDonald (2016): "Textual analysis in accounting and finance: A survey." *Journal of Accounting Research* **54(4)**: pp. 1187–1230.

Ma, C., J. Zhang, J. Liu, L. Ji, & F. Gao (2021): "A parallel multi-module deep reinforcement learning algorithm for stock trading." *Neurocomputing* **449**: pp. 290–302.

Mao, H., S. Counts, & J. Bollen (2011): "Predicting financial markets: Comparing survey, news, twitter and search engine data." *arXiv preprint arXiv:1112.1051* .

Mishkin, F. S. & S. G. Eakins (2006): *Financial markets and institutions*. Pearson Education India.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, & M. Riedmiller (2013): "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* .

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.* (2015): "Human-level control through deep reinforcement learning." *nature* **518(7540)**: pp. 529–533.

Nan, A., A. Perumal, & O. R. Zaiane (2020): "Sentiment and knowledge based algorithmic trading with deep reinforcement learning." *arXiv preprint arXiv:2001.09403* .

Samanidou, E., E. Zschischang, D. Stauffer, & T. Lux (2007): "Agent-based models of financial markets." *Reports on Progress in Physics* **70(3)**: p. 409.

Samuelson, P. A. (1965): "Rational theory of warrant pricing." *IMR; Industrial Management Review (pre-1986)* **6(2)**: p. 13.

Sutton, R. S. & A. G. Barto (2018): *Reinforcement learning: An introduction*. MIT press.

Tetlock, P. C. (2007): "Giving content to investor sentiment: The role of media in the stock market." *The Journal of finance* **62(3)**: pp. 1139–1168.

Tversky, A. & D. Kahneman (1981): "The framing of decisions and the psychology of choice." *Science* **211(4481)**: pp. 453–458.

Vanstone, B. J., A. Gepp, & G. Harris (2019): "Do news and sentiment play a role in stock price prediction?" *Applied Intelligence* **49(11)**: pp. 3815–3820.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, & I. Polosukhin (2017): "Attention is all you need." *Advances in neural information processing systems* **30**.

Watkins, C. J. C. H. (1989): "Learning from delayed rewards." .

Wu, X., H. Chen, J. Wang, L. Troiano, V. Loia, & H. Fujita (2020): "Adaptive stock trading strategies with deep reinforcement learning methods." *Information Sciences* **538**: pp. 142–158.

Xiao, C. & W. Chen (2018): "Trading the twitter sentiment with reinforcement learning." *arXiv preprint arXiv:1801.02243* .

Xiong, Z., X.-Y. Liu, S. Zhong, H. Yang, & A. Walid (2018): "Practical deep reinforcement learning approach for stock trading." *arXiv preprint arXiv:1811.07522* .

Yang, S. Y., Y. Yu, & S. Almahdi (2018): "An investor sentiment reward-based trading system using gaussian inverse reinforcement learning algorithm." *Expert Systems with Applications* **114**: pp. 388–401.

Zhang, J. L., W. K. Härdle, C. Y. Chen, & E. Bommes (2016): "Distillation of news flow into analysis of stock reactions." *Journal of Business & Economic Statistics* **34(4)**: pp. 547–563.

Appendix A

Figures

Figure A.1: Historical Prices (part 2)

