5-2022

Artificial Intelligence and Deep Reinforcement Learning Stock Market Predictions

Andrew W. Brim
Utah State University

ARTIFICIAL INTELLIGENCE AND DEEP REINFORCEMENT LEARNING STOCK MARKET PREDICTIONS

by

Andrew W. Brim

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Computer Science

Approved:

Nicholas Flann, Ph.D. Major Professor	Vicki H. Allan, Ph.D. Committee Member
Tyler Brough, Ph.D. Committee Member	Chad Mano, Ph.D. Committee Member
Ben Blau, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Interim Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2022

ABSTRACT

Artificial Intelligence and Deep Reinforcement Learning Stock Market Predictions

by

Andrew W. Brim, Doctor of Philosophy

Utah State University, 2022

Major Professor: Nicholas Flann, Ph.D.
Department: Computer Science

Billions of dollars are traded automatically in the stock market every day, including algorithms that use artificial intelligence (AI) techniques, but there are still questions regarding how AI trades successfully. The black box nature of these AI techniques, namely neural networks, gives pause to entrusting it with valuable trading funds. This dissertation applies AI techniques to stock market trading strategies, but it also provides exploratory research into how these techniques predict the stock market successfully.

This dissertation presents the work of three research papers. The first paper presented in this dissertation applies a artificial intelligence technique, reinforcement learning, to candlestick pattern trading. This paper utilizes a Double Deep Q-Network (DDQN) to outperform the S&P 500 Index returns. This paper also analyzes how the DDQN trades, through the use of a more recent technique, feature map visualizations. The second paper uses fuzzy logic to arbitrage the causal relationship in a pairs trading strategy. This paper identifies the causal relationship between two cointegrated assets as a fuzzy relationship, and utilizes fuzzy logic to yield higher returns. The third paper utilizes a Double Deep Q-Network (DDQN) to predict the spread of two cointegrated assets in a pairs trading strategy. It also explores the ability to make the DDQN trade more conservatively through

the use of a Negative Rewards Multiplier.

The first paper results show that the DDQN is able to outperform the S&P 500 Index returns. Results also show that the CNN is able to switch its attention from all the candles in a candlestick image to the more recent candles in the image, based on an event such as the coronavirus stock market crash of 2020. The second paper results show fuzzy logic applied to pairs trading strategy for 22 stock pairs, increases annual returns on average from 15% to 17%. The third paper results show a DDQN was able to accurately predict the spread of the Adobe/Red Hat pair, for positive returns. Results also show that the use of a newly introduced variable, Negative Rewards Multiplier (NRM), can cause the DDQN to trade more conservatively and reduce the number of predictions resulting in negative returns. This dissertation shows that AI techniques are successful in predicting the stock market, but more importantly it provides research tools and methods to better understand and implement these techniques in stock market trading.

(85 pages)

PUBLIC ABSTRACT

Artificial Intelligence and Deep Reinforcement Learning Stock Market Predictions

Andrew W. Brim

Billions of dollars are traded automatically in the stock market every day, including algorithms that use artificial intelligence (AI) techniques, but there are still questions regarding how AI trades successfully. The black box nature of these AI techniques, namely neural networks, gives pause to entrusting it with valuable trading funds. This dissertation applies AI techniques to stock market trading strategies, but it also provides exploratory research into how these techniques predict the stock market successfully.

This dissertation presents the work of three research papers. The first paper presented in this dissertation applies a artificial intelligence technique, reinforcement learning, to candlestick pattern trading. This paper also analyzes how the DDQN trades, through the use of a more recent technique, feature map visualizations. The second and third paper analyze AI techniques in a pairs trading strategy. The first paper results show that the DDQN is able to outperform the S&P 500 Index returns. Results also show that the CNN is able to switch its attention from all the candles in a candlestick image to the more recent candles in the image, based on an event such as the coronavirus stock market crash of 2020. The second paper results show fuzzy logic applied to pairs trading strategy for 22 stock pairs, increases annual returns on average from 15% to 17%. The third paper results show a DDQN was able to accurately predict the spread of the Adobe/Red Hat pair, for positive returns. This dissertation shows that AI techniques are successful in predicting the stock market, but more importantly it provides research tools and methods to better understand and implement these techniques in stock market trading.

To my beautiful wife Camey for her years of love, and kindness. I will forever be grateful for your amazing support.

ACKNOWLEDGMENTS

I would like to thank the following individuals for their constant and enormous support. My wife Camey for always loving and believing in me, and pushing me to be better. My parents Greg and Jill Brim. I could not have asked for better parents. Thank you for the examples of strong character that you have always been. My wonderful children, my siblings Doren, Rex and Rachel, and my wonderful in-laws Jody and Karen. The Utah State University Computer Science department, and my extremely talented and intelligent committee members, namely my advisors Dr. Nicholas Flann, Dr. Tyler Brough, and Dr. Vicki Allan, and my committee members Dr. Ben Blau, and Dr. Chad Mano. Thank you Dr. Flann for your love of AI and your wealth of knowledge you so readily dispose to your students. Thank you Dr. Allan and Dr. Mano, for your constant guidance on my path to becoming a researcher. Thank you Dr. Blau for the hours in your office ensuring my statistical and financial rigor. Thank you Dr. Brough. These seven years of educating and enlightening have strengthened and prepared me indeed. Above all I must thank the Almighty, for His endless love and support.

Andrew Brim

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Artificial intelligence (AI) techniques, including neural networks, have proven successful in predicting financial markets. However the black box nature of neural networks creates strong demand for insight into how neural networks accomplish this. This dissertation presents three research papers which utilize AI techniques, including neural networks, for stock market predictions. These papers also explore research questions regarding how AI accomplishes this.

This dissertation utilizes reinforcement learning (RL), in multiple instances, to predict the stock market. RL is a artificial intelligence technique, where an agent interacts with an environment through actions. A state is provided by an environment, and the agent selects an action based on that state to maximize a reward. The agent learns through states and actions to maximize its reward [1]. This dissertation employs Q-learning, a type of temporal difference reinforcement learning, to approximate a policy function for each state in the space of trading parameters [2, 3]. Q-learning is combined with function approximation, utilizing a neural network to approximate a Q-function.

A Deep Q network (DQN) is a multi-layered neural network that for a given input state, outputs a vector of action values [4]. This dissertation employs a specific type of DQN, a Double Deep Q-Network, introduced by Google Deep Mind in 2016 [4] and utilized by the artificial intelligence AlphaGo which defeated the World Go champion Lee Sedol [5]. Van Hasselt, Guez, and Silver (2016), show that the idea behind the Double Q-learning algorithm (Van Hasselt, 2010), which was first proposed in a tabular setting, can be generalized to work with arbitrary function approximation, including deep neural networks. The Double DQN (DDQN), not only yields more accurate value estimates, but leads to better overall

performance of the deep neural network [4].

A more recent tool for analyzing neural networks are feature map visualizations. The Google Brain Team DeepDream project has made recent advancements with feature map visualizations to understand neural networks. They claim these tools are one of the fundamental building blocks that will empower humans to understand neural networks [6]. Feature map visualizations are generated from the input image and the regions of neuron excitement on the input image. Neuron excitement is the activation output from the neurons, in a neural network. While there are multiple uses of feature map visualizations, this work will utilize the values of the feature maps output as a 2D array. A similar approach to Nguyen, Yosinski, Clune, et al. (2019) [7] is used in this work, because it provides the neuron excitement as 2D arrays that can be measured and analyzed.

1.2 Proposed Methods

A DDQN is used in this dissertation since it has been shown to yield more accurate values and give better overall performance than other neural network systems [4]. Many recent applications have employed a DDQN as a result, including autonomous vehicles [8,9], energy reduction and battery optimization [10–12], and robotics and UAV controls [13,14].

In the first paper, a DDQN is used to outperform the S&P500 Index. Additionally feature map visualizations are combined with a DDQN to discover that a neural can switch its attention from a wide focus of all the regions in an input image to a narrower focus, based on an event. In the second paper fuzzy logic is used to increase returns in a pairs trading strategy. In the third paper, a DDQN is used to learn and predict the spread of two cointegrated assets in a pairs trading strategy.

1.3 Research Questions

In addition to utilizing AI techniques to predict the stock market, this dissertation also explores how this is achieved. To explore how AI can be used to predict the stock market the following research questions are pursued:

1. In the first paper, does the evidence yielded from feature map visualizations support that a DDQN uses the most recent days in the candlestick image to determine a trade signal, or are all the days used evenly?

2. In the first paper, does an RL system utilizing a DDQN adjust its attention following an event, such as the coronavirus stock market crash?

3. In the second paper, how can fuzzy logic be used to arbitrage the causal dynamics in a pairs trading strategy?

4. In the third paper, how does an RL system utilizing a DDQN learn to trade more conservatively to reduce the number of trades that result in a negative returns?

1.4 Contributions

The contributions of this dissertation are to bridge the gap between two streams of research: financial and computer science research. This dissertation also substantiates that artificial intelligent techniques, namely an RL system utilizing a DDQN, can be used outperform the S&P 500 Index. It makes direct comparisons of the DDQN returns to financial market returns, where previous works have not. This dissertation also provides exploration of how a DDQN trades, using feature map visualizations, and compares this method to human based trading strategies. Previous works utilizing artificial intelligence for financial markets predictions have not used feature map visualizations to make these comparisons. This dissertation also provides exploration into improving pairs trading strategies, using fuzzy logic to arbitrage causal dynamics. It also utilizes a DDQN to predict the spread of two assets in a pairs trading strategy.

CHAPTER 2

Deep Reinforcement Learning Stock Market Trading, utilizing a CNN with Candlestick Images

Deep Reinforcement Learning Stock Market Trading, utilizing a CNN with Candlestick Images

Andrew Brim[1] and Nicholas S. Flann[2]

[1]Department of Computer Science, Utah State University
[2]Department of Computer Science, Utah State University
andrew.brim@usu.edu, nick.flann@usu.edu

November 2021

2.1 Abstract

Billions of dollars are traded automatically in the stock market every day, including algorithms that use neural networks, but there are still questions regarding how neural networks trade. The black box nature of a neural network gives pause to entrusting it with valuable trading funds. A more recent technique for the study of neural networks, feature map visualizations, yields insight into how a neural network generates an output. Utilizing a Convolutional Neural Network (CNN) with candlestick images as input and feature map visualizations gives a unique opportunity to determine what in the input images is causing the neural network to output a certain action. In this study, a CNN is utilized within a Double Deep Q-Network (DDQN) to outperform the S&P 500 Index returns, and also analyze how the system trades. The DDQN is trained and tested on the 30 largest stocks in the S&P 500. Following training the CNN is used to generate feature map visualizations to determine where the neural network is placing its attention on the candlestick images. Results show that the DDQN is able to yield higher returns than the S&P 500 Index between January 2, 2020 and June 30, 2020. Results also show that the CNN is able to switch its attention from all the candles in a candlestick image to the more recent candles in the image, based on an event such as the coronavirus stock market crash of 2020.

2.2 Introduction

Neural networks have proven successful in predicting financial markets. This includes the use of CNNs [1–4]. However the black box nature of neural networks creates strong demand for insight into how neural networks accomplish this. This work utilizes feature map visualization to generate a visual representation from the CNN weights to analyze how the neural network is able to do this [5–8]. The Google Brain Team DeepDream project has made recent advancements with feature map visualizations to understand neural networks. They claim these tools are one of the fundamental building blocks that will empower humans to understand neural networks [9]. In this work feature map visualizations are used to discover that a CNN can switch its attention from a wide focus of all the regions in an input image to a narrower focus, based on an event.

A specific type of reinforcement learning (RL) system, Double Deep Q-Network (DDQN), is used in this work since it has been shown to yield more accurate values and give better overall performance than other neural network systems [10]. Many recent applications have employed a DDQN as a result, including autonomous vehicles [11, 12], energy reduction and battery optimization [13–15], and robotics and UAV controls [16, 17].

Here a DDQN utilizing a CNN with only candlestick images as input can outperform the S&P 500 Index during one of the most unprecedented stock market crashes in financial history, the coronavirus stock market crash of 2020.

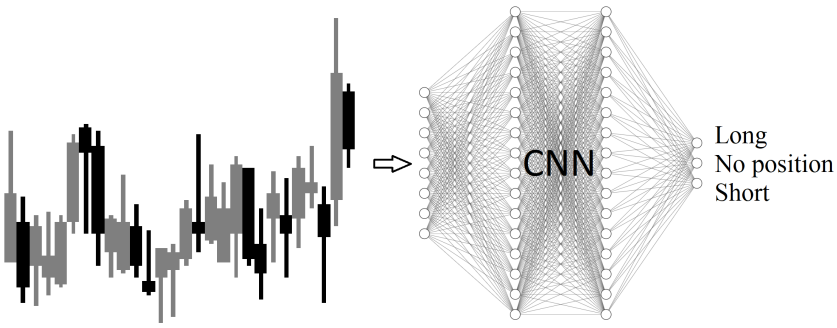


Figure 1: Candle stick images are generated based on the open, close, high, low prices for each day, for each stock. The CNN in the RL system receives candlestick images as input and outputs actions of long, short, or no position.

The motivation for this work is to bridge the gap in computer science research, and financial research, to test whether a RL system utilizing a CNN can outperform the S&P 500 Index. An additional motivation is to determine how the CNN predicts the stock market. There is existing research which trains a CNN to trade financial markets via images, including candlestick images, such as Tsai, Chen et. al 2019 [2] and Selvin, Menon et al 2017 [1]. There is also existing research which utilizes deep neural networks with reinforcement learning

techniques for stock market predictions [18–20]. However no research has yet used feature map visualizations to reconstruct how the CNN predicts the stock market. Additionally there is no existing research which makes specific comparisons to the performance of financial markets, and comparisons to methodology of existing human based trading strategies.

Reinforcement learning is a artificial intelligence technique, where an agent interacts with an environment through actions. A state is provided by an environment, and the agent selects an action based on that state to maximize a reward. The agent learns through states and actions to maximize its reward [21]. In this work the agent is a DDQN, the state it receives is a candlestick image representing the previous 28 days of stock prices. The DDQN action is to take a long, short, or no position on the stock the next day.

This work employs Q-learning, a type of temporal difference reinforcement learning, to approximate a policy function for each state in the space of trading parameters [22, 23]. Q-learning is combined with function approximation, utilizing a CNN to approximate a Q-function. An OpenAI Gym environment [24] is built to simulate the stock market and provide candlestick images to a CNN. The CNN outputs a long, short, or no position action, as shown in Figure 1. The action is sent to the environment. The environment returns the next state and a reward for the action taken.

A Deep Q network (DQN) is a multi-layered neural network that for a given input state, outputs a vector of action values [10]. This work employs a specific type of DQN, a Double Deep Q-Network, introduced by Google Deep Mind in 2016 [10] and utilized by the artificial intelligence AlphaGo which defeated the World Go champion Lee Sedol [25]. Van Hasselt, Guez, and Silver (2016), show that the idea behind the Double Q-learning algorithm (Van Hasselt, 2010), which was first proposed in a tabular setting, can be generalized to work with arbitrary function approximation, including deep neural networks. The Double DQN, not only yields more accurate value estimates, but leads to better overall performance of the deep neural network [10]. For this reason, an RL system utilizing a DDQN is chosen for this work. A CNN is used within the DDQN for function approximation, with candlestick images as input, and a trading position of long, short, or no position as output.

The DDQN is trained on candlestick images generated from stock market prices from 2013 through 2019. It is then tested on candlestick images generated from January 2, 2020 through June 30, 2020. The top 30 stocks in the S&P 500 Index are selected since the data is widely available, and this creates a large enough base of stocks to ensure robustness in results. The DDQN receives an image of 28 candlesticks for each day. This means for 30 stocks, there are a total of 52920 observations in the training data set, and 3780 observations in the testing data set. The training data set of seven years is selected to allow enough time for the DDQN to learn to trade each stock. This training data set also includes the China Tariffs Dispute stock market crash of 2018, which allows the DDQN to learn how to perform during a stock market crash. Six months is selected as the testing data set, allowing for enough time to verify the DDQN

performance and also observe the behavior of the DDQN during the coronavirus stock market crash.

The S&P 500 Index value dropped from $3372.23 to $2234.40 on Mar 23, 2020. A 33.7% loss in 31 days. The S&P 500 Index value began to quickly recover, increasing to $3232.39 on Jun 8, 2020. A 96% recovery in 45 days. The coronavirus stock market crash data is unlike any data in the training data set. The closest event in the training data set is the China tariff dispute stock market crash in 2018, where the S&P 500 declined 20.9% and subsequently recovered as shown in Figure 2.

Figure 2: S&P 500 Index January 2013 - June 2020. Training data consists of stock prices from January 2013 through December 2019. Testing data consists of stock prices from January 2020 through June 2020. The coronavirus stock market crash from Feb 20, 2020 to Mar 23, 2020 is a greater decline, 33%, than any event in the training data. The China tariffs dispute stock market crash in 2018 was closest, a 20.9% decline in 91 days.

Outperforming the S&P 500 Index, defined by this work, will be the DDQN yielding higher geometric returns than the S&P 500 Index for the testing data set. Tests are run with the largest 30 stocks of the S&P 500 including Apple Inc. (AAPL), Amazon.com (AMZN), Microsoft Corporation (MSFT), Google LLC (GOOGL), Facebook Inc. (FB), Adobe Inc. (ADBE), Berkshire Hathaway Inc. Class B (BRK.B), JPMorgan Chase Co. (JPM), Johnson Johnson (JNJ), Visa Inc. Class A (V), United Health Group Incorporated (UNH), Procter Gamble Company (PG), Walt Disney Company (DIS), Home Depot Inc. (HD), Mastercard Incorporated Class A (MA), Bank of America Corp (BAC), Exxon Mobil Corporation (XOM), Coca-Cola Company (KO), Intel Corpora-

tion (INTC), AT&T Inc. (T), Walmart Inc. (WMT), Boeing Airlines Co. (BA), Comcast Corporation (CMCSA), Cisco Systems Inc. (CSCO), Chevron Corporation (CVX), Merck and Company Inc. (MRK), PepsiCo Inc. (PEP), Pfizer Inc. (PFE), Verizon Communications Inc. (VZ), and Wells Fargo Co (WFC).

Human traders use candlestick images to make trading decisions based on calculable trends, but also experience. Candlestick images can allow the trader to see features that are not numeric. Feature map visualizations are used to interpret what in the input images is causing the DDQN to output a given action. The Google Brain Team research on feature visualization, specifically Olah, Mordvintsev, and Schubert (2017) [9] utilizes neural network weight reconstruction to generate images. They claim, in the quest to make neural networks interpretable, feature visualization stands out as one of the most promising and developed research directions.

Feature map visualizations are generated from the input image and the regions of neuron excitement on the input image as show on the far right image in Figure 3. Neuron excitement is the activation output from the neurons, in a neural network. While there are multiple uses of feature map visualizations, this work will utilize the values of the feature maps output as a 2D array. A similar approach to Nguyen, Yosinski, Clune, et al. (2019) [8] is used in this work, because it provides the neuron excitement as 2D arrays that can be measured and analyzed.

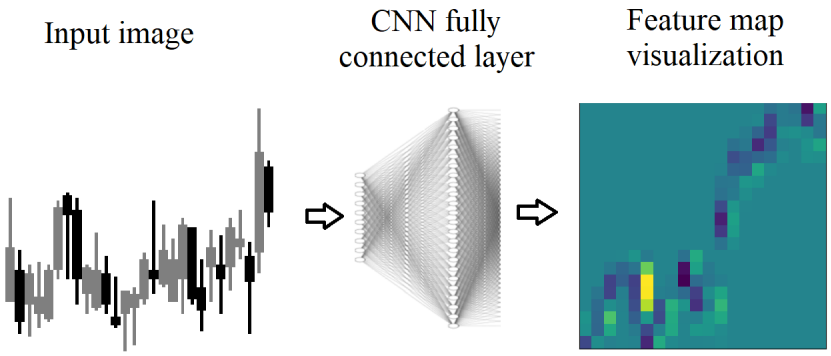


Figure 3: Feature map visualizations are generated from fully connected layers in the DDQN. A candlestick image is input, and each neuron in the fully connected layer receives the image. The neuron is excited on various parts of the image. The excited regions are stored as a 2D array, and shown here as a heatmap.

