

FINANCIAL TRADING USING LEARNING-BASED APPROACH

BY

TAN LI XUE

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2022

REPORT STATUS DECLARATION FORM

Title: Financial Trading Using Learning-based Approach

Academic Session: 202201

I TAN LI XUE

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.



(Author's signature)

Verified by,



(Supervisor's signature)

Address:

No 9, Jalan S/L 7/4, Bandar Sungai Long,

43000 Kajang,

Selangor

Ts Dr Lim Seng Poh

Supervisor's name

Date: 22/04/2022

Date: 22/04/2022

Universiti Tunku Abdul Rahman			
Form Title : Sample of Submission Sheet for FYP/Dissertation/Thesis			
Form Number: FM-IAD-004	Rev No.: 0	Effective Date: 21 JUNE 2011	Page No.: 1 of 1

FACULTY OF INFORMATON AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 22/04/2022

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that Tan Li Xue (ID No: 18ACB01119) has completed this final year project entitled “*Financial Trading Using Learning-based Approach*” under the supervision of Ts Dr Lim Seng Poh (Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology .

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,



Tan Li Xue

DECLARATION OF ORIGINALITY

I declare that this report entitled “**FINANCIAL TRADING USING LEARNING-BASED APPROACH**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.



Signature : _____

Name : _____ TAN LI XUE _____

Date : _____ 22/04/2022 _____

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Ts Dr Lim Seng Poh who has given me the opportunity to work on this research. I also appreciate all the guidance and suggestions that have been given to improve my research.

I would like to say thanks to my family and my friends that have always been supportive for me.

ABSTRACT

Financial trading has been widely studied and many algorithms and approaches have been applied to gain higher profit. In this work, deep reinforcement learning algorithms were applied to automate the trading process. The data used in this work were 1-minute, 5-minute, and 30-minute candlesticks from different asset classes including Foreign Exchange markets (FOREX), equity indexes, and commodities. The proposed framework utilised data from different time intervals to make a trading decision. For each time interval, an autoencoder consisting of InceptionTime and Long Short-Term Memory (LSTM) was trained to perform feature extraction. The reinforcement learning algorithms applied include Advantage Actor-Critic (A2C), Proximal Policy Optimisation (PPO), and Twin Delayed Deep Deterministic Policy Gradient (TD3). Both discrete and continuous action spaces were studied. The performance of the models was evaluated by using expected return and risk-adjusted return such as the Sharpe ratio. Furthermore, the models were trained under different transaction cost settings to identify the effect of transaction cost on the performance of the models. The results showed that the most consistent model is PPO and SAC performs the worst in this setting. Furthermore, the results also showed that the best transaction cost setting should be equal to or higher than the actual transaction cost.

TABLE OF CONTENTS

TITLE PAGE	i
REPORT STATUS DECLARATION FORM	ii
FYP THESIS SUBMISSION FORM	iii
DECLARATION OF ORIGINALITY	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	xi
LIST OF TABLES	xii
LIST OF SYMBOLS	xiii
LIST OF ABBREVIATIONS	xv
CHAPTER 1 INTRODUCTION	1
1.1 Overview	1
1.2 Problem Statement and Motivation	3
1.3 Research Objectives	4
1.4 Research Scope	4
1.5 Impact, Significance and Contribution	5
1.6 Report Organisation	5

CHAPTER 2 LITERATURE REVIEW	7
2.1 Financial Trading	7
2.1.1 Bid Price and Ask Price	7
2.1.2 Candlestick Chart	8
2.1.3 Technical Indicators	9
2.1.4 Evaluation Metrics	12
2.2 Reinforcement Learning	14
2.2.1 Markov Decision Process (MDP)	14
2.2.2 Markov Property	14
2.2.3 Policy	14
2.2.4 Return	15
2.2.5 State-Value Function and Action-Value Function	15
2.2.6 Types of Reinforcement Learning Algorithms	17
2.2.7 Advantage Actor-Critic (A2C)	18
2.2.8 Proximal Policy Optimisation (PPO)	19
2.2.9 Twin Delayed Deep Deterministic Policy Gradient (TD3)	19
2.2.10 Soft Actor Critic (SAC)	20
2.3 Related Works	20
 CHAPTER 3 RESEARCH METHODOLOGY	 23
3.1 Methodologies and General Work Procedures	23
3.1.1 Literature Review and Problem Definition	24
Bachelor of Computer Science (Honours)	
Faculty of Information and Communication Technology (Kampar Campus), UTAR	viii