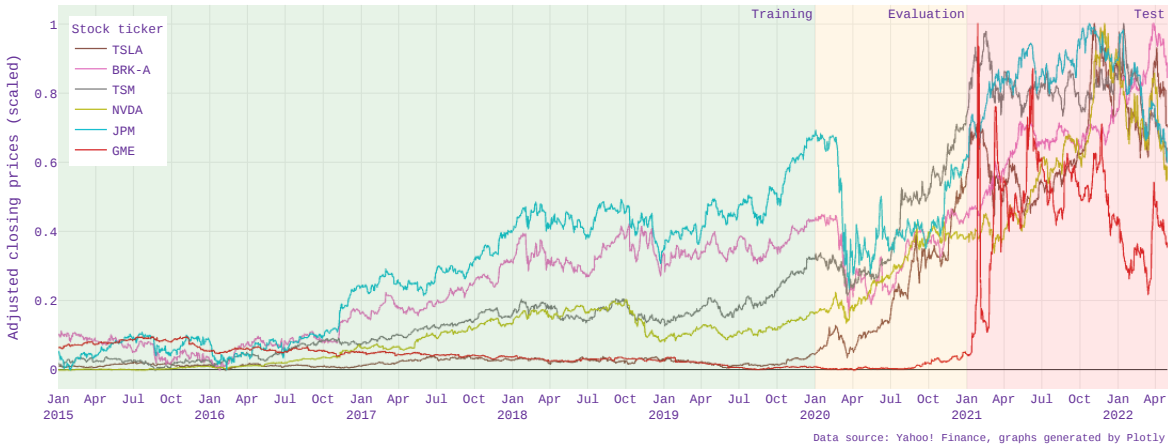


Figure A.2: Train Data: Trading Example

Figure A.3: Test Data: Trading Example

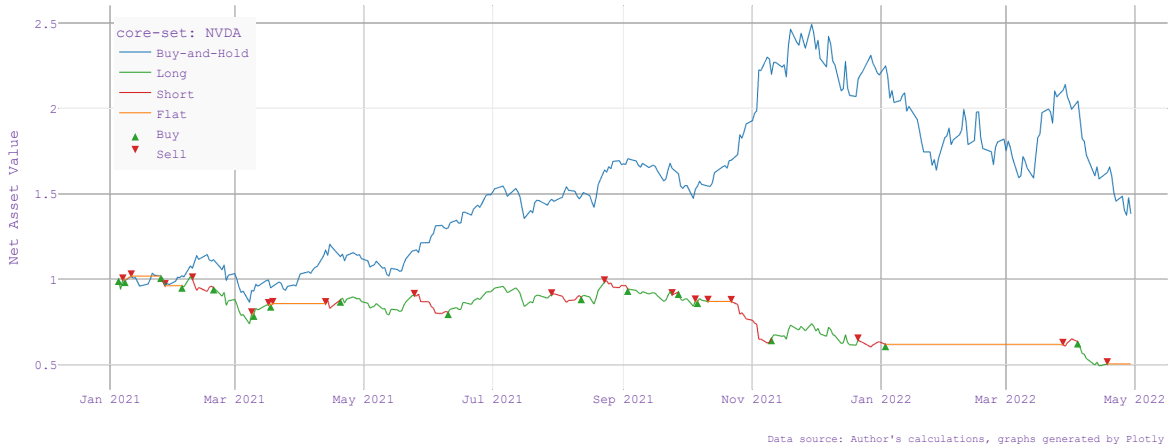


Figure A.4: Train Data: Equally Weighted Portfolio's Value