The region in yellow indicates the highest neuron excitement, meaning the highest value output by the neuron. To generate a feature map visualization, after training, a new network is constructed with only two layers: an input layer for the input image, and a layer constructed from the weights of the fully connected layer. The input image is passed into the second layer and the neurons are excited according to their weights calculated during training, and the output is

stored in a 2D array.

Each candlestick image represents 28 days, while there are 20 regions generated in the feature map. This is due do the reshaping performed by the convolutional layer. Still, the 20 regions provide strong insight into which region provides the highest neuron excitement level, and which region of the candlestick image used by the DDQN to determine a trading signal. For example, ADBE candlestick images can be seen at the beginning of the corona stock market crash in Figure 4. As the stock market crash begins, the neuron excitement shifts from all 28 candlesticks to the most recent candlesticks. The day before the stock market crash, neuron excitement levels are highest on days 23 through 27, and day 9. However the next day, the first day of the crash, the neuron excitement level is highest on the most recent day of the candlestick image. The neuron excitement level is almost double the excitement level of any other day in the image. This excitement level continues on the most recent days as the coronavirus stock market crash continues.

While each of these feature map visualization tools are powerful in revealing how a CNN has learned, this work specifically needs to measure changes in neuron excitement. Therefore a similar approach to Nguyen, Yosinski, Clune, et al. (2019) is used in this work. To generate the output images shown in Figure 4, candlestick images are combined with the neuron activation output, or neuron excitement. The newly created image shows exactly which regions in the input image caused neuron excitement.

The contributions of this work is that it is the first to provide exploration of how a DDQN predicts the stock market, using feature map visualizations. This work also substantiates that artificial intelligent techniques, namely an RL system utilizing a DDQN, can be used to outperform the S&P 500 Index. It makes direct comparisons of the DDQN returns to financial market returns, where previous works have not.

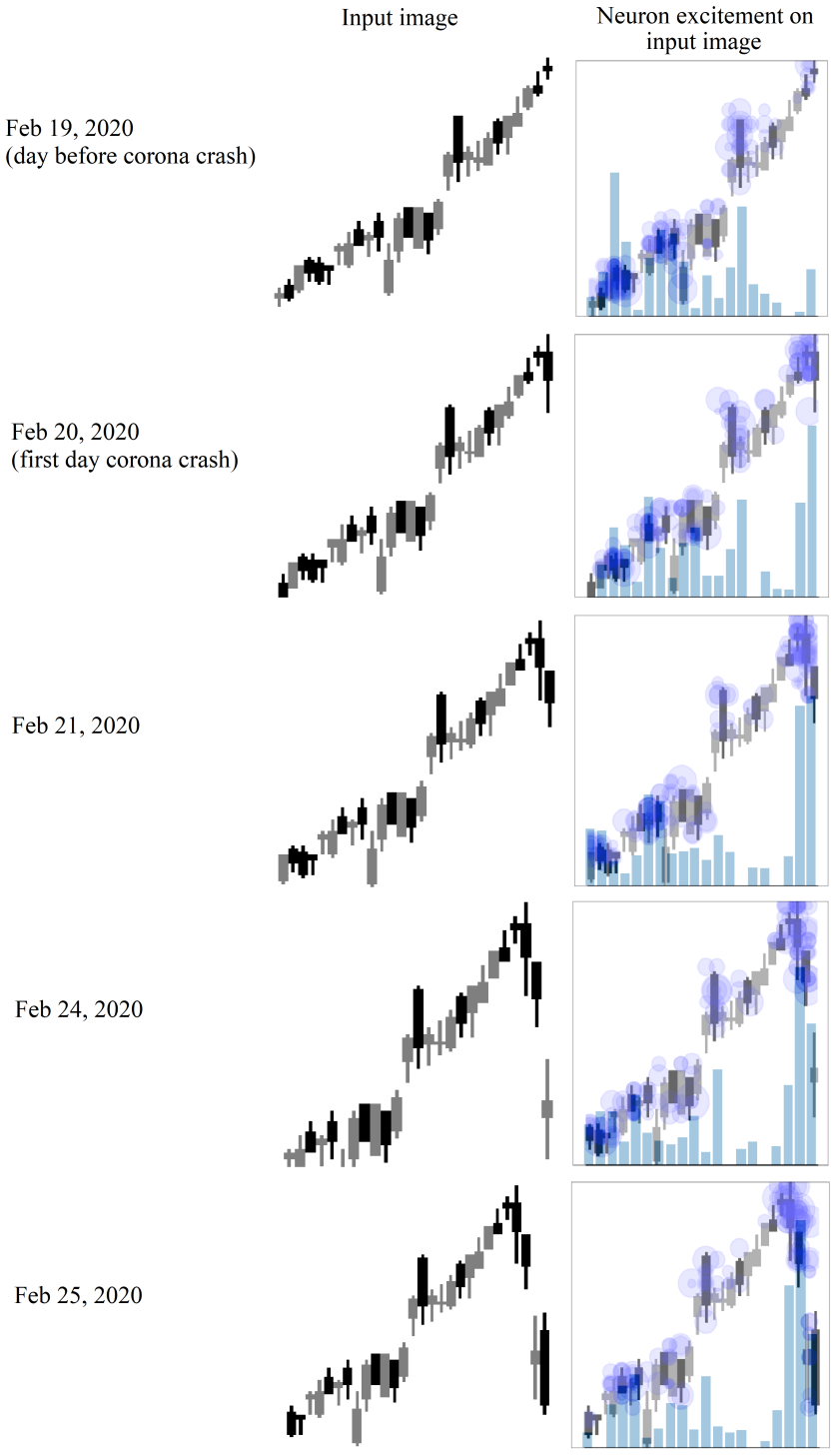


Figure 4: Adobe candlesticks images representing the day before, and the first four days of the coronavirus stock market crash. Each row represents the day the input image was supplied to the DDQN. The first column represents the input image. The second column displays the regions of the image excited by each neuron. The size of each blue dot represents the level of excitement. The blue is made darker by overlapping dots, indicating multiple neurons were excited by the same region of the image. The bars indicate the total excitement value.

2.3 Background

In this section, existing literature is reviewed in two separate streams of research. First, the existing work in financial research on various trading strategies that are related to the tests proposed in this work. The associated profitability of these strategies documented in the literature is reviewed. Second, the prior work in the Computer Science literature that examines both deep learning and prediction in financial markets.

Previous work has shown that a CNN can be effective in stock market trading, but have not been applied with feature map visualization analysis, and have not compared their performance to the S&P 500 Index. In this work, feature map visualizations are used investigate the ability of a CNN to switch its attention from all days in a candlestick image, showing the full 28 day history, to the most recent days. This ability to switch from full history in a candlestick image, to the most recent days is different than existing technical trading indicators like those presented in Brock, Lakonishok and LeBaron (1992) and Skouras (2001). Marshall, Young and Rose (2006) show that candlestick patterns are not effective at predicting the stock market. However they use candlestick images between 1 and 5 days of price history, not 28 days like this RL system.

Financial Markets Literature Review: Price Prediction and Technical Trading

In this section, the existing financial literature that examines technical trading strategies is reviewed. There is a long standing divide between financial fundamental analysis and technical trading analysis. Fama (1965), and also Fama (2021), argue that stock market prices are efficient [26, 27] and therefore technical trading indicators should not outperform the market. However, Brock, Lakonishok and LeBaron (BLL) (1992) [28] are among the first to give strong validation to technical trading, or statistical based trading strategies.

BLL showed the validity of technical trading analysis by testing two of the simplest and most popular trading rules against the Dow Jones Index from 1897 to 1986. This was in direct opposition to the prevailing mindset of the time that technical trading analysis was not a strong research domain(see Malkiel (1981) [29]). Malkiel declared technical analysis was the anathema to the academic world, and its methods are patently false.

BLL results are impressive. The Variable Length Moving Average Rules yields 0.00042 daily returns for Buy signals (12% annual), -0.00025 daily returns for Sell signals (-7% annual), and 0.00067 daily returns for Buy-Sell signals (19% annual).

Skouras (2001) [30] further supports technical trading analysis by improving on

the methods used by BLL. Skouras introduces an Artificial Technical Analyst to dynamically select a technical indicator rather than selecting a signal technical indicator for an entire simulation. The term Artificial Technical Analyst is defined by Skouras as an agent with the ability to change the technical indicator mid simulation, by testing multiple technical indicators on recent data and selecting the best performing. For all time frames t-N to t-1 the technical trading parameters are tested and the Artificial Trading Analyst selects the parameters that produce the highest returns.

Marshall, Young and Rose (MYR) (2006) [31] perform technical trading analysis on the 26 most widely used candlestick patterns, on the Dow Jones Index 1992 to 2002. They give a comprehensive list of the most widely used candlestick patterns and perform a comprehensive performance test of all these patterns. However, they conclude that candlestick patterns are not profitable and have no forecasting power. The candlestick patterns tested by MYR are between one and five candles. This work uses candlestick images with 28 candles. If successful, this work gives support to statistical based trading strategies, and the use of candlestick patterns.

Computer Science Literature Review: Financial Markets Predictions

Selvin, Menon, et al. (2017) propose a deep learning based architecture capable of capturing hidden dynamics to make stock market predictions [1]. Their results show evidence that the CNN is capable of identifying changes in stock market trends. Additionally they compare their CNN to a Long short-term memory network (LSTM). For their proposed methodology a CNN is identified as the best model, as it is capable of identifying inter-relation within the data.

In a related study, by Tsai, Chen, Jun-Hao and Wang 2019 [2] it is shown that traders often decide with news, fundamentals, and technical indicators, but still use their vision and experience. A CNN with candlestick images as inputs is chosen as the best tool to model a trader's judgement. The candlestick images are preprocessed using Gramian Angular Summation Fields encoding (GASF). This format normalizes the data, reduces noise, and preserves temporal relationships. Their work cites Yang, Chen, and Yang 2019 [4] who encode time series data with three separate encoding methods, and GASF is found to be optimal. The CNN is trained and tested on EURUSD data from 2017. The output is a buy or sell signal. The model is trained on nine months of data, tested on 3 months of data, and yields an accurancy of 88% for picking the correct signal. Unlike Selvin (2017), this paper focuses on the best type of image input data, but also contributes to the overall technique that a CNN can predict financial market prices.

Kim, Taewook, Kim, and Ha Young (2019) [3] propose a fusion LSTM which combines both numeric features and image inputs. Inputs include price informa-

tion, time, and multiple stock chart images to predict stock prices. Candlestick images, bar charts, and moving average line plots are all input as images into the LSTM. It is found that candlestick chart performs the best since it has more information compared with other stock chart images, and produces the highest classification accuracy of 91%. Kim et al. (2019) differs from Tsai (2019) and Selvin (2017), by using image and numeric features as CNN input.

These previous works demonstrate the ability of a CNN to make accurate predictions in financial markets. Additionally they show candlestick images work better than other image representations of the market. However they do not perform analysis as to how the CNN trades. Rather they focus on various techniques to improve model accuracy. Also, they claim to be able to predict financial markets, yet make no performance comparisons to the market itself. This work not only compares performance to the S&P 500 Index, but it utilizes feature maps to visualize what the CNN sees, by recreating images from the weights of the neural network. This is a novel contribution.

Computer Science Literature Review: Feature Map Visualizations

Feature map visualizations are used to interpret what and how a CNN has learned. The Google Brain Team research on feature visualization, notably Olah, Mordvintsev, and Schubert (2017) [9] utilizes neural network weight reconstruction to generate images. They note, in the quest to make neural networks interpretable, feature visualization stands out as one of the most promising and developed research directions.

The Google Brain Team argues that there is a growing sense that neural networks need to be interpretable to humans. If we want to understand the individual features, we can search for examples where they have high values. If we want to understand a layer as a whole, we can use the DeepDream objective, searching for images the layer finds "interesting". Images generated by DeepDream are input, and the neurons whose activation function fired, are used to reconstruct images based on the weights. If a tree looks somewhat like a cat, the neurons which fire to classify a cat, are used to reconstruct a new image which appears like the combination of a tree and cat. While Google Deep Dream is largely used to create psychedelic images, the way it works through feature map visualization techniques, can be used for analysis of why images are classified as they are. It is seen as one of the fundamental building blocks that combined with additional tools will empower humans to understand these systems.

Along with the Google Brain Team, Zeiler and Fergus 2014 [5] are among the pioneers of feature visualization research. They argue that large CNNs have recently demonstrated impressive classification performance. However, there is no clear understanding of why they perform so well, or how they might be

improved. In their paper they explore both issues. They introduce a novel visualization technique that gives insight into the function of intermediate feature layers and the operation of the classifier.

Zeiler and Fergus (2014) introduce a feature map visualization technique that reveals the input stimuli that excite individual feature maps. It also allows the observing of the evolution of features during training to diagnose potential problems with the model. Feature activations are put back into the input pixel space and re-inserted into the network. This process reveals which parts of the scene are important for classification.

In a similar study, Grun, Rupprecht, Navab and Tombari (2016) [6] acknowledge that feature visualization is a very young area of research, that begins in 2013 (see Zeiler and Fergus (2013) [5]), and Simonyan et al. (2014) [32]. They divide feature visualization into three areas: (1) input modification methods, (2) deconvolutional methods, and (3) input reconstruction methods. Input reconstruction and modification methods refer to augmenting the input images for either analysis or re-inputting into the network. Deconvolutional methods refer to generating images from the weights of the network itself for analysis.

Additionally Grun et al. (2016) introduce their own library "FeatureVis library for MatConvNet": an extendable, easy to use open source library for visualizing CNNs. This libarary contains implementation from each of the three main classes of visualization methods. This allows feature visualizations to be created and compared from the most widely used CNN classifiers. This method gives insight into how the most popular networks classify images.

Like Zeiler and Fergus (2014) and Grun et al. (2016), Springenberg, Tobias, Dosovitskiy et al. (2014) develop their own method of feature visualization [33]. They create a deconvolutional network (decovnet) to visualize concepts learned by neurons in the layers of the neural network. They propose a new method of visualizing the representations learned by the layers of a convolutional network. This method produces more descriptive images than previously know methods. Using this method, an image is reconstructed to show the part of the input image that produces the most neuron activation output. Thus, allowing the researcher to determine which part(s) of an input image are causing neuron activation functions to fire, and shed light on why the network outputs certain values. In effect, their method produces tiny bits of images that would cause the neurons in the network to activate. Comparing these tiny bits of images to the input images gives researchers an idea on what parts of the input image would cause neurons to activate.

Similar to the Google Deep Dream approach, Dosovitskiy, and Brox (2016) [34] propose a new method to study image representations, and input these images into a CNN. The images are reconstructed from the CNN weights, to create representations. Starting with an input image of an apple, the apple image is

altered to be blue. The blue apple is mis-classified as a croquet ball. The image is input into the network and the activated neurons are used to reconstruct a new image in an attempt to see what the activated neurons were looking for. The reconstructed images appear to be more like a ball, than an apple. Including a ball that appears to be in a green field. These reconstructed images give insight into why an image might have been mis-classified.

Other early feature visualization methods made strong contributions, including these by Donahue, Jia, Yangqing, et al. (2014) [35] and Yu, Yang, Bai, Yalong, et al. (2014) [36]. Donahua et al. (2014) offer a unique approach by extracting feature visualizations from a deep convolutional network, and then inputting those visualizations back into the network. Thus allowing the CNN to learn visually what segment of the image the neurons are activating on. Yu, Yang et al. (2014) admit interesting to the black-box nature on CNNs. They assert that rather than continually increasing with deeper and deeper CNN architectures, understanding the internal work mechanisms is crucial to understanding how a CNN learns. The layers of the CNN are visualized through usage of feature map visualizations. They use this technique to compare the widely used image classification CNNs: VGG and AlexNet.

Nguyen, Dosovitskiy, Clune, et al. (2016) and Nguyen et al. (2019) develop feature visualizations methods which reveal the regions in the input image cause neuron activation output, or neuron excitement [7]. They introduce a method to produce visualizations from a neural network that are synthesized from scratch. Improving our ability to understand which features a neuron has learned to detect. Not only do the images closely reflect the features learned by a neuron, but in addition they are visually interesting.

Nguyen, Yosinski, Clune, et al. (2019) [8] continued their 2016 work by implementing a feature activation technique which reconstructs images based on neuron activation for multiple layers throughout the neural network with the addition of image priors. In addition to the training images synthetic image priors are generated by augmenting training images, with the activation maximization on those images. Subsequently, when an input image is fed into the network the activation on the new image is combined with the image priors to yield an output image. This technique is able to produce a generated image that is very clear. Rather than viewing psychedelic generated images that still require some interpretation, these images generated with image priors are more clear and easier to see why an image was classified a certain way. They conclude that activation maximization techniques enable us to shine light into black-box neural networks. Improving activation maximization techniques aides our ability to understand deep neural networks.

These methods to examine a neural network through feature map visualizations are powerful. However, this work requires the ability to measure neuron excitement. Therefore, a similar approach to Nguyen, Yosinski, Clune, et al.

(2019) is chosen since it gives the ability make precise measurements to changes in neuron excitement. Input images of candlesticks are combined with the neuron activation output, or neuron excitement, and used to create 2D arrays representing the neuron excitement.

2.4 Experiment

To experiment whether an RL system can outperform the S&P 500 Index, and analyze how this is achieved, the following tests are conducted:

Test 1: An RL system utilizing a DDQN will yield higher returns than the S&P 500 Index during the testing period Jan 2, 2020 through Jun 30, 2020.

This is tested by making direct comparisons to the S&P 500 Index returns, and conducting t tests on the results to verify the consistency of the DDQN performance.

Test 2: An RL system utilizing a DDQN will demonstrate the ability to switch its attention from using all price history in 28 days of candlesticks in a candlestick image, to the most recent candles.

This is tested by generating feature map visualizations and analyzing the neuron excitement levels in the candlestick images. Regression tests are conducted to verify the change in neuron excitement in the most recent days of the candlestick image. The corona stock market crash will be used to test whether a DDQN can switch its attention based on a significant event.

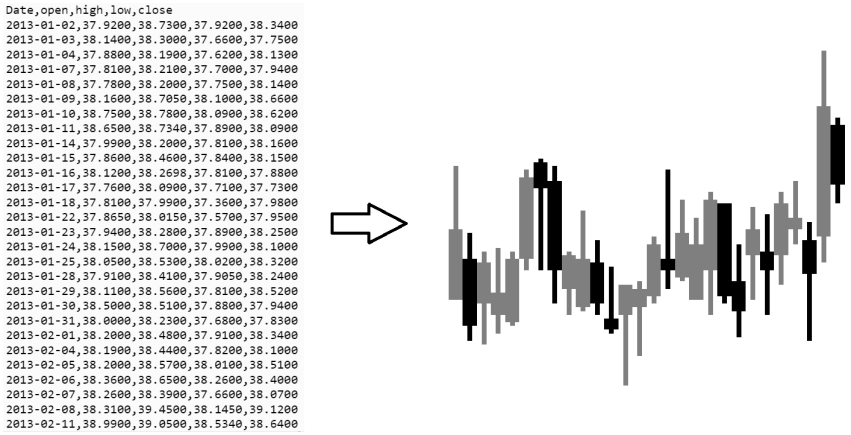
An additional test is run to determine if the ability of a DDQN to yield higher returns than the S&P 500 Index, is due to its ability to switch its attention to various parts of an input image. Logistic regressions are run with the sign of daily returns as the dependent variable, and change in neuron excitement as the independent variable. This analysis will provide insight into how neuron excitement can predict positive returns.