3.1.2	Data Collection	24
3.1.3	Defining reinforcement learning framework	24
3.1.4	Designing Neural Network	25
3.1.5	Analysis and Discussion	25
CHAPTER 4 RESEARCH DESIGN		26
4.1	Data Acquisition and Preprocessing	26
4.2	Markov Decision Process Formalisation	26
4.3	Model Framework	33
4.4	Output Layers and Types of Distribution	37
4.5	Algorithms	37
CHAPTER 5 EXPERIMENT SETUP		43
5.1	Hardware Setup	43
5.2	Software Setup	43
5.3	Setting and Configuration	44
5.3.1	Trading Environment Setting	44
5.3.2	Trading Agent Setting	44

CHAPTER 6 ANALYSIS AND DISCUSSION	47
6.1 Experimental Result	47
6.2 Research Challenges	57
CHAPTER 7 CONCLUSION	58
REFERENCES	59
APPENDIX A	A-1
A.1 Biweekly Report	A-1
A.2 Poster	A-7

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1	Scenario that illustrate bid-ask spread	7
Figure 2.2	Example of a candlestick [1]	8
Figure 2.3	Comparison between traditional candlestick chart vs Heikin-Ashi candlestick chart [22]	9
Figure 2.4	Diagram of Bellman equation for $V^\pi(s)$ [34]	16
Figure 2.5	Diagram of Bellman equation for $Q^\pi(s, a)$ [34]	17
Figure 3.1	Research framework	23
Figure 4.1	Model framework for actor network and critic network	34
Figure 4.2	Design of InceptionTime-LSTM autoencoder	35
Figure 4.3	LSTM repeat vector [46]	35
Figure 4.4	Example of standard convolution block with two layers with kernel size 3 [47]	36
Figure 4.5	Example of standard convolution block with two layers with kernel size 3 [47]	36
Figure 6.1	Account value over time for all methods for EUR/USD	50
Figure 6.2	Account value over time for all methods for USA500.IDX/USD	50
Figure 6.3	Account value over time for all methods for GAS.CMD/USD	50
Figure 6.4	Comparison of different transaction cost settings in reward function for EUR/USD	54
Figure 6.5	Comparison of different transaction cost settings in reward function for USA500.IDX/USD	55
Figure 6.6	Comparison of different transaction cost settings in reward function for GAS.CMD/USD	56

LIST OF TABLES

Table Number	Title	Page
Table 4.1	Action space for A2C, PPO and SAC	28
Table 4.2	Example of common approach	32
Table 4.3	Example of proposed approach	33
Table 4.4	Output layers and types of distribution for each algorithm and action space	37
Table 5.1	Hardware used in this research	43
Table 5.2	Software used in this research	43
Table 5.3	Settings for each dataset	44
Table 5.4	Hyperparameters settings	44
Table 6.1	Models performance measured by various performance metrics	51

LIST OF SYMBOLS

Symbol	Meaning
O_t	Open price at time t
H_t	Close price at time t
L_t	Low price at time t
C_t	Close price at time t
n	Number of periods
T_t	True range at time t
P_t	Price at time t
r_t	Return at time t
R_p	Return of portfolio
R_f	Risk free rate of return
σ_p	Standard deviation of portfolio returns
σ_d	Downside deviation of portfolio returns
S	State space
A	Action space
R	Reward function
T	State transition function
π	Policy
$\pi(a s)$	Probability of taking action a in state s
s	State
s'	Next state
s_t	State at time t
a_t	Action at time t
R_t, r_t	Reward at time t
G_t	Return at time t
γ	Discount rate
$V^\pi(s)$	Value of state s under policy π (Expected return)

$Q^\pi(s, a)$	Value of taking action a in state s under policy π
θ	Parameter of policy
$J(\theta)$	Policy loss
ψ	Parameter of V-function
$J(\psi)$	V-function loss
ω	Parameter of Q-function
$J(\omega)$	Policy loss
$A^\pi(s_t, a_t)$	Advantage of action a_t in state s_t
λ	Parameters of Generalised Advantage Estimation (GAE)
H_t	Entropy of policy
α	Coefficient of entropy
$r_t(\theta)$	Importance sampling weight
ε	Clipping parameter of PPO algorithm
$\pi^{old}(A_t S_t)$	Probability of taking action a in state s under old policy
$\pi^{new}(A_t S_t)$	Probability of taking action a in state s under new or updated policy
τ	Update rate of target network in TD3 algorithm
σ	Standard deviation of Gaussian noise in TD3 algorithm
c	Clip range for gaussian noise
v_t	Account value at time t
p_t	Position at time t
β_1, β_2	Beta values in Adam optimiser

