The University of Hong Kong

Department of Computer Science

COMP4801 – Final Year Project

Interim Report

AI Application on Algorithmic Trading

Supervisor: Dr. S.M. Yiu

Name: Chan Wing Hei

UID: 3035186243

Date: 18/04/2018

Acknowledgement

I would like to present my earnest gratitude to my supervisor Dr. S.M. Yiu, who granted me the valuable opportunity of working on this project about Artificial Intelligence, which is one of research fields that are showing great potential in this generation. Meanwhile, Dr. Yiu has given me his precious opinions which gave me directions to work on this project.

Abstract

Artificial intelligence has been one of the most extensively researched topics in recent decades, and its power has been proven by the victories of the AI Go player AlphaGo over the top human players over the world. In fact, not restricted in playing Go or chess games, artificial intelligence has already been applied to different fields in our societies and daily lives. Algorithmic trading is one of the applications which draw most attention from researchers. In this research, there will be attempts to perform algorithmic trading with mainly three artificial intelligence techniques – Recurrent Neural Networks (RNN), Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL). This projects aim at studying the factors that affect the performances of models built with the three techniques, and explore possible improvements to the models.

Table of Contents

List of Figures

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

Artificial Intelligence:

- AI – Artificial Intelligence
- CNN – Convolutional Neural Network
- DDQN – Double Deep Q Network
- DQN – Deep Q Network
- DRL – Deep Reinforcement Learning
- NN – Neural Network
- MRNN – Multi-layered Recurrent Neural Network
- QL – Q Learning
- RL – Reinforcement Learning
- RNN – Recurrent Neural Network
- RRL – Recurrent Reinforcement Learning
- SRNN – Single-layered Recurrent Neural Network

Stock:

- CADHKD – Canadian Dollar/Hong Kong Dollar Exchange Rate
- NASDAQ – NASDAQ Composite Index

Stock Indicator:

- BB: Bollinger Bands
- CCI: Commodity Channel Index
- MA: Moving Averages
- MA20: 20-day Moving Average
- MA50: 50-day Moving Average
- MACD: Moving Average Convergence Divergence
- %B: Percent B
- RSI: Relative Strength Index

Performance Metric:

- MAE: Mean Absolute Error
- RMSE: Root Mean Squared Error
- SD: Standard Deviation
- Var: Variance

1. Introduction

Artificial Intelligence has been one of the most frequently researched fields among computer scientists and mathematicians, and the potential of AI even surprised everyone all over the world after the epic Go matches between the AI player AlphaGo and the top human Go masters in the world. In fact, AI has been much more powerful than just playing Go or chess games, and AI has had much greater influences than everyone might have expected on us. Melody, the intelligent chatbot developed by Baidu, assists doctors in handling requests from patients and provide useful information and suggestions on diagnostics of patients. In the meantime, there are also many AI's developed to enhance the productivity, or create alerts to potential risks in the workspace [1].

Among the wide variety of applications of AI, algorithmic trading is also one of the most intriguing, and profitable, research topics which has attracted attention from numerous researchers and investors. Under the ever-changing financial market, investors are interested in exploring for some strategies that could assist them to make the right decisions to exploit the maximum benefits. In fact, there has been many attempts to model the fast-paced market with mathematical or statistical models, and adopting machine learning or artificial intelligence has been a common trend to sought for the best solutions. No matter which method is taken, the ultimate goal of this algorithmic trading problem is to develop an algorithm that can learn how to trade intelligently, and then make profits.

In the recent decades, many machine learning or AI techniques has been developed, both addressing the algorithmic trading problem or some other problems, and now people start to notice the potential of the promising deep learning techniques. Attempts have been made to adopt deep learning in predicting market trends with historical prices [2], while there are also attempts in relating market fluctuations with news and social events [3]. Recently, researchers started casting their eye-sights on reinforcement learning, which have revealed its great potential in learning how to play video games [4]. However, there is still not much study done on its application on the field of algorithmic trading.

This research will tackle the algorithmic trading problem with mainly two approaches – stock level prediction and portfolio management. Each of the approaches will be executed with AI learning methods like Recurrent Neural Networks, Reinforcement Learning and Deep

Reinforcement Learning. Efforts will be made to develop the AI models, and the results will be compared.

The structure of this report is organized as the followings. The section which is following immediately mainly reviews on studies made by other researchers on the topics related to artificial intelligence and algorithmic trading (Section 2). Being inspired from those studies, the next sections then describes the objectives of this project (Section 3) and how they will be carried out in this project (Section 4). Then, methodology of algorithms adopted will be thoroughly discussed (Section 5). Also, technical issues including the development tools (Section 6) and information about training data (Section 7) will be covered. After that, the structures and principles of the models built in this project will be explained in details (Section 8), as well as the experiment settings (Section 9). In this project, experiments are divided in the two stages which will be covered afterwards (Section 10 and 11). Finally, this report will be wrapped up by a brief conclusion on the insights gained from the experiments (Section 12).

2. Related Studies

Many studies have been done on the stock market, which usually seems complex to researchers and traders. Therefore, in order to have a better understanding to the market and hence develop strategies to make profit from it, researchers have been formulating the complex market into different models so that it will be easier for them to configure approaches to solve the algorithmic trading problem. The followings summarize some previous studies on how the problem is formalized and what algorithms are adopted.

2.1. Previous Problem Formalisms

After reviewing papers sharing the same interest, it can be observed that there are mainly 4 kinds of approach addressing algorithmic trading problem.

2.1.1. Time Series Forecasting

Forecasting future stock price levels is one of the most common methods used for algorithmic trading. It is not unexpected since one can easily make wise decisions if the future price movements are known. A. Moghaddam, M. Moghaddam and Esfandyari (2016) [5] investigated the optimal number of prior working days used for forecasting NASDAQ index. Wanjama and Muchemi (2014) [2] developed an

Artificial Neural Network model and tested its ability on predicting stocks traded in Nairobi Stock Exchange with more than 5-years of historic data. There are more researches on the capability of deep learning models in stock prediction.

2.1.2. Event-driven Forecasting

Other than exploiting relationships between past and future stock prices, there are also researches which conjecture that the market movements are in fact reflected in news or social events, and they can be studied for prediction. Ding, Zhang, Liu and Duan (2015) [6] extracted events from news texts with Natural Language Processing (NLP) techniques and achieved 6% improvements comparing with simply predicting with a deep Convolutional Neural Network (CNN) model.

2.1.3. Trading Strategy Optimization

The approaches mentioned above aim at making profit with knowledge to future price levels. In the meantime, there are other attempts which emphasize more on the strategy of trading but less on how the prices will change. Moody and Saffell (1999) [7] compared effectiveness of learning such a strategy with different lengths of historical data, while Deng, Bao, Kong, Ren and Dai (2017) [8] attempted in learning such strategy with deep learning methods.

2.1.4. Portfolio Management

Not restricted to trading with a single stock, algorithms have also been developing for managing portfolios with multiple stocks, for example, Jin and El-Saawy [9] trained an improved strategy managing a simple portfolio with 2 stocks by adjusting hyper-parameters.

2.2. Learning Methods

Different algorithms have been adopted to implement the methods mentioned above. The followings are 3 of the most popular algorithms which have been found common, and is also the algorithms being studied in this project.

2.2.1. Deep Neural Networks

Neural Networks (NNs) are commonly used deep learning methods for predicting future stock prices. Poulos [3] compared the performances with different number of

hidden layers of neurons in the deep Recurrent Neural Networks (RNN) model developed for predicting Dow Jones Index. Grigoryan (2015) [10] predicted on Nasdaq OMX Baltic stock exchange with an Artificial Neural Network (ANN) model given with 3 years of historical data. In the meantime, Wanjama and Muchemi (2014) [2] also built an ANN model with 2 layers of neurons for prediction.

2.2.2. Reinforcement Learning

Reinforcement Learning (RL) is an algorithm which is similar to "learning by rewards" and it is suitable for learning trading strategies. Duerson, Khan, Kovalev and Malik (2005) [11] compared 4 different RL models such as Q-learning model and Short-term discounted history reinforcement model. In the studies by Kaur [12], he first formulated the trading problem with Hidden Markov Model, and then learnt to trade with RL.

2.2.3. Deep Reinforcement Learning

Deep Reinforcement Learning describes a class of AI methods with a combination of NN and RL, which attempts to take advantages from both techniques to provide better performances. DeepMind, one of the most successful AI researching institutions, was the first architect to use DRL to learn playing Atari games [13, 14, 15]. Lu (2017) [16] combined RNN with Long-Short Term Memory (LSTM) cells and RL to improve the learning performance in deep neural networks to train a better trading strategy.

There are also many other problem formulations and learning algorithms that are studied by researchers such as transfer learning for trading. In this project, some techniques have been chosen as the main focuses of study, which will be er explained in the following section.

3. Project Objectives

Titled as "AI Application on Algorithmic Trading", this project researches on both AI and trading. The objectives of this project can be summarized as below:

O1) To research and explore on AI techniques with different hypotheses
O2) To design and build AI models for the purpose of algorithmic trading
O3) To test performances of AI models

O4) To study factors affecting the performances and ways to improve

O5) If possible, build a profitable model

Section 2 presented some results from researches on different algorithmic trading and AI techniques (O1). In the next section, it will discuss the outline of how the objectives will be achieved throughout the entire project.

4. Project Outline

Being inspired by the studies of previous researches, 3 models have been built in this project, which used 3 algorithms (See Section 2.2) that adopted 2 approaches (See Sections 2.1.1 and 2.1.4): 1) Stock level prediction with RNN; 2) Portfolio Management with RL; 3) Portfolio Management with DRL.

It is expected to complete each of the models, and the performances will then be compared to find out the reasons behind which contribute to the corresponding performances. With the reason known, modifications could be made to the models built for enhancement.

In the coming section, the principles of the 3 suggested models will be explained.

5. Methodology

This section discusses the principles of models developed to tackle the algorithmic trading problem in this project. It first discusses how the problem will be formalized and the approaches used to manage the problem. Then, algorithms used to develop the models adopting the formalisms will be explained briefly.

5.1. Problem Formalisms

As mentioned in Section 3, this project will give attempts to tackle the algorithmic trading problem with 2 different approaches – 1) Stock level prediction, and 2) Portfolio Management. The following sections describe how are these approaches being carried out in this project.

5.1.1. Stock Level Prediction

In this project, it is considered to be an advantage for traders if they have information about the future stock levels. If precise predictions can be achieved at the end of the project, the difficulty of trading in stock market is expected to reduce much. Within

this project, stock price prediction is achieved by time series forecasting, which takes the historical stock prices as a time series. Then, the algorithms to be adopted should try to seek for a curve or a function which will be a best-fit for all the data entry in the training data set, and then make use of this curve or function and try to predict the actual price levels in the future. Reasonable actions will then be taken according to predictions given.

5.1.2. Portfolio Management

While stock level prediction focuses on the exchange of a single stock, portfolio management enables traders to handle more than one stock at a time. With this approach, situations more realistic and close to real-life trading can be simulated, which include also expenses like transaction costs, or management of risks between assets. Models that are capable to balance all these factors well would be more applicable to trading in real markets.

5.2. Learning Algorithms

The two approaches aforementioned are the general solutions to the problem which need some detailed algorithms to achieve the goals. Among the various variety of algorithms studied in Section 2, three are selected for further studies: 1) Recurrent Neural Network (RNN); 2) Reinforcement Learning (RL), and 3) Deep Reinforcement Learning (DRL). The principles are going to be explained in following sections.

5.2.1. Recurrent Neural Network (RNN)

The idea of the construction of neural networks originates from human thinking and neural networks are mimicking the massive network of neurons in human brain, where each neuron acts as a processing unit which intakes some input signals and generate outputs accordingly.

In a mathematical formalism, in a neural network a neuron of n input parameters can be expressed as a function:

$$h_{W,b}(x) = f(W^T x + b) = f(\sum_{i=1}^{n} (W_i x_i + b_i)) \quad (1)$$

with: x as a vector of n inputs, W as weight vector of length n, b the bias term, and activation function f.

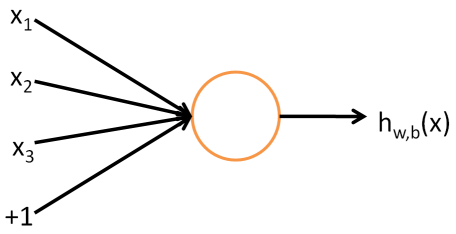


Fig.1 Graphical representation of neuron – the simplest possible neural network [17].

Activation function f here decides whether the neuron will be "activated" or not – whether or how the value would be utilized later, according to the input received [8]. Some of the common choices for activation function f are given as:

- $sigmoid(z) = \frac{1}{(1+\exp(-z))}$
- $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- $ReLU$: $\max(0, z)$

Then, numerous of such neurons are connected together as a neural network, taking outputs from some neurons as the input as one another.

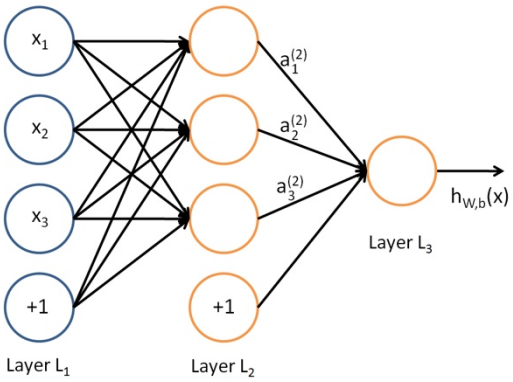


Fig.2 A simple neural network with 1 hidden layer
(L1: Input layer, L2: hidden layer, L3: Output layer) [17].

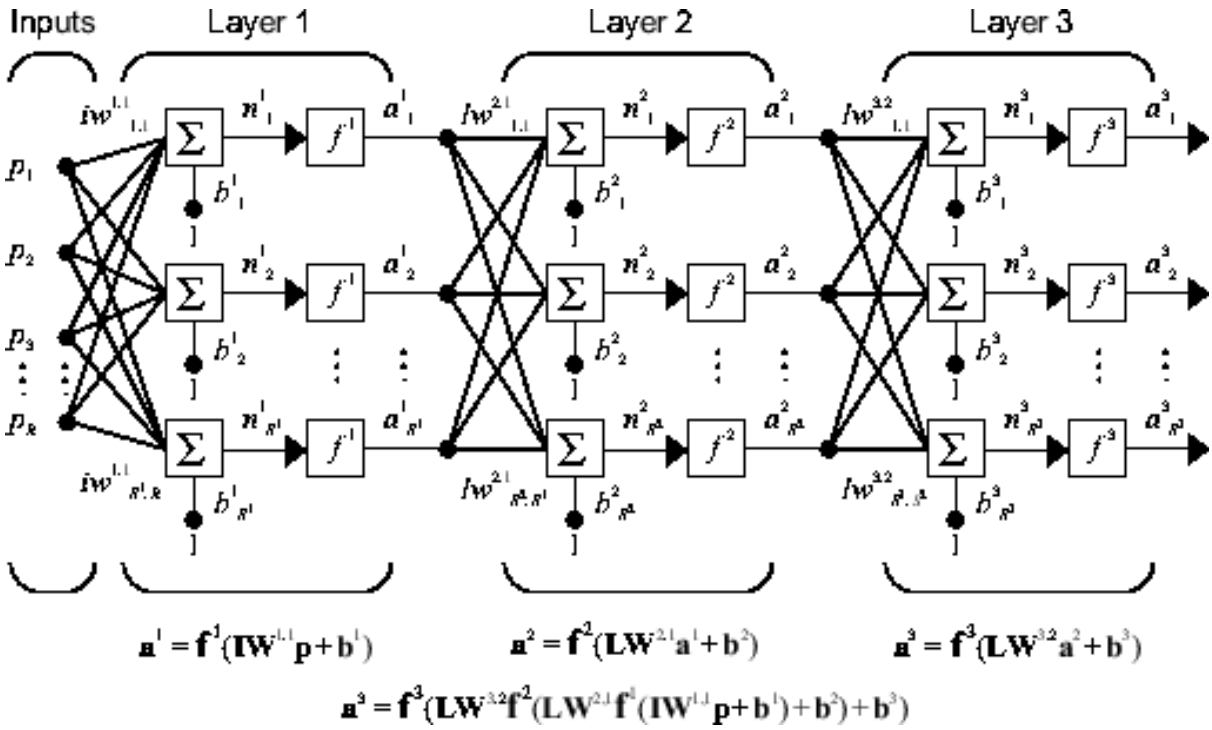


Fig.3 Construction of a deep neural network, in which layers of neurons are connected to each other to form a network.