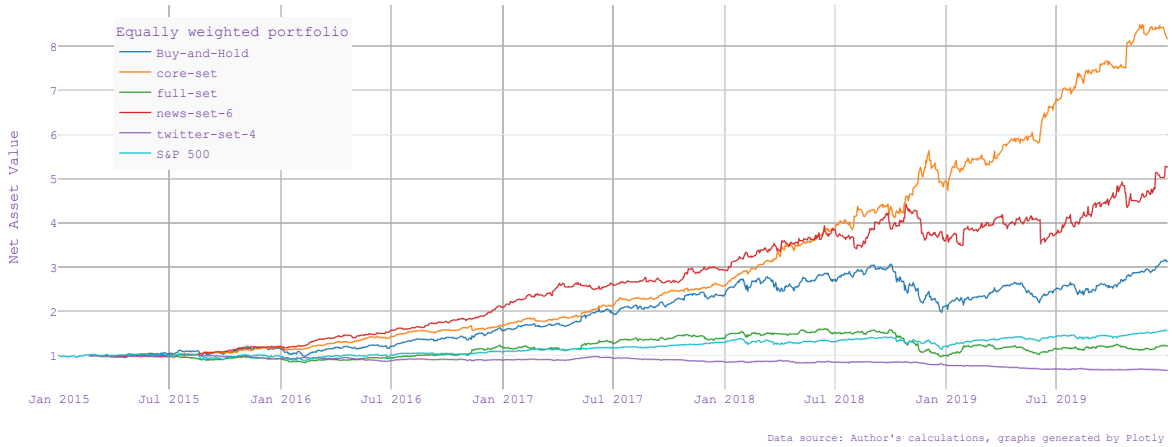


Figure A.5: Train Data: Minimum Variance Portfolio's Value

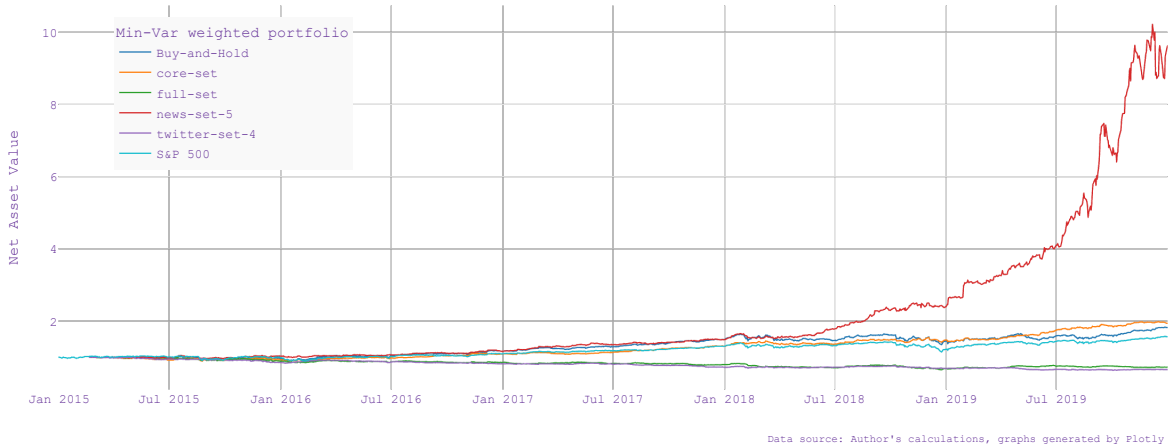


Figure A.6: Development Data: Equally Weighted Portfolio's Value

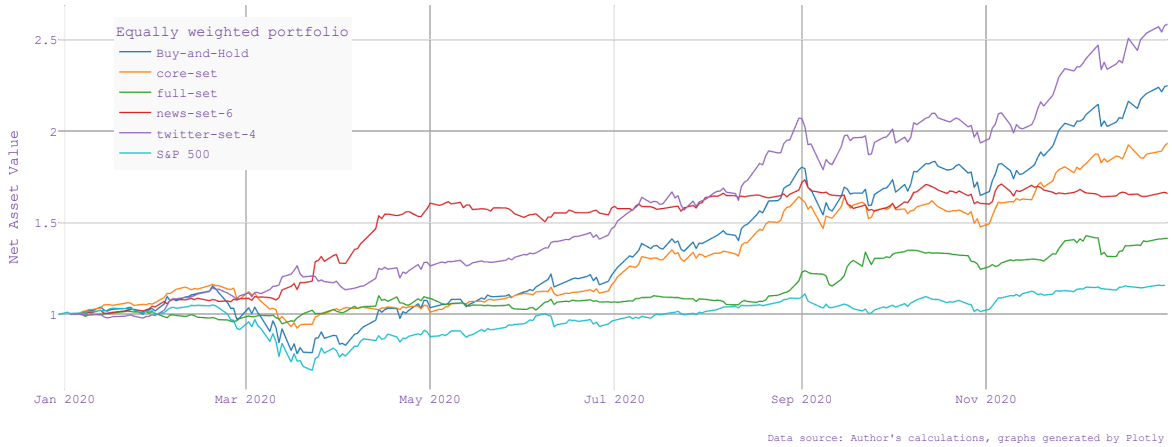