LIST OF ABBREVIATIONS

Abbreviation	Meaning	Page
<i>A2C</i>	Advantage Actor-Critic	4
<i>A3C</i>	Asynchronous advantage Actor-Critic	20
<i>ARIMA</i>	Autoregressive Integrated Moving Average	3
<i>ARMA</i>	Autoregressive Moving Average	3
<i>ATR</i>	Average True Range	11
<i>CCI</i>	Commodity Channel Index	11
<i>CNN</i>	Convolutional Neural Network	4
<i>DDPG</i>	Deep Deterministic Policy Gradient	18
<i>DQN</i>	Deep Q-Network	21
<i>DRQN</i>	Deep Recurrent Q-Network	21
<i>EMA</i>	Exponential Moving Average	1
<i>FOREX</i>	Foreign Exchange Markets	25
<i>FDRNN</i>	Fuzzy Deep Recurrent Neural Network	26
<i>GAE</i>	Generalised Advantage Estimation	19
<i>GARCH</i>	Generalised Autoregressive Conditional Heteroskedasticity	3
<i>LSTM</i>	Long Short-Term Memory	2
<i>MA</i>	Moving Average	11
<i>MACD</i>	Moving Average Convergence Divergence	10
<i>MDP</i>	Markov Decision Process	4
<i>PPO</i>	Proximal Policy Optimisation	4

<i>ROC</i>	Rate of Change	12
<i>RRL</i>	Recurrent Reinforcement Learning	22
<i>RSI</i>	Relative Strength Index	1
<i>SAC</i>	Soft Actor Critic	4
<i>SMA</i>	Simple Moving Average	1
<i>SVM</i>	Support Vector Machine	2
<i>TD3</i>	Twin Delayed Deep Deterministic Policy Gradient	4

CHAPTER 1

Introduction

1.1 Overview

Financial trading approaches can be categorised into three types, fundamental analysis, technical analysis and algorithmic trading. Fundamental analysis [1] is a method of predicting the prices based on economic data. An example of the fundamental analysis method provided was the analogous season method. The analyst first determines past seasons that exhibit similar fundamental characteristics. After that, the price movement of the current season can be predicted based on the price movement of those past seasons with similar characteristics.

Technical analysis, on the other hand, predicts future prices mainly based on historical price data instead of economic data. Technical analysis assumes that the patterns or trends of the price movements are usually repetitive and thus the future prices can be predicted by identifying similar patterns. This can be done by using suitable indicators such as Simple Moving Average (SMA), Exponential Moving Average (EMA) and Relative Strength Index (RSI) to analyse the price pattern and changes. One technical analysis strategy [1] is by using overbought or oversold indicators such as RSI to determine whether the prices are higher or lower than they should be. For example, the value of RSI lies within the range of 0 to 100. A common interpretation by analysts is that value higher than 70 is considered as an overbought condition whereas a value lower than 30 is considered as an oversold condition. Hence, the traders can use this as a trading signal to sell when overbought occurs and vice versa.

Although fundamental analysis and technical analysis are very different from one another, many traders apply both to perform trading, such that fundamental analysis was used to predict the price movements and technical analysis was used to identify the suitable time for entering or exiting the trade [1].

Technology advancement allows financial trading to be carried out electronically. This has attracted more traders to adopt algorithmic trading as it allows the traders to access the markets and trade in a much faster way. Algorithmic trading can be defined as any trading that some part of or the entire trade cycle has been automated by using computers and algorithms [2].

One type of model that has been widely applied in financial trading is the mathematical model. The performance of the ARMA model was studied [3] in predicting monthly and yearly stock returns of the London Stock Exchange and S&P 500. The results showed that the ARMA model can predict medium- or long-term stock returns accurately.

Apart from mathematical models, financial traders also adopted different machine learning algorithms to predict price movements. Support Vector Machine (SVM) was applied in the work [4] in performing financial forecasting on daily prices. The study concluded that SVM is effective in predicting market prices.

The performance between various machine learning and deep learning algorithms were compared [5] which include random forest, deep neural network (DNN), logistic regression and Long Short-Term Memory (LSTM). The study analysed the performance of the models based on a portfolio that consisted of 2000 stocks. It showed that LSTM obtained the highest mean return and Sharpe ratio while logistic regression has achieved the lowest. This might be because the logistic regression model is linear and thus underfitting has occurred.

Apart from making prediction based on historical prices, character-based neural language model [6] was used on financial news to predict the stock prices. A language model was used to embed the character into a vector and for LSTM layer to perform the prediction. The study found that the proposed model can be applied in both interday and intraday predictions.