One can observed the sequential structure in a deep neural network shown in Fig.3, where the output of one neuron is fed to the next neuron as the input. A neural network is "deep" if a number of layers of neurons are used to construct the network. In fact, by feeding the output of a neuron as the input signal to itself will give a recurrent neural network. There are two ways to represent the structure of RNN graphically – either in recursive form or in unfolded form (See Fig.4), where there is no different between the two, only different perspectives to view the network formed.

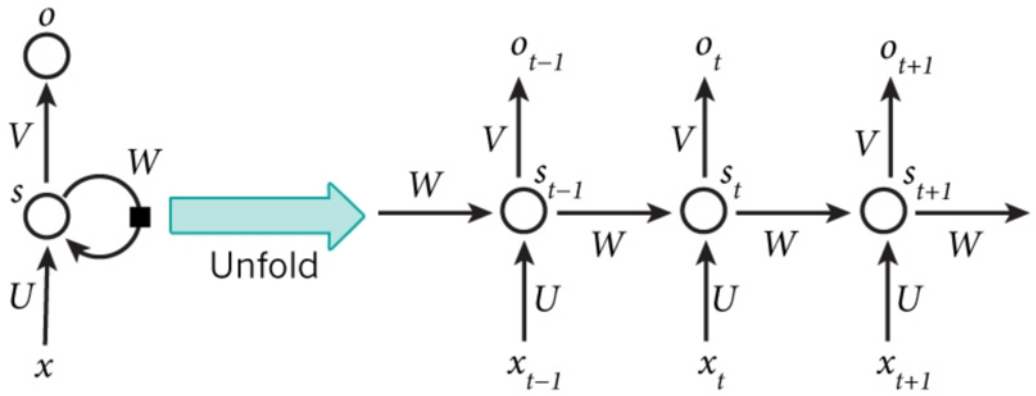


Fig.4 Left: Recursive (left) and unfolded (right) representations of RNN [18].

In Fig.4, the unfolded representation gives a more detailed picture so that the states at each moment is displayed, with x_t be the input at time t; s_t be the hidden state at time t; and o_t be the output at time t. W is the weight to be trained in the network.

RNN is considered to be specialized for sequential data, since the important "parameter sharing" property allows RNN to share parameters across different time index in the network, so during the training process unseen sequences can be generalized and be expected at any time [19]. This is important especially when a specific pattern can occur at anywhere in the entire time series [19] like stocks prices that are not periodic and which are not rare in the market.

Generally, the training process using RNN consists of 4 main steps [18]:
1) Initialization of values for neurons at the beginning. Other than taking 0 or random values, Xavier Initialization is also one of the common choices which take random values from $[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}]$, where n is the number of connections from previous layer
2) Forward propagation with values of hidden states and outputs are evaluated as time proceeds:

- $hidden\ states$: $s_t = f(Ux_t + Ws_{t-1})$
- $outputs$: $o_t = softmax(s_t)$

3) Loss calculation.
4) Back propagation with stochastic gradient descent

5.2.2. Reinforcement Learning (RL)

RL aims at learning strategies that can be used in stochastic and possible unknown environment [20]. Usually the stock market is considered as a black box, where the mechanisms behind or how the stock changes are unknown to the trader. RL attempts to learn the strategy by "Trial and Error" – keep making actions and adjust the strategies based on the rewards received from the environment so that 1) long-run rewards can be maximized, and 2) immediate and future rewards can be balanced [20].

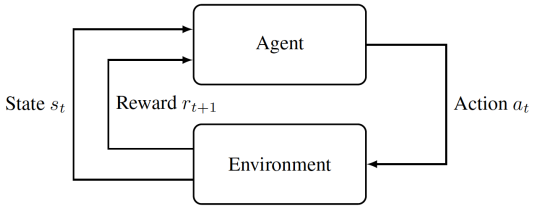


Fig.5 A simple agent-environment structure for RL.

Usually, the problem is modeled by Markov Decision Processes (MDP) [20], which could be represented by tuple of $< S, A, P, R, \gamma >$, where

- $S: measurable\ state\ space$
- $A: meaureable\ action\ space$
- $P: S \times A \times S \longrightarrow \mathbb{R} : Markov\ transition\ kernel$
- $R: S \times A \longrightarrow \mathbb{R} : reward\ function$
- $0 < \gamma < 1: discount\ factor$

It is notable that $R(s, a)$ is the desired reward function which gives the expected rewards that will be returned if action a is taken in state s. With this reward function, strategies can be determined accordingly.

Q-Learning is one of the RL methods that does not require any model structure, or called "model-free") [21]. This will be advantageous especially when the stock market is too complex to be understood and modeled. Q-Learning optimize the expected future rewards with the following assignment [21]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta[r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \qquad (2)$$

where

- $s_t, a_t: state, action\ pair\ at\ time\ t$
- $r(s_t, a_t): reward\ function$
- $\eta: exploration\ probability$
- $\gamma: discount\ factor$

Meanwhile, Q-Learning update the "Q values" $Q(s_t, a_t)$ through time with the following algorithm:

- Initialize $Q(s, a)$
- Repeat:
  - Initialize s_t
  - Repeat:
    - Choose action a_t (e.g. using ε-greedy algorithm)
    - Observe reward r, new state s_{t+1}
    - $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta[r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$
    - $s_t \leftarrow s_{t+1}$
  - end if s_t is terminal

Algo. 1 Algorithm of Q Learning.

Here, table Q stores all the rewards for the entire state space defined by states s, and action a represents the transitions from one state to another state. As the learning process keeps going on, the values in table Q will also keep updating until the final

state is achieved. When the value $Q(s_t, a_t)$ is going to be updated, the original value is adjusted by a learned value which is scaled by the exploration probability η. Inside the learned value, it includes: 1) the rewards at state s; 2) the optimal future reward $\max_a Q(s_{t+1}, a)$ scaled by a discount factor γ; 3) reducing the original value once to get the net change.

5.2.3. Deep Reinforcement Learning (DRL)

Being one of the industry leading research team in the field of machine learning, DeepMind, the team created AlphaGo which is well-known as a super-human Go player, was the first to adopt deep NNs and RL in the same model [13,14,15]. DRL was considered as a breakthrough in the field of RL because DeepMind's DRL model was not only dedicated for playing Atari games but in fact was a general-purposed agent [14]. The model created is called Double Deep Q Network (DDQN) [13].

Before proceeding on explaining how does the state-of-the-art DDQN work, it may be better to take a glimpse of Deep Q Network (DQN) first. As suggested in the name, DQN also adopted the principle of Q Learning discussed above (See Section 5.2.2). Q Learning maintains Q values for each possible state in Q table, and the best action is determined by action associated with the best Q value. Similarly, DQN also learns how to evaluate Q values to choose the best action that produce maximized profit, not using a Q table but a Q network constructed by deep NN.

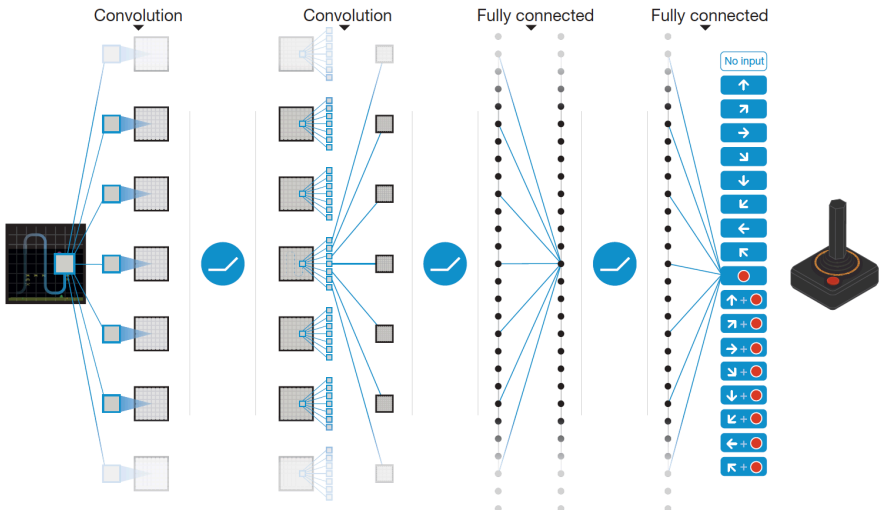


Fig. 6 Policy network constructed with CNN [16].

Fig. 6 illustrated the CNN constructed by DeepMind for predicting Q values to determine the best actions in Atari games [15]. When the agent is requested for action, the agent takes the current state (e.g. screen pixels) as the input of CNN. The CNN then compute Q values and suggests the best action to be taken. The learning procedures continues like the ordinary Q learning (See Section 5.2), but DDQN takes a step further by not only taking one but two DQNs – Online DQN and Target DQN, which is also the reason of being named as Double DQN. Meanwhile, DeepMind also introduced the Memory Replay technique [15]. Before explaining the two innovative techniques, first consider the algorithm of DDQN which is given as the followings:

- Construct Online DQN and Target DQN
- Repeat:
  - Observe states s_t
  - Choose action a_t with Online DQN
  - Take action a_t and calculate rewards r_t
  - Observe next states s_{t+1}
  - Memorize experience (s_t, a_t, r_t, s_{t+1})
  - If training not yet started (1):
    - Continue to next iteration
  - Periodically do the training (2):
    - Sample batches of past experience (s_t, a_t, r_t, s_{t+1})
    - Calculate next max Q values q_{t+1} with Target DQN
    - Use $y \leftarrow r_t + \gamma q_{t+1}$ as output labels for training Online DQN
  - Periodically do (3):
    - Copy Online DQN to Target DQN

Algo. 2 Algorithm of training with DDQN.

Fundamental Q Learning Algorithms update Q Tables/DQN examining rewards after every steps taken. However, this might make the network unstable since it might lead to divergence, cycles or loops of feedbacks [22]. DeepMind then devised DDQN to stabilizes reinforcement signals and stochastic gradient descent [15]. From the above algorithm (Algo. 2), given the states, actions are determined by Online DQN and rewards are evaluated afterwards. One can observe that the training process (2) is not performed for every step, but only once per period of training. Meanwhile, Target DQN is also updated once per period of copying (3).

Memory Replay is also one of the key technique adopted by DeepMind. According to Algo. 2 (1), the training process is not started from the very beginning. It is because in

some applications the first steps are comparatively less important than the subsequent steps [22], and one should ensure that there are enough experiences for sampling. Meanwhile, for each training period in (2), a batch of experiences is sampled from all experiences memorized, or from a queue of experiences with limited size. This helps DQN to learn from both past and recent experiences, and hence to learn to a various of situations [23].

6. Development Tools

This section mainly introduces the tools utilized, i.e. the programming language used and libraries included, during the development of the deep learning models introduced in previous sections (See Section 3,4).

6.1. Python

Python was chosen as the developing programming language of the deep learning models (RNN, RL) in this pro ject. For the implementations of deep learning models discussed in the methodology section, a vast amount of arithmetic calculations with matrices is essential and inevitable. In the meantime, there are sufficiently many libraries built for Python that support fast and easy matrix calculations, and more importantly, deep learning computations. The libraries adopted in this project will be discussed in the following sub-sections.

6.2. TensorFlow

TensorFlow is an open-source library which is designed for building machine learning models, especially deep learning models. TensorFlow achieves high-speed deep learning computations by introducing the concept of “tensors” and “computational graphs” [9]. The models of RNN and RL are mostly built with TensorFlow.

Enabled by TensorFlow, one can visualize the models built with TensorFlow using TensorBoard. In the following sections, all figures demonstrating structure of models built are generated by TensorBoard.

7. Training Data Sets

This section introduces the stock market indices chosen for testing the completed models.

7.1. Stock Market Data

The exchange rates of CAD to HKD (CADHKD, See Fig.7), and Nasdaq Composite Index (NASDAQ) (See Fig.10) have been chosen as the targets for prediction. NASDAQ has been considered as one of the most common indices in US, and long periods of historical data are available (usually more than 20 years), which is the same for the exchange rates of CADHKD. Meanwhile, these two series are selected because they share different ranges and magnitudes of rates, and the series also experiences different characteristics as shown in Table 1. Therefore the models can be tested under different situation.

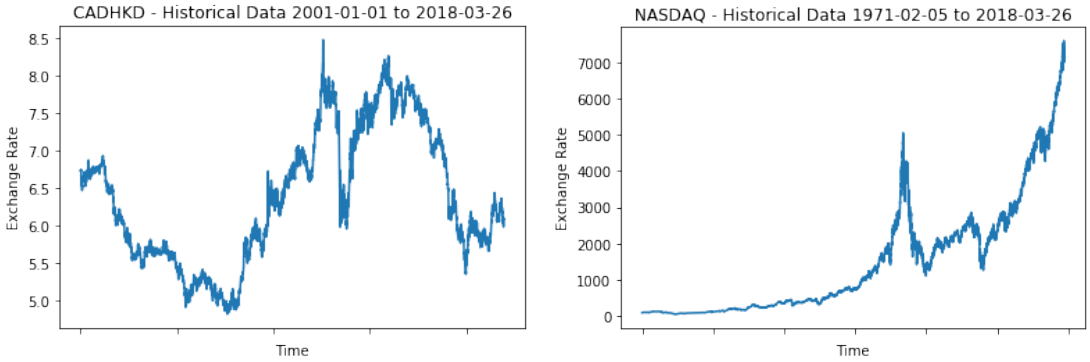


Fig.7 Historical trend of CADHKD (left), NASDAQ (right).

Stock	CADHKD	NASDAQ
Period (Trading days)	1990-01-02 to 2018-03-26 (8786 days)	1971-02-05 to 2018-03-26 (11890 days)
Mean	6.3852	1945.9950
Mean Daily Fluctuation	0.01863	14.1044
Max.	8.4790	7588.3198
Min.	4.8243	54.8699
Variance	0.7877	2518346.4547
Std. dev.	0.8875	1586.9298
Max. Profit	1431.00	3097.67
Min. Profit	-1154.13	-1199.00
Profit Range	2858.13	4296.67

Table 1 Statistics of stock price series used.

Table 1 reveals that the two stocks are having price levels in two completely different ranges. Therefore, performances of models under different ranges can be compared. Other than the

difference in their magnitudes of price levels, the two data sets also represent different trends. One can observe that CADHKD is more turbulent with ups and downs all over the time. On the other hand, NASDAQ mainly reveals a uptrend with one small peak near the middle of the entire set. In other words, learning with CADHKD may visits more similar patterns, while learning with NASDAQ may lead to a lack of evidences. Therefore, different characteristics of training sets may also differentiate the performances of models.