2.5 Methods

Data

The data is pre-processed by creating custom candlestick images. To simplify the images and reduce erroneous information the body of each candle, representing open and close prices, is three pixels wide. The stick representing the high and low prices is one pixel wide. For preliminary results, each candlestick image represents 28 days of prices. A study analyzing the different number of candles per image, can be found in the results section. Gray candles indicate an upward price movement, and black candles indicate a downward price movement. The candlestick are generated for 30 stocks for the training dataset Jan,

2 2013 to Dec 31, 2019, and the testing dataset Jan 1, 2020 to Jun 30, 2020. An example candlestick image is shown in Figure 5.



```
Date,open,high,low,close
2013-01-02,37.9200,38.7300,37.9200,38.3400
2013-01-03,38.1400,38.3000,37.6600,37.7500
2013-01-04,37.8800,38.1900,37.6200,38.1300
2013-01-07,37.8100,38.2100,37.7000,37.9400
2013-01-08,37.7800,38.2000,37.7500,38.1400
2013-01-09,38.1600,38.7050,38.1000,38.6600
2013-01-10,38.7500,38.7800,38.0900,38.6200
2013-01-11,38.6500,38.7340,37.8900,38.0900
2013-01-14,37.9900,38.2000,37.8100,38.1600
2013-01-15,37.8600,38.4600,37.8400,38.1500
2013-01-16,38.1200,38.2698,37.8100,37.8800
2013-01-17,37.7600,38.0900,37.7100,37.7300
2013-01-18,37.8100,37.9900,37.3600,37.9800
2013-01-22,37.8650,38.0150,37.5700,37.9500
2013-01-23,37.9400,38.2800,37.8900,38.2500
2013-01-24,38.1500,38.7000,37.9900,38.1000
2013-01-25,38.0500,38.5300,38.0200,38.3200
2013-01-28,37.9100,38.4100,37.9050,38.2400
2013-01-29,38.1100,38.5600,37.8100,38.5200
2013-01-30,38.5000,38.5100,37.8800,37.9400
2013-01-31,38.0000,38.2300,37.6800,37.8300
2013-02-01,38.2000,38.4800,37.9100,38.3400
2013-02-04,38.1900,38.4400,37.8200,38.1000
2013-02-05,38.2000,38.5700,38.0100,38.5100
2013-02-06,38.3600,38.6500,38.2600,38.4000
2013-02-07,38.2600,38.3900,37.6600,38.0700
2013-02-08,38.3100,39.4500,38.1450,39.1200
2013-02-11,38.9900,39.0500,38.5340,38.6400
```

Figure 5: Daily high, low, open, and close stock prices are converted to a candlestick image. Gray candles indicate an upward price movement, and black candles indicate a downward price movement.

Double Deep Q-Network

A reinforcement learning system is used with a Double Deep Q-Network for learning. The two neural networks of the DDQN are CNNs implemented in Tensorflow [37]. The architecture of the CNN consists of: input layer (84 x 84 pixels), convolutional layer (128 neurons), a second convolutional layer (256 neurons), a third convolutional layer (512 neurons), and an output layer (3 neurons for outputs long, short, or no position). The DDQN Target network is used for training on the candlestick images, and an Evaluation network for producing actions that are sent to the RL environment, implemented as an OpenAI Gym. The Evaluation network's weights are not adjusted throughout training. The Evaluation network weights are copied from the Target network every 100 steps. The RL system training workflow is show in Figure 6. The DDQN is able to fit the training data, as shown in Figure 7

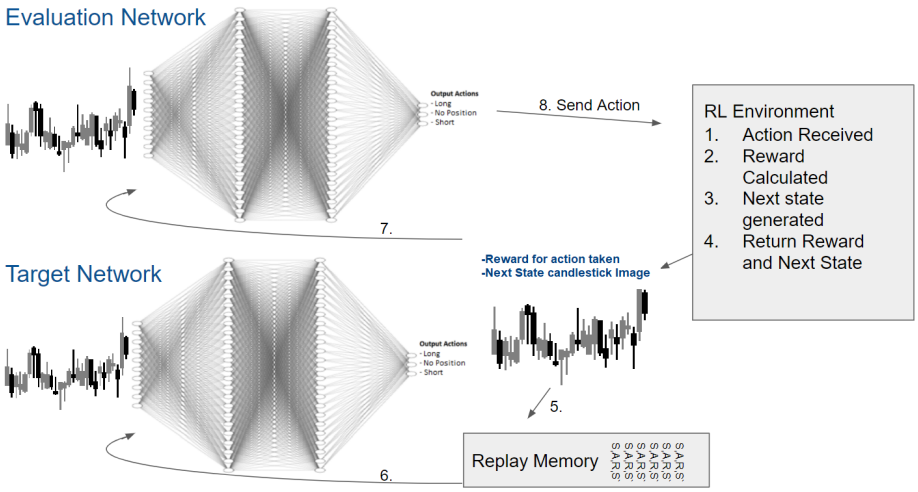


Figure 6: DDQN structure and workflow with RL Environment. 1. An action is received by the RL environment. 2. The reward is calculated. 3. the next state candlestick image, is generated. 4. The reward and next state are returned by the RL Environment. 5. The next state is stored in the Replay memory. 6. The replay memory stores 1000 previous state, action, reward, next state observations. The target network is trained using these randomly selected previous states. Every 100 steps,the target network weights are copied to the evaluation network. 7. The next state is also directly inputted into the evaluation network. 8. The next action is outputted by the DDQN and sent to the RL Environment.

The DDQN training rewards in the OPENAI Gym are calculated as follows:

$$T = a \times r \times N$$

Where:

T: Training rewards
a: action output by DDQN, action = {1, -1, 0}
r: daily returns
N: Negative Rewards Multiplier

The action space of {1, -1, 0} represents a long, short, or no position. Negative Rewards Multiplier (NRM) is used for training to increase the ability of the DDQN to take a no position action. The Negative Rewards Multiplier is a variable used to assist in training, introduced by Brim (2020) [38]. It is a constant that is multiplied, to the returns supplied from the environment. Training rewards are different from returns. Training rewards are received by the DDQN from the environment, for learning. Returns are used to calculate training rewards, and also used to compare the performance of the DDQN to the returns of the S&P 500 Index. It is used to decrease training time, as the CNN learns more quickly to output no position during training. The CNN learns to take

long or short position in spite of a penalty for an incorrect output. It has no bearing on a better fit of the CNN, it simply decreases training time. Figure 7 shows the DDQN training curves for 10 of the 30 stocks. It can be seen that after approximately 400 training episodes the DDQN begins to consistently receive a positive reward form the environment. 1000 training episodes is selected as both a high enough number of training episodes to fit a CNN which is able to perform well, and also not so high as to over fit the CNN to the training data set. The training rewards on the final episode, for the 30 stocks, are shown in Table 1.

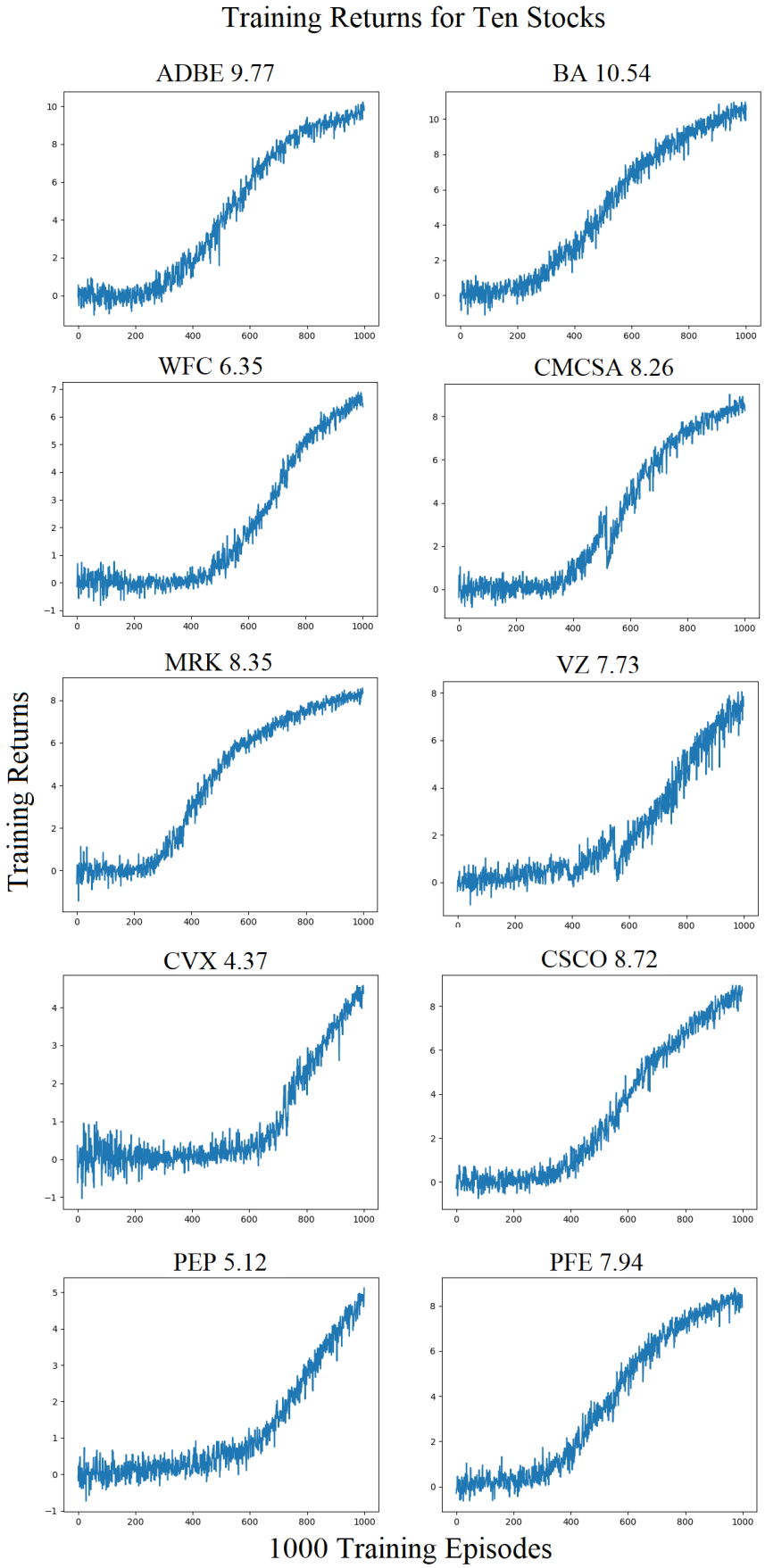


Figure 7: Each DDQN is able to fit a function on the training data. The training data consists of candlestick images representing stock market prices from Jan 01, 2013 through Dec 31, 2019. Final episode training rewards are shown above each training curve.

ADBE	9.77	XOM	7.78
BA	10.54	UNH	12.39
WFC	6.35	T	7.58
CMCSA	8.26	INTC	6.64
MRK	8.35	BAC	9.78
VZ	7.73	PG	6.99
CVX	4.37	MA	9.2
CSCO	8.72	JPM	10.77
PEP	5.12	VZ	9
PFE	7.94	WMT	10.18
KO	0	JNJ	9.16
DIS	7.87	BRK.B	9.62
HD	8.21	MSFT	10.1
AAPL	10.2	AMZN	17.03
GOOGL	13.9	FB	18.5

Table 1: DDQN training rewards on final episode, for 30 stocks.

After training and testing, the Evaluation network is used to generate feature map visualizations for analysis as shown in Figures 8 and 9. The OpenAI Gym Environment [24] provides a candlestick image as input into the Evaluation Network. The weights are used to create feature map visualizations which are analyzed to determine which regions of the input image are generating neuron excitement.

Feature Map Visualizations

Feature map visualizations are generated from the input image and the regions of neuron excitement on the input image as show on the far right image in Figure 8. Neuron excitement is the activation output on the input image, and can be seen and analyzed in a feature map visualization. A similar approach to Nguyen, Yosinski, Clune, et al. (2019) is used in this work. Input images of candlesticks are combined with a trained CNN and used to create a 2D array of the neuron excitement.

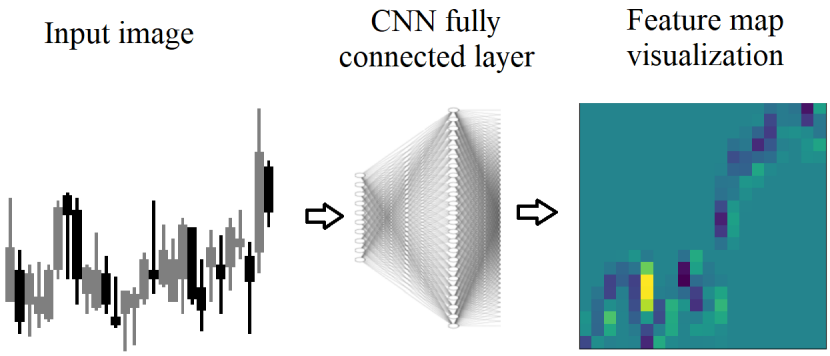


Figure 8: Feature map visualizations are generated from fully connected layers in the DDQN. A candlestick image is input, and each neuron in the fully connected layer receives the image. The neuron is excited on various parts of the image. The neuron is excited on various parts of the image.The excited regions are stored as a 2D array, and shown here as a heatmap. The region in yellow indicates the highest neuron excitement value on the image.

To generate a feature map visualization, a new network is constructed with only two layers: an input layer for the input image, and a layer constructed from the weights of the fully connected layer. It would also be possible to start with a trained network and remove layers. However, utilizing Tensorflow and its built in functions, it is more suited to creating a new network and adding the layers. The input image is passed into the second layer and the neuron excitement is generated and stored in a 2D array. A feature map is generated for every neuron. In order to investigate the neuron excitement level, the highest excitement regions for each feature map are summed, as shown in the bar charts of Figure 9.

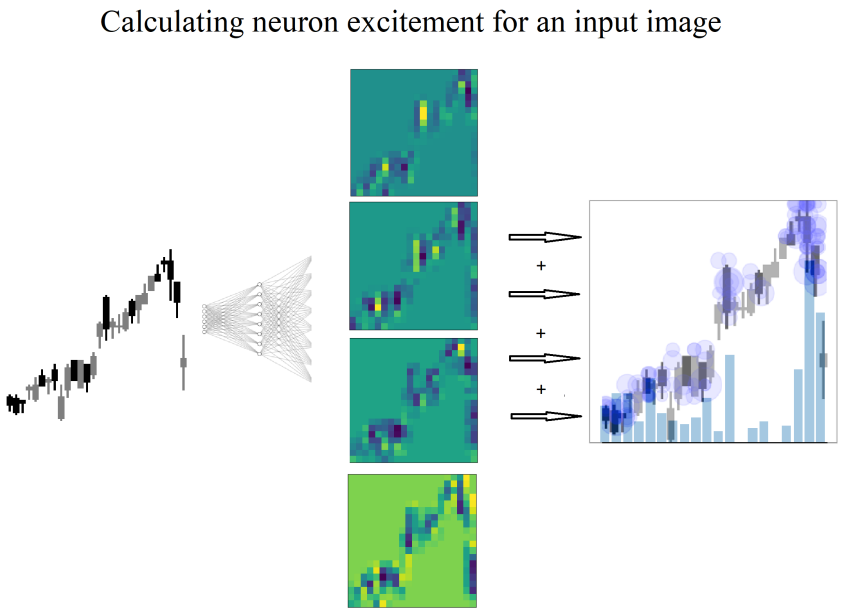


Figure 9: The highest level of neuron excitement, shown in yellow, for each the feature map generated from each neuron, is summed revealing the highest levels of neuron excitement by day for a single candlestick image. The size of the blue dot corresponds to the neuron excitement value. The darkness of the blue dots is caused by multiple dots overlapping. This indicates multiple neurons have produced neuron excitement on the same region. Dark blue dot clusters indicate high levels of neuron excitement. The level of neuron excitement can be seen by the overlaid blue bar chart.

2.6 Results and Discussion

DDQN Returns vs the S&P 500 Results

The DDQN testing results on 30 stocks yield an average of 13.2% geometric returns in 124 trading days from January 2, 2020 through June 30, 2020. The stocks with the top five geometric returns are Walmart (57.7%), Adobe (47.5%), Pepsi Co. (44.7%), Wells Fargo (34.1%) and JPMorgan (28.7%). The S&P 500 Index geometric returns during this same time yield -4% as show in Figure 10.

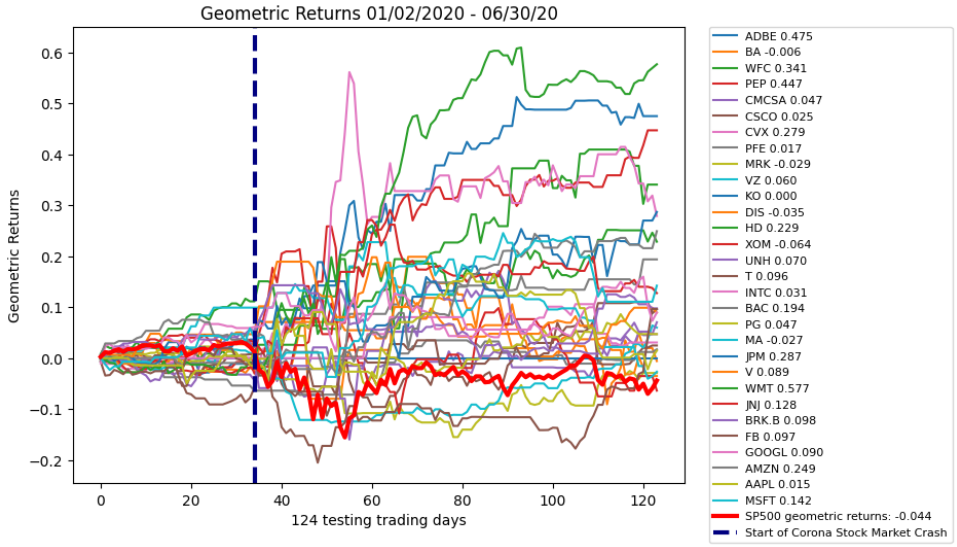


Figure 10: Testing geometric returns for 30 stocks. The average geometric returns is 13.2%. All but one (Exxon Mobile -6%) yield higher geometric daily returns than the S&P 500 Index. S&P 500 Index geometric daily returns are shown in the bold red line. S&P 500 Index geometric daily returns for the testing data set are -4%.

To verify the DDQN is able to yield higher returns than the S&P 500 Index, a T test is conducted between the geometric returns of the DDQN on testing day 124 and the geometric returns of the S&P 500 Index. The T value of -5.95 and near zero p-value of 0.0000018 clearly reject the null hypothesis that the population means are the same, indicating the difference in the DDQN returns and S&P 500 Index returns is statistically significant. This gives strong statistical evidence to support Test 1.

To further analyze the ability of the DDQN to yield higher returns than the S&P 500 Index, 20 day cross sectional t tests are conducted on daily returns of the DDQN, and the daily returns of the S&P 500 Index. The daily returns, rather than the geometric returns, allow for a day to day evaluation of the DDQN performance. Figure 11 shows the p-values of the cross sectional T tests plotted on the geometric returns. It is clear that the days immediately following the corona stock market crash are where the DDQN yields the highest returns over the S&P 500 Index. This is also the same time when the neuron excitement of the DDQN increases, as discussed in the next section.

