12-2019

Deep Reinforcement Learning Pairs Trading

Andrew Brim
Utah State University

Deep Reinforcement Learning Pairs Trading

Andrew Brim

May 2019

Abstract

This research applies a deep reinforcement learning technique, Deep Q-network (DQN), to a stock market pairs trading strategy for profit. Artificial intelligent methods have long since been applied to optimize trading strategies. This work trains and tests a DQN to trade cointegrated stock market prices, in a pairs trading strategy. The results demonstrate the DQN is able to consistently produce positive returns when executing pairs trading strategy.

1 Introduction

Pairs Trading is a statistical based trading strategy involving a pair of cointegrated financial assets [4, 3, 1]. This work presents a reinforcement learning system, utilizing a DQN and an RL environment in which to interact, to learn a trading strategy for a cointegrated pair of stocks.

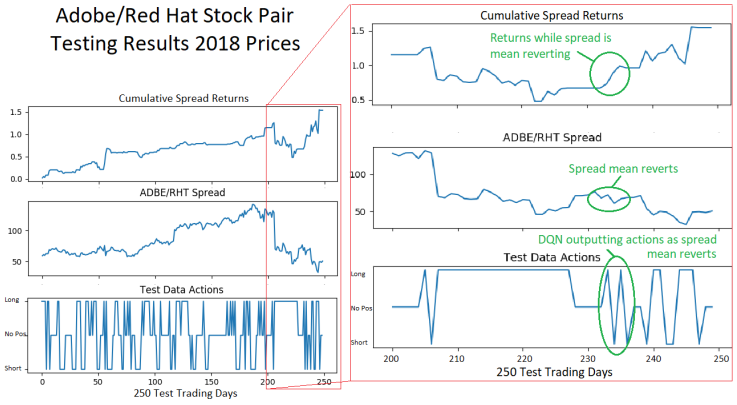

Figure 1: Pairs Trading Testing Results for the Adobe/Red Hat stock pair. The RL System is able to train a DQN with training data 2014-2017, and then test it's predictive ability on 2018 data. The DQN outputs actions of long, short, or no position on the spread, producing cumulative spread returns of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

A pairs trading strategy is a mean reversion strategy on the spread, or price difference, of two financial assets. 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 and then correctly predict it.

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 [8]. If the spread decreases to a given threshold the stock pair is traded by simultaneously entering into a short position for the lower price stock and a long position for the higher price stock as shown in Algorithm 1. When the spread of the cointegrated pair, reverts back to the mean, one or both positions will be profitable [4, 1]. Each position is held until the opposite position is entered into, or no position. Then the position is exited. 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.

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

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

$UPPER\_THREASHOLD \leftarrow 1.05$
$LOWER\_THREASHOLD \leftarrow 0.95$
if current_spread > mean_spread * UPPER_THREASHOLD **then**
  short higher_price_stock
  long lower_price_stock
else if current_spread < mean_spread * LOWER_THREASHOLD **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 [9, 7], to predict and trade the stock market. Liang, Chen, Zhu, Jiang, Li 2018 train a DQN to hedge portfolio risk [6]. Ding, Zhang, Liu, Duan 2015, train a Deep Learning system for event-driven stock prediction [2]. Li, Jiang, Li, Chen 2015 and Wu 2015 use Neural Networks for stock market predication as well[5, 10].

This RL system specifically utilizes a DQN to run a pairs trading strategy on 38 cointegrated stock pairs. The DQN interacts with a RL environment by taking the actions to long, short, or enter no position, on the spread of the stock pair. The DQN produces a Q-function, which learns the trading parameters to maximize the profit of the pairs trading strategy. This pairs trading strategy will execute based on 10 trading parameters which represent the spread, with a possibility of 30 values or more for each parameter, making the space of possible combinations of pairs trading parameters at least 5.9e+14. As the DQN 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 in the space.

2 Experiment

This system trains and tests a DQN 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 2014-2017, 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 10 input features for each day. These 10 features represent the state of the environment for that day. The 10 input features are received by the OPENAI Gym environment's step function, as a state. The features are inputted into the DQN, and the DQN outputs an action of long, short, or no position on the spread of the pair. The step function is called with this new action, and the environment returns the next state as well as the reward for the action taken on the previous state. The DQN updates its input weights to maximize reward, and the process is repeated. The DQN utilizes RELU non-linear activation functions and an Adam optimizer to update network weights. The DQN trains for 300 episodes. Figure 4 illustrates, for the training data set for the Adobe/Red Hat stock pair. The DQN 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.