7.2. Training, Testing Data Sets

Table 1 only shows statistics about the entire data set. In the testing conducted later in this project, training and testing set of different lengths extracted from the two data sets were used. For example, in the experiments which will be discussed in Section 10.1, the lengths of training data increases with intervals of 3 years. The statistics of training data with different ranges is given as follows:

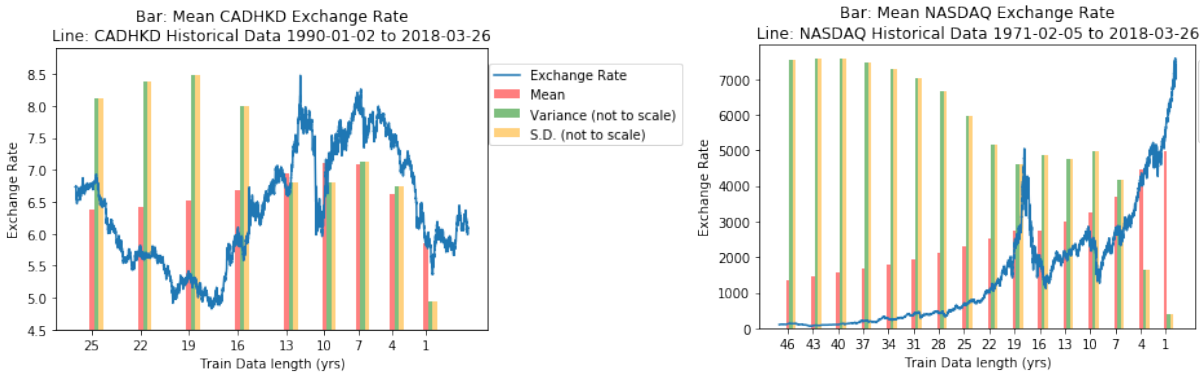
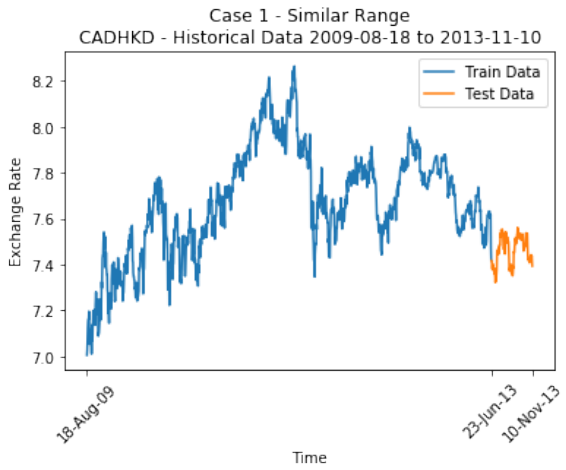


Fig. 8 Train data statistics of different lengths (Left: CADHKD, Right: NASDAQ).

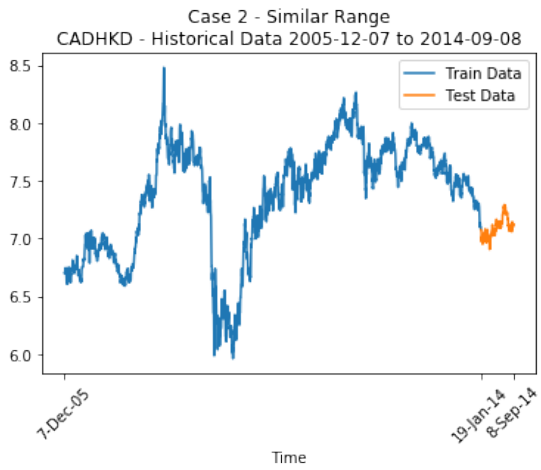
Fig. 8 displays statistics of training data with different lengths by bars of different colors. Bars on the left represents data with longer lengths with bars on the right refers to shorter data lengths. There are some common features among the two data sets, for example, variances (green) and standard deviation (yellow) grows with data length. On the other hand, their means (red) are showing different patters, where CADHKD have similar means for different data ranges, while means of NASDAQ drops when data length grows. This is related to the characteristics of the data sets described in Section 7.1. Subsequent experiments and analyses are expected to investigate on the relation between these characteristics and the model performances.

7.3. Test Cases

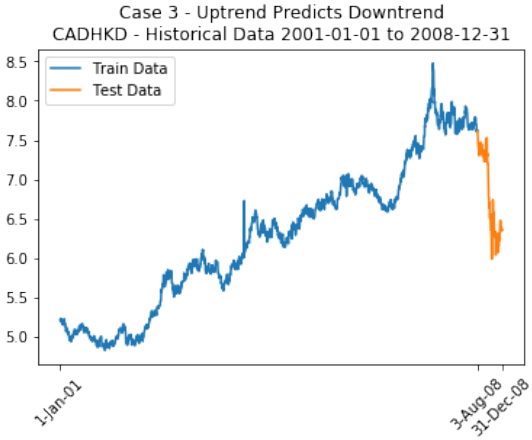
Other than testing against different lengths of training data, stock trends showing special patterns are selected in order to test whether the models perform well under these situations. In total, 6 cases were designed:



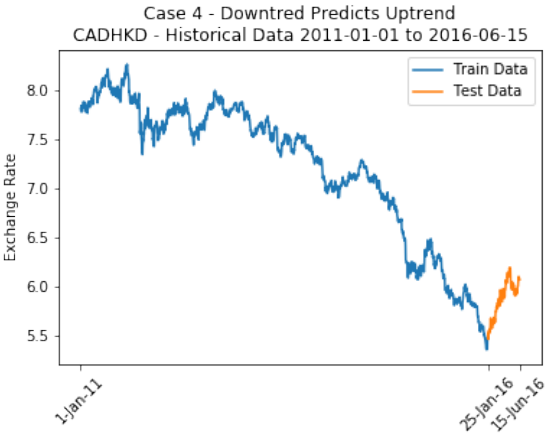
(a) Case 1 – Similar range



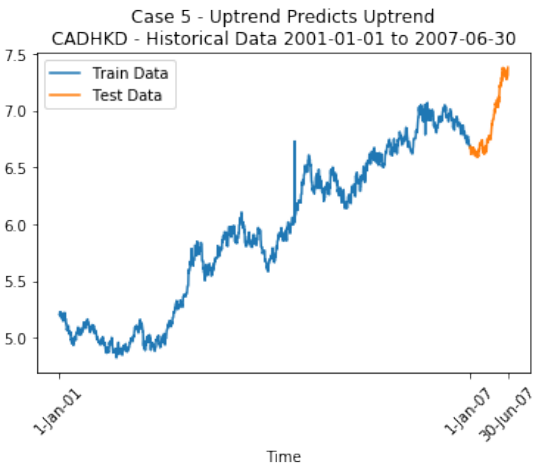
(b) Case 2 – Similar range



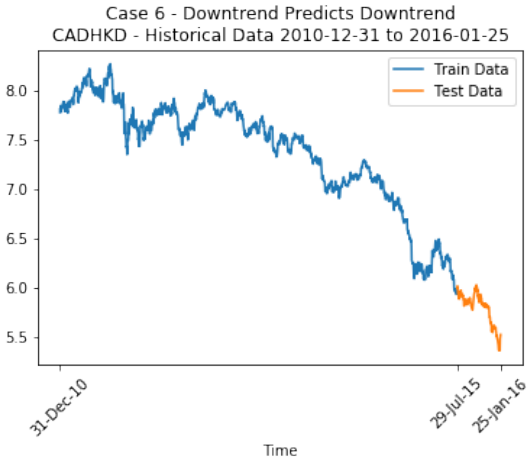
(c) Case 3 - Up to down



(d) Case 4 – Down to up



(e) Case 5 – Up to up



(f) Case 6 – Down to down

Fig. 9 Trends of 6 test cases (Blue: train data, Orange: test data).

Case	1	2	3	4	5	6
Train len (days)	1380/140	2940/230	2527/150	1814/142	1947/180	1640/180
Max	8.26/7.56	8.47/7.29	8.47/7.61	8.26/6.19	7.06/7.37	8.26/6.02
Min	7.00/7.32	5.29/6.90	4.82/5.98	5.36/5.46	4.82/6.58	5.93/5.36
Mean	7.67/7.46	7.36/7.09	6.26/6.81	7.22/5.88	5.90/6.88	7.38/5.78
Var	0.054/0.004	0.24/0.007	0.83/0.27	0.50/0.033	0.47/0.07	0.30/0.02
SD	0.233/0.062	0.49/0.08	0.91/0.52	0.78/0.18	0.69/0.26	0.54/0.16
Mean daily diff.	0.025/0.014	0.024/0.013	0.019/0.042	0.019/0.024	0.017/0.016	0.019/0.017
Max. Prof	+1465.36	+2580.68	+4712.63	+4075.50	+2951.42	+2650.08

Table 2. Statistics of test case data (shown in: train/test).

The test cases are designed to capture the following situations:

- Case 1,2: Future prices fluctuates within range of train data.
- Case 3: Train data with uptrend stops at global max and test data starts to drop.
- Case 4: Train data shows a uptrend and test data continues to rise.
- Case 5: Train data with downtrend stops at global min and test data starts to rise.
- Case 6: Train data shows a downtrend and test data continues to drop.

The maximum profits for each cases are given in Table 2 as a reference for comparing performances of models.

8. AI Models

Section 4 outlined this project with three models: RNN, RL and DRL. This section presents the structures of all the models built for studies.

8.1. Recurrent Neural Networks (RNNs)

8.1.1. Base Model – Single-layered RNN (SRNN)

The first model built with RNN is a simple model with one layer of RNN cells, with the structure illustrated by Fig. 10.

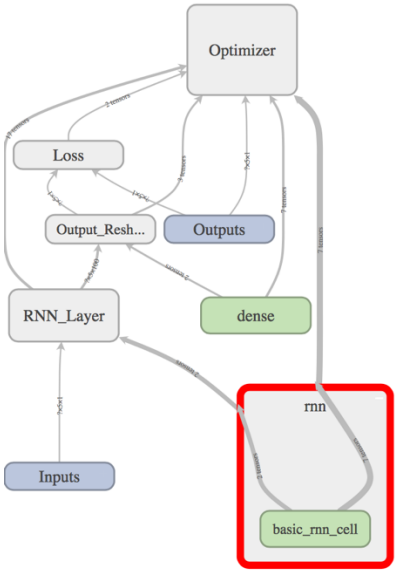


Fig. 10 Structure of single-layered RNN model.

Fig. 10 gives the outline of the SRNN model. The model is mainly composed by one layer of RNN cells ("basic_rnn_cell"), where the number of hidden units can be specified later. The RNN layer ("RNN_Layer") uses the hidden cells and calculates the outputs, which is also a sequence of price levels but shifted by one trading day and this is the prediction desired. Therefore, there is a layer ("Output_Reshape") to reshape the output and extract the predicted price. The loss is then evaluated by the difference between predicted and actual price levels. Accuracy is improved by adjusting weights used in RNN cells through iterations.

8.1.2. Modification 1 – Multi-layered RNN (MRNN)

After constructing the SRNN, it is intuitive to devise a modification which is multi-layered RNN, where the structure is demonstrated with Fig. 11. One can observe that the structure is in fact comparable with SRNN (See Fig. 10). The only difference is that MRNN adopts a MultiRNNCell defined by TensorFlow, which is constructed with a list of BasicRNNCell. For illustration purpose, Fig. 11 shows a MRNN model with 2 layers only ("multi_rnn_cell"), but the number of layers are subject to change during construction of the model.

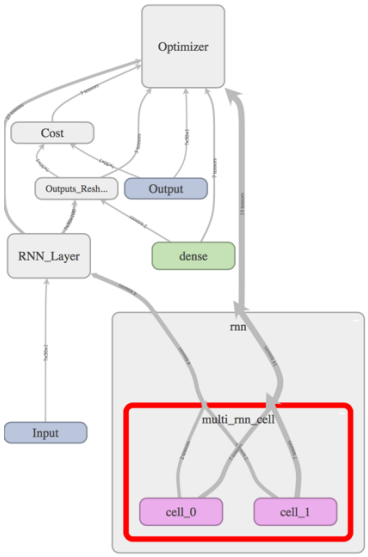


Fig. 11 Structure of multi-layered RNN model.

8.1.3. Modification 2 – Convolutional Output Layer

For both SRNN and MRNN, the predictions are given by the last entry of outputs of the RNN layer, with all other entries simply ignored. It is conjectured that accuracy of prediction may be improved by using predictions that considered other outputs also. Therefore, a layer of Convolutional Neural Network (CNN) is attached as the output layer of MRNN (where SRNN is equivalent to MRNN with one layer), in order to capture some useful patterns to improve predictions.

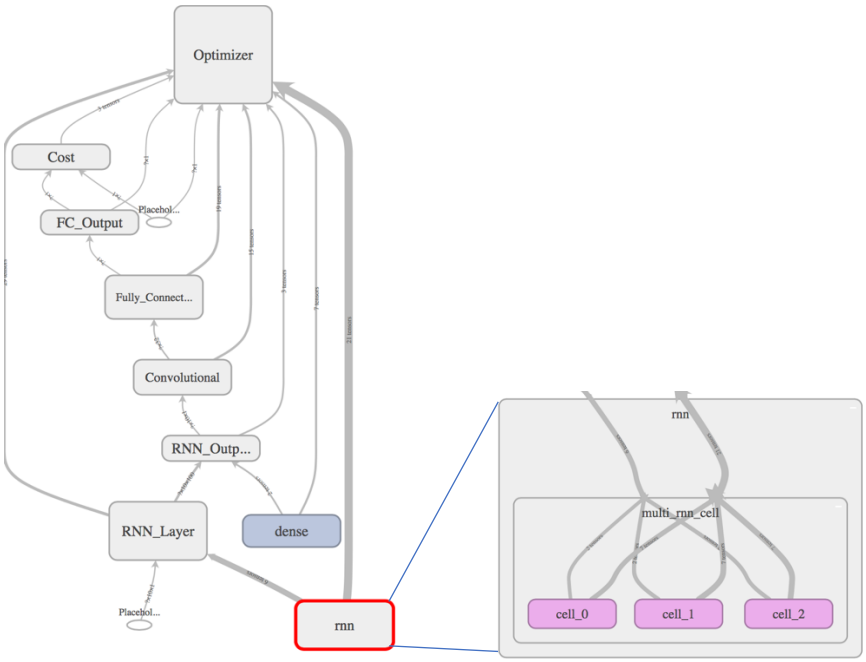


Fig. 12 Structure of MRNN model with Convolutional output layer.
(Left: Overall structure; Right: Details of MultiRNNCell)

Fig. 12 shows a MRNN model with three hidden layers. Note that after the RNN output layer ("RNN_Outputs"), it is followed by a convolutional layer ("Convolutional") to process the outputs. Finally, outputs from the convolutional layer are further processed by a fully-connected layer ("Fully_Connected") compute a single-valued prediction, which is used for computing accuracy.

8.2. Reinforcement Learning (RL)

8.2.1. Base Model – Q Learning

The base model simply adopts algorithm described by Algo. 1 in Section 5.2.2, which can also be described with the following figure:

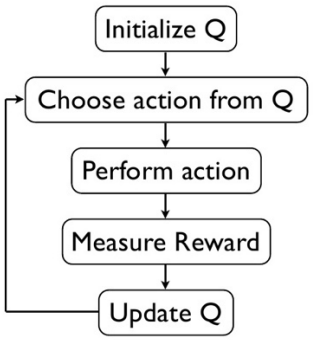


Fig. 13 Flow chart showing process of RL base model.

Q learning gives the foundation of all RL-related learning models in this project. The main difference between models in Section 8.2 is the states adopted, which also defined the structure of Q tables. Therefore, many of the modifications made to RL were based on defining states through different ways.

8.2.2. Modification 1 – States

Q learning relies much on the definition of Q table, which is involved in determining the best action under different situations. Since the situations are described by states, the definition of states would probably have influences on the performance of QL. In this project, models have been built using the following definitions of states:

- Price-related:
    - e.g. Price-levels, Moving Averages (MA), …
- Trend-related:
    - e.g. Change of prices/MA
- Others:

- e.g. Stock indicators (RSI/CCI/MACD)

It is conjectured that with different information provided by the three states definitions, QL will give different performances, which will be discussed in Sections related to experiment results.