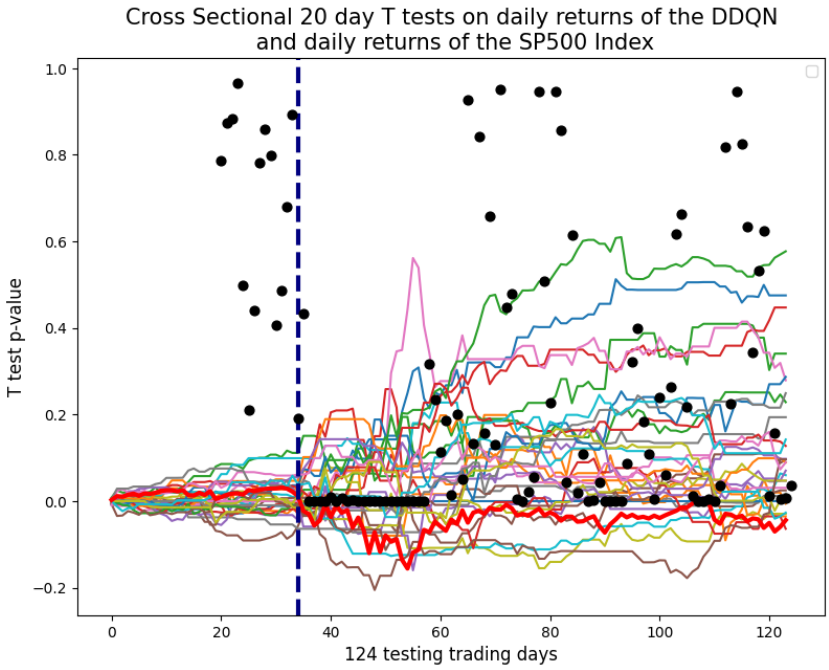


Figure 11: Cross Sectional 20 day T tests on daily returns of the DDQN, minus daily returns of S&P 500 Index. Each black dot is the 20 day T test p-value. The line of 22 black dots indicates that during the 22 day period following the corona stock market crash, the daily returns of the DDQN are statistically significant and different than the daily returns of the S&P 500 Index.

DDQN Feature Map Visualizations Analysis Results

To determine if a DDQN can switch from using all days in a candlestick image to the most recent days, feature map visualizations are used to measure the areas of neuron excitement in candlestick images. The re-shaping of the neural network transforms the 28 candles in the candlestick image, into 20 regions. The neuron excitement values of these 20 regions is measured and analyzed. This analysis will show what part of the image the DDQN is placing attention. Additionally it can be determined if the attention is switching based on an event.

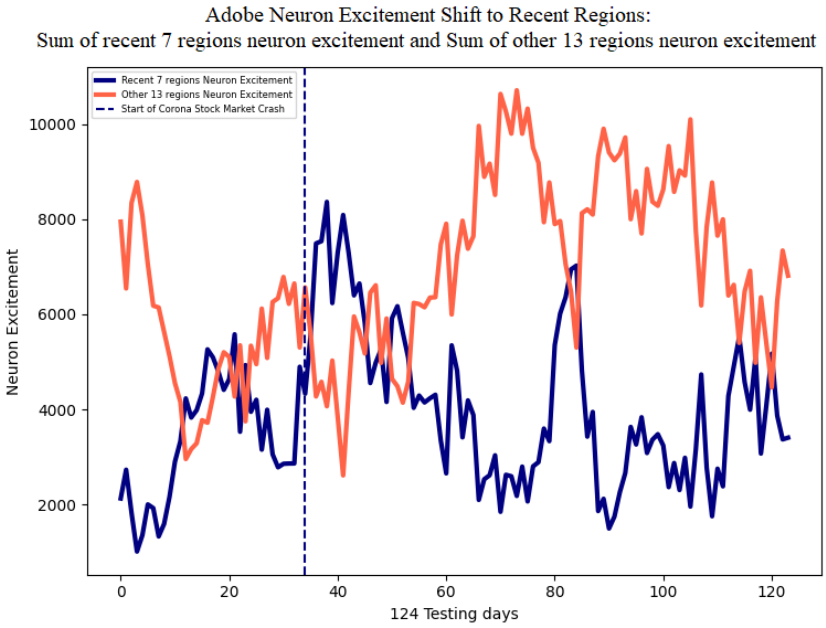


Figure 12: Adobe sum of neuron excitement on the recent seven regions, and the other 13 regions. Following the corona stock market crash, neuron excitement increases on the most recent seven regions and decreases on the other 13 regions.

Figure 12 illustrates Adobe sum of neuron excitement for the seven recent regions in the candlestick image in blue, and the other 13 regions in red. The average sum of neuron excitement level for the seven recent regions is 3873.07, and the other 13 regions is 6825.05. The greater average of the non-recent 13 regions is expected since these regions constitute 85% more area of the image. On day 34 of the testing data, Feb 20, 2020, the corona stock market crash begins. This event is followed by an increase in neuron excitement in the most recent seven candles and a decrease in the other 21 regions. The recent seven regions reaches a value of 8366 and the other 13 regions decreases to a value of 2610.99, indicating the DDQN is shifting its attention from all regions to the most recent regions.

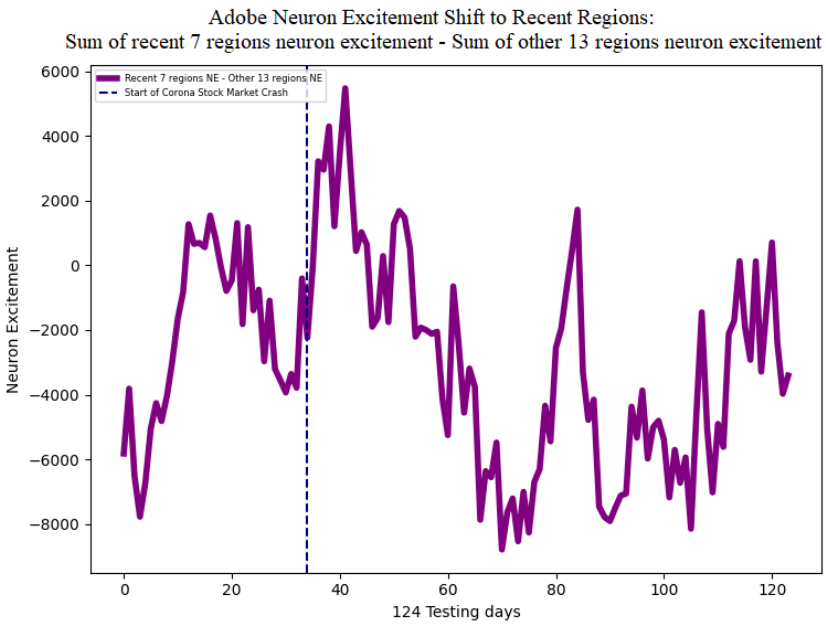


Figure 13: Adobe neuron excitement 7 recent regions - 13 older regions. Following the corona stock market crash the sum of neuron excitement increases on the 7 recent regions and decreases on the 13 other regions.

The single line in Figure 13 shows the difference of the sum of neuron excitement of the seven recent regions neuron excitement minus the other 13 regions for Adobe. This metric indicates a shift in neuron excitement from all regions in the image, to the most recent regions. The average difference is -2951.98. The negative value is expected since the average sum of neuron excitement of the other 13 regions is greater than the seven recent regions. Following the corona stock market crash the difference in neuron excitement increases to a maximum value of 5480.27. This behavior of the DDQN shifting attention to the recent regions in the image can also be seen in the average of all 30 stocks. Figure 14 shows the difference of the sum of neuron excitement of the seven recent regions neuron excitement minus the other 13 regions for 30 stocks. The average neuron excitement difference between the seven recent regions and the other 13 is -2643.71. Following the corona stock market crash the difference in neuron excitement increases to a maximum value of 3184.71.

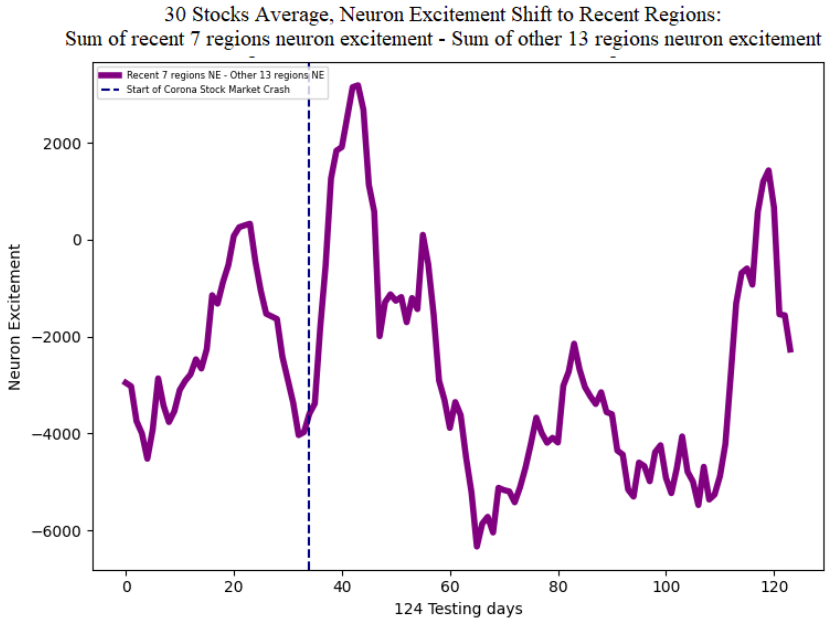


Figure 14: 30 stocks average neuron excitement 7 recent regions - 13 other regions. Following the corona stock market crash neuron excitement increases on the 7 recent regions and decreases on the 13 other regions, on average for the 30 preliminary stocks.

This shift in neuron excitement following the corona stock market crash, for 30 stocks, shows the ability of the DDQN to switch its attention. To further analyze this behavior, and determine more exactly which regions in the image are involved, regression tests are run to statistically verify this ability of a DDQN The use of dummy variables in testing for equality between sets of coefficients in linear regressions methodology is presented in Gujarati (1970) [39]. This method allows a set of regression coefficients to be applied to each of the 20 regions in the image. The coefficients will allow for greater sensitivity testing for changes in neuron excitement. The dummy variables, of value 0 or 1, are multiplied to the coefficients in the regression. A dummy variable trap is possible here, since the regions of neuron excitement for 19 regions can be predictive of the other 1 region. To solve the dummy variable trap, one region is removed as outlined in Hirschberg (2001) [40]. Comparing the coefficients for the entire testing data set, to the 22 days following the stock market crash, will reveal the changes in neuron excitement that occur following the crash.

The first regression is run with neuron excitement as the dependent variable and the neuron excitement levels of each region as the independent variables. Dummy variables are included as coefficients to determine which regions are

effecting neuron excitement. Region 1 is removed from the regression to avoid the dummy variable trap. Region 20 is the most recent region in the candlestick image, and region 2 is the oldest region. The first regression is run as follows:

$$NeuronExcitement_t = \alpha + Z_t\beta_2 NE.2_t + Z_t\beta_3 NE.3_t + Z_t\beta_4 NE.4_t + Z_t\beta_5 NE.5_t + Z_t\beta_6 NE.6_t + Z_t\beta_7 NE.7_t + Z_t\beta_8 NE.8_t + Z_t\beta_9 NE.9_t + Z_t\beta_{10} NE.10_t + Z_t\beta_{11} NE.11_t + Z_t\beta_{12} NE.12_t + Z_t\beta_{13} NE.13_t + Z_t\beta_{14} NE.14_t + Z_t\beta_{15} NE.15_t + Z_t\beta_{16} NE.16_t + Z_t\beta_{17} NE.17_t + Z_t\beta_{18} NE.18_t + Z_t\beta_{19} NE.19_t + Z_t\beta_{20} NE.20_t + \epsilon$$

Where:

- $NeuronExcitement_t$: Neuron excitement sum for all regions in a candlestick image at day t
- $NE.Region_t$: Neuron excitement value for that region in the candlestick image at day t
- Z_t : Dummy variable, 1 for each region in candlestick image, 0 otherwise.

The second regression is run to compare the coefficients of the overall testing data set, to the 22 days following the corona stock market crash. An additional dummy variable is included for the 22 days following the corona stock market crash. The 22 days following the corona stock market crash, are chosen for this test, since this is the region where the cross sectional T tests yield near 0 p-values. The coefficients in the second regression will indicate whether the recent regions have a greater effect on neuron excitement, following the corona stock market crash. The second regression is run as follows:

$$NeuronExcitement_t = \alpha + Y_tZ_t\beta_2 NE.2_t + Y_tZ_t\beta_3 NE.3_t + Y_tZ_t\beta_4 NE.4_t + Y_tZ_t\beta_5 NE.5_t + Y_tZ_t\beta_6 NE.6_t + Y_tZ_t\beta_7 NE.7_t + Y_tZ_t\beta_8 NE.8_t + Y_tZ_t\beta_9 NE.9_t + Y_tZ_t\beta_{10} NE.10_t + Y_tZ_t\beta_{11} NE.11_t + Y_tZ_t\beta_{12} NE.12_t + Y_tZ_t\beta_{13} NE.13_t + Y_tZ_t\beta_{14} NE.14_t + Y_tZ_t\beta_{15} NE.15_t + Y_tZ_t\beta_{16} NE.16_t + Y_tZ_t\beta_{17} NE.17_t + Y_tZ_t\beta_{18} NE.18_t + Y_tZ_t\beta_{19} NE.19_t + Y_tZ_t\beta_{20} NE.20_t + \epsilon$$

Where:

- $NeuronExcitement_t$: Neuron excitement sum for all regions in a candlestick image at day t
- $NE.Region_t$: Neuron excitement value for that region in the candlestick image at day t
- Z_t : Dummy variable, 1 for each region in candlestick image, 0 otherwise
- Y_t : Dummy variable, 1 for 22 days following corona stock market, 0 otherwise

Table 2 shows regression coefficients for regions 2 through 20 range from 0.0440 to 0.0518 indicating a slight increase in attention toward the recent regions for the overall test data set. A large non-zero F-statistic of 15.75 indicates

volatility and significance among the coefficients. Table 3 shows the regression coefficients for the 22 days following the corona stock market crash. Regions 2 through 20 range from 0.0278 to 0.0776. A large non-zero F-statistic of 379.0 indicates volatility and significance among the coefficients following the corona stock market crash. Coefficients for neuron excitement increase for regions 9 through 20, and decrease for regions 2 through 8 as shown in Table 4. Region 19 has the greatest increase in coefficient from 0.055 to 0.0822. Figure 15 displays the values of Column 4 from Table 4.

It can be seen from the change in coefficients, the 22 days following the corona stock market crash, that regions 10 through 20 increase while regions 2 through 9 decrease. This indicates the neuron excitement in the 10 recent regions has a greater effect on total neuron excitement, than the older regions. The single region that most effects overall neuron excitement is region 19, representing the candles from two days ago. Both analysis of neuron excitement values, and change in regression coefficients provide evidence that a DDQN is able to switch its attention. However, these metrics do not yet confirm if the ability of DDQN to switch attention, is the reason the DDQN is able to outperform the S&P 500 Index. This is pursued in the following section.

Dep. Variable:	y	R-squared:	0.004
Model:	OLS	Adj. R-squared:	0.004
Method:	Least Squares	F-statistic:	15.75
No. Observations:	70680	Prob (F-statistic):	1.56E-49
Df Residuals:	70661	Log-Likelihood:	1.11E+05
Df Model:	18	AIC:	-2.22E+05
Covariance Type:	nonrobust	BIC:	-2.22E+05

	coef	std err.	t	P>|t|	[0.025	0.975]
x1	0.044	0.001	53.288	0	0.042	0.046
x2	0.0465	0.001	56.223	0	0.045	0.048
x3	0.0479	0.001	57.962	0	0.046	0.05
x4	0.0465	0.001	56.305	0	0.045	0.048
x5	0.0472	0.001	57.048	0	0.046	0.049
x6	0.0479	0.001	57.977	0	0.046	0.05
x7	0.0479	0.001	57.919	0	0.046	0.049
x8	0.0491	0.001	59.375	0	0.047	0.051
x9	0.0495	0.001	59.882	0	0.048	0.051
x10	0.049	0.001	59.286	0	0.047	0.051
x11	0.0527	0.001	63.763	0	0.051	0.054
x12	0.0532	0.001	64.408	0	0.052	0.055
x13	0.0523	0.001	63.326	0	0.051	0.054
x14	0.0532	0.001	64.412	0	0.052	0.055
x15	0.0541	0.001	65.486	0	0.053	0.056
x16	0.0533	0.001	64.466	0	0.052	0.055
x17	0.0547	0.001	66.155	0	0.053	0.056
x18	0.055	0.001	66.592	0	0.053	0.057
x19	0.0518	0.001	62.682	0	0.05	0.053

Table 2: Regression results for neuron excitement. Region 20 is the most recent region, and region 2 is the oldest region. Dummy variables are used to isolate the effect of neuron excitement for each region. Region 1 is removed to avoid the dummy variable trap. The largest coefficient is region 19 (x18), the second most recent region, with a value of 0.055.

Dep. Variable:	y	R-squared:	0.092
Model:	OLS	Adj. R-squared:	0.092
Method:	Least Squares	F-statistic:	379
No. Observations:	70680	Prob (F-statistic):	0.00
Df Residuals:	70661	Log-Likelihood:	8.98E+04
Df Model:	19	AIC:	-1.80E+05
Covariance Type:	nonrobust	BIC:	-1.79E+05

	coef	std err.	t	P>|t|	[0.025	0.975]
x1	0.0278	0.003	10.037	0	0.022	0.033
x2	0.0308	0.003	11.109	0	0.025	0.036
x3	0.0316	0.003	11.388	0	0.026	0.037
x4	0.0329	0.003	11.872	0	0.027	0.038
x5	0.0344	0.003	12.387	0	0.029	0.04
x6	0.0331	0.003	11.952	0	0.028	0.039
x7	0.0328	0.003	11.846	0	0.027	0.038
x8	0.0429	0.003	15.469	0	0.037	0.048
x9	0.0501	0.003	18.052	0	0.045	0.055
x10	0.0519	0.003	18.717	0	0.046	0.057
x11	0.0557	0.003	20.088	0	0.05	0.061
x12	0.0553	0.003	19.958	0	0.05	0.061
x13	0.0583	0.003	21.02	0	0.053	0.064
x14	0.0603	0.003	21.737	0	0.055	0.066
x15	0.0638	0.003	23.015	0	0.058	0.069
x16	0.724	0.003	26.103	0	0.067	0.078
x17	0.767	0.003	27.652	0	0.071	0.082
x18	0.0822	0.003	29.655	0	0.077	0.088
x19	0.0776	0.003	27.978	0	0.72	0.083

Table 3: Regression results for the second regression, the 22 days following the corona stock market crash. Coefficients for neuron excitement increase for regions 9 through 20, and decrease for regions 2 through 8. The largest coefficient is region 19 (x18), the second most recent region, with a value of 0.0822. An increase of 0.0272 from the overall testing data.

	A Neuron Excitement Region Coefficients All Testing Data	B Neuron Excitement Region Coefficients Corona Crash	Column B - A
region2	0.044	0.0278	-0.0162
region3	0.0465	0.0308	-0.0157
region4	0.0479	0.0316	-0.0163
region5	0.0465	0.0329	-0.0136
region6	0.0472	0.0344	-0.0128
region7	0.0479	0.0331	-0.0148
region8	0.0479	0.0328	-0.0151
region9	0.0491	0.0429	-0.0062
region10	0.0495	0.0501	0.0006
region11	0.049	0.0519	0.0029
region12	0.0527	0.0557	0.003
region13	0.0532	0.0553	0.0021
region14	0.0523	0.0583	0.006
region15	0.0532	0.0603	0.0071
region16	0.0541	0.0638	0.0097
region17	0.0533	0.0724	0.0191
region18	0.0547	0.0767	0.022
region19	0.055	0.0822	0.0272
region20	0.0518	0.0776	0.0258

Table 4: Summary table of neuron excitement region regression coefficients. Column A consists of the regression coefficients for neuron excitement for all testing data. Column B consists of the regression coefficients for neuron excitement for the 22 days following the corona stock market crash. The third column shows the increase in regression coefficients following the corona stock market crash.

