Financial Stock Trading based on Deep Reinforcement Learning

LiangLiang ZHENG
Vrije Universiteit Brussel
zheng.liangliang@vub.be

Abstract

There are multiple reinforcement learning (RL) algorithms that have been used and reached successful results. The first part of this report described a customized stock market environment based on the environment design of OpenAI. The second part introduced and tested MACD, DQN, Dueling DQN, DDPG, DDPG with LSTM based on the stock price from 2012 to 2019. The experiments without any strategies indicate that the 3-layer linear layer DDPG can not capture useful information from the historical data. While DQN and DDPG with LSTM follow the stock trend. With the help of Long Short strategy and 53 strategy, it is found that DDPG outperformed all of the algorithms by delivering almost 30% of profits, the training with Long and Short strategy make the investing actions more conservative than without it. The overall experiments suggest that LSTM based DDPG is the most robust model and CNN based DQNs are sensitive to external strategies.

1 Introduction

Efficient Market Hypothesis (EMH) is introduced by Fama in the domain of finance, decades later, Lo (2004) proposed Adaptive markets hypothesis (AMH) which includes the idea that: Individuals made decisions based on their own benefit

- Individuals in the market will make mistakes
- Individuals in the market will learn and adapt
- Competition leads to individual adaptation and renewal
- Natural selection shapes the market ecology, and evolution determines market dynamics.

According to EMH and AMH, individuals can react according to the changes or states of the market. The actions or strategies that could be taken are going long, going short, or going short position, also means the action spaces hold, buy, and sell. Based on the properties, reinforcement learning could be used in this context.

In the financial market, it is of importance to take wise and strategic actions to buy or sell when participating the stock trading since it directly influences the overall profit. There is no single perfect approach that fits all of the trading situations, therefore finding a sound and optimal strategy is significantly tricky even for stock trading experts.

2 Literature Review

Generally, in financial stock trading, technical analysis and algorithmic trading are mostly used in the academia and business area, in the algorithmic trading area specifically where scientists are using a mathematical model to execute the stock trading automatically.

Reinforcement Learning was first being applied to the stock marketing by John Moody [1998] who built a trading and portfolio management system by using recurrent reinforcement learning (RRL) algorithms. In 2006, Nevmyvaka and his colleagues used Q-learning and Dynamic programming in the financial markets based on 1.5 years of millisecond time-scale limit order data from NASDAQ[Nevmyvaka *et al.*, 2006].

Due to the extensive use of deep learning, many scientists have begun to apply deep reinforcement learning in automatic stock trading. In 2016, researchers from Tsinghua university introduced a recurrent deep neural network for real-time financial signal representation and combining it with reinforcement learning for online trading [Deng *et al.*, 2016]. In 2018, Huang we propose a Markov Decision Process (MDP) model suitable for the financial trading task and solve it with the state-of-the-art deep recurrent Q-network (DRQN) algorithm employed with a substantially small replay memory inside [Huang, 2018]. Recently, researchers from Oxford adopted deep reinforcement learning algorithms to design new trading strategies for continuous futures contracts and investigate how performance varies across different asset classes including commodities, equity indices, fixed income and FX markets [Zhang *et al.*, 2020].

3 Stock Environment

3.1 Structure of the Environment

In this report, a customized stock trading environment is built, which composed of 3 functions. The first is *reset* function, in which the information of the stock and account such as initial money in the account, number of stock user holds, number of stock bought so far and total profit is initialized. The

second function defined as *next_observation*, which will return a certain state of a given time. The final function is the most important one, defined as *step*, the working pattern of the function *step* is similar as most of the environments defined in Gym, which takes an action and return next state, reward, and termination.

3.2 Strategies

There are 2 basic strategies used inside the environment, but these will not be used by default. the first one is to Long and Short, which compares the difference between the close market price of the day and the close market price in the morning if the close market price of the day is higher than that of the morning, then sending the Long signal, if not, then sending Short signal.

The second one is called 53-strategy. 5 means if the profit percentage is higher or equal to 50 percent of the initial account cash, then stop gaining profit in this way to avoid stock price drop. 3 means if the percentage of loss is higher or equal to 30 percent of the initial account cash, stop losing more money.