8.2.3. Modification 2 – Exploration Algorithms

Given with some well-defined states, in order to harness full potential given by the states, there has to be enough exploration on different states, or else the QL model will easily face some unexperienced situations which make it hard to make good responses. Therefore, models adopting different exploration algorithms have been built to test how the performances change with different degree of exploration. The algorithms used are namely:

- Exploration by random actions.
- Exploration by Epsilon-greedy Algorithm.

By exploring with random actions, each time when the RL model needs to choose an action in learning phase, it chooses by random so that states associated with each action will share equal probability to be visited. Note that Q table is not involved in choosing this action.

Meanwhile, by exploring with Epsilon-greedy Algorithm, the RL model chooses the best action observed so far with a probability of ε, and a random action with probability of $1-\varepsilon$. The best action here is then determined by the Q table which is continuously updating. By keep exploiting the states with best actions, it is expected to refine the learning results of states will probably be considered more often in practice.

8.3. Deep Reinforcement Learning (DRL)

8.3.1. Model 1 – Deep Q Network (DQN)

For the RL models discussed in Section 8.2, all states are represented by Q tables, defined by states and actions. For the next model, Q table is replaced by a deep NN (called DQN), which takes the states as input and gives an action in return. Similar to Q tables, DQN can be considered to be representing a policy, and the aim of this model is to learn the best policy to give best actions in response.

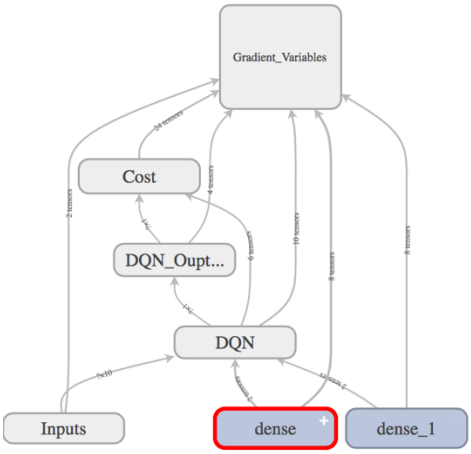


Fig. 14 Structure of DQN.

Fig 14 shows the structure of DQN built for this project. The DQN layer ("DQN") takes the states given by the environment (See Section 8.2.2) as inputs. The DQN layer in this project is constructed with a dense layer (with tf.layers.dense() defined by TensorFlow). DQN is kept simple considering that stock trends (Up/Down) or indicators define states spaces with limited dimensions. In the output layer ("DQN_Output"), the Q values associated to each possible action are evaluated, and the fitness of such Q values is then refined by gradient descent algorithms, where the gradients of each action (Sell/Hold/Buy) are calculated separately (in "Gradient_Variables"), so the corresponding Q values are also refined separately.

8.3.2. Model 2 – Double Deep Q Network (DDQN)

The model of DDQN discussed in Section 5.2.3 was attempted to modify for algorithmic trading, which also adopted the use of memory replay and the use of Online and Target DQN. Fig. 15 describes the structure of the DDQN model, in which one can observe the learning procedures in the figure. While in the stage of obtaining experiences, input states are fed to Online DQN ("Online_DQN"). Meanwhile, given input states in the learning phase, next Q values are determined by Target DQN ("Target_DQN"). Besides, Online DQN is periodically being copied to Target DQN ("Copy_DQN").

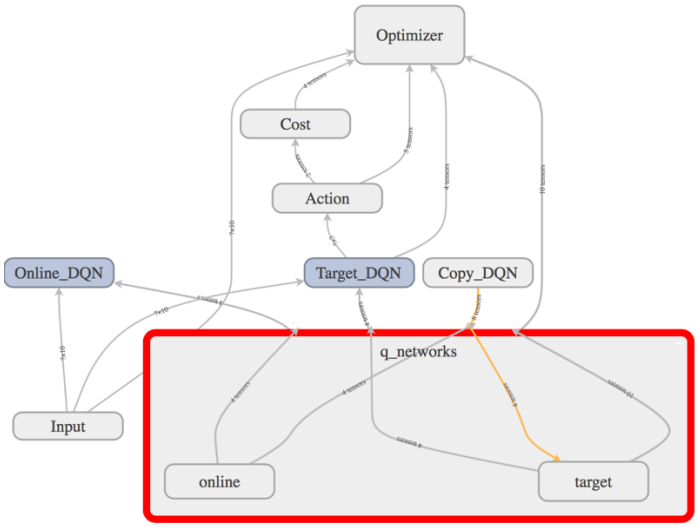


Fig. 15 Structure of DDQN model.

The paragraphs above give the designs of models built for this project. Hyper-parameters used for each models were subject to change and they will be discussed in coming sections.

9. Experiment Settings

In order to test the performances of the models constructed, experiments have been conducted and this section describes the procedures and details of them.

9.1. General Procedure

Most of the experiments follows the procedures given below:

1) Determine length of training data and extract data accordingly.
2) Determine values of hyper-parameters and build model accordingly.
3) Train model with prepared training data.
4) Test model with test data.
5) Continue to next data sets/values.

By testing the models on different lengths of training data and values of hyper-parameters, it is expected to find out some factors affecting the performances of models. Therefore, one might be able to determine the "best" or at least "good" hyper-parameters that are likely to produce good results. The analyses are going to be discussed in Section 10.

9.2. Experiment by Cases

Given with the information of "good" or "best" hyper-parameters, some models of the specified hyper-parameters have been built. These models were tested against some pre-

defined scenarios, which are designed by selecting price levels from certain parts of the entire data set (See Section 7.3). The procedures are given as:

1) Design scenarios from entire data set.
2) Construct models with "best" hyper-parameters.
3) Testing of models on scenarios.

Since the scenarios are designed to cover some common situations of trading in real-life, these experiments are designed to determine whether the models could be used in practice, or when the models should be used. The results will be discussed in Section 11.

9.3. Performance Metrics

The following metrics were used to evaluate the performance of models:

(a) Mean Absolute Error (MAE):

It is measured by the absolute difference between values by prediction and actual data. This would measure how the predictions are variating from the actual values.

(b) Root Mean Squared Error (RMSE):

It is measured by the mean of squared errors, but taking root at last. While errors by every predicted values are equally influential to MAE, in RMSE large errors are weight more heavily since all errors are squared. Hence, it has more implications to the existence of large errors by the predictions.

(c) Predicted Profit:

A series of predictions gives a series of suggested actions. By making actions corresponding to the suggestion actions, one can simulate the process of investment. The returns after simulation is taken as the predicted profit. In the following experiments, the test data is the price levels in the latest 100 trading days, and the predicted profit is the return after simulation of 100 days.

(d) Empty Q Values:

Some of the RL models built involved the use of Q table, which stores Q values which are related to the likelihood of actions to be chosen under different situations. Therefore, if during the training stage some states are not visited and the Q values are just kept at their initial values, the no information can be read from the Q values and hence the decision may be misleading. There, a fewer number of unaltered Q values

shows a higher degree of exploration, which makes the models prepared for more situations.

It is noted that (a) MAE and (b) RMSE were only used for measuring accuracy of predictions. It is difficult to apply (a) and (b) for RL since RL gives a policy (e.g. Q Table) in return which accuracy is ambiguous in this situation. Similarly, (d) is only used with models using Q tables. Meanwhile, (c) Predicted Profit is used for all models. Moreover, in order to analyse the correlations between the performance and a specific hyper-parameter, aggregations like mean, median and standard deviation (SD) will be used.

10. Experiment Results – Overall Tests

The models built are first put under a general test as described in Section 9.1. The models will be test against different data lengths and values of hyper-parameters in order to examine how are the performances of the models being affected. Section 10.1 gives the performances of models of RNN, Section 10.2 discusses performances of RL models, and Section 10.3 focuses on models of DQN. Section 10.4 gives a summary on the analyses.

10.1. RNN Models

In this project, models built with RNN mainly performs time-series prediction that the RNN model read the previous price levels and predicts the price on the next trading day. The action to buy or sell is then based on the prediction. The higher the accuracy of prediction, the more profits to be earned. In this project, three models with RNN were designed, and the experiments results on the models will be presented in following sections. At the end of this section, there will be a summary that compares the performances of the models.

10.1.1. Stock Price Level Prediction with SRNN

The SRNN model with only one hidden layer is the simplest RNN model built project. It will be used as a baseline to compare with other RNN models.

10.1.1.1. Test Cases Information

SRNN model was tested with two sets of training data: CADHKD and NASDAQ. The followings show the domains of hyper-parameters used:

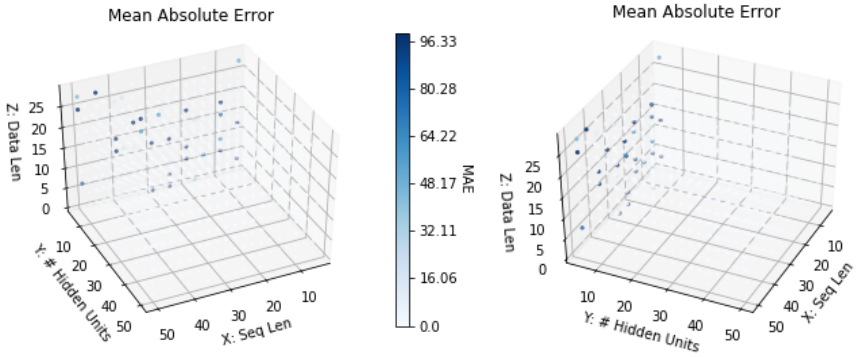
- Training data size (in years, approx.)
    - CADHKD: { 1, 4, 7, 10, 13, 16, 19, 22, 25, 28 }

- NASDAQ: {1, 4, 7, 10, …, 40, 43, 46}

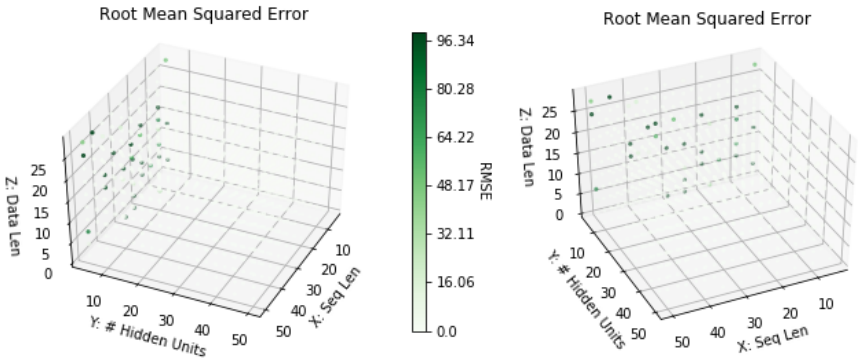
- Sequence length = { 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 }
- Number of hidden units = { 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 }

10.1.1.2. Overall Performance

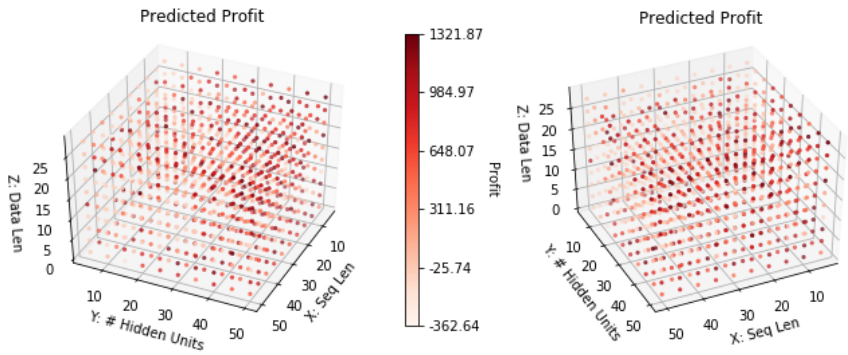
The overall performance of SRNN model is visualized with some 3D scatter plots in Fig. 16(a), (b). Every scatter point in the above plots are colored in different shades. Darker colors represent higher values of the performance metrics while lighter colors show the converse (refer to the color bars). For each metric, there are 2 3D scatter plots (which are in same color) which are basically identical, but only differ from the viewing angle to the 3D plots, which forester understanding to the high-dimensional plots.



(i) MAE on CADHKD

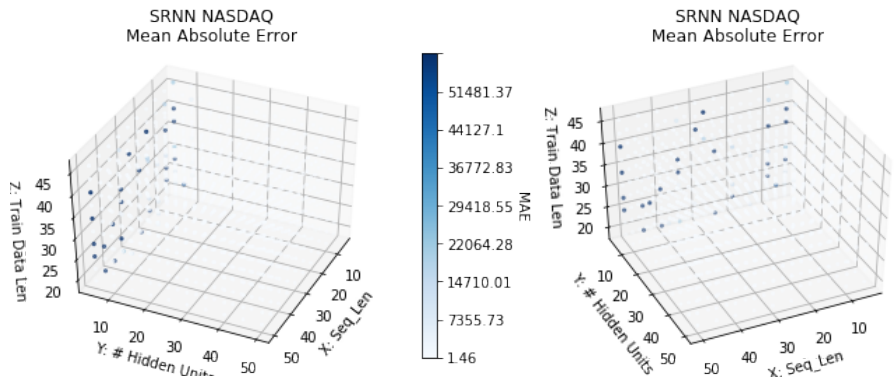


(ii) RMSE on CADHKD

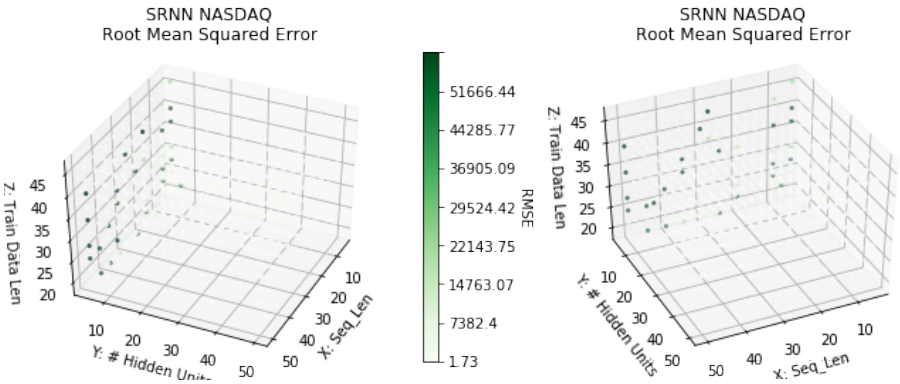


(iii) Predicted Profit on CADHKD

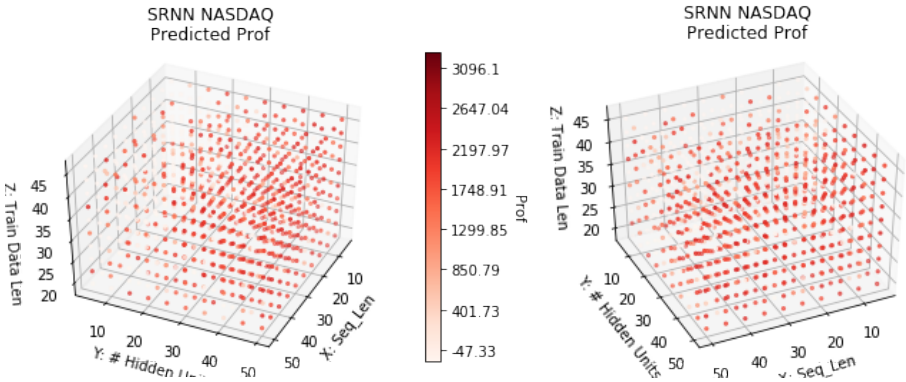
Fig.16(a) 3D scatter plots on SRNN model performances on CADHKD



(i) MAE on NASDAQ



(ii) RMSE on NASDAQ



(iii) Predicted Profit on NASDAQ

Fig. 16(b) 3D scatter plots on SRNN model performances on NASDAQ

The plots above in Fig. 16 (mainly (iii)) give a brief picture to the performance of the model – better performances ((i),(ii): lighter colors; (iii): darker colors) with a larger size of training data and more hidden units in the hidden layer. On the other hand, performances are comparatively better for shorter sequence lengths, with longer sequence lengths give more darker dots (more obvious in (i), (ii)). This observation will be examined separately in the coming sections.