Figure A.7: Development Data: Minimum Variance Portfolio's Value

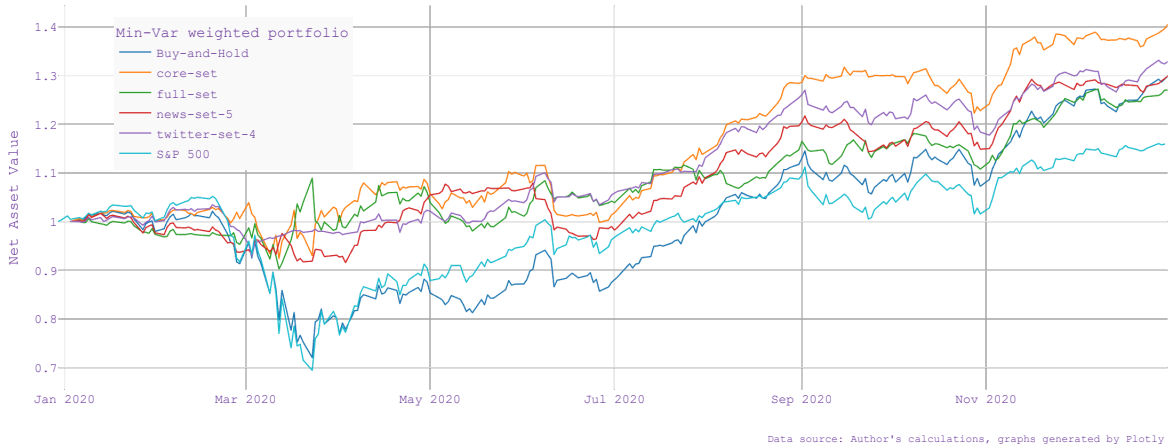


Figure A.8: Test Data: Minimum Variance Portfolio's Value

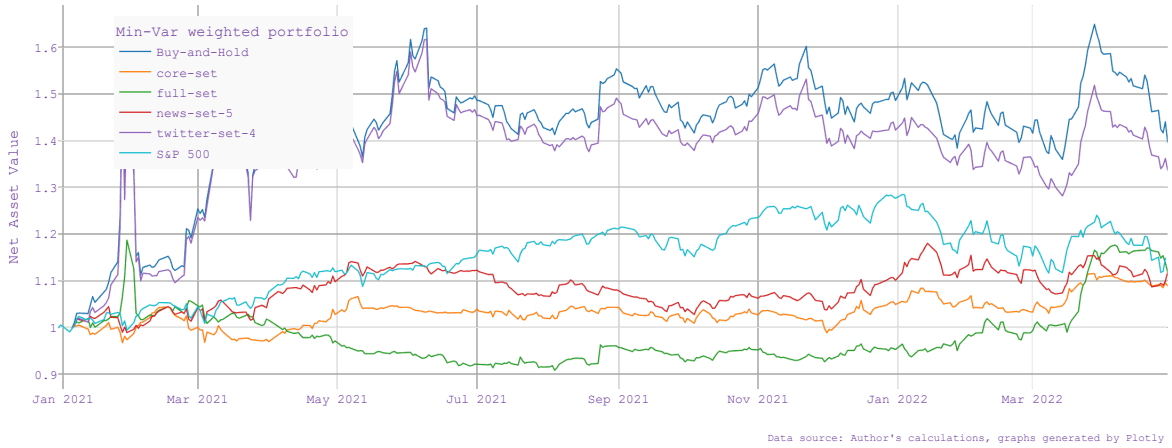
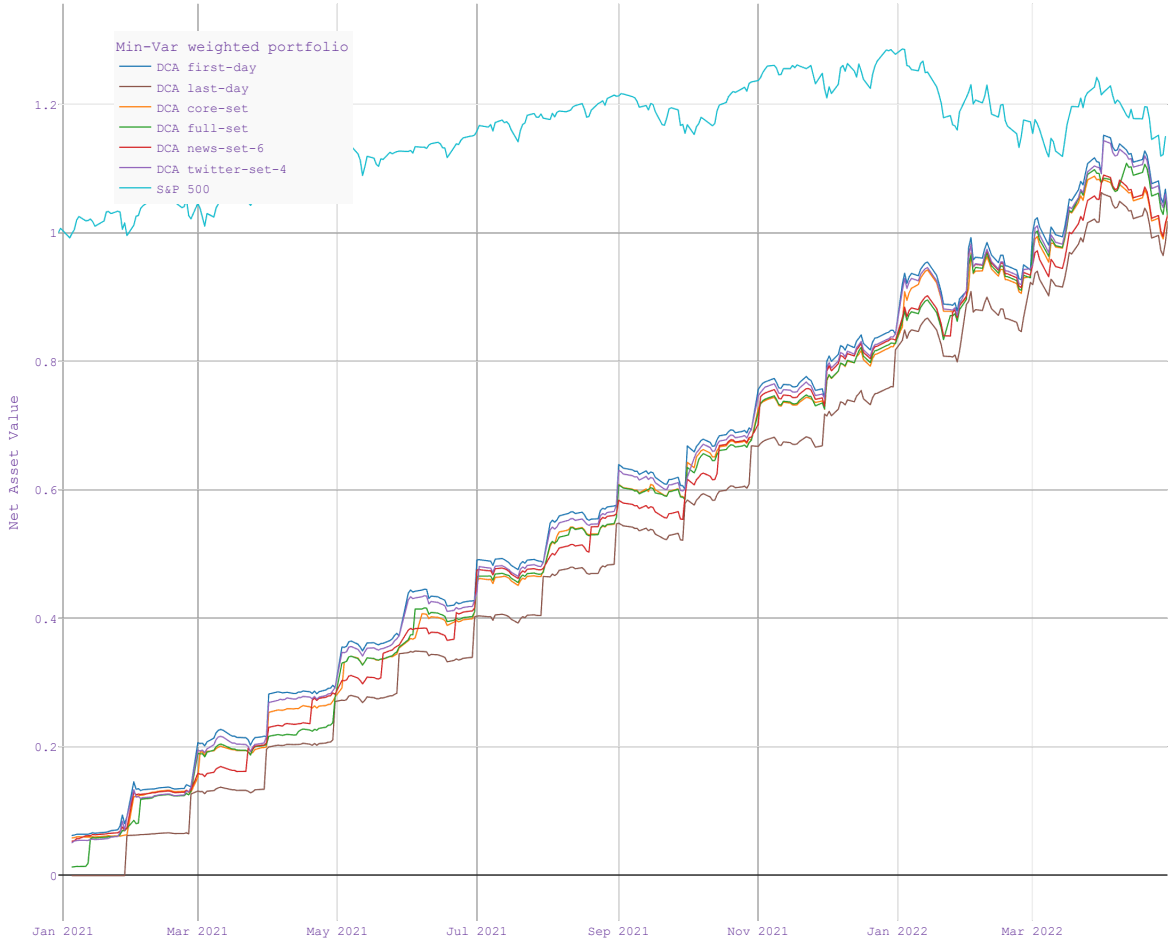