4 Methodology

4.1 MDP Formalisation

The stock trading process could be formalized as Markov Decision Process (MDP), the action space, state-space and reward function in the MDP will be introduced as follows.

Action Space

For the sake of clarification, it is of vital importance to mention the action spaces. According to typical stock trading actions, discrete action spaces are used in this report. Hold, buy, sell are denoted as $\{0, 1, 2\}$ in this environment. The *step* function will run corresponding actions provided by the agent.

State Space

The states are the everyday closing price of a certain stock, in this experiment, the state is a list of 3-month prices, therefore a state space of 90 will be used.

Reward Function

The reward of taking action a_t is gained by taking the expectation of the rewards $r\left(s_t, a_t, s_{t+1}\right)$, plus the expected reward in the next state s_{t+1}. Based on the assumption that the returns are discounted by a factor of γ.

Reward returned from Environment

The reward returned from in the environment is not purely the amount of money made so far. First of all, the reward is calculated by the percentage of the price trend, specifically it is defined as the equation .

$$r_{t+1} = \frac{p_{t+1} - p_t}{p_t} \tag{1}$$

Besides, considering that there are some meaningless rise and fall in the trading market, for example, after a small percentage of rising, the price starts falling sharply. The large percentage of rising or falling will also influence the agent's judgment. Therefore it is necessary to wipe out those situations and extract reasonable trends. The reward is calculated as the code below, b1, b2, b3 are the discount(discount) value, defined as 0.2, 0.7, and 0.1 respectively.

```
def reward(reward):
    r = np.abs(reward)
    if r<=0.015:
            r = reward * b1
    elif r<=0.03:
            r = reward * b2
    elif r>=0.05:
        if reward < 0 :
            r = (reward+0.05) *b3 - 0.05
        else:
            r = (reward-0.05) *b3 + 0.05
```

5 Data Description

Stock data could be retrieved from Chinese stock online and saved in a CSV file. The stock used in this experiment is NORINCO International Cooperation Ltd (000065), the stock price is from Jan 2012 to Dec 2019, 1796 days in total, information such as date, open price, close price, and morning close price is extracted. The first 1500 days will be used for training and the rest 296 days will be used for backtesting.

6 Baseline and RL Models

6.1 Moving Average Convergence Divergence (MACD)

Moving Average Convergence Divergence, also known as MACD, is an indicator that represents the market trend. It shows the relationship between 2 main moving averages of price. The MACD is calculated as shown in Equation 2:

$$\begin{aligned} E_{12} &= E'_{12} \cdot \frac{11}{33} + P \cdot \frac{2}{13} \\ E_{26} &= E'_{26} \cdot \frac{25}{27} + P \cdot \frac{2}{27} \\ \mathrm{d} &= E_{12} - E_{26} \\ D &= D' \cdot \frac{8}{10} + \mathrm{d}\,\frac{2}{10} \\ M &= 2(\mathrm{d} - D) \end{aligned} \tag{2}$$

where E_{12} denotes as the 12-period Exponential Moving Average (EMA) and E_{26} is the 26-period EMA. d is the difference value between E_{12} and E_{26}, which is also the standard of MACD, and D denotes as the average of DIF in 9 days. And m denotes as MACD value. When MACD is greater than 0, then a buy signal will be sent, otherwise, a sell signal will be sent. The MACD will be used as a baseline in the experiment setting.

6.2 DQN

The first Reinforcement Learning algorithm is Deep Q Learning. The policy of this problem has been replaced by a neural network, the DQN is designed as a 5-layer of 1D convolutional neural network with batch normalization and dropout

Layers	Configuration	Output
input		(?, 1, 90)
conv1	K: 2 D: 1 S: 1	(?, 2, 89)
bn1		(?, 2, 89)
dropout1	0.25	(?, 2, 89)
conv2	K: 2 D: 2 S: 1	(?, 4, 87)
bn2		(?, 4, 87)
dropout2	0.25	(?, 4, 87)
conv3	K: 2 D: 4 S: 1	(?, 4, 83)
bn3		(?, 4, 83)
dropout3	0.25	(?, 4, 83)
conv4	K: 2 D: 8 S: 1	(?, 4, 75)
bn4		(?, 4, 75)
dropout4	0.25	(?, 4, 75)
conv5	K: 2 D: 16 S: 1	(?, 32, 59)
bn5		(?, 32, 59)
dropout5	0.25	(?, 32, 59)
fc1	(1888, 1024)	(?, 1024)
dropout6	0.25	(?, 1024)
fc2	(1024, 512)	(?, 512)
dropout7	0.25	(?, 512)
fc3	(512, 3)	(?, 3)
dropout8	0.25	(?, 3)