10.1.1.3. Number of Hidden Units

Fig. 17 groups the profits made by SRNN model by the number of hidden units, and aggregates profits in the same group by taking mean, median, etc.. The table on the left shows the profits made on CADHKD. One can observe the trend that the mean and median profits, as well as the min profit, increase with the number of hidden units. Moreover, the same trend seems to be observable also in the table of NASDAQ on the right. Therefore, the two table show that increasing the number of hidden units can help to improve the performance of SRNN model.

num_hiddens	mean_prof	max_prof	min_prof	median_prof	std_prof
5	211.48	1304.40	-121.72	0.00	429.36
10	238.48	1279.31	-304.01	0.00	426.02
15	296.10	1270.01	-362.64	127.34	426.73
20	411.20	1312.45	-417.37	298.88	470.04
25	441.48	1259.41	-332.57	329.23	463.58
30	543.24	1270.47	-109.85	610.61	427.31
35	590.00	1294.43	-109.85	634.21	471.57
40	620.21	1321.87	-109.85	649.84	426.99
45	757.18	1321.07	-78.08	847.09	390.22
50	648.47	1310.50	-97.67	705.52	437.92

num_hidden	mean_prof	max_prof	min_prof	median_prof	std_prof
5	1386.72	3086.49	0.00	1527.25	1007.12
10	1956.97	3070.19	-47.33	2140.64	983.25
15	2163.26	3092.65	0.00	2394.45	879.02
20	2552.77	3095.53	0.00	2831.57	653.87
25	2715.08	3089.01	317.59	2942.74	519.96
30	2779.31	3095.53	344.79	2994.35	511.62
35	2870.23	3094.92	73.01	3018.25	431.50
40	2921.73	3096.10	387.66	3014.03	326.35
45	2878.09	3096.10	0.00	3031.51	446.73
50	3006.60	3095.53	2372.17	3046.01	126.71

Fig. 17 Predicted profits by SRNN on CADHKD (left) and NASDAQ(right) with different aggregations grouped by number of hidden cells.

10.1.1.4. Number of Training Data Length

Fig. 18 aggregates profits by grouping lengths of training data. From the table of CADHKD on the left, it seems that SRNN does not perform well with 7-13 years of data. Also, the performance with 1-4 years of training data is comparatively better than that with more than 16 years. Even though, one should note that SRNN gives positive profits for most of the time.

data_year	mean_prof	max_prof	min_prof	median_prof	std_prof
1	573.82	1266.39	-166.12	643.57	322.80
4	922.85	1321.93	-417.37	1090.39	398.29
7	379.28	1322.50	-109.85	139.21	505.80
10	229.47	1321.07	-109.56	0.00	449.26
13	240.25	1279.31	-109.85	139.21	390.65
16	314.48	1264.22	-107.32	139.21	432.53
19	446.94	1312.45	-362.64	397.91	458.83
22	547.68	1300.28	-95.43	580.69	460.28
25	635.67	1321.87	-209.93	768.12	469.26
28	571.99	1310.50	-304.01	672.62	454.00

data_year	mean_prof	max_prof	min_prof	median_prof	std_prof
19	2530.69	3091.16	0.00	2959.57	813.15
22	2535.10	3096.10	0.00	2964.28	817.89
25	2519.59	3095.53	0.00	2957.66	831.73
28	2496.16	3094.57	0.00	2968.86	908.69
31	2451.52	3094.42	-47.33	2901.26	826.37
34	2395.07	3096.10	0.00	2888.35	908.59
37	2636.50	3095.53	0.00	2981.37	675.03
40	2537.07	3095.13	0.00	2895.55	802.11
43	2589.50	3095.53	0.00	2936.25	732.59
46	2539.57	3096.10	0.00	2933.82	835.67

Fig. 18 Predicted profits by SRNN on CADHKD (left) and NASDAQ(right) with different aggregations grouped by lengths of training data.

However, the performances on NASDAQ is rather stable over years. Yet, considering the max possible profit to obtain is 4296 (See Table 1 in Section 7), SRNN could be considered as well-performed by obtaining mean profits around 2500.

10.1.1.5. Sequence Length

Each input to the RNN model is in the shape of a sequence, which is a sequence of n previous price levels. By taking inputs with different sequence lengths, the model is expected to learn patterns of different lengths, which may affect the performance of prediction. However, according to the heat maps in Section 10.1.1.2, the effect of change sequence length is not obvious. Fig. 19 also agrees with the observation from the heat maps. The table on the left shows that the mean or max profits on CADHKD are change variating too much with different sequence lengths. However, the min profits are decreasing with larger sequence lengths, which shows that the performance could go worse with larger window.

num_periods	mean_prof	max_prof	min_prof	median_prof	std_prof
5	618.39	1322.50	-78.08	660.66	487.07
10	533.85	1296.67	-126.01	543.96	471.73
15	512.07	1290.26	-80.87	525.60	480.86
20	603.45	1319.29	-97.67	664.42	492.84
25	607.40	1302.09	-95.43	726.01	492.27
30	489.48	1298.51	-107.32	442.92	479.51
35	522.78	1292.59	-209.93	541.55	499.66
40	481.01	1285.11	-332.57	514.02	476.78
45	488.73	1304.40	-362.64	496.71	464.34
50	500.34	1321.07	-166.94	509.59	494.05

num_periods	mean_prof	max_prof	min_prof	median_prof	std_prof
5	2564.97	3096.10	0.00	3032.41	911.90
10	2552.20	3094.23	0.00	2900.57	734.89
15	2379.68	3089.03	0.00	2779.55	928.15
20	2503.38	3094.92	-47.33	2927.81	870.96
25	2506.04	3095.53	0.00	2902.61	807.38
30	2505.05	3086.50	0.00	2943.98	829.94
35	2563.52	3094.62	0.00	2919.29	723.81
40	2575.19	3096.10	0.00	2979.81	735.20
45	2525.27	3094.92	0.00	2899.19	761.33
50	2555.45	3086.25	0.00	2950.23	851.19

Fig. 19 Predicted profits by SRNN on CADHKD (left) and NASDAQ(right) with different aggregations grouped by sequence length.

10.1.1.6. Discussion

Section 10.1.1.4 suggested that SRNN has worse performance with 7-13 years of historical data. It is suspected to be related to the trend of training data used. Fig. 20 shows the entire historical data of CADHKD. The red section marks the prices within the period of 7-13 years of training data, while the yellow section marks the test data. The remaining blue sections are the rest of the training data. It can be observed that both of the blue and yellow sections are showing an uptrend in mainstream. However, different from that, the red section shows a downtrend. The difference here may be the reason that hinder the learning of SRNN model. With similar reason, the good performance of SRNN on NASDAQ may be due to its uptrend across the entire data.

To sum up, for SRNN model, more hidden units are required for better performances in general (Section 10.1.1.3), but sequence length is comparatively less influential here (Section 10.1.1.5). Moreover, the performance is also affected by the training data (Section 10.1.1.4). Even though differences of features in train and test data are obstacles to the performance, SRNN was still able to give positive profits most of the time.

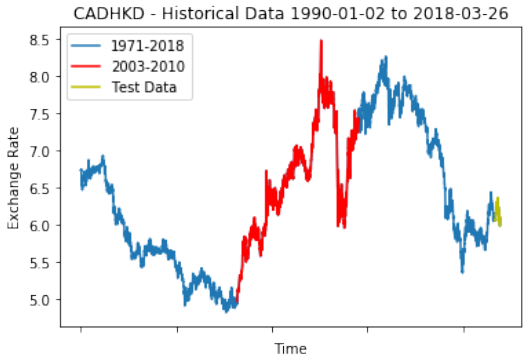


Fig. 20 Training data of CADHKD partitioned by up-/down-trends.

10.1.2. Stock Price Level Prediction with MRNN

MRNN attempts to improve the performance of SRNN by adding more hidden layers. Except the experiments of number of hidden layers, the experiments on MRNN are similar to that on SRNN which will be discussed in this section.

10.1.2.1. Test Cases Information

Information about the test cases on MRNN model are given as follows:

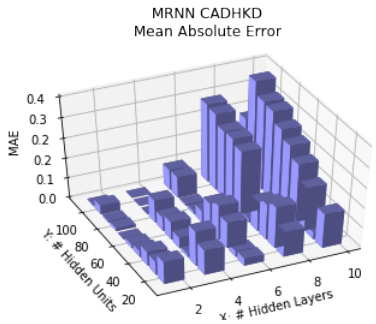
- Train data size (in years, approx.)

- CADHKD, NASDAQ: 10 years

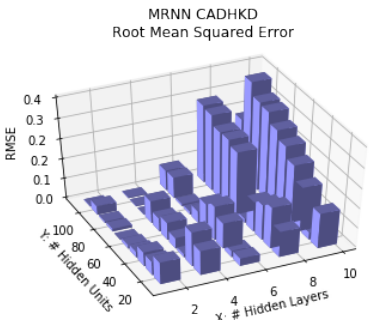
- Test data size: 100 days
- Sequence length = 20
- Number of hidden units = {10, 20, …, 100}
- Number of hidden layers = {2, 4, 6, 8, 10}

10.1.2.2. Overall Performance

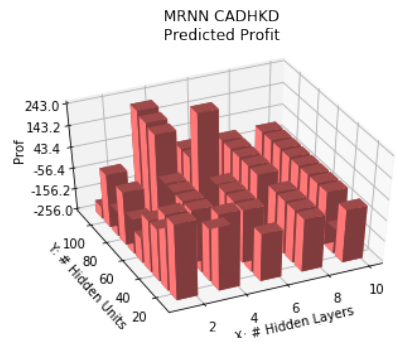
Analyses of MRNN model begins by visualizing performances with 3D bar charts.



(i) MAE on CADHKD

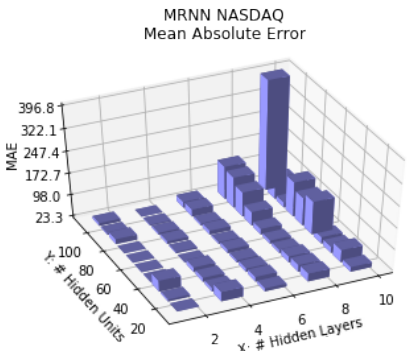


(ii) RMSE on CADHKD

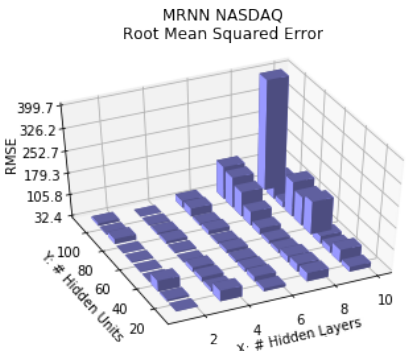


(iii) Predicted Profit on CADHKD

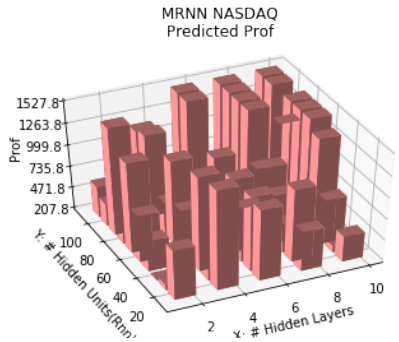
Fig. 21(a) 3D bar chars on MRNN model performances on CADHKD



(i) MAE on NASDAQ



(ii) RMSE on NASDAQ



(iii) Predicted Profit on NASDAQ

Fig. 21(b) 3D bar chars on MRNN model performances on NASDAQ

Considering the measurements of MAE (i) and RMSE (ii) shown in Fig. 21 (a) and (b), MRNN gives smaller error on both CADHKD and NASDAQ with fewer number of hidden units and number of hidden layers. The large errors on the contrary might give a sign to overfitting.

However, the predicted profits made by MRNN on the two stocks do not match with each other. For CADHKD, it follows that more profits made with less error with few number of layers. On the other hand, for NASDAQ, the variation of profits becomes larger, and it seems the only trend is that more profits are made with more hidden layers and units, unlike that in CADHKD.

10.1.2.3. Number of Hidden Units

In order to examine the effects induced by the number of hidden units, the performance metrics are now aggregated by taking means of values with same number of hidden units:

num_cells	mean_prof	max_prof	min_prof	median_prof	std_prof
10	56.15	138.68	-53.35	111.71	66.96
20	6.70	138.33	-159.37	0.00	87.20
30	-20.80	111.71	-164.73	0.00	63.19
40	-10.98	111.71	-209.42	0.00	64.66
50	-39.87	0.00	-255.98	0.00	85.74
60	1.41	313.27	-201.54	0.00	102.21
70	-11.92	111.71	-159.09	0.00	58.34
80	-1.34	242.99	-217.96	0.00	109.58
90	-6.05	199.66	-230.96	0.00	111.09
100	-17.46	216.25	-198.95	0.00	87.53

num_cells	mean_prof	max_prof	min_prof	median_prof	std_prof
10	710.19	1527.25	0.00	542.47	517.90
20	641.05	1137.09	207.78	626.50	341.37
30	934.42	1527.25	207.78	889.53	395.68
40	779.43	1511.02	0.00	697.96	441.02
50	918.27	1527.25	304.75	862.62	334.55
60	1016.76	1527.25	458.18	939.32	386.81
70	817.96	1527.25	0.00	618.76	509.22
80	1105.19	1527.75	550.36	1101.32	398.42
90	1180.26	1546.23	487.18	1527.25	429.61
100	1222.26	1531.15	336.58	1527.25	456.07

Fig. 22 Predicted profits by MRNN on CADHKD (left) and NASDAQ(right) with different aggregations grouped by number of hidden cells.

From the table of NASDAQ on the right of Fig. 22, it can easily be observed, e.g. from increaseing median and mean profits, that the performance improves with a larger number of hidden units. However, the performances on CADHKD on the left of Fig. 22 is more fluctuating with different numbers of units. With more hidden units, even MRNN model give the highest max profits, at the same time it gives the lowests min profits.

10.1.2.4. Number of Hidden Layers

num_layers	mean_prof	max_prof	min_prof	median_prof	std_prof
2	-13.82	313.27	-230.96	0.0	130.06
4	2.38	216.25	-209.42	0.0	105.92
6	-4.76	242.99	-159.37	0.0	68.08
8	1.39	197.47	-255.98	0.0	68.75
10	-7.27	0.00	-135.14	0.0	26.53

num_layers	mean_prof	max_prof	min_prof	median_prof	std_prof
2	680.81	1527.25	0.00	610.67	389.50
4	808.97	1546.23	0.00	715.13	457.67
6	988.14	1527.75	478.29	904.95	386.95
8	1023.55	1527.25	0.00	1103.90	459.05
10	1161.43	1527.75	285.21	1404.95	432.06

Fig. 23 Predicted profits by MRNN on CADHKD (left) and NASDAQ(right) with different aggregations grouped by number of hidden layers.

Table of the left in Fig. 23 shows that influence of changing the number of hidden layers is not obvious on CADHKD, similar to the effects of changing the number of hidden units (See Section 10.1.2.3). Although the max profits is highest with just 2 layers, the max profits decrease with more layers and the min profits increase at the same time.

Meanwhile, it is clear for NASDAQ that better performances with more hidden layers, by observing from the increasing mean and median profits.

10.1.2.5. Discussion

Section 10.1.2 attempts to explore how does the structure of MRNN affects its performances. Although it is clear that more profits can be made on NASDAQ with more hidden layers and units, this effect is not obvious on CADHKD. For example, increasing more hidden layers do improved the worst case performance, but the average performance is not improved too much.

With the effects of changing the structure being unclear, one should still note that no matter how did the hyper-parameters are changed, generally MRNN does not provide as much profits as SRNN does. In fact, the differences in profits are obvious while considering their range of profits (e.g. on CADHKD, SRNN: -362 - +1321, MRNN: -256 - +243). Therefore, from the results above, MRNN fails to improve SRNN by adding more hidden layers.