The rewards in the OPENAI Gym are calculated as follows:

$$training\ rewards = action \times spread\ returns \times negative\ returns\ multiplier$$

$$testing\ rewards = action \times spread\ returns$$

As seen in Figure 6, the DQN outputs actions to maximize reward. Another parameter is introduced, during training, the Negative Returns Multiplier, which multiplies any negative spread returns, to make the rewards much more negative. This causes the DQN 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.

Figure 8 shows the total spread cumulative returns for all 38 stock pairs, for various Negative Returns Multiplier values from 1 to 1000. The higher the Negative Returns Multiplier, the more likely the DQN system will take a no position action when the state is less predictable. This decreases the total spread cumulative returns, but it results in a DQN which more consistently takes actions that produce positive returns. Figure 9 displays total cumulative returns decreasing as Negative Returns Multiplier increases to 1000. Figure 10 shows how the DQN takes an action of no position more often, until it receives a state it can more likely predict to gain a positive reward.

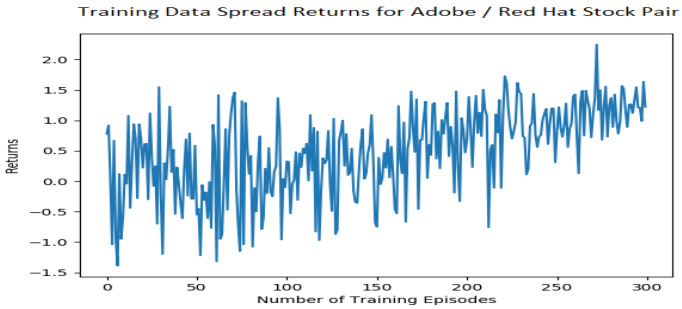

Figure 3: Training Results, 300 episodes.

3 Results

This method is applied to all 38 stock pairs, and tested on 2018 stock prices, consisting of 250 daily prices. As shown in the first column of Figure 8, the total cumulative returns for all 38 stock pairs is 131.33. It is clear the RL system, utilizing a DQN interacting with an RL environment, is able to learn and predict spread movements. Figure 4 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.

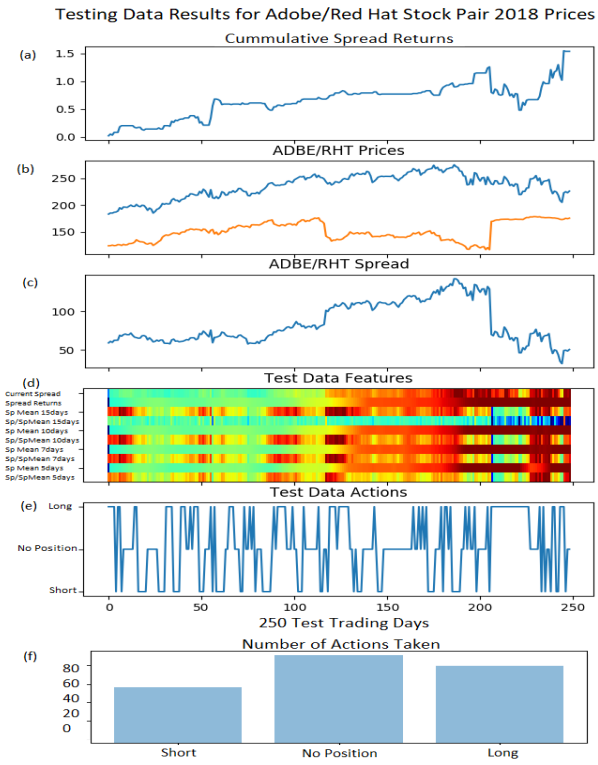

Figure 4: Testing Results for ADBE/RHT. (a) shows DQN trading returns of 1.58. (b) shows the prices of ADBE and RHT. (c) shows the spread of ADBE and RHT. (d) Heatmap of DQN input features for test data ADBE/RHT. (e) shows the actions output by the DQN. (f) shows the number of each action taken.

As illustrated in Figure 4(c) ADBE/RHT Spread, the spread mean reverts at least 3 times in the last 50 trading days. The DQN 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 8, 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 6. The lowest 2 performers were ESV/RRC -0.78 and ESV/GNW -9.64 as shown in Figure 7.

4 Methods

This work utilizes Q-learning, to build an optimized policy function for each state in the space of trading parameters. It utilizes a Deep Q-Network, to build

a Q-function which will take a state of 10 features as input, and output the best action on the spread: long, short or no position. The RL Environment simulates the stock market, and returns a reward tomorrow on the action taken today.