Table 1: Architecture of DQN

technique. The architecture configuration is shown in table 1, where K stands for kernel size, D is dilation size and S is all used default value 1. In the layer part, bn is the abbreviation of the batch normalization layer. The activation function used after every convolutional layer is Rectified Linear Units (ReLU) and scaled exponential linear units (SELU) after every linear layer.

6.3 Dueling DQN

The second one is Dueling DQN [Wang *et al.*, 2015], which is similar to the DQN but using 2 branches, 1 for predicting the state value with 1 dimension and the other for predicting the action advantage value. The reason Dueling DQN is used here is that the state function and advantage function is extracts could be used to predict the importance of the predicted the importance of the state and action. Therefore the definition of Q function is divided into state function and advantage function. The equation is defined in 3.

$$\begin{aligned} Q(s,a;\theta,\alpha,\beta) &= V(s;\theta,\beta) + (A(s,a;\theta,\alpha) - m) \\ m &= \sum_{a'} A\left(s,a';\theta,\alpha\right)/|A| \end{aligned} \tag{3}$$

where V is the state function, the output is a scala and A is the advantage function with the output size of action space. θ denotes as the parameter of the convolutional layer and α and β denotes as the parameter of the fully connected layers in the end.

6.4 Deep Deterministic Policy Gradient

Introduction of DDPG

Deep Deterministic Policy Gradient [Lillicrap *et al.*, 2015], also known as DDPG, is an improved version of the Deterministic Policy Gradient because it uses neural networks as a function approximator. The key reason that DDPG is used in the stock trading in this report is that it used a greedy action to maximize Q function for the next state as shown in equation 4.

$$Q_\pi = E_{s_{t+1}}\left[r\left(s_t,a_t,s_{t+1}\right) + \gamma \max_{a_{t+1}} Q\left(s_{t+1},a_{t+1}\right)\right] \tag{4}$$

In detail, there are actor-network and critic network inside DDPG. Actor-network, defined as $\mu\left(s|\theta^\mu\right)$ is responsible to use state s_0 to predict next action a_1, where θ^μ denotes parameters of actor-network. The critic-network $Q\left(s,a|\theta^Q\right)$ generate the value of action under next state s_1 and next action a_1. During training, the target network of actor and critic, denoted as Q' and μ', are updated by soft copying the parameters of actor and critic respectively.

Similar to the DQN mentioned before, an experience buffer will be used to store transitions when training. At each iteration, the network takes an action takes an action a_t on s_t, and then receives a reward based on s_{t+1}. The parameter information (s_t, a_t, s_{t+1}, r_t) is then stored in experience buffer E. The critic network is then updated by applying Mean squared error over the expected difference $M\left(\theta^Q\right)$ between outputs of the target critic network Q' and the critic network Q. The actor network is updated by optimizing the loss function defined in equation 5.

$$\nabla_{\theta^\mu} J \approx E_{s_t\sim\rho^\beta}\left[\nabla_{\theta^\mu} Q\left(s,a|\theta^Q\right)\big|_{s=s_t,a=\mu(s_t|\theta^\mu)}\right] \tag{5}$$

The intuition behind the equation is that to get larger Q value, the loss should be as small as possible, therefore simply taking the nagative of the expected value of critic network based on the state s_t and action a_t produced by actor function μ.

DDPG Architecture used in this report

In this report, there are 2 versions of DDPG architectures . The first version, denoted as DDPG_V1, is composed of pure 3 linear layers following by ReLU activation. The size of the hidden layer is 128. The configuration is the same for actor and critic network.

Layers	**Configuration**	**Output**
Input		(1, ?, 90)
LSTM1	l=2 h=64	(1, ?, 64)
LSTM2	l=2 h=32	(1, ?, 32)
LSTM3	l=2 h=3	(1, ?, 3)
output	squeeze(dim=0)	(?, 3)

Table 2: Architecture of the DDPG_V2

The second version, denoted as DDPG_V2, used 3 2-layer Long short-term memory (LSTM) for both critic and actor

network. The architecture is shown in table 2, where l stands for number of layers of the LSTM, and h stands for the size of the hidden layer.