10.1.3. Stock Price Level Prediction with MRNN and Convolutional Output Layer (MRNN+CNN)

The previous section shows that the MRNN model built in this project failed to improve performance of the SRNN model mentioned in Section 10.1.1. In this section, a CNN output layer is added to the MRNN model to see whether the performance of MRNN can be improved. In the following sections, this model is referred as the MRNN+CNN model.

10.1.3.1. Test Cases Information

Information about the test cases on MRNN+CNN model are given as follows:

- Train data size (in years, approx.)
    - CADHKD, NASDAQ: 10 years
- Test data size: 100 days
- Sequence length = 20
- Number of hidden units (RNN)= {10, 20, …, 100}
- Number of hidden layers (RNN) = {2, 4, 6, 8, 10}
- Size of stride window (CNN) = {2, 5, 10}
- Number of hidden units (CNN) = {8, 16, 32, 64}

10.1.3.2. Effects of Adding CNN Output Layer

Since the main difference of the models in Sections 10.1.2 and 10.1.3 is that the latter adds a CNN output layer to the former one, by comparing with the results in Sections 10.1.2.3 and 10.1.2.4 may give information on whether the CNN output layer improves the performances.

num_cells	mean_prof	max_prof	min_prof	median_prof	std_prof	num_cells	mean_prof	max_prof	min_prof	median_prof	std_prof
10	77.58	276.53	-159.49	111.71	90.83	10	702.15	1910.27	0.00	763.77	411.44
20	119.38	400.09	-222.41	111.71	118.79	20	711.55	1478.78	0.00	695.90	370.75
30	75.12	338.29	-120.18	111.71	89.85	30	596.92	1809.85	0.00	554.08	462.80
40	57.13	427.43	-162.70	81.92	91.08	40	628.21	1615.82	0.00	658.49	476.72
50	74.36	405.40	-190.99	91.17	116.25	50	553.48	1527.25	-44.46	356.80	473.57
60	77.03	382.19	-220.30	111.71	89.63	60	556.15	1638.86	0.00	426.92	477.92
70	100.57	322.54	-81.19	111.71	79.08	70	605.55	1527.25	0.00	481.53	530.78
80	58.50	366.33	-262.39	105.01	97.02	80	689.74	1855.34	0.00	645.54	539.62
90	79.33	341.55	-229.13	111.71	118.15	90	611.33	1807.95	-27.77	500.92	556.51
100	86.38	436.04	-140.82	111.71	93.74	100	570.53	1536.25	0.00	459.63	503.90

Fig. 24 Predicted profits by MRNN+CNN on CADHKD (left) and NASDAQ(right) with different aggregations grouped by number of hidden cells.

num_layers	mean_prof	max_prof	min_prof	median_prof	std_prof	num_layers	mean_prof	max_prof	min_prof	median_prof	std_prof
2	81.00	364.52	-220.30	111.71	104.92	2	603.97	1855.34	-44.46	585.81	442.60
4	78.24	436.04	-262.39	111.71	122.15	4	626.78	1807.95	0.00	564.81	433.36
6	71.25	391.00	-229.13	111.71	94.34	6	640.09	1910.27	0.00	583.47	504.85
8	92.46	384.24	-176.72	111.71	94.20	8	652.32	1809.85	-27.77	606.63	530.17
10	79.74	366.33	-144.68	111.71	80.85	10	589.66	1542.12	0.00	469.37	502.58

Fig. 25 Predicted profits by MRNN+CNN on CADHKD (left) and NASDAQ(right) with different aggregations grouped by number of hidden layers.

Recall that the influences of changing the number of hidden units and layers are obscure on CADHKD as discussed previously. This is similar for MRNN+CNN since there is not a clear trend observed in tables for CADHKD in both Fig. 24 and 25. On the other hand, considering the performances on NASDAQ, the strong positive correlation observed between the profits made and the numbers of hidden cells or units for MRNN models seems to be weakened in MRNN+CNN models. Here, it is difficult to decide the best numbers to be used for both CADHKD and NASDAQ.

Regardless how the changes on strucure affect the performances of MRNN+CNN, it is notable that generally the performances improved as values of both mean, median and max profits increased as compared to that are made by the MRNN-only models. Therefore, adding a CNN output layer seems to be an improvement to MRNN.

10.1.3.3. Size of Stride Window and Number of Hidden Units (CNN)

Given with the output from RNN layer which is in the shape of a sequence, the CNN output layer is designed to capture some patterns in the RNN output. By capturing

patterns with stirde windows of different lengths, or different numbers of hidden units, CNN may give different performance.

strides	conv_dims	mean_prof	max_prof	min_prof	median_prof	std_prof
2	8	102.70	284.23	-190.99	111.71	87.60
	16	90.50	436.04	-222.41	111.71	122.03
	32	62.19	338.29	-162.70	110.78	96.38
	64	63.28	427.43	-229.13	9.30	121.25
5	8	94.14	322.54	-151.05	111.71	93.86
	16	77.01	391.00	-144.86	111.71	92.69
	32	97.57	366.33	-35.83	111.71	85.48
	64	49.14	314.08	-221.45	37.25	115.44
10	8	73.35	250.47	-97.65	111.71	73.81
	16	68.72	142.99	-201.97	111.71	80.50
	32	111.03	405.40	-31.00	111.71	100.15
	64	76.82	384.24	-262.39	111.71	110.12

strides	conv_dims	mean_prof	max_prof	min_prof	median_prof	std_prof
2	8	552.51	1527.25	30.72	561.01	434.14
	16	550.09	1748.97	0.00	481.95	477.06
	32	729.91	1542.12	0.00	744.10	497.19
	64	829.02	1910.27	22.81	866.87	526.11
5	8	581.49	1527.25	-27.77	650.23	395.17
	16	532.14	1536.25	0.00	551.91	420.72
	32	508.43	1582.13	0.00	374.60	442.66
	64	720.58	1807.95	0.00	618.72	558.61
10	8	609.75	1527.25	0.00	593.22	430.03
	16	752.49	1855.34	0.00	718.11	507.44
	32	536.08	1527.25	-44.46	385.98	486.86
	64	568.26	1809.85	0.00	377.51	515.58

Fig. 26 Predicted profits by MRNN+CNN on CADHKD (left) and NASDAQ(right) with different aggregations grouped by number of stride window lengths and hidden units.

Fig. 26 gives performances of MRNN+CNN with diffenent stride window lengths and different number of hidden units in CNN layer. First, consider the table of CADHKD. Although there does not seems to be a general trend, one can examine the best number of units (conv_dims) for each stride window sizes. For stride windows of sizes 2, 5 and 10, the best numbers of units with highest mean profits are given by 8, 32 and 32 respectively. This may suggest that more hidden units are required for larger stride windows for CNN layer to provide better results.

However, while considering NASDAQ, this observation may not be applicable. For stride windows of sizes 2, 5 and 10, the best numbers of units with highest mean profit, as well as max profit, are given by 64, 64 and 16 respectively.

This may be related to the fluctuations of stock prices within different periods (e.g. 2, 5, 10 days). For CADHKD which is more fluctuating, there may be more different patterns for longer periods, so more number of units are required to do the matching between patterns and predictions. On the other hand, with a longer period length, it will have a higher chance for the sequence to be an increasing trend, thus less variation with longer windows. So fewer units are required for best performance.

10.1.3.4. Discussion

It is observed in Section 10.1.3.2 that the CNN output layer did improve the performance of MRNN. With such insight, it is natural for one to consider how to configure the CNN layer in order to provide greater improvements. Although the results in Section 10.1.3.3 was not clear, it does suggest that the stride window size and the number of hidden units in CNN layer may be related and hence should be considered together.

10.1.4. RNN Summary

For the three models built with the RNN base model, it is found that the SRNN performed the best among all three models, with the greatest profits produced in average. Although MRNN was not performing well, it is found that adding a CNN layer could make an improvement. However, SRNN was not experimented together with a CNN output layer, which should be done in the future to examine whether CNN makes the same boosting effect to SRNN.

10.2. RL Models

While RNN models perform predictions on price levels, RL models built attempt to learn a best policy to reacts to different situations in the market. Since Q Learning is the learning algorithm chose in this project, where the learning operations depend heavily on the state space defined for the model, different state definitions and exploration algorithms have been tested, where the results will be presented in the following sections.

10.2.1. QL Model 1 – Price-related States and Random Exploration

The first Q Learning model adopts state space that is defined by partitioning the price state. More information is given in following sections.

10.2.1.1. Test Cases Information

The followings give information about the QL model built, including the state definitions, exploration algorithm and other related hyper-parameters:

- Train data lengths (yrs):
  - CADHKD: {1, 4, 7, …, 25}
  - NASDAQ: {1, 4, 7, 10, …, 40, 43, 46}
- Test data size: 100 days

- States:
    - Price-related: 100 uniform intervals from min to max price
    - Trend-related: Up, Down
- Actions:
    - Sell/Hold/Buy
- Q Table Shape (= #states * #actions):
    - 600 = (100*2) * 3
- Exploration: Random action
- Discount Factors γ:
    - CADHKD: {0, 0.1, 0.2, …, 1.0}
    - NASDAQ {0, 0.2, 0.4, …, 1.0}
- Exploration Probability η:
    - CADHKD: {0, 0.1, 0.2, …, 1.0}
    - NASDAQ {0, 0.2, 0.4,…, 1.0}

10.2.1.2. Training Data Length

data_year	mean_prof	max_prof	min_prof	median_prof	std_prof
1	145.63	632.03	-385.26	134.22	157.31
4	1.70	632.57	-448.21	5.32	190.49
7	134.93	615.58	-391.18	116.75	186.11
10	130.57	632.05	-371.90	114.02	174.40
13	126.81	678.17	-435.36	111.65	175.84
16	81.66	646.88	-500.87	102.18	188.92
19	79.42	623.72	-517.35	99.26	189.28
22	78.00	566.03	-483.01	93.74	184.29
25	90.50	723.91	-489.92	111.65	197.82

data_year	mean_prof	max_prof	min_prof	median_prof	std_prof
1	704.51	733.94	675.08	704.51	30.40
6	700.57	733.94	675.08	686.02	27.07
11	715.55	733.94	675.08	733.94	26.91
16	717.17	733.94	675.08	723.00	22.54
21	715.55	733.94	675.08	733.94	26.91
26	706.82	733.94	675.08	723.00	27.90
31	708.19	733.94	675.08	723.00	26.45
36	723.16	733.94	675.08	728.47	17.55
41	711.18	733.94	675.08	728.47	27.78
46	727.53	733.94	686.02	733.94	12.21

Fig. 27 Predicted profits by QL Model 1 on CADHKD (left) and NASDAQ(right) with different aggregations grouped by lengths of training data.

Model 1 has been tested with training data of different lengths, which is shown in Fig. 27 For CADHKD, considering the mean profits, Model 1 is performing better with short data lengths (<=13 years), but the performance starts to drop with more train data (>=16 years). On the other hand, for NASDAQ, Model 1 provides similar performance for all data lengths. This is believed to be related to the definition of state space, which will be discussed in the discussion section of Model 1 (See Section 10.2.1.4).

10.2.1.3. Empty Q Values

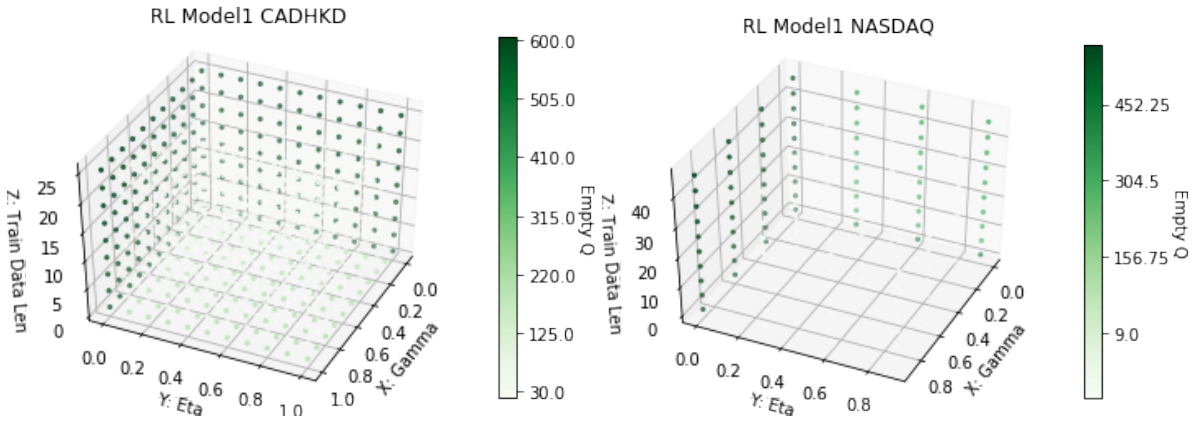


Fig. 28 Empty Q values by QL Model 1 on CADHKD (left) and NASDAQ (right).

As discussed in Section 9.3(d), the number of empty Q values is an indicator of degree of exploration. Fig. 28 shows the number of empty Q values produced by different sets of parameters, where a darker color means more Q values are unaltered, and a ligher color means the contrary. The correlations here is easy to observe – the smaller the values of discount factors γ or exploration probability η, the larger the number of empty Q values.

10.2.1.4. Discussion

The above results suggest that the performance is greatly influenced by the state space definition. First note that Model 1 defines states by partitioning the price range into intervals. Consider also the characteristics of the two data sets, even CADHKD shows more fluctuations, the variations lie in similar ranges over times, as the min and max prices were not changing too much. However, as NASDAQ seems to be increasing in one way, the max price also keeps increasing, and the price range gets larger. Therefore, for each iteration, states are visited repeatedly in CADHKD, but states in NASDAQ are only visited with few times per iteration, no matter how long the data set is.

Besides, Section 10.2.1.3 suggested that low values of γ or η contributes to a larger number of unvisited states. In fact, the formula for updating Q values itself already suggested the reason:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta[r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

If the values of γ or η is small, then the latter term diminish very quickly so the previous price levels at each state are imposing less influence to the Q values.

Therefore, the Q values will become similar to each other, hindering the model from selecting the best actions.

10.2.2. QL Model 2 – Price-related States and ε-Greedy Exploration

For Model 2, the state definition is the same as Model 1. However, the mean to explore the state space is changed to ε-Greedy Algorithm.

10.2.2.1. Test Cases Information

The followings give information about the QL model built, including the state definitions, exploration algorithm and other related hyper-parameters:

- Train data lengths (yrs):
  - CADHKD: {1, 8, 15, 22}
  - NASDAQ: {1, 8, 15, 22, …, 43}
- Test data size: 100 days
- States:
  - Price-related: 100 uniform intervals from min to max price
  - Change-related: Up, Down
- Actions:
  - Sell/Hold/Buy
- Q Table Shape (= #states * #actions):
  - 600 = (100*2) * 3
- Exploration: ε-Greedy Algorithm
  - ε: {0.3, 0.6, 0.9}
- Discount Factors γ: {0.3, 0.6, 0.9}
- Exploration Probability η: {0.3, 0.6, 0.9}

10.2.2.2. Training Data Length

Fig. 29 gives the peformance of Model 2 on CADHKD and NASDAQ with different lenghts of training data. The result is similar to that in Section 10.1.3 – better performance with around 15 years on CADHKD and similar performances on different lengths of NASDAQ. However, Model 2 generally gives much higher (about a double) profits on NASDAQ than Model 1, while slightly lower profits on CADHKD than Model 1.

data_year	mean_prof	max_prof	min_prof	median_prof	std_prof
1	7.15	441.69	-491.57	5.01	256.88
8	212.66	1364.19	-570.89	216.38	404.04
15	337.48	1020.63	-326.35	388.53	343.71
22	178.73	896.02	-859.35	111.49	366.65

data_year	mean_prof	max_prof	min_prof	median_prof	std_prof
1	1513.45	1563.59	1504.73	1504.73	20.34
8	1512.08	1563.59	1504.73	1504.73	17.68
15	1521.36	1563.59	1504.73	1504.73	25.47
22	1511.27	1563.59	1504.73	1504.73	16.57
29	1511.27	1552.65	1504.73	1504.73	15.31
36	1523.54	1563.59	1504.73	1504.73	26.49
43	1559.13	1563.59	1552.65	1563.59	5.48

Fig. 29 Predicted profits by QL Model 2 on CADHKD (left) and NASDAQ(right) with different aggregations grouped by lengths of training data.