1.2 Problem Statement and Motivation

Financial trading has been a hot topic for decades. Different kinds of approaches for financial trading have been proposed such as fundamental analysis, technical analysis and algorithmic trading. Algorithmic trading has become popular due to several advantages as compared to fundamental and technical analysis. This includes helping the traders to make the trading decision and submit orders much faster and risk diversification [7]. To perform algorithmic trading, financial traders apply mathematical models and learning-based models to predict future prices. Examples of mathematical models are Autoregressive Moving Average (ARMA) [8], Autoregressive Integrated Moving Average (ARIMA) [9], and Generalised Autoregressive Conditional Heteroskedasticity (GARCH) [10]. However, the financial data have a low signal-to-noise ratio, causing mathematical models to not be effective in predicting future prices. [7].

Machine learning has been used in many different areas. Some applications of machine learning include natural language processing [11], time series prediction and image classification [12]. These tasks utilise a set of labelled data for the model to learn. In recent years, reinforcement learning has gained a lot of popularity and attention. Unlike supervised learning methods which require labelled data, a reinforcement learning agent learns by interacting with an environment and receiving the feedback through specified reward functions. Deep reinforcement learning has demonstrated top-level human performance in playing video games. Some examples are AlphaZero [13] and AlphaStar [14] developed by DeepMind. Apart from that, reinforcement learning can also be applied in robotic control optimisation according to [15] or even autonomous driving according to [16].

Machine learning algorithms that have been employed in financial trading include random forest, Support Vector Machine (SVM), and Long Short-Term Memory (LSTM). However, the predictions produced by machine learning models cannot be used as a trading signal directly. This is because they fail to consider the transaction

cost that might incur to the traders [17]. Besides that, risk management is also an important aspect of financial trading and thus requires complex trading strategies [18]. Hence, it is important to have a model that can learn trading strategies and make trading decisions directly, instead of merely predicting the price series. Therefore, reinforcement learning algorithms were adopted as the learning-based approach in this work as it is a more suitable choice than supervised learning algorithms for sequential decision-making tasks.

1.3 Research Objectives

The objectives of this research are:

- To propose a deep reinforcement learning model to perform financial trading.
- To identify the best transaction cost settings for training the deep reinforcement learning model.

1.4 Research Scope

In this research, the financial trading problem has been formulated by using Markov Decision Process (MDP). This involved designing the state space, action space and the reward function. InceptionTime, which is a type of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) have been applied as an autoencoder model to perform the feature extraction on the time series data. The reinforcement learning algorithms that were applied include Advantage Actor-Critic (A2C) [19], Proximal Policy Optimisation (PPO) [19], Soft Actor Critic (SAC) [20] and Twin Delayed Deep Deterministic Policy Gradient (TD3) [21].

The data that is used for the training are 1-minute bid candlestick data obtained from Dukascopy. The data consists of different asset classes which are Foreign Exchange markets (FOREX), equity indexes, and commodities. For the CNN-LSTM autoencoder, the training period starts from 1st Jan 2017 until 31st Dec 2019. Lastly, the models are tested with data between 1st Jan 2020 and 31st Dec 2020. For the reinforcement learning model, the training period starts from 1st Jan 2017 until 31st Dec 2018 and the testing period starts from 1st Jan 2020 until 31st Dec 2020. The

proposed method is benchmarked against Buy and Hold strategy, Short and Hold strategy and Fuzzy Deep Recurrent Neural Network (FDRNN) [17]. The evaluation metrics included expected return, Sharpe ratio, Sortino ratio, Maximum Drawdown.

1.5 Impact, Significance and Contribution

The main contribution of this research was proposing a reinforcement learning approach for financial trading. This can assist the traders especially those who are inexperienced to perform financial trading to gain profit in the financial market. Moreover, experienced traders can also utilise this approach to assist them to automate or improve their trading.

Besides that, an InceptionTime-LSTM autoencoder has been improved to be more suitable in performing feature extractions in financial time series data. as was done by applying the concept of denoising autoencoder to ensure the autoencoder was robust to noisy financial market data. Standard convolution operation in CNN is replaced with causal convolution to be more suitable for extracting time-series features.

Furthermore, a reinforcement learning approach for financial trading using multiple time intervals is proposed. This allowed the models to make trading decisions based on the trading signals based on the financial market data of different time intervals to increase the profit.

Lastly, this research also studied how the transaction cost in the reward function can affect the models. The expected outcomes of the research proved that the models are affected by the transaction cost in the reward function.