7 Experiments and Backtest results

The first experiment is to test the baseline in the training data and test data, noticed since the strategy is fixed, there is no need to train the data in multiple episodes. Figure 2 is the MACD graph.

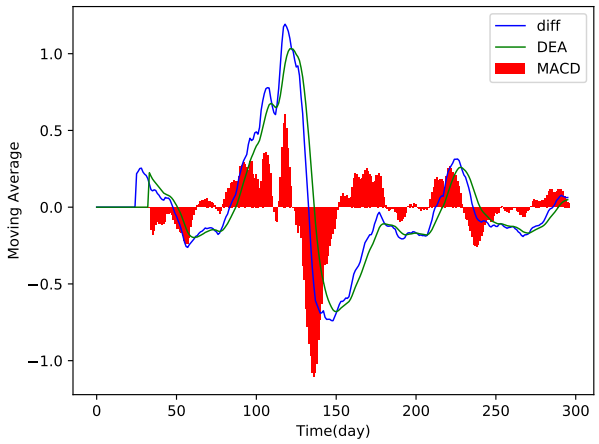

Figure 2: Action taken by MACD in 296 days

The MACD strategy regulates the action taken situation that if the value is greater than 0, then a buy signal will be sent, otherwise a sell signal will be sent. Even though it is a naive strategy, it is verified that MACD an indicator to show if an action is well chosen in the later experiments. The action taken by MACD is shown in figure 1.

The experiment setting for the RL algorithms is shown below in table

Model	Optim	batch	γ	Buff	α_{critic}	α_{actor}
DQN	Adam	200	0.9	1000	0.02	–
DDQN	Adam	200	0.9	1000	0.02	–
DDPG	Adam	20	0.99	500	0.001	0.001

Table 3: Experiment Setting for RL algorithms

The 4 different RL algorithms, DQN, Dueling DQN, DDPG, and DDPG using LSTM are trained in 500 episodes based on the first 1500 days of stock data. The result of the average cumulative reward in every episode is shown in figure 3. In the training set, it is obvious that DDPG with LSTM algorithms and DQN algorithms performed the best, in other words, almost making money all the time. Specifically, the reward of DDPG LSTM and DQN made more than 1.5 times more money (reward has already been shrunk) after 500 episodes. The pure 3-layer linear DDPG algorithms continue to perform the worst, the average cumulative reward goes to the negative. For the sake of clarification, it is necessary to mention that the alphabet 'w' in the following labels inside the graph stands for 'without strategies'.

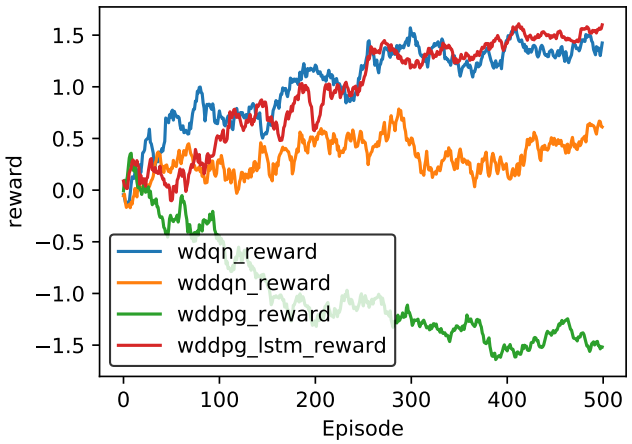

Figure 3: Average Cumulative Reward vs Episodes based on 1500 days of data

Backtesting is a way to evaluate the effectiveness of a trading strategy by running the strategy in history data or new data. Figure 4 shows the result of the backtest, the line labeled as 'stock' stands for the trend of the stock, it can be found that MACD is losing money by strictly followed the MACD graph. It is noticeable that DDPG keeps holding the money until day 200, which indicates that the 3-layer linear layer is weak in capturing useful information from the historical data. While DQN and DDPG with LSTM (denoted as 'wddpg1') follow the stock trend and make more than 60% of profit compared to the initial money in the account around day 75. Same as DDPG, Dueling DQN does not follow the stock trend and the profit stays in 0.2 in this backtest.