10.2.2.3. Empty Q Values

Fig. 30 shows that ε-Greedy Algorithms explored the state space in a different way as compared to choosing random actions in Model 1. The heat map of CADHKD shows that the large number of empty Q values only occurs with short data length. Also, the range of empty Q values also decreased (Model 1: 30-600; Model 2: 21-192). Even though the profits are not improved, the state spaces was visited more thoroughly.

On the other hand, the behaviour in NASDAQ seems to be the same as in Model 1, which the reason may be due to the simple patterns in NASDAQ. However, considering the improved profits of Model 2 suggested in Section 10.2.2.2, exploitation on the best Q values by ε-Greedy may be helpful in exploring the state space where revisiting is less likely.

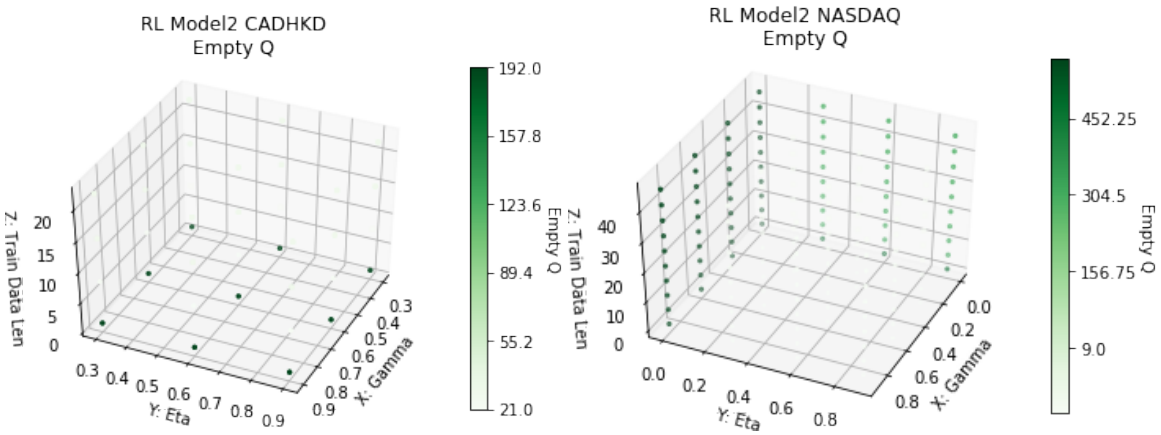


Fig. 30 Empty Q values by QL Model 2 on CADHKD (left) and NASDAQ (right).

10.2.2.4. ε-Greedy

eps	mean_prof	max_prof	min_prof	median_prof	std_prof	eps	mean_prof	max_prof	min_prof	median_prof	std_prof
0.3	82.32	720.42	-288.95	11.60	233.81	0.3	1514.66	1563.59	1504.73	1504.73	21.05
0.6	154.99	1364.19	-859.35	152.12	436.87	0.6	1517.29	1563.59	1504.73	1504.73	23.06
0.9	314.70	1020.63	-570.89	334.75	355.35	0.9	1533.24	1563.59	1504.73	1552.65	26.33

Fig. 31 Predicted profits by QL Model 2 on CADHKD (left) and NASDAQ(right) with different aggregations grouped by ε values.

From the table of CADHKD in Fig. 31, it can be observed that profits increases with higher ε values. Meanwhile, such trend cannot be observed for NASDAQ, same as the performances in Section 10.2.2.2 with different train data length.

10.2.2.5. Discussion

The profits made by Model 2 are more than that by Model 1 significantly generally (about a double for both CADHKD and NASDAQ). It shows that ε-Greedy does help to learn a policy that gives better performance by exploiting information about the best actions learnt on the fly.

10.2.3. QL Model 3 – Indicator-related States and Random Exploration

Instead of defining the state space by partitioning the price range, stock indicators are used to construct the state space and the signals given by the indicators are used to distinguish states between each other.

10.2.3.1. Indicators

This section gives a brief discussion on the indicators used, and the meanings of signals generated by the indicators.

(a) 20-day/50-day Moving Average (MA20/MA50)

By taking averages in a moving window, MA attempts to removes noises and give the direction of price changes. Therefore, MA can acts as a support or resistance line such that the price levels are bounded by the line which can also indicate where the price is going to change its movements.

Meanwhile, when the price level runs over MA, i.e. a price crossover occurs, there is a potential change of trend in near future.

In Model 3, MA with two different window sizes are used – one short-term average (MA20) and one long-term average (MA50). The crossover of the long-/short-term MAs are usually considered by investors since it also give signs to bearish ("Golden Cross") and bullish ("Death Cross") markets.

(b) Moving Average Convergence Divergence (MACD)

MACD considers the relationship between two MAs, which are the 12-day and 26-day Exponential MAs (EMAs) according to common practice. The difference between the two MAs is compared with a signal line (usually 9-day EMA) and give a signal for buy or sell when MACD crosses the signal line (Crossover).

(c) Relative Strength Index (RSI)

Relative strength is calculated by dividing the average gain in a time window by the average loss in the same window. It measures the momentum of the price changes. The value of RSI ranges from 0 to 100. In usual practice, if RSI exceeds the value of 70 or 80, then it gives a signal of overbought which tells the price may drop back in the near future. On the contrary, if the value goes below 30 or 20, an undervalued signal is given. Therefore, in Model 3, the states corresponding to RSI are partitioned into ranges in 0 to 100, which informs different momentums and signals in each range.

(d) Commodity Channel Index (CCI)

CCI measures the variation from the mean price. It is usually considered as an indicator for extreme points. The calculation of CCI is designed to make 70~80% of its values fall into the range from -100 to +100. Going above or below this range gives a signal of being overbought or undervalued. Similar to RSI, in Model 3, the range from -100 to +100 is partitioned into intervals to form a state space.

(e) Bollinger Band® (BB)

Bollinger Band® consists of three bands – a 21-day MA as Middle Band, and two bands that differ by 2 SD from Middle Band as Upper Band and Lower Band. The trend of the stock price can be reflected by which side Middle band is close to, which is measured by Percent B (%B). %B as a percentage ranges from 0 to 1,

with 0.5 means the level of Middle Band. %B is considered as an indicator of uptrends or downtrends.

10.2.3.2. Test Cases Information

The following information describes the states space and parameters used for constructing Model 3:

- Train data lengths (yrs):
  - CADHKD: {1, 8, 15, 22}
  - NASDAQ: {1, 8, 15, 22, …, 43}
- Test data size: 100 days
- States:
  - MA(20/50) Change: Up/Down
  - Long-/Short-term MA Crossovers: MA20>MA50 or MA20<MA50
  - MA(20/50)-Price Crossovers: MA>Price or MA<Price
  - MACD Change: Up/Down
  - MACD-Price Crossovers: MACD>0 or MACD<0
  - RSI Level: {(≤0),[20,40], …,[80,100], (≥100)} (6 states)
  - CCI Level: {(≤-100),[-100,-80], …,[80,100], (≥100)} (12 states)
  - %B: {(≤0),[0.2,0.4], …,[0.8,1], (≥1)} (6 states)
- Q Table size: 165888
- Actions:
  - Sell/Hold/Buy
- Q Table Shape (= #states * #actions):
  - 600 = (100*2) * 3
- Exploration: ε-Greedy Algorithm
- Discount Factors γ: {0.3, 0.6, 0.9}
- Exploration Probability η: {0.3, 0.6, 0.9}

10.2.3.3. Training Data Length

The table on the left in Fig. 32 points to similar conclustion as in Section 10.2.1.2 and 10.2.2.2 that the performance on CADHKD is the best with about 15 years of training data. However, one should note that this time all the mean profits are negative, which shows that the states spaces is not applicable to CADHKD.

data_year	mean_prof	max_prof	min_prof	median_prof	std_prof
1	-69.83	186.01	-361.74	-69.61	126.85
8	-73.72	280.20	-316.41	-97.23	150.27
15	-24.78	223.98	-239.69	-28.57	104.26
22	-57.74	165.69	-348.64	-86.28	134.62

data_year	mean_prof	max_prof	min_prof	median_prof	std_prof
1	308.31	834.70	-236.05	102.24	353.64
8	228.22	709.65	-408.98	395.15	369.10
15	251.14	709.45	-397.19	391.56	308.64
22	47.85	478.68	-622.64	215.90	338.99
29	-51.41	419.75	-612.13	-140.76	334.62
36	35.26	626.30	-438.53	-78.27	349.40
43	10.64	540.80	-469.87	-106.95	333.16

Fig. 32 Predicted profits by QL Model 3 on CADHKD (left) and NASDAQ(right) with different aggregations grouped by lengths of training data.

Meanwhile, different to the conclusion in Section 10.2.1.2 and 10.2.2.2, the performance on NASDAQ shows variations that the performance is better with shorter length of training data. However, same as the profits on CADHKD, generally the profits dropped as compared to Model 1 and 2.

10.2.3.4. Empty Q values

Fig. 33 shows the empty Q values of Model 3 after training. Considering the entire space size is 165888, Fig. 34 suggests that Model 3 fails to make enough exploration to the entire state space.

gamma	eta	mean_empty_q	max_empty_q	min_empty_q	median_empty_q	std_empty_q
0.3	0.3	165078.92	165509	164832	164988.5	271.38
	0.6	165079.92	165515	164832	164986.5	272.17
	0.9	165079.08	165513	164832	164987.5	272.29
0.6	0.3	165079.25	165512	164832	164987.5	271.97
	0.6	165078.58	165512	164832	164987.0	272.23
	0.9	165078.42	165509	164832	164986.5	271.83
0.9	0.3	165078.25	165509	164832	164986.5	271.85
	0.6	165078.00	165507	164832	164986.5	271.57
	0.9	165078.25	165510	164832	164986.5	272.00

gamma	eta	mean_empty_q	max_empty_q	min_empty_q	median_empty_q	std_empty_q
0.3	0.3	165036	165615	164733	164982	276.04
	0.6	165035	165612	164733	164982	275.80
	0.9	165035	165610	164733	164982	275.46
0.6	0.3	165035	165610	164733	164982	275.67
	0.6	165034	165608	164733	164982	275.42
	0.9	165035	165611	164733	164982	275.76
0.9	0.3	165035	165614	164733	164982	276.09
	0.6	165034	165611	164733	164982	275.77
	0.9	165034	165609	164733	164982	275.57

Fig. 33 Empty Q values by QL Model 3 on CADHKD (left) and NASDAQ(right) with different aggregations grouped by values of γ and η.

10.2.3.5. Discussion

Since Model 3 is having a poorer performance with lower profits and more unexplored states, it can be concluded that the state space definition is not desirable. However, it does show that it is possible for a suitable definition, QL models can still learn stock data with simple patterns like NASDAQ (See Section 10.2.3.3).

10.2.4. RL Summary

The definition of state space greatly influences the performance of RL models. Model 1 adopts a state space that is simply defined by partitioning between the min and max price in training set. The performance of Model 1 on NASDAQ (See Section 10.2.1.2) reveals the risk of such definition – train data that mainly in one direction leads to insufficient exploration on the state space. Even though there seems to be a higher degree of exploration on CADHKD by Model 1, the performance is worse than that achieved by SRNN.

Model 2 shows that a better exploration algorithm can improve the performance of Q Learning model, even the state space is not well-designed. Model 3 attempts to modify the state definition, but the result is not desirable. Therefore, one can see that the state definition impose great influence on the performance, and more researches should be made on how to define the state space in the future.

10.3. DRL Models

Models in Section 10.2 shows that the state spaces should be defined delicately in order to give good performance. Therefore, instead of using a Q table, DRL models capture the state space with a deep neural network, the model is then expected to learn the best policy in the network. In the following sections, the performance of DQN and DDQN models will be discussed.

10.3.1. Deep Q Network (DQN) Model

DQN represents the state space with a deep neural network. In this project, it is designed to be a dense layer. The following experiments include testing on the structure of the dense layer, and other hyper-parameters on training procedures and data:

10.3.1.1. Test Cases Information

The DQN model developed was tested with different lengths of training data, and was tested with different structures of the dense layer:

- Train data lengths (yrs):
  - CADHKD: {1, 8, 15, 22}
  - NASDAQ: {1, 8, 15, 22, …, 43}
- Test data size: 100 days

- Signals:
    - MA(20/50) Change: Up/Down
    - Long-/Short-term MA Crossovers: MA20>MA50 or MA20<MA50
    - MA(20/50)-Price Crossovers: MA>Price or MA<Price
    - MACD Change: Up/Down
    - MACD-Price Crossovers: MACD>0 or MACD<0
    - RSI Level: {(≤0),[20,40], …,[80,100], (≥100)} (6 signals)
    - CCI Level: {(≤-100),[-100,-80], …,[80,100], (≥100)} (12 signals)
    - %B: {(≤0),[0.2,0.4], …,[0.8,1], (≥1)} (6 signals)
- Actions:
    - Sell/Hold/Buy
- Exploration: ε-Greedy Algorithm
- DQN Hidden Units: {5, 10, 15, 20, 25, 30}

10.3.1.2. Training Data Length

data_year	mean_prof	max_prof	min_prof	median_prof	std_prof
1	458.76	1234.63	-205.67	387.73	475.60
8	1026.60	1366.38	286.15	1285.60	464.76
15	527.02	1461.89	-35.02	390.52	606.97
22	142.42	655.94	-354.34	200.85	347.59

data_year	mean_prof	max_prof	min_prof	median_prof	std_prof
1	735.89	755.59	705.79	740.76	20.56
8	442.68	755.59	0.00	572.46	365.98
15	525.44	755.59	86.02	675.06	279.78
22	768.46	832.94	693.94	759.25	51.43
29	644.90	755.59	305.09	702.45	171.27
36	585.92	755.59	148.56	669.11	226.59
43	703.57	792.31	416.37	755.59	141.69

Fig. 34 Predicted profits by DQN on CADHKD (left) and NASDAQ(right) with different aggregations grouped by lengths of training data.

The influence of training data on DQN model can be observed from its performance on CADHKD which is shown in Fig. 34. The performance is better with training data of 8 years. However, there is not a clear trend for profits on NASDAQ.

Besides, comparing to QL Model 1 (See Section. 10.2.1.2), DQN has a similar performance on NASDAQ (on mean profits, Model 1: ~700, DQN: ~500-700), but DQN greatly improved from Model 1 on NASDAQ (on mean profits: Model 1: ~1-140, DQN: ~140-1000). DQN also performs better than Model 2 (See Section. 10.2.2.2) on CADHKD (on mean profits: Model 2: ~10-340, DQN: ~140-1000), but

worse than Model 2 on NASDAQ (on mean profits: Model 2: ~1500, DQN: ~140-1000). Generally, DQN performs better on CADHKD.