Figure A.9: Test Data: Minimum Variance Portfolio's Value (DCA)



Appendix B

Tables

Table B.1: Correlations of Features and Future Returns

	AAPL	MSFT	GOOGL	AMZN	TSLA	FB	NVDA	BRK-A	TSM	JPM	GME
Returns											
returns_1	-0.09	-0.19	-0.07	-0.03	0.00	-0.06	-0.09	-0.13	-0.13	-0.12	0.08
returns_2	-0.06	-0.13	-0.04	-0.02	0.02	-0.04	-0.04	-0.04	-0.07	-0.03	0.10
returns_5	-0.04	-0.10	-0.05	-0.02	0.01	-0.06	-0.01	-0.04	-0.05	-0.04	0.07
returns_10	-0.01	-0.09	-0.05	-0.03	0.02	-0.06	-0.02	-0.01	-0.04	-0.03	0.06
returns_21	0.01	-0.08	-0.03	-0.03	0.01	-0.01	-0.01	-0.03	-0.00	-0.04	-0.03
Indicators											
STOCH	0.03	0.03	0.01	0.01	-0.03	0.02	0.02	0.01	0.03	0.01	-0.05
ULTOSC	-0.00	-0.08	-0.04	-0.02	0.02	-0.05	-0.00	-0.02	-0.04	-0.00	-0.04
RSI	-0.03	-0.04	-0.03	-0.01	0.01	-0.05	0.01	-0.00	-0.03	-0.02	0.06
MACD	-0.01	-0.05	-0.03	-0.04	-0.01	-0.00	-0.01	-0.03	0.01	-0.01	-0.05
ATR	-0.00	0.00	-0.01	-0.04	-0.00	-0.02	-0.04	0.00	-0.02	-0.02	0.01
News											
newsCount	0.01	0.01	0.01	-0.04	-0.04	-0.04	0.01	0.02	-0.16	-0.07	-0.11
opinionsCount	0.02	0.02	-0.01	-0.03	-0.03	0.01	0.09	-0.18	0.01	-0.03	-0.15
sent_title_news	-0.03	0.01	0.02	0.04	0.02	0.02	-0.01	-0.04	-0.03	-0.01	0.05
sent_title_opinions	-0.02	0.00	0.02	-0.01	-0.04	-0.01	0.02	0.13	0.17	0.03	0.02
sent_content_news	-0.00	0.00	0.04	0.04	0.02	0.03	-0.04	-0.11	-0.04	0.04	0.04
sent_content_opinions	0.01	0.00	-0.00	-0.02	0.02	0.02	-0.08	-0.07	-0.04	0.06	0.05
Twitter											
tweetCount_cashtags	0.03	0.02	0.03	0.01	0.00	-0.07	-0.04	-0.02	-0.01	-0.05	0.10
retweetCount_cashtags	0.04	0.00	0.04	0.01	-0.03	-0.07	-0.01	0.01	0.03	-0.00	0.13
tweetCount_keywords	0.05	0.02	0.02	0.01	-0.04	0.01	0.01	-0.00	-0.03	0.05	0.08
retweetCount_keywords	0.01	0.04	0.02	0.00	-0.06	-0.00	-0.03	-0.01	0.04	0.05	0.09
sent_cashtags	0.01	0.04	0.02	0.05	0.03	0.03	0.05	0.03	0.04	-0.01	0.01
sent_keywords	-0.02	-0.03	-0.00	0.02	0.00	0.00	0.03	0.05	0.00	0.00	0.00