1.6 Report Organisation

This report was organised as follow. Chapter 2 discussed concepts about financial trading and reinforcement learning. Chapter 2 also included related works that are reviewed. Chapter 3 discussed the general work procedures of this research, the hardware and software that are used, system design, together with implementation

issues and challenges. Chapter 4 discussed the preliminary works that have been done and the results. Chapter 5 discussed about the conclusion of this research.

CHAPTER 2

Literature Review

2.1 Financial Trading

This section discussed some of the concepts and formulas used in financial trading.

2.1.1 Bid Price and Ask Price

A bid price of an asset is the price that the seller will receive whereas an ask price is the price that the buyer has to pay for buying the asset. Bid-ask spread is the difference between the bid price and the ask price at a point in time. It is also the transaction cost for the traders. Figure 2.1 shows a scenario that illustrate bid ask spread. If a trader bought the asset at 9.00, he or she will have to pay 1.2875 dollars. If the trader then sold that asset immediately, he or she will sell it at 1.2872, losing 0.0001 dollars.



Figure 2.1 Scenario that illustrate bid-ask spread

2.1.2 Candlestick Chart

A candlestick chart is a type of chart for visualising the price movements of an asset. Each candlestick shows 4 price points which are open, high, low and close. Open price and close price show the price of an asset at the beginning and ending of a period respectively. High price and low price, on the other hand, represents the highest and lowest point that the price series has reached throughout the period. Candlesticks that increase and decrease will be represented by using different colours. Figure 2.2 as referred by [1] shows an example of a candlestick. In this case, the increasing candlestick is represented by white while the decreasing candlestick is represented by black.

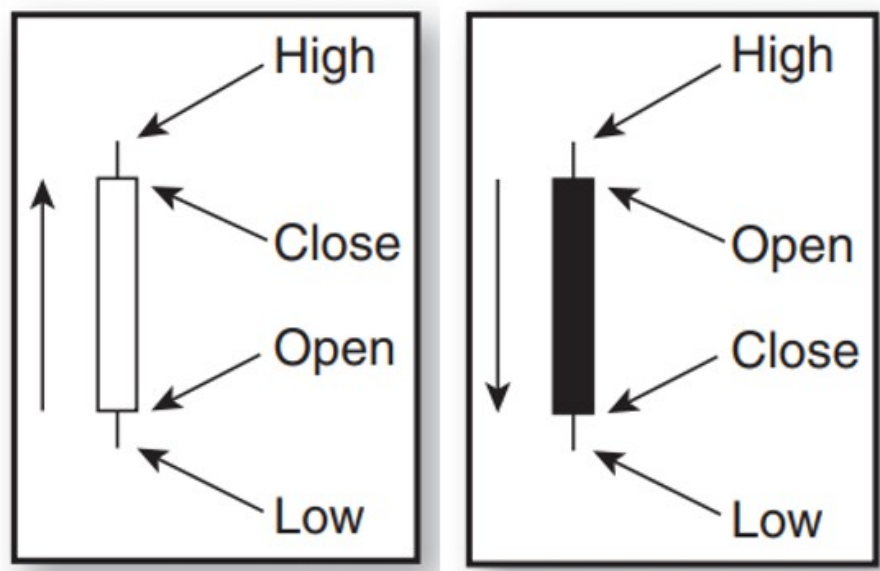


Figure 2.2 Example of a candlestick [1]

Heikin-Ashi candlestick is a type of chart that visualise the price movements by showing its open, high, low and close prices. It is a chart that is derived by using the prices from the standard candlestick chart. Equation 2.1, Equation 2.2, Equation 2.3 and Equation 2.4 are the equations for calculating close price, open price, high price and low price in Heikin-Ashi candlesticks respectively.

$$Close_t = \frac{1}{4}(Open_t + High_t + Low_t + Close_t) \quad (2.1)$$

$$Open_t = \frac{1}{2}(Open_{t-1} + Close_{t-1}) \quad (2.2)$$

$$High_t = Max(High_t, Open_t, Close_t) \quad (2.3)$$

$$Low_t = Min(Low_t, Open_t, Close_t) \quad (2.4)$$

Figure 2.3 [22] shows the Comparison between traditional candlestick chart vs Heikin-Ashi candlestick chart. It shows that Heikin-Ashi candlestick chart can help to reduce the noise in the price movements compared to normal candlestick chart.