4.1 Q-Learning

Q-Learning is a reinforcement learning technique which optimizes the best action for a given state. Here, an OPENAI Gym environment is built to simulate the stock market pairs trading strategy, and allow the DQN to take long, short, or no position actions on the spread, to learn the state space. The DQN outputs an action, that is sent to the Gym environment, which then returns the next state and the reward for the action taken.

4.2 Deep Q-Network

DQNs utilize a neural network to generate a Q-function. The input features for this DQN 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 3. 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 DQN outputs the action to take at that point in time: long, short, or take no position on the spread of the stock pair. The DQN is able to build a Q-function, to maximize reward based on the input features it receives.

DQN replay memory is utilized to store 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 DQN.

4.3 Neural Network Structure

This experiment utilizes a Pytorch NN 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 3 .

5 Summary and Future Work

The DQN was able to output actions which earned a total cumulative returns for all 38 stock pairs of 131.33. The DQN 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.

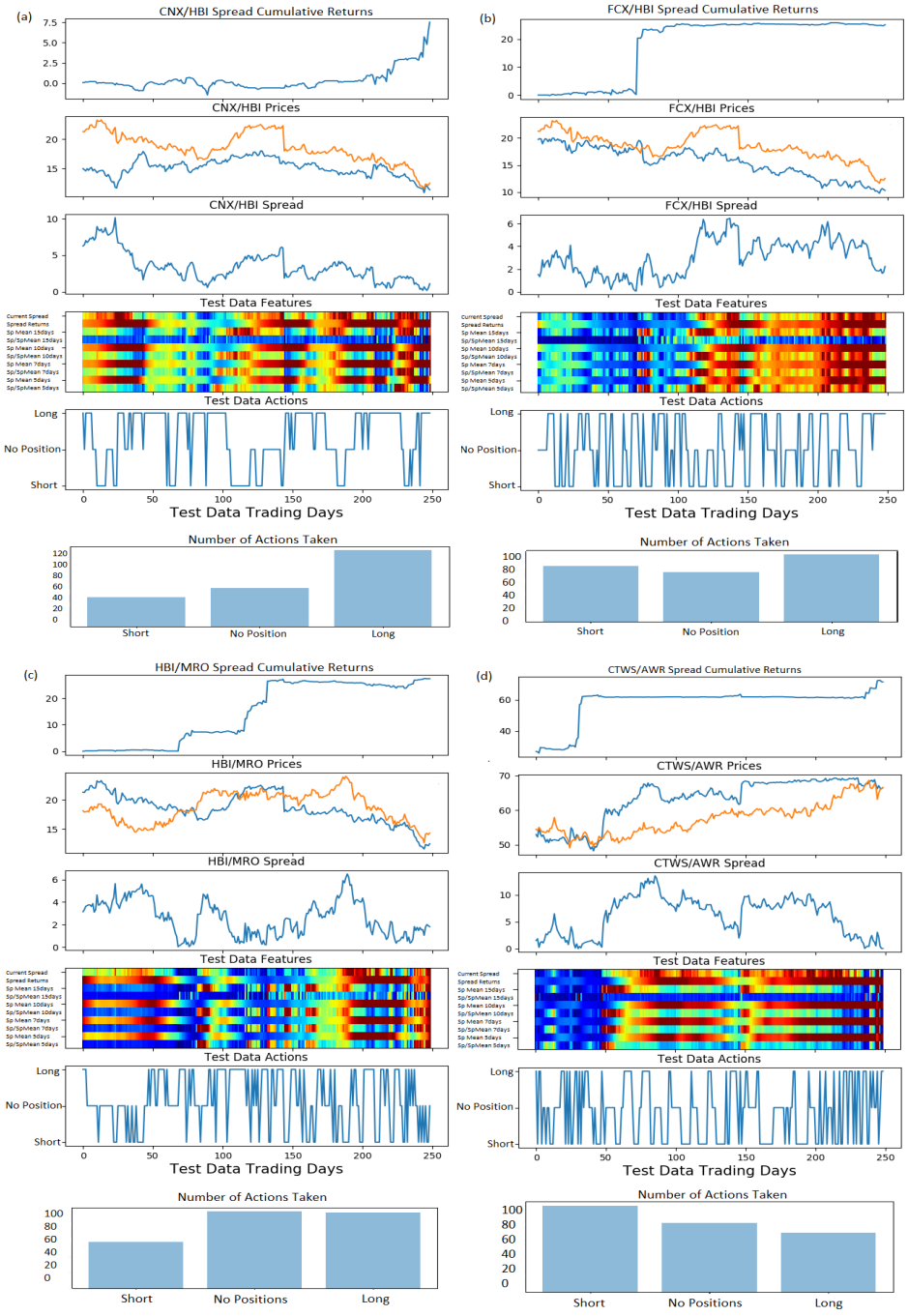

Figure 5: Top 4 DQN Pairs Trading performers: CNX/HBI 7.52 (a), FCX/HBI 25.67 (b), HBI/MRO 27.41 (c), CTWS/AWR 71.28 (d)