All values correspond to Pearson's correlation with respect to future returns. Lagged averages of sentiments were not included.

Source: Author's calculations.

Table B.2: Train Data: Stock's Final Net Asset Value

	AAPL	MSFT	GOOGL	AMZN	FB	TSLA	BRK-A	TSM	NVDA	JPM	GME
Buy-and-Hold	2.265	3.592	2.470	4.812	2.566	1.925	1.521	2.375	10.522	2.329	0.162
DDQN Agent											
core-set	6.730	3.596	3.523	27.083	0.273	25.306	1.292	2.168	16.093	0.603	3.151
	(1000)	*(1000)*	*(350)*	*(1000)*	*(100)*	*(425)*	*(100)*	*(1000)*	*(1000)*	*(100)*	*(1000)*
news-set-1	0.631	1.304	0.891	32.212	4.176	7.987	0.651	2.144	18.879	3.132	4.998
	(1000)	*(1000)*	*(200)*	*(1000)*	*(1000)*	*(1000)*	*(800)*	*(250)*	*(325)*	*(525)*	*(1000)*
news-set-2	0.282	0.721	1.441	4.424	5.018	3.636	0.560	4.479	40.754	0.768	0.698
	(475)	*(250)*	*(150)*	*(450)*	*(1000)*	*(1000)*	*(150)*	*(800)*	*(1000)*	*(125)*	*(250)*
news-set-3	8.296	4.223	5.349	2.264	0.414	78.831	0.487	7.127	2.426	0.870	10.840
	(450)	*(1000)*	*(225)*	*(400)*	*(175)*	*(1000)*	*(125)*	*(925)*	*(850)*	*(100)*	*(275)*
news-set-4	4.370	2.210	1.157	8.669	4.275	1.015	0.413	3.107	10.216	0.507	0.261
	(1000)	*(1000)*	*(250)*	*(1000)*	*(875)*	*(150)*	*(100)*	*(1000)*	*(1000)*	*(175)*	*(200)*
news-set-5	3.364	3.531	2.337	10.842	5.115	40.257	1.030	3.550	20.690	0.720	296.250
	(1000)	*(400)*	*(325)*	*(1000)*	*(225)*	*(1000)*	*(200)*	*(1000)*	*(1000)*	*(175)*	*(1000)*
news-set-6	1.790	4.630	1.616	3.407	0.852	10.471	0.827	0.741	0.523	1.031	31.869
	(1000)	*(550)*	*(225)*	*(725)*	*(200)*	*(1000)*	*(250)*	*(700)*	*(250)*	*(250)*	*(1000)*
twitter-set-1	0.347	0.866	0.977	0.477	0.784	0.975	0.207	4.913	0.038	0.469	10.033
	(625)	*(300)*	*(450)*	*(1000)*	*(375)*	*(1000)*	*(1000)*	*(1000)*	*(1000)*	*(100)*	*(1000)*
twitter-set-2	0.419	1.899	8.293	14.733	2.569	3.137	0.808	3.325	28.462	1.595	0.148
	(1000)	*(1000)*	*(525)*	*(1000)*	*(1000)*	*(800)*	*(275)*	*(1000)*	*(950)*	*(275)*	*(175)*
twitter-set-3	3.020	3.181	6.381	7.674	4.092	1.927	0.461	2.377	9.871	1.258	0.173
	(275)	*(1000)*	*(1000)*	*(1000)*	*(1000)*	*(1000)*	*(125)*	*(1000)*	*(850)*	*(125)*	*(400)*
twitter-set-4	0.703	0.892	1.555	1.060	0.801	0.240	0.488	1.042	0.235	0.302	0.107
	(500)	*(100)*	*(425)*	*(1000)*	*(800)*	*(125)*	*(150)*	*(1000)*	*(425)*	*(100)*	*(525)*
full-set	0.377	0.807	0.195	0.676	0.182	0.497	0.595	1.948	6.836	1.037	0.310
	(1000)	*(150)*	*(425)*	*(1000)*	*(1000)*	*(1000)*	*(675)*	*(1000)*	*(1000)*	*(300)*	*(175)*