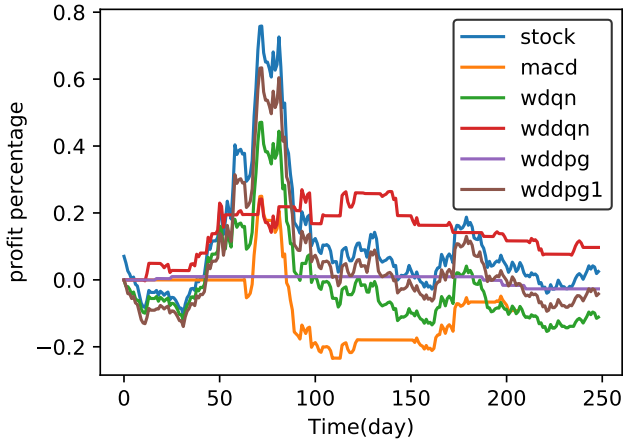

Figure 4: Profit percentage vs Day in backtest

The trading strategies rely only on the output of the model,

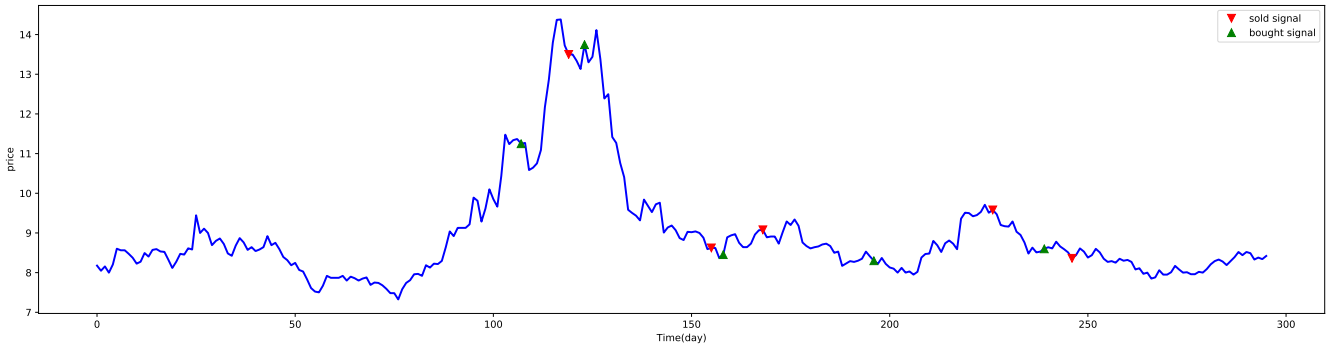

Figure 1: MACD indicator for the rest 296 days

as mentioned before, there are 2 strategies implemented in the environment, one is use Long and Short strategy and the other is 53 strategy. The next experiment is to test how the RL algorithms will perform with the strategies embedded.

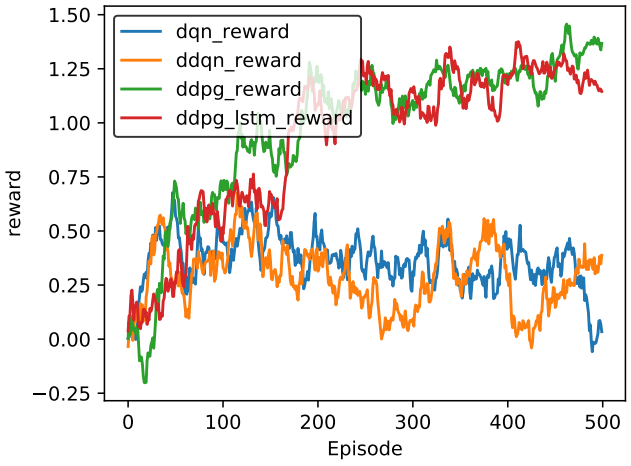

Figure 5: Average Cumulative Reward vs Episodes based on 1500 days of data with strategy

Figure 5 shows the average cumulative rewards of the 4 algorithms using Long and Short strategy and 53 strategy. After using the strategies, DDPG and DDPG with LSTM performed the best. Part of the reasons is that the influence of Long and Short strategy could guide the algorithm to converge the right direction by returning higher rewards. And the 53 strategy work as a safety net to prevent the agent to lose more money. After introducing strategies, the DQN and Dueling DQN both performed mediocrely, compared with the results from figure 3, it can be found that CNN based DQN algorithms are sensitive to the external changes since CNNs are more used for image-based tasks.