10.3.1.3. Number of Hidden Units (DQN)

num_cells	mean_prof	max_prof	min_prof	median_prof	std_prof
5	927.93	1250.22	272.66	1094.42	457.53
10	225.69	598.65	-121.43	212.76	305.54
15	352.55	1337.23	-354.34	213.66	737.46
20	621.32	1366.38	-205.67	662.29	643.55
25	192.33	292.63	0.00	238.34	136.45
30	912.38	1461.89	211.16	988.23	584.98

num_cells	mean_prof	max_prof	min_prof	median_prof	std_prof
5	613.38	820.85	148.56	699.45	242.49
10	540.00	792.31	261.46	542.65	217.81
15	553.63	832.94	0.00	737.57	351.74
20	616.93	759.54	0.00	693.94	274.19
25	717.28	762.91	648.20	727.78	46.10
30	736.09	755.59	657.75	755.59	35.88

Fig. 35 Predicted profits by DQN on CADHKD (left) and NASDAQ(right) with different aggregations grouped by number of hidden units.

From Fig. 35, it is hard to observe a strong correlation between the number of hidden units and the profits made on CADHKD. For NASDAQ, the correlation is weak but still observable.

10.3.1.4. Discussion

Tables in section 10.3.1.2 and 10.3.1.3 both shows that DQN works better than QL Model 1 and 2 on CADHKD. Therefore, for trading with a more fluctuating stock, DQN may be more preferable than QL models. It shows the potential of DQN model which is worthwhile for further researches. However, on NASDAQ, even DQN works better than Model 1, Model 2 already provides better performance by a fix of ε-Greedy Algorithm.

10.3.2. Double Deep Q Network (DDQN) Model

DDQN also uses a deep neural network to represent the state space. Unlike the DQN model in Section 10.3.1, DDQN uses two DQNs in the learning phase.

10.3.2.1. Test Cases Information

The information of experiments on DDQN model is given as followings:

- Train data lengths (yrs):
  - CADHKD: {1, 8, 15, 22}
  - NASDAQ: {1, 8, 15, 22, …, 43}
- Test data size: 100 days

- Signals:
    - MA(20/50) Change: Up/Down
    - Long-/Short-term MA Crossovers: MA20>MA50 or MA20<MA50
    - MA(20/50)-Price Crossovers: MA>Price or MA<Price
    - MACD Change: Up/Down
    - MACD-Price Crossovers: MACD>0 or MACD<0
    - RSI Level: {(≤0),[20,40], …,[80,100], (≥100)} (6 signals)
    - CCI Level: {(≤-100),[-100,-80], …,[80,100], (≥100)} (12 signals)
    - %B: {(≤0),[0.2,0.4], …,[0.8,1], (≥1)} (6 signals)
- Actions:
    - Sell/Hold/Buy
- Exploration: ε-Greedy Algorithm
- DQN Hidden Units: {10, 20, 30}
- Training Batch Size: {10, 50, 200}

10.3.2.2. Training Data Length

Fig. 36 shows the performances of DDQN with different lengths of training data. Unfortunately, it seems that there is no strong correlation between the two observed from both tables of CADHKD and NASDAQ.

data_year	mean_prof	max_prof	min_prof	median_prof	std_prof
1	25.67	433.19	-258.86	-12.80	185.79
8	-3.27	302.25	-294.17	-10.81	186.75
15	91.22	473.08	-319.45	72.35	217.45
22	51.67	540.46	-243.46	83.26	201.65

data_year	mean_prof	max_prof	min_prof	median_prof	std_prof
1	451.86	760.40	182.42	432.80	167.66
8	343.52	786.70	20.80	340.02	174.96
15	410.41	574.04	149.57	409.13	116.70
22	329.13	632.39	2.13	332.19	143.81
29	296.57	656.59	13.71	293.02	192.43
36	386.77	636.20	94.97	370.81	167.93
43	392.08	885.44	161.53	356.70	180.73

Fig. 36 Predicted profits by DDQN on CADHKD (left) and NASDAQ(right) with different aggregations grouped by training data length.

10.3.2.3. Number of Hidden Units

num_hidden	mean_prof	max_prof	min_prof	median_prof	std_prof
10	-30.63	266.44	-294.17	-12.89	157.88
20	80.21	540.46	-319.45	49.35	231.90
30	74.39	473.08	-223.14	56.71	182.36

num_hidden	mean_prof	max_prof	min_prof	median_prof	std_prof
10	392.94	786.70	2.13	397.24	189.75
20	340.32	633.84	91.42	337.65	148.20
30	385.45	885.44	19.28	378.35	164.04

Fig. 37 Predicted profits by DDQN on CADHKD (left) and NASDAQ(right) with different aggregations grouped by number of hidden units.

According to Fig. 37, more profits are given on CADHKD by more hidden units. However, changing the number of hidden units does not change the profits made on NASDAQ too much.

10.3.2.4. Number of Learning Batch Size

In DDQN model, the learning procedures are controlled by memory replay. In each replay, a small batch of memories is sampled from the whole memory bank. The following studies whether the batch size affects the performance of learning. Nonetheless, from Fig. 38, since the profits are not variating obviously, the effect of changing the batch size seems to be insignificant.

batch_size	mean_prof	max_prof	min_prof	median_prof	std_prof
10	45.63	540.46	-243.66	22.75	192.54
50	50.14	410.88	-243.46	44.61	182.66
200	28.20	433.19	-319.45	37.09	222.17

batch_size	mean_prof	max_prof	min_prof	median_prof	std_prof
10	384.69	885.44	2.13	387.45	189.12
50	344.89	682.76	13.71	341.99	144.25
200	389.12	786.70	43.62	371.34	169.55

Fig. 38 Predicted profits by DDQN on CADHKD (left) and NASDAQ(right) with different aggregations grouped by batch size.

10.3.2.5. Discussion

From the experiment results above, it is difficult to conclude that what are the factors that affect the performance of DDQN model. Meanwhile, compared with the experiment results of DQN model, DDQN is outperformed by DQN model that DDQN generally makes less profit.

Created by the DeepMind team, DDQN was designed for playing Atari games, which is an application that has little intersection with time series nor trading. Therefore, in order to make DDQN to produce good performance, more adjustments should be made to the model. More studies should be made on DDQN models.

10.3.3. DQN Summary

Although DDQN is considered as an enhancement to DQN, DDQN fails to improve the performance of DDQN. In fact, SRNN and QL Model 1 and 2 also perform better than DDQN.

On the other hand, even not performing as well as SRNN and QL Model 2, DQN shows better performance than QL Model 1. Also, according to Section 10.3.1.2, since DQN is showing better performance on CADHKD, using a deep NN might be better to capture the state space in a more fluctuating market.

10.4. Summary on Experiment Results

The experiments above aim at discovering the factors that affect the performances of models. The following summarizes some key findings from the above sections.

For RNN models, it is observed that the performance is greatly influenced by the training data set. However, in RL and DRL models, such influence is weaker than that on RNN models.

Besides, generally RNN models, especially SRNN and MRNN+CNN, perform better then RL and DRL models. RL and DRL models are not performing well since the performances were restricted by the state space definition. DQN shows that with a better way to capture the state space, the performances can be improved. Also, with a better way to explore the state space, for example using ε-Greedy Algorithm, the performance could also be improved.

Even though the above experiments may suggest ways to improve the models, it is still difficult to determine whether the models can be used in practice. In Section 11, some test cases were designed to create possible situations in real market, and the performances of models will be tested.

11. Experiment Results – Test Cases

In order to test whether the models are practicable for real-life trading, some test cases were designed to simulate different situations in real market. According to the performances of models at each situation, one may be able to determine which model to use given with a situation.

11.1. Model Specification

Unlike experiments in Section 10, for the coming experiments, it is attempted to design the "best" models by utilizing the information given in previous sections. The following gives the information of the model used.

Models	Specification
SRNN	Sequence length: 15; Hidden units: 45
MRNN	Sequence length: 15; Hidden units: 45; Layers: 3;
MRNN+CNN	Sequence length: 15; RNN hidden units: 45; RNN Layers: 3; CNN hidden units: 16
QL Model 1	States: Price; Exploration: Random; γ: 0.9; η: 0.9
QL Model 2	States: Price; Exploration: ε-Greedy; γ: 0.9; η: 0.9; ε: 0.6
QL Model 3	States: Indicators; Exploration: ε-Greedy; γ: 0.9; η: 0.9; ε: 0.6;
DQN	States: Indicators; Exploration: ε-Greedy; Hidden units: 128;
DDQN	States: Indicators; Exploration: ε-Greedy; Hidden units: 128; Batch size: 100;

Table 3 Specifications of models designed for case tests.

11.2. Test Results

6 cases have been designed to capture different situations (See Section 7.3). The following gives the profits made by each model at different cases.

Case	SRNN	MRNN	MRNN +CNN	QL M1	QL M2	QL M3	DQN	DDQN	Max
C1	**258.1**	129.0	-40.5	14.0	-87.0	81.9	-67.2	9.7	1465.3
C2	**2047.8**	1507.6	100.8	296.7	-149.9	-72.2	282.1	175.0	2580.6
C3	**3018.1**	2601.7	0.00	324.5	98.4	438.5	348.9	53.9	4712.6
C4	**1000.3**	**1000.3**	261.4	324.5	125.5	313.4	224.8	0.00	4075.5
C5	0.00	28.0	460.7	-437.7	463.7	314.8	328.4	**781.9**	2951.4
C6	-720.4	**1571.0**	-775.7	-484.6	-702.7	106.1	-20.9	-477.3	2650.0

Table 4 Profits of models in 6 test cases

(Bold: Highest profit per case; Green: Positive profit; Red: Negative Profit).

Case 1 and Case 2 are designed by choosing test data that is within the range of train data. Also, the train data and test data does not appear with some specific trend and includes as much features as possible (See Section 7.3). Therefore, it is expected that the features in training data is imposing the minimum effects to the learning results. For both Cases 1 and 2, SRNN achieved the best profits. However, for case 1, all of the models could only product at most 17.6% of the max possible profit. In case 2, SRNN and MRNN achieved profits which

are 79.3% and 58.2% of max profit, while the profits from other models are less than about 11.5% of the max profit.

For Case 3, the train data is an uptrend which stops at the global max and the test data is a downtrend immediately afterwards. Meanwhile, for Case 4, the train data is an downtrend which stops at the global min and the test data is a uptrend immediately afterwards. These two cases are designed to be challenges to the models since the test data do not follow the main trend in train data. However, it is unanticipated that there no loss made among all models. SRNN is the best for Case 3 and gives a profit of 64.0% of the max profit. Meanwhile, SRNN and MRNN give the best profit in Case 4. However, generally all models give less profit in Case 4 than Case 3. It may be more difficult to predict an uptrend from a downtrend.

Case 5 and 6 are designed to be cases opposite to Case 3 and Case 4. In Case 5, there is an uptrend all the way from the train data and test data. Case 6 describes a downtrend from train data to test data instead. These two cases, on the contrary to Case 3 and Case 4, are designed to be the easy cases since the test data follows the trend of train data. Nonetheless, most of the models are not performing well unexpectedly.

For Case 5, DDQN gives the best profit which is about 26% of the max profit. However, at the same time, compared with Case 3 which the train data is also an uptrend, most of the models give less profits, where the performance of SRNN declined the most. Situation for Case 6 is similar when compared to Case 4 that almost all models give worse performance, except MRNN which give the best profit of 59.2% of the max profit at the same time.

11.3. Discussion

The result above give an unanticipated result that the main trend of train data is not the most influential factor to the performance of models. Even though test data are not following the main trend in Case 3 and 4, the models are providing good performances generally. On the contrary, models are not performing as expected in Case 5 and 6 even though the test data follows the main trends. Considering all the cases together, one can conclude that the models fail easily if the test data is out of the range of the train data.

Besides, SRNN provides the best performance for Case 1 to Case 4, which are the cases that have the test data within the range of train data. Therefore, SRNN requires the future prices to be inside the range of historical data for good performances. Otherwise, the performance of SRNN significantly drops. In this situation, MRNN may be a good substation since it is giving positive profits in both Cases 5 and 6. However, DDQN is also one of the potential candidates which is the best for Case 5. Moreover, since QL Model 3 and DQN model are generally giving profits that are about 10% of the maximum, and little losses sometimes, more study should be made to exploit their potential.

12. Conclusion

In this project, 3 types of models have been built for the purpose of algorithmic trading – namely RNN, RL and DRL models. In Section 10, experiments have been conducted with the models to find out the factors that affect their performances, hence finding out ways to improve their performances. In Section 11, the models were tested against test cases which simulate different situations in real market. It is discovered that SRNN excels in most of the test cases. However, when future price levels go beyond the range of historical data, SRNN fails to produce accurate predictions. MRNN and DDQN is then the candidates of substitutions in such situations. Nonetheless, more researches should be done in order to ensure the models to be applicable in real trading market. For example, making enhancement of RNN models by adding neural networks other than a simple CNN output layer, or the study of state definitions for RL models. Although unfortunately this project fails to create a promising model, it does give information about the factors that affect the performances and give insights to improve each of the models being studied.

References

[1] T. Greenwald (2017, Mar 10). How AI is Transforming the Workplace. The Wall Street Journal [Online]. Available: https://www.wsj.com/articles/how-ai-is-transforming-the-workplace-1489371060

[2] Wanjawa, B. W., & Muchemi, L. (2014). ANN Model to Predict Stock Prices at Stock Exchange Markets. arXiv preprint arXiv:1502.06434.

[3] J. Poulos. Predicting Stock Market Movement with Deep RNNs. Available: https://bcourses.berkeley.edu/files/70257274/download.

[4] J Liang, Z Jiang, D Xu (2017, Jun 30). A Deep Reinforcement Learning Framework for Financial Portfolio Management Problem. Available: https://arxiv.org/pdf/1706.10059.pdf

[5] Hedayati Moghaddam, A., Hedayati Moghaddam, M., & Esfandyari, M. Stock market index prediction using artificial neural network.

[6] Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015, July). Deep Learning for Event-Driven Stock Prediction. In Ijcai (pp. 2327-2333).

[7] Moody, J. E., & Saffell, M. (1999). Reinforcement learning for trading. In Advances in Neural Information Processing Systems (pp. 917-923).

[8] Deng, Y., Bao, F., Kong, Y., Ren, Z., & Dai, Q. (2017). Deep direct reinforcement learning for financial signal representation and trading. IEEE transactions on neural networks and learning systems, 28(3), 653-664.

[9] Jin, O., & El-Saawy, H. Portfolio Management using Reinforcement Learning.

[10] Grigoryan, H. (2015). Stock Market Prediction using Artificial Neural Networks. Case Study of TAL1T, Nasdaq OMX Baltic Stock. Database Systems Journal BOARD, 14.

[11] Duerson, S., Khan, F. S., Kovalev, V., & Malik, A. H. (2005). Reinforcement learning in online stock trading systems.

[12] Kaur, S. Algorithmic Trading using Reinforcement Learning augmented with Hidden Markov Model.

[13] Van Hasselt, H., Guez, A., & Silver, D. (2016, February). Deep Reinforcement Learning with Double Q-Learning. In AAAI (Vol. 16, pp. 2094-2100).

[14] DeepMind. Human-level control through Deep Reinforcement Learning. Available from https://deepmind.com/research/dqn/

[15] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529.

[16] Lu, D. W. (2017). Agent Inspired Trading Using Recurrent Reinforcement Learning and LSTM Neural Networks.
[17] Computer Science Department, Stanford University. Multi-Layer Neural Network. UFLDL Tutorial. Available from http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/
[18] Singhal, D., & Swarup, K. S. (2011). Electricity price forecasting using artificial neural networks. International Journal of Electrical Power & Energy Systems, 33(3), 550-555.
[19] Terna, P., D'Acunto, G., & Caselle, M. A Deep Learning Model to Forecast Financial Time-Series.
[20] Necchi, P. G. Reinforcement Learning For Automated Trading.
[21] Lee, J. W., Park, J., Jangmin, O., Lee, J., & Hong, E. (2007). A Multiagent Approach to $ Q $-Learning for Daily Stock Trading. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 37(6), 864-877.
[22] Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.".
[23] Juliani A (2016). Simple Reinforcement Learning with Tensorflow. Medium. Available from https://medium.com/@awjuliani/simple-reinforcement-learning-with-tensorflow-part-4-deep-q-networks-and-beyond-8438a3e2b8df