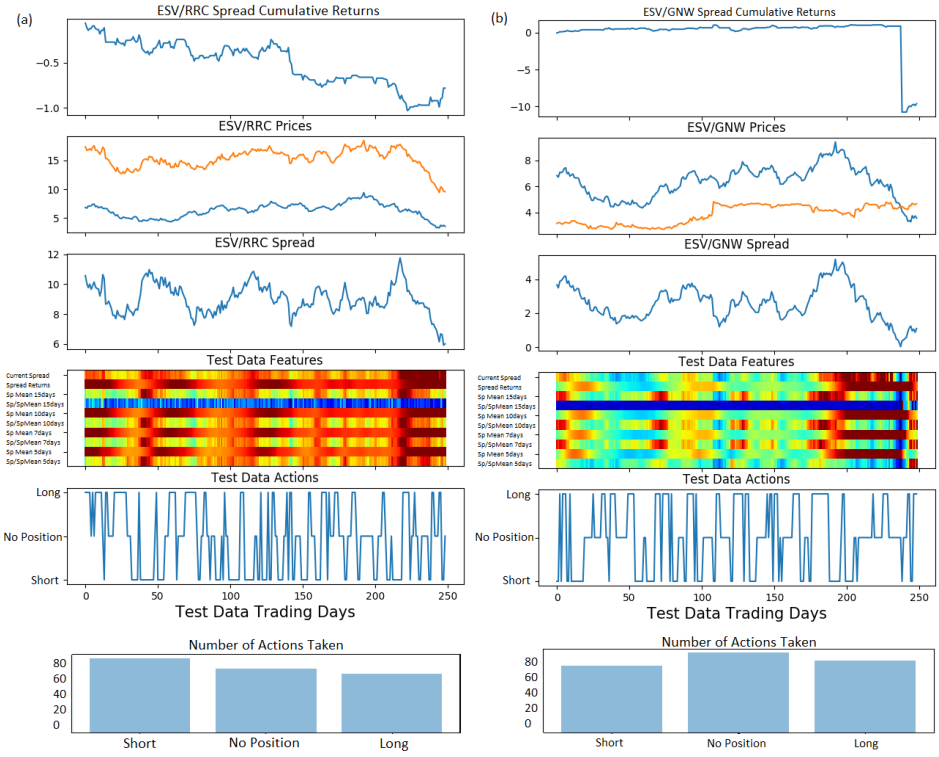

Figure 6: Bottom 2 DQN Pairs Trading performers: ESV/RRC -0.78 (a), ESV/GNW -9.64 (b)

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 7: 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 factor in row 1. The higher the negative rewards multiplier, the more often the DQN will take a conservative action of no position. A value of 0 returns indicates the DQN took no trading actions.

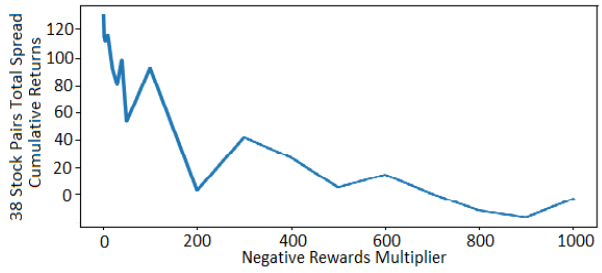

Figure 8: Returns approach zero as the multiplier increases, causing the DQN to take more no position actions.