Figure 2.3 Comparison between traditional candlestick chart vs Heikin-Ashi candlestick chart [22]

2.1.3 Technical Indicators

Technical indicators are signals that are produced by price patterns or volume patterns which were usually used in technical analysis [23]. Technical indicators can be categorised into two types which are overlays and oscillators. Overlays have the same scaling as the prices such that they can be plotted on top of the price chart to make a comparison or identify signals. Examples of overlays are Simple Moving Average (SMA) and Exponential Moving Average (EMA). On the other hand, oscillators have a different scaling and are plotted in a separate chart. Examples of oscillators are

CHAPTER 2

Relative Strength Index (RSI) and Rate of Change (ROC). The technical indicators that will be used in this work have been listed out in this section. The symbols O_t , H_t , L_t , C_t were defined as open, high, low and close price at time t , and n represented the number of periods.

Simple Moving Average (SMA) is one of the simplest technical indicators. It computes the average price of the last n time steps by using arithmetic mean. Equation 2.5 is the formula for calculating SMA [24].

$$SMA_t = \frac{(C_{t-(n-1)} + C_{t-(n-2)} + \dots + C_t)}{n} \quad (2.5)$$

Exponential Moving Average (EMA) is a type of moving average indicator. However, unlike Simple Moving Average, the weight assigned for previous time steps will decrease exponentially. Equation 2.6 [24] shows the formula for calculating EMA.

$$EMA_t = (C_t - EMA_{t-1})K + EMA_{t-1} \quad (2.6)$$

, where

$$K = \frac{2}{n+1} \quad (2.7)$$

Moving Average Convergence Divergence (MACD) is an indicator that is used to determine the relationship between 2 moving average indicators. One common choice for the moving average indicators is EMA of 12-periods and EMA of 26-periods. MACD can be calculated by taking the difference between 12-periods EMA and 26-periods EMA [24].

Bollinger bands consists of three different values. One is a n -periods moving average that has been chosen, one upper band represents price that are k standard deviation

larger than the average and a lower band that are k standard deviation smaller than the average. Equation 2.8 [25] shows the formula for calculating upper Bollinger band while Equation 2.9 shows the formula for calculating lower Bollinger band.

$$BOLU_t = MA_t(M, n) + k \sigma_t(M, n) \quad (2.8)$$

$$BOLD_t = MA_t(M, n) - k \sigma_t(M, n) \quad (2.9)$$

$$M_t = \frac{1}{3} (H_t + L_t + C_t) \quad (2.10)$$

where $BOLU_t$ is upper Bollinger band, $BOLD_t$ is lower Bollinger band and $MA_t(M, n)$ is n -periods moving average of M , and $\sigma_t(M, n)$ is n -periods standard deviation of M .

Commodity Channel Index (CCI) is a technical indicator that measures the deviation of the prices from its mean price of past n -periods. Equation 2.11 [24] shows the formula for calculating CCI

$$CCI_t = \frac{M_t - A}{0.015 D_t} \quad (2.11)$$

, where

$$M_t = \frac{1}{3} (H_t + L_t + C_t) \quad (2.12)$$

$$A = \frac{(M_{t-(n-1)} + M_{t-(n-2)} + \dots + M_t)}{n} \quad (2.13)$$

$$D_t = \sum_{i=t-(n-1)}^t \frac{|M_t - A_i|}{n} \quad (2.14)$$

Average True Range (ATR) is an indicator for measuring the volatility of the asset price. Equation 2.15 as referred from [26] is the formula for calculating true range, T while Equation 2.16 is the formula for calculating Average True Range.

$$T_t = \max(H_t - L_t, |H_t - C_{t-1}|, |C_{t-1} - L_t|) \quad (2.15)$$

$$ATR_t = \frac{(T_{t-(n-1)} + T_{t-(n-2)} + \dots + T_t)}{n} \quad (2.16)$$

Rate of Change (ROC) indicator represents the percentage change of the price as compared to the price of n time steps ago. Equation 2.17 as referred from [27] shows the formula for calculating ROC indicator.

$$ROC_t = 100 \left(\frac{C_t}{C_{t-n}} - 1 \right) \quad (2.17)$$

2.1.4 Evaluation Metrics

This section elaborates on some of the evaluation metrics that have been used in this work.

Simple return and log return are two of the most commonly used formula for calculating return. Equation 2.18 is the formula for calculating simple return whereas Equation 2.19 is the formula for calculating log return [28].

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}} \quad (2.18)$$

$$\ln(1 + r_t) \quad (2.19)$$

, where p_t = price of the asset at time t .