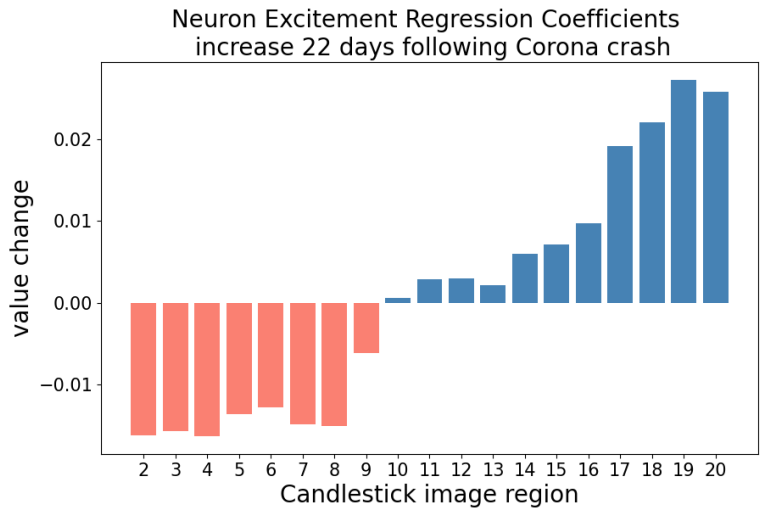


Figure 15: The increase in regression coefficients following the corona stock market crash. Values shown are Column B - Column A from Table 4. The neuron excitement regression coefficients for the recent regions increase while the older regions decrease. It is also notable that the neuron excitement regression coefficient with the highest increase is region 19. This region consists of the candles representing yesterday and two days ago.

Logistic Regressions, Returns and Neuron Excitement Results

To determine if the ability of a DDQN to outperform the S&P 500 Index is caused by its ability to switch attention from all candles in a candlestick image to the most recent, logistic regressions are run with daily returns sign as the dependent variable and change in neuron excitement as the independent variable. These regression tests will determine if positive returns are predicated by change in neuron excitement. The logistic regressions are run as follows:

$Returns.Sign_t = \alpha + \beta(NE.Rec_t - NE.Oth_t) + \epsilon$

Where:

$Returns.Sign_t$: Daily returns sign, 1, 0 or -1
$NE.Rec_t$: Neuron excitement recent regions
$NE.Other_t$: Neuron excitement non-recent regions

The change in neuron excitement to the most recent regions is calculated as: recent regions - other regions. 20 logistic regressions are run for every value of recent and other regions, as the independent variable as shown in Table 5. All p-values are near zero, and all Z statistics are greater than 2 or less than -2 indicating the logistic regressions are able to successfully fit a function with daily returns sign as the dependent variable, and change in neuron excitement as the independent variable. The two regressions with the highest coefficients are the

seven most recent regions at 1.5082 and six most recent regions at 1.48. This shows the change in neuron excitement in the most six or seven recent regions, is the strongest indicator of positive returns. The most negative coefficient is 13 most recent regions at -2.36. It does not indicate these regions can be used as a signal to short a stock. This simply means these regions are the least predictive to positive returns.

20 Logistic Regression Results

Num of Recent Regions	Num of Other Regions	coefficient	std err	Z statistic	p-value
1	19	0.0001	0.00	26.414	0
2	18	0.2996	0.012	25.81	0
3	17	0.6402	0.026	25.017	0
4	16	0.9739	0.04	24.064	0
5	15	1.1664	0.051	22.684	0
6	14	1.5082	0.074	20.49	0
7	13	1.4838	0.087	17.073	0
8	12	1.1176	0.098	11.436	0
9	11	0.4036	0.117	3.458	0.001
10	10	-0.7143	0.129	-5.528	0
11	9	-1.6547	0.128	-12.906	0
12	8	-2.2301	0.125	-17.788	0
13	7	-2.3634	0.113	-20.967	0
14	6	-2.2663	0.098	-23.062	0
15	5	-2.0877	0.086	-24.349	0
16	4	-1.9614	0.078	-25.214	0
17	3	-1.8147	0.07	-25.773	0
18	2	-1.664	0.063	-26.34	0
19	1	-1.6709	0.063	-26.682	0
20	0	-1.6098	0.06	-26.756	0

Table 5: Logistic regressions are run with the daily returns sign as the dependent variable and change in neuron excitement as the independent variable. The first and second columns show the most recent regions and other regions. Near-zero p-values and Z statistics greater than 2 or less than -2 indicate the regressions are able to successfully fit a function with daily returns sign as the dependent variable, and change in neuron excitement as the independent variable. The two regressions with the highest coefficients are the seven most recent regions at 1.5082 and six most recent regions at 1.48. Change in neuron excitement in the six or seven most recent regions of the candlestick image, is the best predictor of positive returns among those tested.

2.7 Conclusion

The results of Test 1 show the DDQN tested on the largest 30 stocks of the S&P 500, yield an average of 13.2% geometric returns in 124 trading days from January 2, 2020 through June 30, 2020. The S&P 500 Index returns during this

same time yield -4%. On average, the DDQN yields higher geometric returns than the S&P 500 Index, by 17.2% in six months. Cross sectional t tests shows the DDQN most out performs the S&P 500 Index 22 days following the coronavirus stock market crash.
The results of Test 2 show, through the use of feature map visualizations, that neuron excitement during the 22 days following the crash, increases on the recent regions in the candlestick image and decrease on the other regions. Results also show, through the use of dummy variables in testing for equality between sets of coefficients, that a DDQN is able to switch its attention from all days in a candlestick image to the most recent days. To test whether the ability of a DDQN to outperform the S&P500 Index is due to its ability to shift its attention, 20 logistic regressions are run. The results of the logistic regressions shown that changes in neuron excitement can forecast positive returns. In this experiment, the shift in neuron excitement to the recent 7 regions is the strongest predictor of positive returns. This work not only validates statistical based trading strategies, but also provides a successful use case for candlestick images.

References

[1] Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647. IEEE, 2017.

[2] Yun-Cheng Tsai, Jun-Hao Chen, and Chun-Chieh Wang. Encoding candlesticks as images for patterns classification using convolutional neural networks. *arXiv preprint arXiv:1901.05237*, 2019.

[3] Taewook Kim and Ha Young Kim. Forecasting stock prices with a feature fusion lstm-cnn model using different representations of the same data. *PloS one*, 14(2), 2019.

[4] Chao-Lung Yang, Zhi-Xuan Chen, and Chen-Yi Yang. Sensor classification using convolutional neural network by encoding multivariate time series as two-dimensional colored images. *Sensors*, 20(1):168, 2020.

[5] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[6] Felix Grün, Christian Rupprecht, Nassir Navab, and Federico Tombari. A taxonomy and library for visualizing learned features in convolutional neural networks. *arXiv preprint arXiv:1606.07757*, 2016.

[7] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks

via deep generator networks. In *Advances in neural information processing systems*, pages 3387–3395, 2016.

[8] Anh Nguyen, Jason Yosinski, and Jeff Clune. *Understanding Neural Networks via Feature Visualization: A Survey*, pages 55–76. Springer International Publishing, Cham, 2019.

[9] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

[10] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[11] Takafumi Okuyama, Tad Gonsalves, and Jaychand Upadhay. Autonomous driving system based on deep q learnig. In *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*, pages 201–205. IEEE, 2018.

[12] Yi Zhang, Ping Sun, Yuhan Yin, Lin Lin, and Xuesong Wang. Human-like autonomous vehicle speed control by deep reinforcement learning with double q-learning. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1251–1256. IEEE, 2018.

[13] Xuefeng Han, Hongwen He, Jingda Wu, Jiankun Peng, and Yuecheng Li. Energy management based on reinforcement learning with double deep q-learning for a hybrid electric tracked vehicle. *Applied Energy*, 254:113708, 2019.

[14] Van-Hai Bui, Akhtar Hussain, and Hak-Man Kim. Double deep q-learning-based distributed operation of battery energy storage system considering uncertainties. *IEEE Transactions on Smart Grid*, 11(1):457–469, 2019.

[15] Qingchen Zhang, Man Lin, Laurence T Yang, Zhikui Chen, Samee U Khan, and Peng Li. A double deep q-learning model for energy-efficient edge scheduling. *IEEE Transactions on Services Computing*, 12(5):739–749, 2018.

[16] Kai Li, Wei Ni, Eduardo Tovar, and Abbas Jamalipour. On-board deep q-network for uav-assisted online power transfer and data collection. *IEEE Transactions on Vehicular Technology*, 68(12):12215–12226, 2019.

[17] Hikaru Sasaki, Tadashi Horiuchi, and Satoru Kato. A study on vision-based mobile robot learning by deep q-network. In *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 799–804. IEEE, 2017.

[18] Yong Shi, Wei Li, Luyao Zhu, Kun Guo, and Erik Cambria. Stock trading rule discovery with double deep q-network. *Applied Soft Computing*, 107:107320, 2021.

[19] Yuming Li, Pin Ni, and Victor Chang. Application of deep reinforcement learning in stock trading strategies and stock forecasting. *Computing*, pages 1–18, 2019.

[20] Salvatore Carta, Anselmo Ferreira, Alessandro Sebastian Podda, Diego Reforgiato Recupero, and Antonio Sanna. Multi-dqn: An ensemble of deep q-learning agents for stock market forecasting. *Expert Systems with Applications*, 164:113820, 2021.

[21] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[22] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[23] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

[24] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[25] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[26] Eugene F Fama. The behavior of stock-market prices. *The journal of Business*, 38(1):34–105, 1965.

[27] Eugene F Fama. Efficient capital markets a review of theory and empirical work. *The Fama Portfolio*, pages 76–121, 2021.

[28] William Brock, Josef Lakonishok, and Blake LeBaron. Simple technical trading rules and the stochastic properties of stock returns. *The Journal of finance*, 47(5):1731–1764, 1992.

[29] Burton Malkiel. A random walk down wall street, nueva york, 1981.

[30] Spyros Skouras. Financial returns and efficiency as seen by an artificial technicaleditor mode. analyst. *Journal of Economic Dynamics and Control*, 25(1-2):213–244, 2001.

[31] Ben R Marshall, Martin R Young, and Lawrence C Rose. Candlestick technical trading strategies: Can they create value for investors? *Journal of Banking & Finance*, 30(8):2303–2323, 2006.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[33] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[34] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4829–4837, 2016.

[35] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[36] Wei Yu, Kuiyuan Yang, Yalong Bai, Hongxun Yao, and Yong Rui. Visualizing and comparing convolutional neural networks. *arXiv preprint arXiv:1412.6631*, 2014.

[37] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[38] Andrew Brim. Deep reinforcement learning pairs trading with a double deep q-network. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0222–0227. IEEE, 2020.

[39] Damodar Gujarati. Use of dummy variables in testing for equality between sets of coefficients in linear regressions: A generalization. *The American Statistician*, 24(5):18–22, 1970.

[40] Joe Hirschberg and Jenny Lye. The interpretation of multiple dummy variable coefficients: an application to industry effects in wage equations. *Applied Economics Letters*, 8(11):701–707, 2001.

CHAPTER 3

Pairs Trading - Arbitraging Causal Dynamics using Fuzzy Logic

DECISION SCIENCES INSTITUTE

Pairs Trading - Arbitraging Causal Dynamics using Fuzzy Logic

Andrew Brim
Utah State University
4205 Old Main Hill, Logan, UT 84322
andrew.brim@usu.edu
435-797-0643

3.1 ABSTRACT

In the 1980s, Morgan Stanley, became very successful trading pairs of stocks simultaneously. This stock market trading strategy is referred to as a ”pairs trading” strategy[3]. Pairs trading in the stock market consists of finding a pair of stocks whose prices are highly cointegrated. Two stock prices, which are cointegrated, will move together. If one price goes up, you can expect the other price to go up as well[1]. Previous works show the simultaneous buys and sells that make up a traditional pairs trade as equal dollar amounts. In other words you engage in a pairs trade by simultaneously buying $1.00 of one stock, and selling $1.00 of the other stock in the pair[3].
However, the causal relationship of the stocks in a pairs trade can vary. In this paper we propose a method, using fuzzy logic, to allocate more money to the stock which does not show causality between the two. We will show there is a more profitable allocation of $2.00 than placing $1.00 on each stock in the pair, as we analyze the causal relationship between the two stocks.

KEYWORDS: Pairs Trading, Fuzzy Logic, Stock Market Predictions

3.2 INTRODUCTION

The basis of pairs trading is finding stocks which are cointegrated. Cointegration is more than correlation, as it also implies a causality factor between the two stock prices. A common analogy to cointegration is a drunk man walking a dog[2]. A drunk man performs a random walk, but the dog will always follow the drunk man because of the leash. Just as the man and the dog, might separate in the short term, in the long term they will come back together. If we understand which stock in the pair is the leader and which is the follower, we can make the pairs trading strategy more profitable.
Two stocks which exhibit this cointegrated behavior can be traded in the following way: when the prices of the two stocks diverge away from the mean of the price difference (or spread), you can simultaneously buy the less expensive stock, and sell the more expensive stock. This way, when the prices converge back together, either one or both positions will be profitable[3]. Conversely, if the two stock prices converge from the mean of the spread you can simultaneously sell the less expensive stock and buy the more expensive stock. When the prices revert back to the mean, either one or both positions will be profitable.
The logic above shows a pairs trading strategy which sells the more expensive stock, and buys the less expensive stock when the current price difference (the spread) reaches 105% of the mean,

```
if current_spread > mean_spread * 1.05:
    sell(more_expensive_stock)
    buy(less_expensive_stock)
elif current_spread < mean_spread * 0.95:
    buy(more_expensive_stock)
    sell(less_expensive_stock)
else:
    pass # do not buy or sell
```

and conversely buys the more expensive stock and sells the less expensive stock when the current spread reaches 95% of the mean. Historically the simultaneous buying and selling of the two stocks has been for same dollar amount[3]. For example, a very commonly traded stock pair is Visa and Mastercard[11]. A traditional pairs trading strategy would involve buying and selling the pair with the same dollar amount. However, if the causality between Visa and Mastercard is not equal, why should the amounts allocated be equal? The basis of this paper is to analyze the causality dynamics between pairs of stocks and rather than allocate equal amounts to the buy and sell of cointegrated stocks, we will experiment with putting more money on the stock which does not exhibit causality, or the follower stock. However, the causality between two stocks is not always clear. In fact, as you will see, the causality can be shared between the two stocks. Or there can be very little causality exhibited between the two stocks.

Using fuzzy logic, we make can take very noisy inputs like the cointegration level, and causality between stocks, and make a precise decision on how to allocate money to the buys and sells in a pairs trading strategy to make it more profitable. We will use the statistical test Augmented Dickey-Fuller to test for cointegration, and Granger causality to test for causality. In addition to Granger Causality, we introduce an original algorithm Mean Push (MP) as an additional test for causality. Mean push measures the stock price in the pair, whose movements are effecting the change in the mean of the spread. The algorithm produces a ratio (MPR) of how much one stock pushes the mean of the spread, in comparison to the other stock in the pair. In short, we will try to arbitrage the differences in causality between two stocks, to make a pairs trading strategy more profitable.

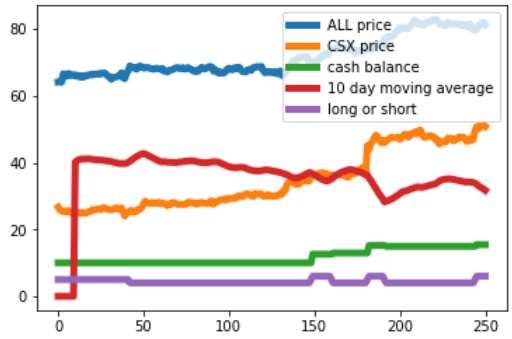
The Granger Causality test is a statistical test used to determine if one time series data set can be used to forecast another[10]. As we run the test, we are looking for a p-value less than 0.05 to indicate that one time series Granger causes the other.

Here are the results of applying fuzzy logic function to the buy and sells of a pairs trading strategy, for the pair of stocks: Allstate Corporation (ticker symbol ALL), and CSX Corp (ticker symbol CSX).

Augmented Dickey-Fuller cointegration test for pair ALL/CSX, p-value less than 0.05 indicates cointegration.

ALL/CSX p-value: 0.001
ALL/CSX ADF Statistic: -10.8209
ALL/CSX 1% Conf: -3.4982
ALL/CSX 5% Conf: -2.8912
ALL/CSX 10% Conf: -2.5826

Grange Causality test, p-value less than 0.05 indicates granger causality.
ALL GC p-value: 0.000028
CSX GC p-value: 0.266936
Mean Push Ratios:
ALL MPR: 1.03
CSX MPR: 0.97



You can see from above, Allstate and CSX Corp. have strong cointegration with a p-value of 0.001. Also Allstate shows strong Granger causality with a p-value of 0.000028. In our analogy of a drunk man walking a dog, Allstate is the man and CSX Corp. is the dog. The blue line shows the stock price of the more expensive stock, in this case ALL, and the orange line shows the stock price of the less expensive stock, CSX. The red line shows the 10 day moving average of the spread. The green line shows the cash balance. The purple line shows a buy or sell signal of the more expensive stock. When the purple line goes down it means the spread has widened and the more expensive stock is being sold. Conversely when the purple line goes up it means the spread has narrowed and the expensive stock is being bought.

3.3 PREVIOUS WORK

Fuzzy Logic has been applied to financial markets via stock market prediction and forecasting prices, detecting trends or patterns, and detecting or clarifying buy and sell signals. Fuzzy logic has also been applied specifically to pairs trading strategies with respect to classifying signals as strong buy/sell signals or weak buy/sell signals, rather than just buy/sell signals. Our approach is new, in that we are using fuzzy logic to specifically analyze the causal relationships in pairs trading, and adjust fund allocation based on these relationships, to increase profits.

Atsalakis and Valavanis, 2009, use fuzzy logic to improve stock market predictions, “A neuro-fuzzy system composed of an Adaptive Neuro Fuzzy Inference System controller used to control the stock market process model...demonstrating much improved and better predictions, compared to other approaches”[6]. Additionally Boyacioglu and Avci, 2010, analyze fuzzy logic within a Adaptive Neuro Fuzzy Inference System (ANFIS) to predict stock market returns, “...ANFIS algorithm is capable of accurately predicting stock market return”[7]. Zarandi, Rezaee, Turksen and Neshat, 2009 use fuzzy logic within a fuzzy model to predict stock market prices on a single company, “The proposed type-2 fuzzy model applies the technical and fundamental indexes as the input variables. This model is tested on stock price prediction...”[9]. While a pairs trading strategy, like the one we are using, can be used for stock market prediction, we are not using fuzzy logic to predict stock market prices. We are using fuzzy logic to change the amounts allocated to each stock in a pair, based on the causal relationships in the pair, and then modifying our pairs trading strategy accordingly.

Fuzzy logic has also been used to clarify when exactly to buy or sell a stock. If there are multiple factors involved in the decision to buy or sell a stock, fuzzy logic can be applied to add clarity to the buy/sell signals. Kuo, Chen and Hwang, 2001, apply fuzzy logic to generate rules to add to clarity to buy/sell points, “Thus, this study develops a genetic algorithm based fuzzy neural network

(GFNN) to formulate the knowledge base of fuzzy inference rules which can measure the qualitative effect on the stock market. Next, the effect is further integrated with the technical indexes through the artificial neural network (ANN)...Evaluation results indicate that the neural network considering both the quantitative and qualitative factors excels the neural network considering only the quantitative factors both in the clarity of buying-selling points and buying-selling performance."[8] Daisuke Yoshikawa, 2017, also uses fuzzy logic to develop more specific buy/sell signals. Rather than simply a buy signal, it generates a strong buy or weak buy signal using fuzzy logic. "Thus, the introduction of fuzzy logic in deriving an optimal strategy may lead to a sophisticated transaction flag, such as strong sell (buy), sell (buy), weak sell (buy) and hold, when the pair value touches exit or entry points."[4] While this approach does attempt to improve pairs trading it is still very different from our method. We are using fuzzy logic to better analyze the leaders and followers in pairs trading strategy, and make the strategy more profitable accordingly.