Final NAV of agent's trading strategy when applied on train data. Values in parentheses represent the number of train episodes.
Source: Author's calculations.

Table B.3: Train Data: Portfolio's Final Net Asset Value

	Equal weights					Minimum Variance weights				
Feature set	B&H	Trade	DCA-F	DCA-L	DCA-T	B&H	Trade	DCA-F	DCA-L	DCA-T
core-set	3.140	8.165	1.837	1.805	1.843	1.822	1.937	1.450	1.441	1.453
news-set-1	3.140	7.000	1.837	1.805	1.839	1.822	1.523	1.450	1.441	1.451
news-set-2	3.140	5.707	1.837	1.805	1.838	1.822	1.534	1.450	1.441	1.454
news-set-3	3.140	11.011	1.837	1.805	1.828	1.822	2.683	1.450	1.441	1.449
news-set-4	3.140	3.291	1.837	1.805	1.840	1.822	1.288	1.450	1.441	1.453
news-set-5	3.140	35.244	1.837	1.805	1.841	1.822	9.616	1.450	1.441	1.453
news-set-6	3.140	5.251	1.837	1.805	1.836	1.822	1.766	1.450	1.441	1.450
twitter-set-1	3.140	1.826	1.837	1.805	1.825	1.822	1.232	1.450	1.441	1.451
twitter-set-2	3.140	5.944	1.837	1.805	1.839	1.822	1.920	1.450	1.441	1.451
twitter-set-3	3.140	3.674	1.837	1.805	1.837	1.822	1.573	1.450	1.441	1.451
twitter-set-4	3.140	0.675	1.837	1.805	1.830	1.822	0.658	1.450	1.441	1.446
full-set	3.140	1.224	1.837	1.805	1.825	1.822	0.726	1.450	1.441	1.446

Final NAV of agent's portfolio with respect to strategy and initialization when applied on training data.
Source: Author's calculations.

Table B.4: Development Data: Portfolio's Final Net Asset Value

	Equal weights					Minimum Variance weights				
Feature set	B&H	Trade	DCA-F	DCA-L	DCA-T	B&H	Trade	DCA-F	DCA-L	DCA-T
core-set	2.245	1.935	1.755	1.668	1.724	1.296	1.404	1.324	1.303	1.313
news-set-1	2.245	1.652	1.755	1.668	1.749	1.296	1.448	1.324	1.303	1.340
news-set-2	2.245	2.165	1.755	1.668	1.774	1.296	1.505	1.324	1.303	1.334
news-set-3	2.245	2.047	1.755	1.668	1.751	1.296	1.661	1.324	1.303	1.319
news-set-4	2.245	2.328	1.755	1.668	1.739	1.296	1.290	1.324	1.303	1.309
news-set-5	2.245	1.330	1.755	1.668	1.745	1.296	1.298	1.324	1.303	1.321
news-set-6	2.245	1.659	1.755	1.668	1.766	1.296	1.382	1.324	1.303	1.325
twitter-set-1	2.245	1.045	1.755	1.668	1.697	1.296	0.873	1.324	1.303	1.305
twitter-set-2	2.245	2.404	1.755	1.668	1.752	1.296	1.390	1.324	1.303	1.321
twitter-set-3	2.245	2.190	1.755	1.668	1.752	1.296	1.419	1.324	1.303	1.320
twitter-set-4	2.245	2.583	1.755	1.668	1.746	1.296	1.328	1.324	1.303	1.310
full-set	2.245	1.415	1.755	1.668	1.746	1.296	1.269	1.324	1.303	1.319

Final NAV of agent's portfolio with respect to strategy and initialization when applied on development data.
Source: Author's calculations.