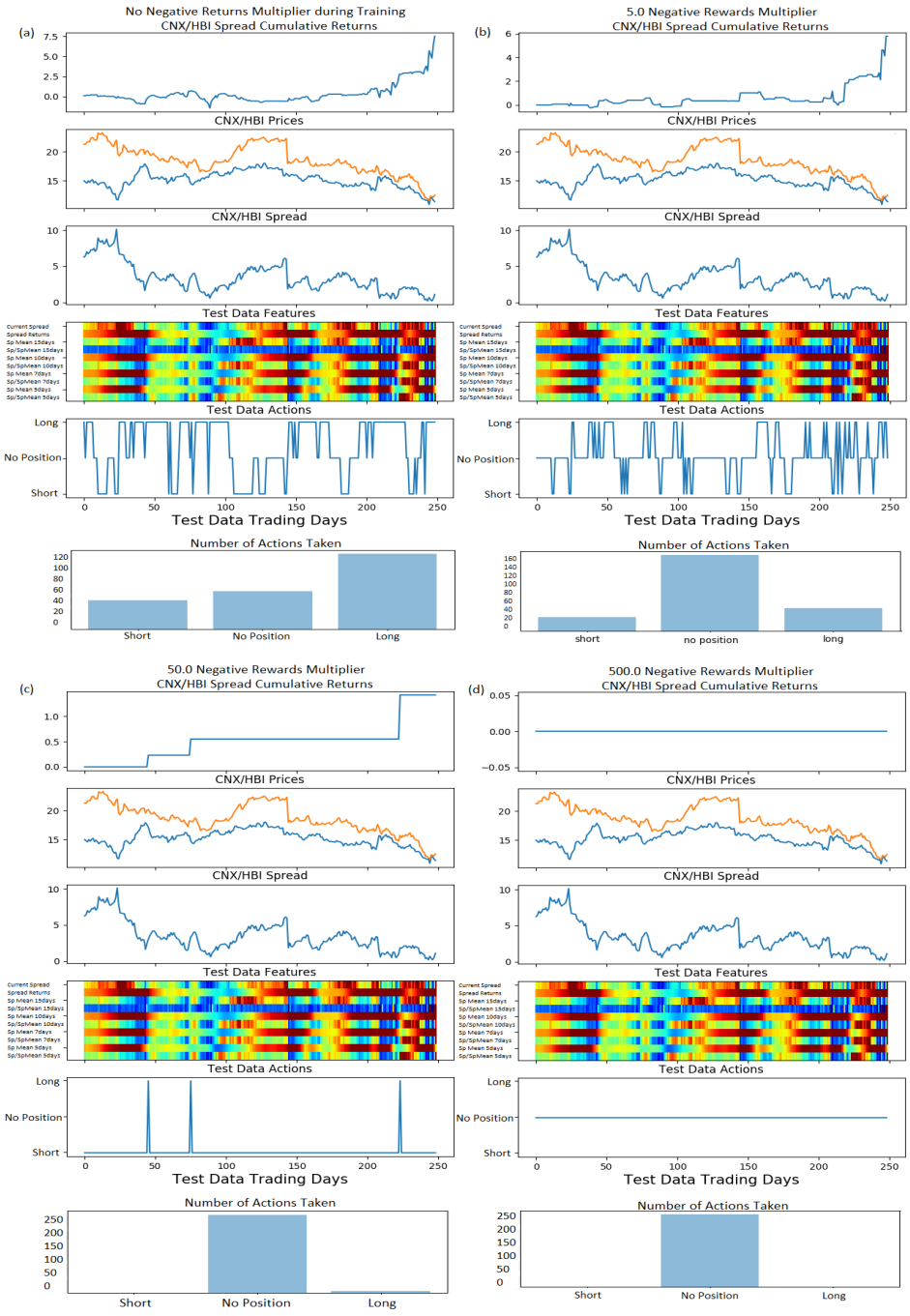

Figure 9: CNX/HBI returns decline as negative returns multipliers increase, as the DQN actions become more conservative. NR Multiplier of 50.0 (c), causes DQN to only make successful predictions and yields returns of 1.42.

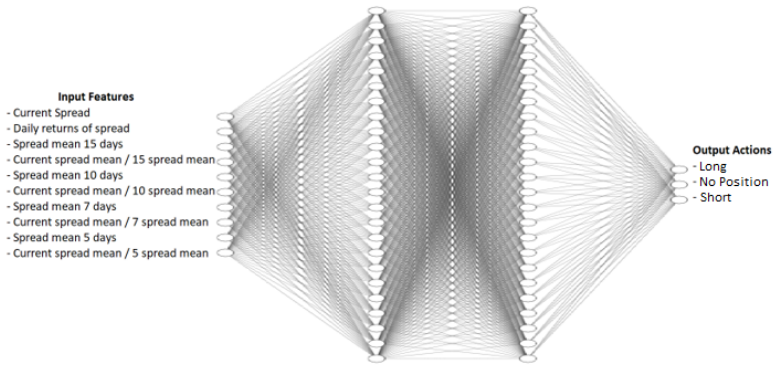

Figure 10: DQN NN structure

References

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

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

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

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

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

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

[7] 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.

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

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

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