3.4 PROPOSED METHOD

To pairs trade, we need to identify cointegrated stock pairs. To be successful with our approach to arbitrage causality relationship, we also need to determine which stock is the leader stock, and which is the follower. We will start with Granger causality (GC) test. We will run GC on the first stock with respect to the second stock, and vice versa. This will give us a measurement of which stock is the leader and which is the follower, or possibly, if both stocks exhibit causality.
Next we will use the mean push algorithm (MP) as a second test to determine the leader follower relationship between the stocks. MP measures the moving average over time, and which stock is more responsible for the changes to the moving average, when compared to the other stock. Essentially it is a measurement of the ratio of who is moving the spread average more, stock P compared to stock Q, or stock Q compared to stock P. The stock which is moving the average more over time is the leader. Since pairs trading is a mean reversion strategy, the mean push algorithm allows us to look directly at the mean of the spread and measure which stock between the two is more responsible for changing the mean, as a ratio (MPR):

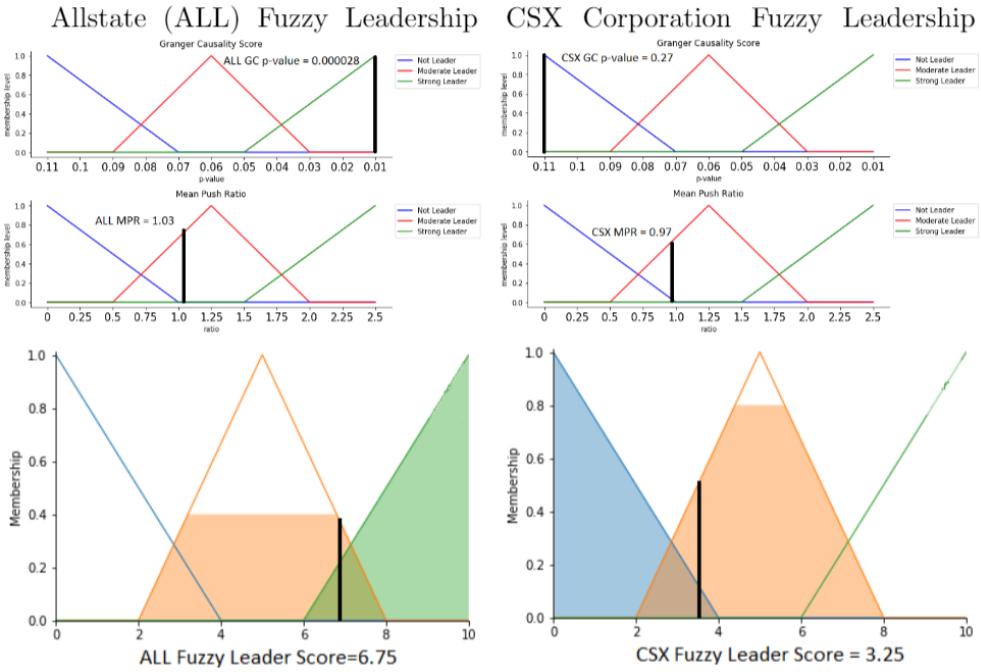
$$meanpushratio = \frac{\sum_{i=j}^{n} \frac{p_i - (\sum_{k=1}^{j} p_{i-k})/j}{p_i - p_{i-j}}}{\sum_{i=j}^{n} \frac{q_i - (\sum_{k=1}^{j} q_{i-k})/j}{q_i - q_{i-j}}}$$

Where:

p: price of first stock in the pair
q: price of second stock in the pair
j: number of observations used to calculate moving average
n: total number of observations

Fuzzy logic is used to handle this noisy relationship of who is the leader and who is the follower. A fuzzy membership function calculates a leadership membership score for GC p-value, then calculates a leadership membership score for the MPR, and then calculates a consolidated fuzzy leader score. We will take this value, and use it to create a leader-follower ratio and assign this to each stock in the pair. This ratio will then be used to determine how much money will be allocated to the leader and how much will be allocated to the follower. For example If the GC score p-value is very low, and the MPR is high it indicates this stock is a strong leader. The leader-follower ratio might be 9/10, indicating 10% of the cash would be allocated to the leader, and 90% would be allocated to the follower. Instead of the traditional pairs trading strategy, which would allocate $1.00 to both

stocks, we would allocate $1.80 to the follower, and $0.20 to the leader. We will apply fuzzy logic this to all pairs, run a trading strategy and compare the results to a traditional pairs trading strategy. Here are the results of the the example used in the introduction Allstate and CSX Corporation.



You can see the membership for the ALL GC score is a strong leader with membership level 1.0, and CSX is a No leader with membership score of 1.0. The ALL Mean push ratio membership is also calculated with a Moderate leader membership score of 0.8, and CSX has a Moderate leader membership score of 0.6. These two membership scores are inputs for the fuzzy function which calculates a leader score.

Visually you can see the levels of membership are shaded in, and the centroid function uses a center of gravity, of the entire shaded region to calculate a final fuzzy leader score. From these two fuzzy leader scores 6.75 and 3.25, we create a leader ratio we can use to allocate cash in our pairs trading strategy. 67.5% of the cash will be put into the follower CSX, and 32.5% of the cash will be put into the leader ALL.

We hypothesize that a follower stock exhibits more price movements, and if the pair is cointegrated the price will come follow the leader. We will show through simulations that this price movement can be arbitraged to increase the profits of a pairs trading strategy.

3.5 EXPERIMENT

We run a series of stock market trading simulations to test a traditional pairs trading strategy, versus our strategy which uses fuzzy logic. We are using medium sized companies from the Russell Index to search for pairs, as it will give a large base of 66 stocks, to pick pairs from. We test all 2145 possible pair combinations for correlation above 90%, to give us a group of stocks where cointegration may exist. Correlation is simply used as a test to remove stocks which are not correlated, and therefore not cointegrated. After non-correlated stocks are removed, we test for

cointegration, Granger Causality, and a Mean Push ratio. Cointegration is tested with Augmented Dickey-Fuller[13]. We are looking for p-values less than 0.05, and a critical value lower than 1% confidence interval, typically -3.5, so we can reject the null hypothesis that cointegration does not exist.

After we have the inputs, GC and MPR, for our fuzzy logic function we calculate a fuzzy leader score for each stock in the pair. The trading simulation steps through the daily prices for each pair, and calculates the 10 day spread moving average and the current spread. If the spread widens past 105% of the average, we sell the high price, and buy the low price. If the spread narrows closer than 95% of the average we buy the high price and sell the low price. The simulation will allocate the entire cash balance according the leader-follower ratios, and trading fees are not included. Each simulation begins with $1000 dollars. The green line in each graph shows the cash balance throughout the simulation.

3.6 RESULTS

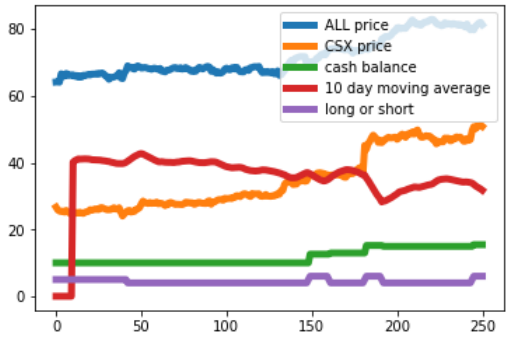
From 66 stocks, there are 2145 possible pairs of stocks. After all possible pairs of stocks are tested for correlation above 90%, we select 22 pairs of stocks with high correlation, cointegration, and one or both stocks exhibiting causality. The trading simulation was run twice, for each stock pair. Once with traditionaL 50/50 ratios, meaning equal buy and sell positions. Then run a second time with our new ratios, with more money allocated to the follower stock, decided by our fuzzy function. As you can see our leader-follower ratios adjusted by the fuzzy function yielded 17% over one year. The traditional pairs trading strategy yielded 15% over one year. Below, the highlighted stocks are the strongest performers using our leader-follower method.

Pairs Trading Simulation Results Possible Pairs 2145 Pairs Found 22

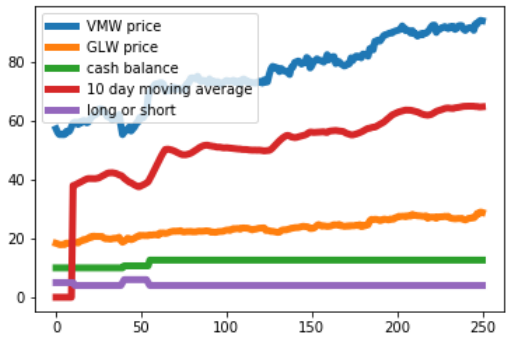
Stock1	Stock2	Returns w/Fuzzy Ratios	Returns w/traditional strategy 50/50 Ratios	GC1	GC2	MPR1	MPR2	FuzzLead1	FuzzLead2	Lead/Follow1 %	Lead/Follow2 %
ADP	WM	0.07	0.10	0.00	0.54	0.62	1.62	8.65	3.85	30.80	69.20
AVGO	**GLW**	**0.21**	**0.07**	**0.03**	**0.55**	**0.13**	**7.65**	**1.62**	**5.00**	**75.58**	**24.42**
GLW	**MU**	**0.23**	**0.19**	**0.04**	**0.01**	**1.41**	**0.71**	**8.73**	**7.46**	**46.05**	**53.95**
VMW	**GLW**	**0.26**	**0.10**	**0.09**	**0.02**	**0.35**	**2.86**	**1.00**	**9.00**	**90.00**	**10.00**
CMI	GLW	0.09	0.09	0.02	0.10	0.26	3.82	4.71	5.00	51.48	48.52
ITW	VMW	0.12	0.12	0.01	0.08	0.95	1.05	6.66	3.83	36.54	63.46
CMI	VMW	0.27	0.27	0.01	0.02	0.70	1.42	7.49	8.73	53.84	46.16
VLO	BBT	0.27	0.25	0.00	0.19	0.59	1.69	8.82	4.24	32.47	67.53
VLO	DFS	0.07	0.14	0.01	0.11	0.61	1.64	8.71	4.00	31.48	68.52
CME	MPC	0.09	0.13	0.09	0.00	0.88	1.13	3.13	6.89	68.76	31.24
ALL	MMC	0.10	0.11	0.44	0.00	0.84	1.18	3.07	6.94	69.31	30.69
CME	BBT	0.04	0.04	0.00	0.05	0.56	1.77	8.28	8.90	51.82	48.18
DFS	BBT	0.06	0.04	0.17	0.31	0.70	1.43	2.32	1.64	41.48	58.52
CME	**NSC**	**0.18**	**0.12**	**0.02**	**0.08**	**1.34**	**0.75**	**7.71**	**3.67**	**32.24**	**67.76**
CMI	CME	0.07	0.09	0.03	0.09	0.80	1.24	5.00	3.08	38.13	61.87
ALL	**CSX**	**0.54**	**0.41**	**0.00**	**0.27**	**1.03**	**0.97**	**6.75**	**3.25**	**32.50**	**67.50**
DE	CSX	0.28	0.37	0.04	0.06	0.69	1.45	5.00	8.73	63.59	36.41
CMI	CSX	0.05	0.02	0.01	0.01	0.49	2.05	7.01	9.00	56.21	43.79
NSC	DE	0.02	0.16	0.12	0.03	0.66	1.51	1.97	8.73	81.62	18.38
DE	CBS	0.44	0.43	0.02	0.01	0.85	1.18	6.51	6.85	51.25	48.75
CMI	ITW	0.11	0.08	0.03	0.01	0.83	1.20	5.00	6.72	57.35	42.65
CMI	NSC	0.08	0.07	0.02	0.00	1.06	0.94	6.19	6.78	52.26	47.74
	Mean Annual Returns	**0.17**	**0.15**								

Here are some visuals of the strongest performers, that were allocated with leader-follower ratios. As mentioned above, the blue line shows the stock price of the more expensive stock, and the orange line shows the stock price of the less expensive stock. The red line shows the 10 day moving average of the spread. The green line shows the cash balance. The purple line shows the buy or sell signal of the more expensive stock, to visualize when the simulation is making trades. When the purple line goes down it means the spread has widened and the more expensive stock is being sold. Conversely when the purple line goes up it means the spread has narrowed and the expensive stock is being bought.

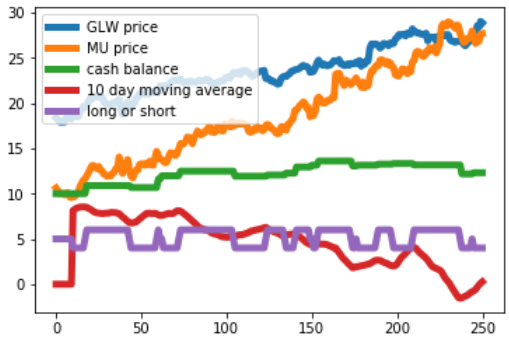
ALL - CSX Pairs trading results...
total buys: 6.0
total sells: 5.0
Beginning cash: 1000.00
Final cash: 1544.58



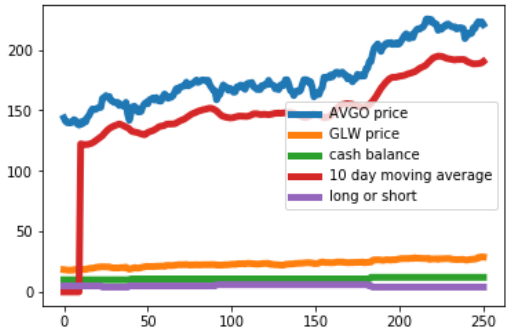
VMW - GLW Pairs trading results...
total buys: 3.0
total sells: 2.0
Beginning cash: 1000.00
Final cash: 1263.77



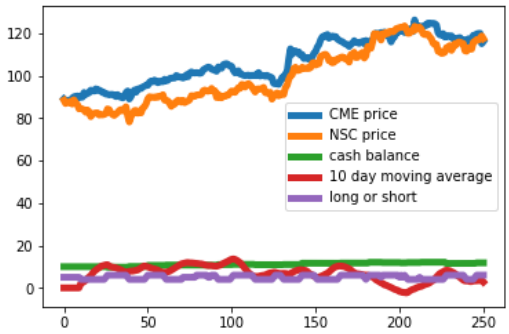
GLW - MU Pairs trading results...
total buys: 19.0
total sells: 18.0
Beginning cash: 1000.00
Final cash: 1230.80



AVGO - GLW Pairs trading results...
total buys: 3.0
total sells: 2.0
Beginning cash: 1000.00
Final cash: 1208.02



CME - NSC Pairs trading results...
total buys: 25.0
total sells: 24.0
Beginning cash: 1000.00
Final cash: 1180.23



3.7 CONCLUSION

It is apparent that pairs trading strategies can be improved by analyzing the causal dynamics in a highly cointegrated pair. Our trading strategy, which allocated funds based on a leader follower relationship, yielded 17% annual returns, versus the traditional strategy's 15%.

3.8 FUTURE WORK

This approach looked particularly at handling causality on a more individual case, based on the stock pair. The same approach could be taken when looking at a pair's volatility. Better buy and sell signals could be generated by looking more closely at the different volatility's that arise in different pairs. In other words, a better trading strategy could be applied, based on the specific behavior of a pair's volatility.

3.9 CODE

The code can be viewed on github. Please recall that all financial trading includes risks and past performance is never a guarantee for future performance. Recreate this experiment here: https://github.com/abrim24/PairsTrading_Causality/blob/master/PairsTrading_Causality.ipynb

REFERENCES

[1] Engle, Robert F., Granger, Clive W. J. (1987) ”Co-integration and error correction: Representation, estimation and testing”, Econometrica, 55(2), 251–276. done

[2] Michael P. Murray (1993) A Drunk and Her Dog: An illustration of Cointegration and Error Correction. The American Statistician, February 1994, Vol. 48, No. 1

[3] Gatev, G., Goetzmann, N., Rouwenhorst, K. (2006). Pairs trading: Performance of a relative value arbitrage rule. Review of Financial Studies, 19(3), 797–827.

[4] Yoshikawa, Daisuke 2017, An Entropic Approach for Pair Trading, Entropy, Vol. 19, No. 7. (30 June 2017)

[5] Cheung, W., Kaymak, U.: A fuzzy Logic Based Trading System, Technical Report, Erasmus Institute ,The Netherlands (2007)

[6] Atsalakis, G. S., Valavanis, K. P. (2009). Forecasting stock market short-term trends using a neuro-fuzzy based methodology. Expert Systems with Applications, 36(3), 10696–10707.

[7] Boyacioglu, M., Avci, D.: An adaptive network-based fuzzy inference systems (ANFIS) for prediction of stock market return: the case of Istanbul stock exchange. In: Expert Systems with Applications, vol. 37, pp. 7902–7912 (2010)

[8] R. J. Kuo, C. H. Chen, Y. C. Hwang, An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network, Fuzzy Sets Syst., vol. 118, no. 1, pp. 21-45, Feb. 2001.

[9] Zarandi MHF, Rezaee B, Turksen IB, Neshat E (2009) A type-2 fuzzy rule-based expert system model for stock price analysis. Expert Syst Appl 36(1):139–154.

[10] Granger, C.W.J. (1980), Testing for causality: a personal viewpoint. Journal of Economic Dynamics and Control, 2, 329-352.

[11] Stocks For The Ultimate Pairs Trade

[12] Guide to Pairs Trading Investopedia, Investopedia LLC 2016 `http://www.investopedia.com/university/guide-pairs-trading/`

[13] Dickey David A, Wayne A. Fuller (1979): Distribution of the Estimators for Autoregressive Time Series With a Unit Root, Journal of the American Statistical Assoc., 74, 427-431.

CHAPTER 4

Deep Reinforcement Learning Pairs Trading with a Double Deep Q-Network

Deep Reinforcement Learning Pairs Trading with a Double Deep Q-Network

Andrew Brim

May 2019

4.1 Abstract

This research applies a deep reinforcement learning technique, Deep Q-network (DQN), to a stock market pairs trading strategy for profit. There is a need for this work, not only to further the use of reinforcement learning in stock market trading, but in many other areas of financial markets. The work utilizes a specific type of DQN, a Double Deep Q-Network to learn a pairs trading strategy. The DDQN is able to learn a cointegrated stock pair's mean reversion pattern, and successfully make predictions based on this pattern. Attesting that a reinforcement learning system, can effectively learn and execute a pairs trading strategy in the stock market. It also introduces a parameter, Negative Rewards Multiplier, during training that adjusts the system's ability to take more conservative actions. Based on the results, the next steps would be to employ this method in other financial markets, or perhaps use a DDQN to learn additional trading strategies.

4.2 Introduction

Pairs Trading is a statistical based trading strategy involving a pair of cointegrated financial assets [5, 4, 1]. This work presents a reinforcement learning system, utilizing a DDQN and an RL environment in which to interact[14, 13], to learn a trading strategy for a cointegrated pair of stocks.

A pairs trading strategy targets a pair of cointegrated stocks whose spread, or price difference, is mean reverting. When the spread increases or decreases away from the mean, this strategy predicts the spread of the cointegrated pair of stocks will revert back to the mean. As shown in Figure 1, a reinforcement learning system can learn the spread mean reversion of the Adobe/Red Hat stock pair from 2014 to 2017 and then correctly predict it for 2018.