Figure 6 illustrates the profit percentage in 250 test days. After using the Long and Short strategies, the MACD stays the same because the Long and Short also follows the principle of diff and DAE line. It is interesting to see that DQN performed more conservative than without using the strategies, holding money longer than without Long Short strategies. From this graph, it proved again that Long Short strategy provides a good direction for DDPG to learn the trend of the stock, which is the key that the reward of DDPG goes up. Same as without using the strategy, DDPG with LSTM (denoted as 'ddpg1' in the graph) is robust to the external strategies.

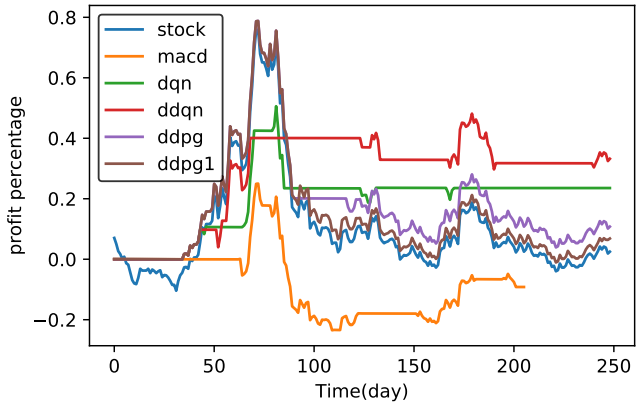

Figure 6: Profit percentage vs Day in backtest

8 Conclusion

In this report, a stock environment ensemble gym environment was first introduced and built, the reward inside the stock environment is calculated by multiplying a shrink factor to avoid falling into a 'short-sighted' trap. In the experiment part, 1 traditional trading algorithm and 4 RL algorithms are built and tested. The results suggest without the external strategies, that the 3-layer linear layer DDPG can not capture useful information from the historical data. While DQN and DDPG with LSTM can follow the stock trend perfectly. With the Long and Short strategy and 53 strategy, the DQN a Dueling DQN become more conservative than without it. The overall experiments suggest that LSTM based DDPG performed the best in both with and without external strategies.

However, from the experiment results, it seems CNN based

DQNs are sensitive to external strategies, this might come from the fact that the use of CNN in DQN is not a wise decision because CNN is more used in image-based tasks, besides, the intuition behind the convolutional computation over the 90 days of close market price, that is to say, recognize the historical data as an image, is not clear enough to justify the use of it.

There are many other indicators or strategies (eg: Relative Strength Index) could be used in the environment, while the purpose of this experiment is to investigate whether different deep reinforcement learning algorithms can learn the features and provide a correct signal when conducting trading in the market, therefore only 2 simple strategies are applied. The results also suggest that LSTM based algorithm can better capture the trend of the stock price, the example of LSTM being applied in time series could be generalized or used in similar tasks. (eg: Foreign Exchange (FX) market markets)

The disadvantage of using Deep RL for predicting the price based simply on the previous close market price is obvious. For example, the lack of global market analysis or that of considering natural disasters (eg: COVID-19, earthquake) will result in losing money even though with some safety net strategies, therefore, the combination of using Natural Language Processing techniques to investigate sentiment analysis in the social media and Deep RL could be the next research direction.

References

[Deng *et al.*, 2016] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664, 2016.

[Huang, 2018] Chien Yi Huang. Financial trading as a game: A deep reinforcement learning approach. *arXiv preprint arXiv:1807.02787*, 2018.

[Lillicrap *et al.*, 2015] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[Moody *et al.*, 1998] John Moody, Lizhong Wu, Yuansong Liao, and Matthew Saffell. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, 17(5-6):441–470, 1998.

[Nevmyvaka *et al.*, 2006] Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, pages 673–680, 2006.

[Wang *et al.*, 2015] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[Zhang *et al.*, 2020] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deep reinforcement learning for trading. *The Journal of Financial Data Science*, 2(2):25–40, 2020.