Sharpe ratio is a type of risk-adjusted return to compare the return and the risk for an investment. Equation 2.20 shows the formula for calculating Sharpe ratio [28].

$$Sharpe\ Ratio = \frac{E[R_p] - R_f}{\sigma_p} \quad (2.20)$$

CHAPTER 2

, where R_f represents risk free rate of return, R_p represents return of portfolio while σ_p represents the standard deviation of the returns of portfolio.

Sortino ratio is also a risk-adjusted return similar to the Sharpe ratio. However, it only considers downside deviation, which is the standard deviation of negative returns. Hence, a large standard deviation of positive return will not be penalised when using the Sortino ratio. Equation 2.21 is the formula for calculating the Sortino ratio [29].

$$\text{Sortino Ratio} = \frac{E[R_p] - R_f}{\sigma_d} \quad (2.21)$$

where σ_d represents downside deviation of the returns.

Maximum drawdown (MDD) shows the maximum loss throughout the trading period. In other words, it is the loss that has occurred to the portfolio from the highest point to the lowest point before another highest point is achieved. Equation 2.22 [30] is the formula for calculating MDD.

$$\text{MDD} = \frac{\text{ThroughValue} - \text{PeakValue}}{\text{PeakValue}} \quad (2.22)$$

2.2 Reinforcement Learning

The concept of reinforcement learning was discussed in this section.

2.2.1 Markov Decision Process (MDP)

A reinforcement learning problem is usually formulated as an MDP, which is a tuple of (S, A, T, R) . The S in the tuple represents the state space, which is the set of all possible state. The A in the tuple represents the action space, which is the set of all possible actions that can be taken. The T in the tuple is a function represents the probability of arriving in state s' after taking action a in state s , i.e. $T : S \times A \times S \rightarrow [0,1]$. Lastly, R is the reward function of the MDP. There are two possible definitions for R in the tuple which are interchangeable. The first definition is $R: S \times A \rightarrow R$ while the second one is $R : S \times A \times S \rightarrow R$. The first one means that the reward was given based on the action that has been taken in a state, while the second definition gives reward based on the action that has been taken and also the transition from one state to another [31].

2.2.2 Markov Property

Markov property indicates that the future only depends on the present but not the past. It means that the current state has “captured” all information from the past that will affect the future [32].

2.2.3 Policy

For every MDP problem, the solutions of them are called a policy π . It represents the probability of selecting each action in a particular state. For example, $\pi(a|s)$ represents the probability of taking action a in state s [32].

2.2.4 Return

In every reinforcement learning task, the general objective for the agent is to maximise its cumulative rewards. Let $R_{t+1}, R_{t+2}, R_{t+3}, \dots, R_T$ be a reward sequence received by the agent after time step t up until final time step T , the return, G_t can be defined as in Equation 2.23 referred from [32].

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots + R_T \quad (2.23)$$

Although this return is easy to be calculated, it is only suitable for task that can be easily and naturally broken into independent episodes that will end when the agent arrives in a terminal state. For continuing task that can continue for infinite number of steps, its return can diverge easily to be infinite too. Hence, discounted return was used to prevent return of continuing task to diverge. Equation 2.24 as referred from [32]. shows the formula of discounted return

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (2.24)$$

where $\gamma \in [0, 1]$ is the discount rate of the return. When $\gamma = 1$, the discounted return reduced to Equation 2.23. When $\gamma = 0$, the agent only maximises its immediate reward.

2.2.5 State-Value Function and Action-Value Function

A state-value function $V_{\pi}(s)$ for a given state s and a policy π , is the expected return of beginning in state s , and following π afterwards. Equation 2.25 as referred from [32] is the formula of state-value function.

$$V^{\pi}(s) = E_{\pi}[G_t | S_t = s] = E_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right], \text{ for all } s \in S \quad (2.25)$$

CHAPTER 2

An action-value function on the other hand, is denoted by $Q_\pi(s, a)$. It is defined to be the expected return by first taking action a in state s at time t , then continue taking action according to the policy π afterwards. Equation 2.26 as referred from [32] is the formula of action-value function.

$$Q^\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right] \quad (2.26)$$

Bellman equations relate the value of successor states to the value of current state. Equation 2.27 and Equation 2.28 as referred from [33] shows the Bellman equation for v_π while Equation 2.28 shows the Bellman equation for q_π for a given policy π .

$$V^\pi(s) = E_{a \sim \pi, s' \sim p}[r(s, a) + \gamma V^\pi(s')] \quad (2.27)$$

$$Q^\pi(s, a) = E_{s' \sim p}[r(s, a) + \gamma E_{a' \sim \pi}[Q^\pi(s', a')]] \quad (2.28)$$

Figure 2.4 and 2.5 as referred from [34] shows diagram of Bellman equation for $V^\pi(s)$ and diagram of Bellman equation for $Q^\pi(s, a)$. The Bellman equations can be interpreted as such: the value of state s is the immediate reward that was received plus the value of the next state s' .