A naive pairs trading strategy is executed in the following way: When the spread increases to a given threshold, the stock pair is traded by simultaneously entering into a short position (sell) for the higher price stock and a long position (buy) for the lower price stock [9]. If the spread decreases to a given threshold the stock pair is traded by simultaneously entering into a short position for the

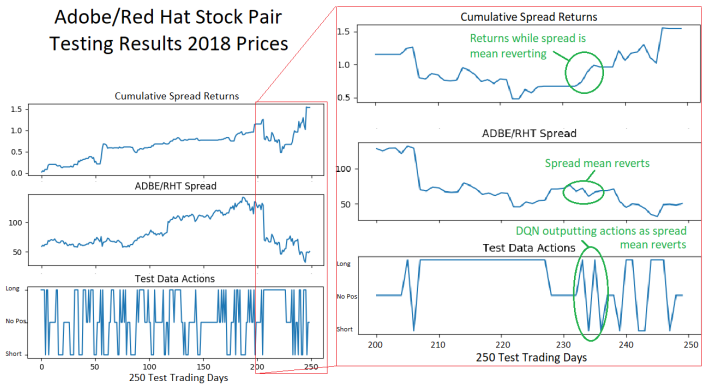


Figure 1: Pairs Trading Testing Results for the Adobe/Red Hat stock pair. The RL System is able to train a DDQN with training data 2014-2017, and then test it's predictive ability on 2018 data. The DDQN outputs actions of long, short, or no position on the spread, producing cumulative spread returns of 1.58.

lower price stock and a long position for the higher price stock as shown in Algorithm 1 and illustrated in Figure 2. When the spread of the cointegrated pair, reverts back to the mean, one or both positions will be profitable [5, 1]. In this work, the system only learns and predicts movements of the spread. It does not attempt to predict the prices of the individual stocks. Therefore if the system predicts the spread will increase, it will output a long signal. Conversely if the system predicts a spread will decrease it would output a short signal. Figure 2 demonstrates the prices of two cointegrated stocks, PepsiCo, Inc.(PEP) and Coca-Cola Co.(KO). Temporary spread divergences are eventually corrected as cointegrated prices move back together.

Algorithm 1 Naive Pairs Trading Strategy with a spread threshold of +/- 0.05

```
UPPER_THRESHOLD ← 1.05
LOWER_THRESHOLD ← 0.95
if current_spread > mean_spread * UPPER_THRESHOLD then
    short higher_price_stock
    long lower_price_stock
else if current_spread < mean_spread * LOWER_THRESHOLD then
    short lower_price_stock
    long higher_price_stock
else
    no position
end if
```

Various approaches have used deep reinforcement learning techniques, such as a DQN [10, 8], to predict and trade the stock market. Liang, Chen, Zhu, Jiang, Li 2018 train a DQN to hedge portfolio risk [7]. Ding, Zhang, Liu, Duan 2015, train a Deep Learning system for event-driven stock prediction [3]. Li, Jiang, Li, Chen 2015 and Wu 2015 use Neural Networks for stock market

Figure 2: Cointegrated stock pair PepsiCo, Inc. (PEP) and The Coca-Cola Co (KO) demonstrating price spread mean reversion

predication as well[6, 16]. This work advances these previous works, as it utilizes a Double Deep Q-Network (DDQN) to decorrelate training samples, reduce error, and achieve better performance[14]. It is also different from previous works, as it extracts features with a specific trading strategy, pairs trading, in mind. The previously mentioned works use input features including the portfolio assets itself, or news feed information. Here, the input features represent the spread mean for different time lags, enabling the system to learn at what time interval the spread mean reverts. The system then outputs actions to predict the spread. The DDQN interacts with an RL environment by taking the actions to long, short, or enter no position, on the spread of the stock pair. As the DDQN takes actions in the environment and receives reward for each action, it optimizes a Q-function which outputs the best action for any given state of input features.

4.3 Experiment

This system trains and tests a DDQN on 38 stock pairs. The pairs are selected from the SP500 Stock Index in the following manner. From a possible 78000 pairs, each pair must have a Augmented Dickey-Fuller p-value between 0 and 0.05, indicating the pair is cointegrated. This test reduces the number of possible pairs to 145. The pair must also have enough variance to generate trade signals. This is achieved by selecting a pair where each stock in the pair must have a standard deviation divided by the mean of 0.5 or greater. This second test reduces the number of possible pairs to 38. These two statistical tests verify the pair's spread will be mean reverting, with enough variance to be trade-able.

The training data consists of daily prices for 4 years from 2014 to 2017, and the testing data consists of daily prices for 2018. The daily prices data for the training set is pre-processed, and used to generate the input features for the state of the environment for that day. The DDQN trains for 300 episodes. Figure 11 illustrates, for the training data set for the Adobe/Red Hat stock pair. The DDQN is able to converge on an near optimal set of weights for the input features, and produce a Q-function to maximize reward for any given state in the training data set. The testing data results for this pair are shown in Figure

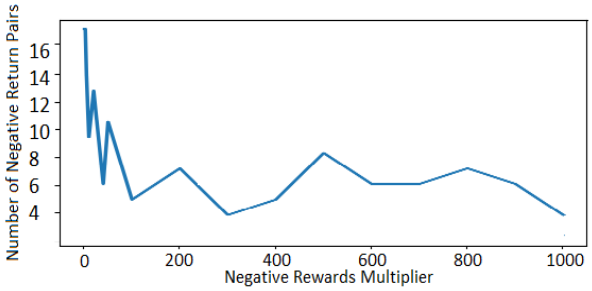


Figure 3: From 38 stock pairs, the number of pairs with negative returns decreases as NRM increases, illustrating the DDQN taking more conservative actions.

6. The Figure 6(d) heat-map shows the input features received from the RL environment and inputted into the DDQN, providing a visualization of what the DDQN sees. Figure 6(e) shows the actions output by the DDQN during testing. Figure 6(f) provides a histogram of the short, no position, and long actions.

Negative Rewards Multiplier

During training, it was discovered that while total cumulative returns were more often positive, the system was learning that more actions of long or short yielded higher reward overall. While promising on the macro level, it left many day to day actions with negative returns. A new parameter was introduced during training, the Negative Rewards Multiplier (NRM), which multiplies any negative spread returns to make the rewards much more negative. This causes the DDQN to take actions more conservatively, and to take an action of no position more often as no position will always result in a reward of 0. This may reduce total cumulative returns, but it also reduces the number of actions which produce a negative return, as illustrated in Figure 3. Introducing NRW during training teaches the system to be more risk averse. This work does not attempt to give specific measurement to the risk aversion, as defined by financial industry standards. It simply defines the system's risk aversion more generally as an attempt to teach the system to reduce it's number of actions that result in a negative return. Future work could development a more rigorous measurement as to what levels the system is able to reduce risk.

4.4 Methods

Q-Learning

This work employs Q-learning, a type of temporal difference learning, to optimize a policy function for each state in the space of trading parameters[15, 13].

Q-learning is combined with function approximation, utilizing a neural network to approximate a Q-function. An OPENAI Gym environment[2] is built to simulate the stock market pairs trading strategy, and allow the DDQN to take long, short, or no position actions on the spread. The DDQN outputs an action, that is sent to the Gym environment, which then returns the next state and the reward for the action taken.

Deep Q-Network

A Deep Q network (DQN) is a multi-layered neural network that for a given input state, outputs a vector of action values[14]. The input features for this DDQN are designed for the system to learn the spread mean reversion including: current spread of the pair, daily returns of the spread, spread mean for various time intervals, and spread / spread mean for the same time intervals as shown in Figure 4. Spread / spread mean for a spread at equilibrium will be 1.0. A spread / spread mean of 1.05 would be high suggesting the spread value will decline, and 0.95 would be low suggesting the spread value will rise. The DDQN outputs the action to take at that point in time: long, short, or take no position on the spread of the stock pair. The DDQN is able to optimize a Q-function, to maximize reward based on the input features it receives. The DDQN structure utilizes a Pytorch NN[11] consisting of an input layer of 10 features, a fully connected layer of 50 nodes, another fully connected layer of 50 nodes, utilizing a RELU non-linear activation function, and an output layer of 3 nodes, as seen in Figure 4. The DDQN utilizes Adam optimizer to update network weights.

The training rewards in the OPENAI Gym are calculated as follows:

$$T = a \times r \times N$$

Where:
- T: Training Rewards
- a: action output by DDQN, action = {1, -1, 0}
- r: spread returns
- N: Negative Rewards Multiplier

The testing rewards in the OPENAI Gym are calculated as follows:

$$R = a \times r$$

Where:
- R: Testing Rewards
- a: action output by DDQN, action = {1, -1, 0}
- r: spread returns

The action space of {1, -1, 0} represents a long, short, or no position on the spread. NRW is only used for training, to make the DDQN more risk averse. During testing the rewards are simply calculated as spread returns * action. Figure 6(a) shows the cumulative returns for the actions taken in 6(e) during testing for the Adobe/Red Hat pair.

Double Deep Q-Network and Replay Memory

This work employs a specific type of DQN, a Double Deep Q-Network, introduced by Google Deep Mind in 2016 [14] and utilized by the artificial intelligence AlphaGo which defeated the World Go champion Lee Sedol [12].

Van Hasselt, Guez, and Silver 2016, show that the idea behind the Double Q-learning algorithm (van Hasselt, 2010), which was first proposed in a tabular setting, can be generalized to work with arbitrary function approximation, including deep neural networks. Their new algorithm we call Double DQN, not only yields more accurate value estimates, but leads better overall performance of the deep neural network[14].

DDQN's ability to yield more accurate value estimates comes from it's separating the neural network into two networks: evaluation and target networks. The evaluation network is used to generate actions and the target network is used to train from randomly selected observations from replay memory[14]. The DDQN replay memory stores the state transitions that are received from the environment, allowing this data to be reused. By sampling from it randomly, the transitions that build up a batch are decorrelated, stabilizing the DDQN[14].

As shown in Figure 5 the workflow of the DDQN is as follows: 1. An action is received by the RL environment consisting of long, short, or no position on the spread. 2. The reward from this action is calculated based on the spread returns for that day. 3. the next state is generated by the RL environment. The state consists of the next set of input features for the DDQN. 4. The reward and next state are returned by the RL Environment. 5. The next state is stored in the the DDQN replay memory. 6. The replay memory stores 200 previous State, action, reward, next State observations. The target network is trained using these randomly selected previous states. Every 100 observations, the target network weights are copied to the evaluation network. The target network utilizes Adam optimization. The evaluation does use any type of backpropagation as the network weights are manually copied over from the target network 7. The next state is directly inputted into the evaluation network and the next action is outputted. 8. The next action is sent to the RL Environment.

4.5 Results

This method is applied to all 38 stock pairs, and tested on 2018 stock prices. Figure 9 shows the total spread cumulative returns for all 38 stock pairs, for various NRM values from 1 to 1000. As shown in the first column of Figure 9 shows the total cumulative returns for all 38 stock pairs is 131.33. Figure 12 shows how the DDQN takes an action of no position more often, until it receives a state it can more likely predict to gain a positive reward.

Perhaps the most interesting result is found in Figure 12(c). With an NRM of 50.0, the DDQN outputs no position actions for 249 out of the 252 test days. However, the 3 days where a long action was output, yielded returns of 1.42. It is clear the system can learn to be more conservative and still yield strong results. Figure 10 displays total cumulative returns decreasing as NRM increases

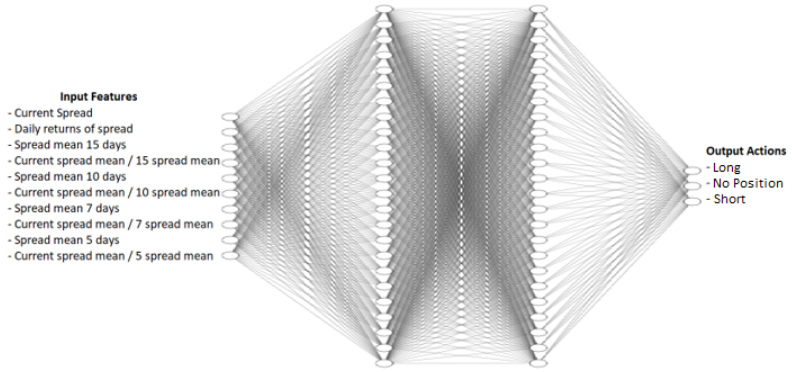


Figure 4: DQN NN structure

to 1000. Figure 3 illustrates how the number of negative return pairs decreases as the NRM increases, showing precisely what the NRM was introduced for. Figure 6 shows the results for the pair Adobe/Red Hat, which produces an annual spread cumulative returns of 1.58. This figure a more complete version of the results shown in the Introduction.

As illustrated in Figure 6(c) ADBE/RHT Spread, the spread mean reverts at least 3 times in the last 50 trading days. The DDQN is able to generate a Q-function during training, and successfully take actions to long, short, and no position on the spread of the pair in the testing data. As shown in the first column of Figure 9, the total cumulative returns for all 38 stock pairs is 131.33. The highest 4 pairs' spread returns were CNX/HBI 7.52, FCX/HBI 25.67, HBI/MRO 27.41 and CTWS/AWR 71.28. as shown in Figure 7. The lowest 2 performers were ESV/RRC -0.78 and ESV/GNW -9.64 as shown in Figure 8.

4.6 Summary and Future Work

The DDQN was able to output actions which earned a total cumulative returns for all 38 stock pairs of 131.33. The DDQN was able to learn to take actions more conservatively, based on adding a Negative Reward Multiplier. The features provided to the system, allowed it to learn a mean reversion strategy for intervals of 15 days or less.

DQNs are able to learn and execute trading strategies for positive returns, as shown by this application. Future applications could include a system learning different types of trading strategies or opportunities. Reinforcement learning systems could also be applied to different time frames including high frequency trading, or other financial markets.

References

[1] Dickey David A and Wayne A. Fuller. Distribution of the estimators for autoregressive time series with a unit root, 1979.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[3] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction, 2015.

[4] Robert F. Engle and Clive W. J. Granger. Cointegration and error correction: Representation, estimation and testing, 1987.

[5] G. Gatev, N. Goetzmann, and K. Rouwenhorst. Pairs trading: Performance of a relative value arbitrage rule, 2006.

[6] Qing Li, LiLing Jiang, Ping Li, and Hsinchun Chen. Tensor-based learning for predicting stock movements, 2015.

[7] Zhipeng Liang, Hao Chen, Junhao Zhu, Kangkang Jiang, and Yanran Li. Adversarial deep reinforcement learning in portfolio management. https://arxiv.org/abs/1808.09940v3.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[9] Michael P. Murray. A drunk and herdog: An illustration of cointegration and error correction, 1993.

[10] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn, 2016.

[11] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch. *Computer software. Vers. 0.3*, 1, 2017.

[12] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[13] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

[14] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[15] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[16] Jiayu Wu. A pairs trading strategy for goog/googl using machine learning. http://cs229.stanford.edu/proj2015/028$_r$*eport.pdf*, 2015.

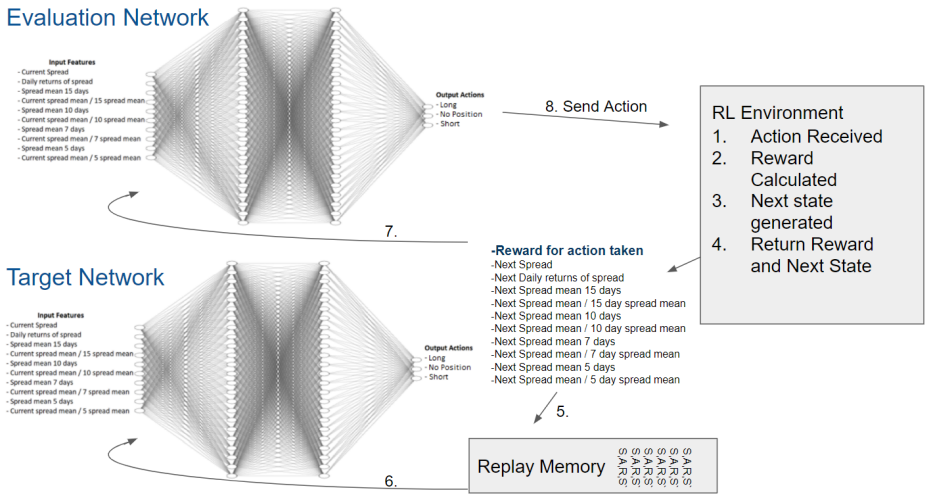


Figure 5: DDQN structure and workflow with RL Environment. 1. An action is received by the RL environment. 2. The reward is calculated. 3. the next state is generated. The state consists of the next set of input features for the DDQN. 4. The reward and next state are returned by the RL Environment. 5. The next state is stored in the Replay memory. 6. The replay memory stores 200 previous State, action, reward, next State observations. The target network is trained using these randomly selected previous states. Every 100 observations, the target network weights are copied to the evaluation network. 7. The next state is also directly inputted into the evaluation network. 8. The next action is outputted by the DDQN and sent to the RL Environment.

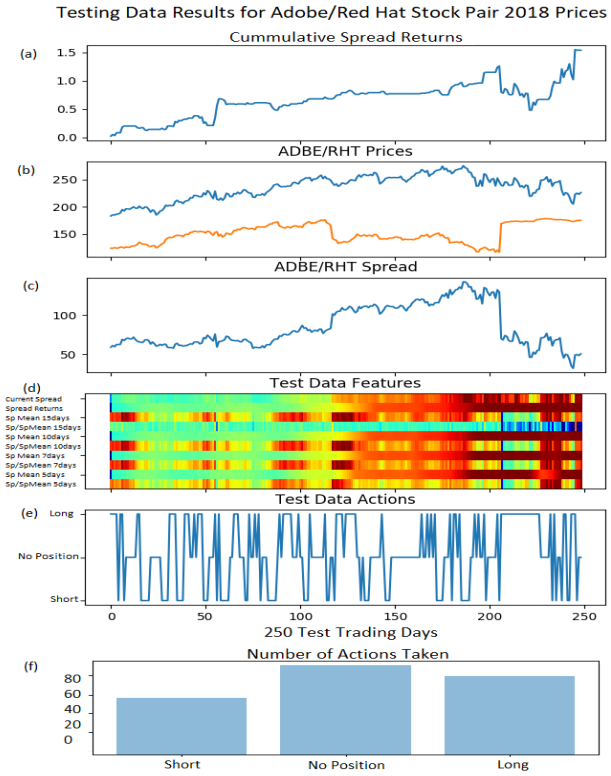


Figure 6: Testing Results for ADBE/RHT. (a) shows DDQN trading returns of 1.58. (b) shows the prices of ADBE and RHT. (c) shows the spread of ADBE and RHT. (d) Heatmap of DDQN input features for test data ADBE/RHT. (e) shows the actions output by the DDQN. (f) shows the number of each action taken.

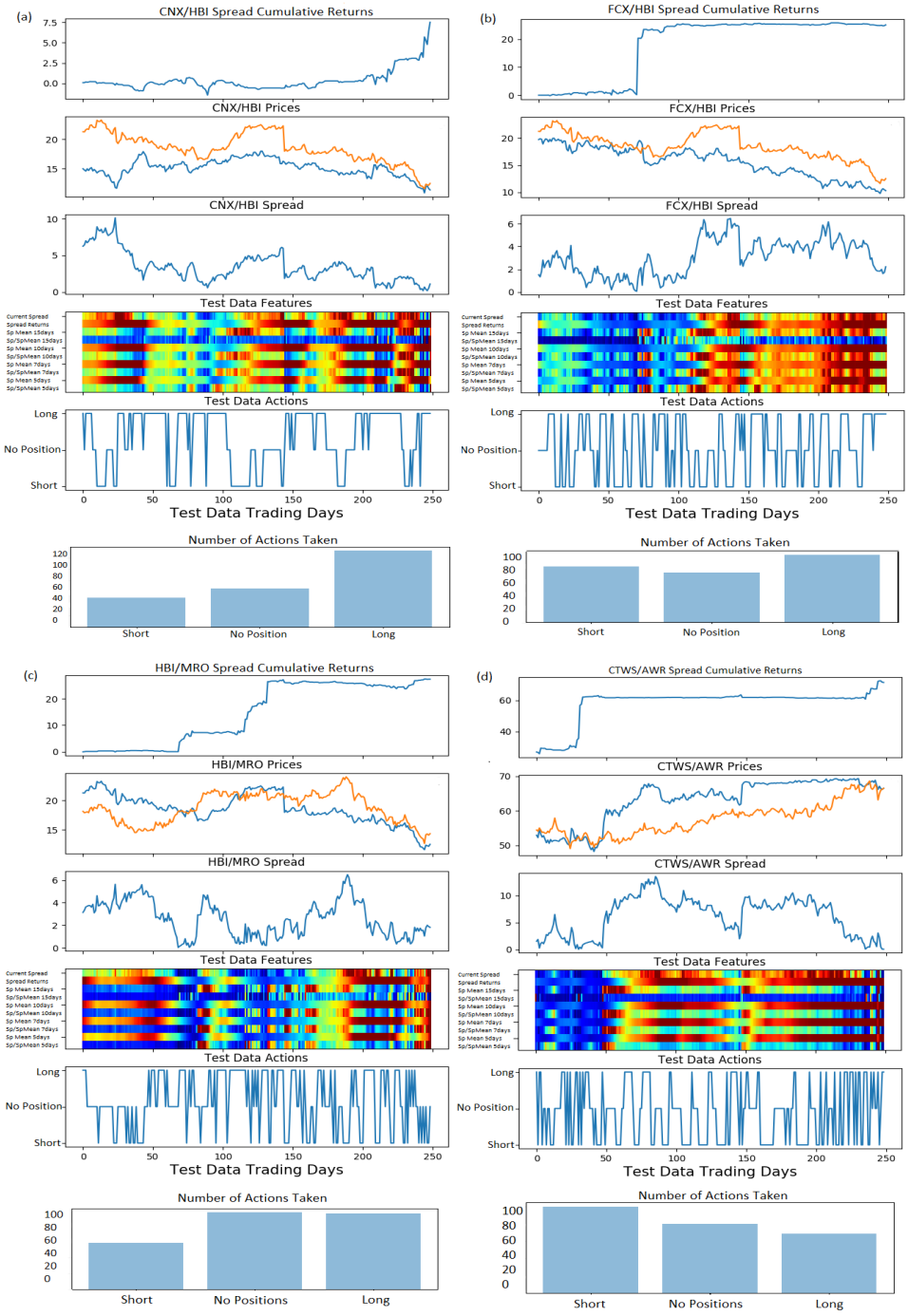


Figure 7: Top 4 DDQN Pairs Trading performers: CNX/HBI 7.52 (a), FCX/HBI 25.67 (b), HBI/MRO 27.41 (c), CTWS/AWR 71.28 (d)

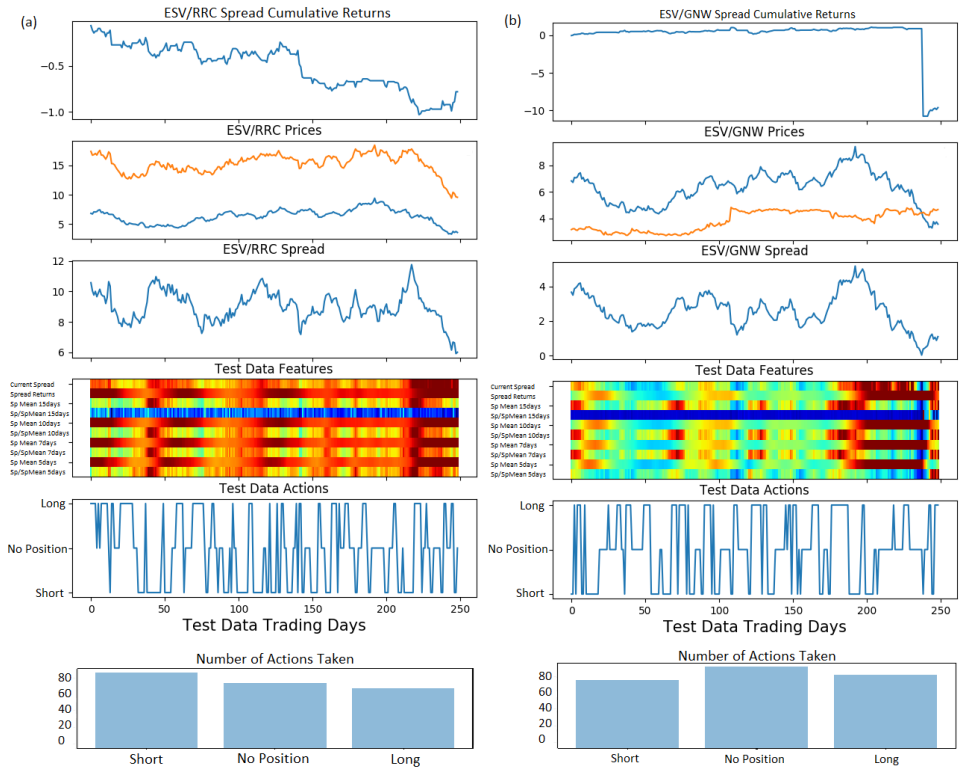


Figure 8: Bottom 2 DDQN Pairs Trading performers: ESV/RRC -0.78 (a), ESV/GNW -9.64 (b)

38 Stock Pairs Total Spread Cumulative Returns, by Negative Returns Multiplier during Training

Negative Returns Multiplier	1	2.5	5	10	20	50	100	200	500	700	1000
Stock Pair											
BEN_COG	1.23	-0.27	0	1.13	0.27	0.18	0	0	-0.16	0	0.48
DISCA_RIG	-0.22	-0.54	-0.53	0	0	0	0.01	-0.14	0	0	0
DISCK_RIG	1.25	0	1.11	0.45	0.01	-0.14	0	0	-0.14	0	0
ADBE_CRM	1.07	0.09	-0.02	0	0	0	0	0	0	0	0
CF_HBI	0.25	-0.04	-0.14	0.31	0	0	0	0	0	0	0
ESV_GNW	-9.64	-11.54	-11.62	-2.3	-0.32	-11.5	0.89	-9.95	-11.5	11.67	-0.13
CNX_HBI	7.52	8.46	5.78	8.5	11.9	1.42	4.61	3.42	0.44	0	0
AMZN_CRM	0.56	-0.03	-0.01	-0.01	0	-0.07	0	0	0	0	0
MA_VFC	0.01	-0.18	-0.22	-0.26	-0.27	-0.06	0	0	0	0	0
FCX_GNW	-0.19	0	0	0	0	0	0	0	-0.05	0	0
CRM_NVDA	-0.54	-1.22	-3.29	2.94	-0.43	-0.96	-3.56	-3.15	-3.15	0	0
CF_FOSL	-1	-0.04	0.12	0	-0.01	0	0	0	0.02	0.12	0
FCX_HBI	25.67	26.55	17.34	16.12	20.87	22.52	22.27	21.28	20.92	-1.62	0.54
DISCK_ESV	0.08	0	0	-0.18	0	0	0	0	0	0	0
DISCK_NE	-0.08	0	0	0	0	0	0.09	0	0	0	0
DISCA_NE	-0.25	-0.17	0	0	0.12	0	0	0	0	0.03	0
DISCA_ESV	-0.56	-0.16	-0.21	0	0	0	0	0	0	0	0
ESV_RRC	-0.78	0.29	0.15	0.05	-0.28	-0.03	0.2	0	0.11	0.17	0
NBL_RIG	1.14	0.24	-0.04	-0.02	-0.02	0	0.04	0	0	0	0
CNX_GNW	-0.04	0.19	-0.06	0.14	-0.02	0	0.01	0.29	0.03	0	0
COG_DO	2.32	-0.71	0.51	0.54	0.14	-0.97	-0.22	-0.63	-0.41	0	1.06
HBI_NBL	0.24	0.01	0	0.07	0.03	0	0	0	0	0	0
HBI_MRO	27.41	8.6	16.15	29.83	10.68	3.3	13.16	0.6	-0.5	-6.23	0
GNW_NBL	0.09	-0.05	0	-0.02	0.01	0	0	0	0	0	-0.07
DISCA_MA	0.49	0.03	0.01	0.25	-0.04	-0.03	-0.03	0	0	-0.01	0.02
DISCK_MA	-0.21	-0.01	0	0	0	0	0	0	0	0.02	0.03
RIG_RRC	1.32	0.48	0.28	0.31	0	0.44	-0.08	-0.07	0.77	0.58	0.5
CF_CNX	0.09	-0.12	0	0.07	0	0	0	0	0	0	0
CF_GNW	0.12	0.06	0.41	0	0	0.06	0	0	0	0	0
ESV_HBI	0.03	0.17	0	0.1	0	0	0	0	0	-0.02	0
NE_RRC	-0.08	-0.05	0.07	-0.02	0	0	0	0	0	0.01	0
ADBE_RHT	1.58	1	0.55	0	0.07	-0.24	0.05	0	0	0	0
MA_RIG	0.45	0.23	-0.03	0	-0.06	0.08	0	-0.05	0	0	0
NBL_SWN	0.02	0.02	-0.03	-0.32	-0.02	0	0	0	0	0	-0.05
CTWS_AWR	71.28	80.79	87.1	53.6	50.2	44.45	55.92	0.51	1.65	1.39	0.41
CTWS_WTR	-0.48	-0.06	0.19	-0.04	-0.26	0	0.03	0	0	-0.03	0
AWR_WTR	0.62	0.17	0.08	0	0	0	0	0	0	0.15	0
SLB_PFE	0.56	2.73	-2.06	5.75	-0.52	-4.83	-0.72	-6.23	-4.38	-5.08	-5.22
Total Cumulative Returns	**131.33**	**114.92**	**111.59**	**116.99**	**92.05**	**53.62**	**92.67**	**5.88**	**3.65**	**1.15**	**-2.43**

Figure 9: DQN Pairs Trading Results for all 38 pairs. Each column shows the returns for all pairs where any negative returns, during training, are multiplied the NRM factor in row 1. The higher the NRM, the more often the DDQN will take a conservative action of no position. A value of 0 returns indicates the DDQN took no trading actions.

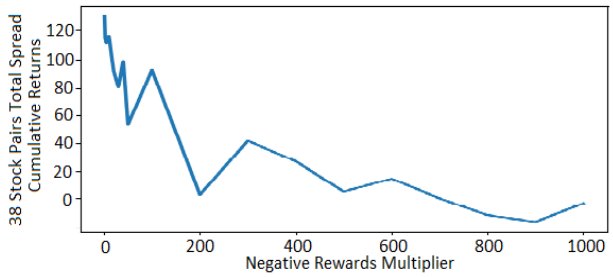


Figure 10: Returns approach zero as the NRM increases, causing the DDQN to take more no position actions.

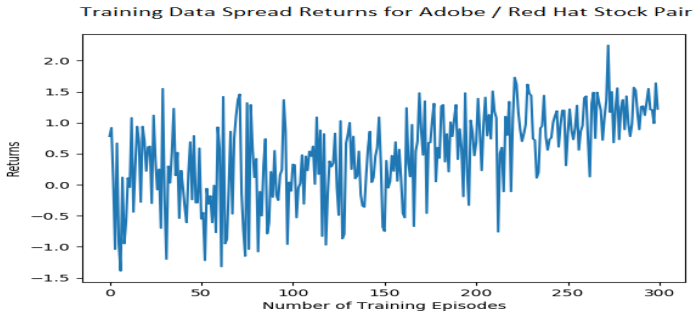


Figure 11: Training Results, 300 episodes. As the number of training episodes approaches 300 on the X axis, the system is able to converge on returns per episode of 1.2 on the Y axis.

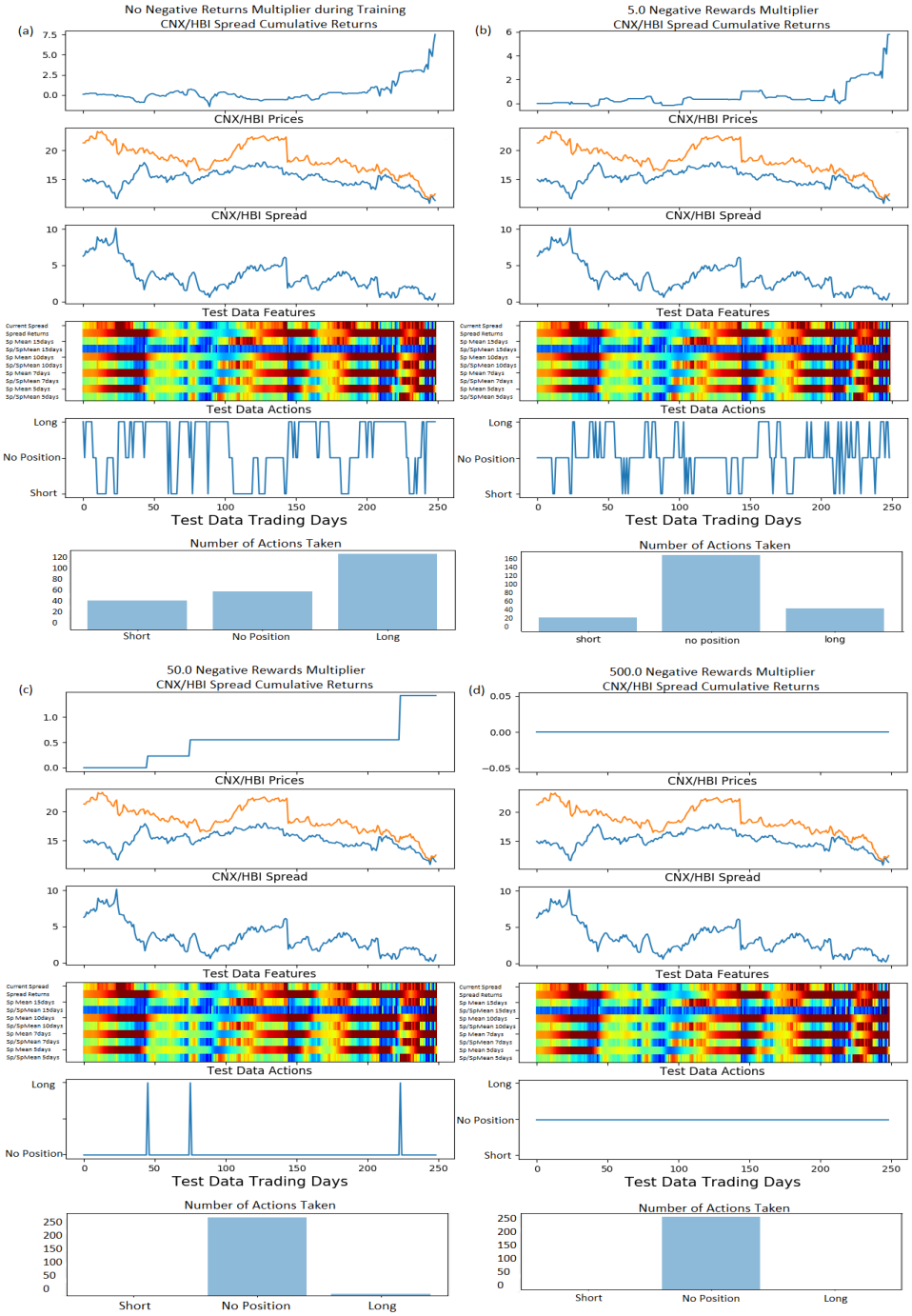


Figure 12: CNX/HBI returns decline as NRM increase, as the DDQN actions become more conservative. NR Multiplier of 50.0 (c), causes DDQN to only make successful predictions and yields returns of 1.42.

CHAPTER 5

Conclusion

The results of the first research paper show the DDQN tested on the largest 30 stocks of the S&P 500, yield an average of 13.2% geometric returns in 124 trading days from January 2, 2020 through June 30, 2020. The S&P 500 Index returns during this same time yield -4%. On average, the DDQN yields higher geometric returns than the S&P 500 Index, by 17.2% in six months. Results also show, through the use of feature map visualizations, that neuron excitement during the 22 days following the coronavirus stock market crash, increases on the recent regions in the candlestick image and decrease on the other regions. Results also show, through the use of dummy variables in testing for equality between sets of coefficients, that a DDQN is able to switch its attention from all days in a candlestick image to the most recent days. To test whether the ability of a DDQN to outperform the S&P500 Index is due to its ability to shift its attention, 20 logistic regressions are run. The results of the logistic regressions shown that changes in neuron excitement can forecast positive returns. In this paper, the shift in neuron excitement to the recent 7 regions is the strongest predictor of positive returns. This work not only validates statistical based trading strategies, but also provides a successful use case for candlestick images.

It is shown in the second paper, that a pairs trading strategies can be improved by analyzing the causal dynamics in a highly cointegrated pair. Our trading strategy, which allocated funds based on a leader follower relationship, yielded 17% annual returns, versus the traditional strategy's 15%. In the third paper, the DDQN was able to predict the spread of the Adobe/Red Hat pair for positive returns in the testing data set. The DDQN learned to take actions more conservatively by adding a Negative Reward Multiplier to the training process. The NRM reduced the number of trades resulting in negative returns. The input features provided to the DDQN allowed it to learn the spread mean reversion pattern for intervals of 15 days or less.

This dissertation has shown that AI techniques are effective at predicting the stock market. Specifically, it has shown that Double Deep Q-Networks are able to learn and execute trading strategies for positive returns. Additionally this dissertation has provided methods to explore how a DDQN has learned, namely feature map visualizations and the NRM training parameter. The implications of this dissertation will be to improve, and better understand, AI techniques in stock market trading strategies.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[3] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 2, no. 4.

[4] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[6] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, vol. 2, no. 11, p. e7, 2017.

[7] A. Nguyen, J. Yosinski, and J. Clune, *Understanding Neural Networks via Feature Visualization: A Survey*. Cham: Springer International Publishing, 2019, pp. 55–76. [Online]. Available: https://doi.org/10.1007/978-3-030-28954-6_4

[8] T. Okuyama, T. Gonsalves, and J. Upadhay, "Autonomous driving system based on deep q learnig," in *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*. IEEE, 2018, pp. 201–205.

[9] Y. Zhang, P. Sun, Y. Yin, L. Lin, and X. Wang, "Human-like autonomous vehicle speed control by deep reinforcement learning with double q-learning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1251–1256.

[10] X. Han, H. He, J. Wu, J. Peng, and Y. Li, "Energy management based on reinforcement learning with double deep q-learning for a hybrid electric tracked vehicle," *Applied Energy*, vol. 254, p. 113708, 2019.

[11] V.-H. Bui, A. Hussain, and H.-M. Kim, "Double deep q-learning-based distributed operation of battery energy storage system considering uncertainties," *IEEE Transactions on Smart Grid*, vol. 11, no. 1, pp. 457–469, 2019.

[12] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, S. U. Khan, and P. Li, "A double deep q-learning model for energy-efficient edge scheduling," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 739–749, 2018.

[13] K. Li, W. Ni, E. Tovar, and A. Jamalipour, "On-board deep q-network for uav-assisted online power transfer and data collection," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12 215–12 226, 2019.

[14] H. Sasaki, T. Horiuchi, and S. Kato, "A study on vision-based mobile robot learning by deep q-network," in *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. IEEE, 2017, pp. 799–804.

CURRICULUM VITAE

Andrew W. Brim

Education

- PhD: Computer Science, Utah State University, 2022
- MS: Financial Economics, Utah State University, 2019
- BS: Computer Science, Utah State University, 2006

Publications

- Brim, Andrew, Flann, Nicholas, S. Deep Reinforcement Learning Stock Market Trading, utilizing a CNN with Candlestick Images. PloS One Journal, 2022.
- Brim, Andrew. "Pairs Trading - Arbitraging Causal Dynamics using Fuzzy Logic", Decision Sciences Institute Annual Conference Proceedings, November 2021, Houston, TX.
- Brim, Andrew. "Deep Reinforcement Learning Pairs Trading with a Double Deep Q-Network." 2020 10th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2020.
- Brim, Andrew. "Deep reinforcement learning pairs trading." Master Thesis, Master of Science Financial Economics, Utah State University Library, 2019.
- Brim, Andrew. "Machine Learning Pairs Trading using Google Tensorflow", Decision Sciences Institute Annual Conference Proceedings, November 2018, Chicago, IL.