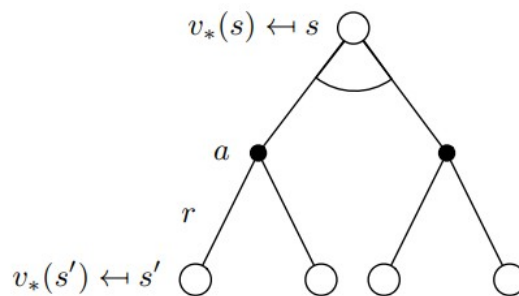
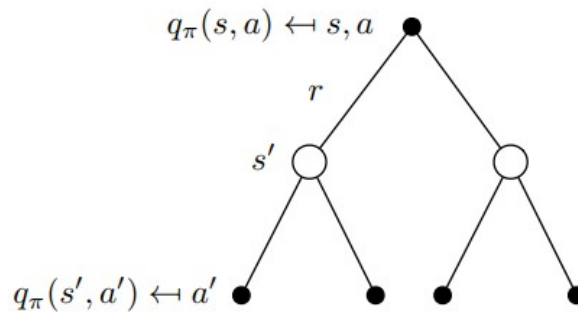


Figure 2.4 Diagram of Bellman equation for $V^\pi(s)$ [34]Figure 2.5 Diagram of Bellman equation for $Q^\pi(s, a)$ [34]

2.2.6 Types of Reinforcement Learning Algorithms

Reinforcement learning algorithms can be grouped into three types. This includes critic-only algorithm, actor-only algorithm, and actor-critic algorithm. Critic-only algorithms work by learning $Q^\pi(s, a)$. The policy can then be constructed based on the learned value function. For example, one of the most used policies is ϵ -greedy policy. It has a probability of $(1 - \epsilon)$ to select the action a with highest $Q^\pi(s, a)$ in a given state s , and a probability of ϵ to select all actions with equal probability [19]. Some algorithms that belong to this family include State-Action-Reward-State-Action (SARSA), Deep Q-Networks (DQN), and Double DQN.

Actor-only algorithm learns a policy directly instead of $Q^\pi(s, a)$ [19]. Hence, for each given state, the actor can output the probability distribution to sample an action. Furthermore, since actor-only algorithm does not have to learn the value for all state-action pairs, actor-only algorithm can be used with both discrete and continuous action space. Currently, the most well-known actor-only algorithm is REINFORCE algorithm.

CHAPTER 2

Assume that the policy $\pi_{\theta}(A|S)$ is represented by using a set of parameters θ , which are the parameters of the neural network model. This policy can then be used to interact with the environment to sample a trajectory that consists of sequence of rewards. Then, parameters θ can be updated to maximise the objective function in Equation 2.29 as referred from [19] by performing gradient ascent on the parameters θ .

$$J(\theta) = E[G_t \ln \pi_{\theta}(a_t | s_t)] \quad (2.29)$$

Equation 2.30 as referred from [19] shows the gradient for the objective function in Equation 2.29. Equation 2.30 is also known as Policy Gradient Theorem.

$$\nabla J(\theta) = E[G_t \nabla \ln \pi_{\theta}(a_t | s_t)] \quad (2.30)$$

Actor-critic algorithm learns both policy network (actor) and a value network (critic). Some algorithms uses both actor and critic are Advantage Actor-Critic (A2C), Proximal Policy Optimisation (PPO), Deep Deterministic Policy Gradient (DDPG) and Twin Delayed DDPG (TD3). During the training process, the actor is used to output and action or probability distribution of actions while the critic was used to provide a better feedback for the actor.

2.2.7 Advantage Actor-Critic (A2C)

Advantage Actor Critic is an algorithm that is based on Policy Gradient Theorem. Compared to REINFORCE algorithm (actor-only), A2C uses a critic and a function called advantage function to provide lower variance feedback for the actor.

An advantage function measures how much better or worse an action is compared to the average action of the policy. The Advantage function can help us to prevent the policy from being falsely penalised (rewarding) if the agent was in a bad state (good state). Equation 2.31 is the advantage function referred from [19], which is defined to be:

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \quad (2.31)$$

In A2C, the advantage can be estimated by using Generalised Advantage Estimation (GAE) [19]. The objective function of A2C is shown in Equation 2.32 as referred from [19] while the gradient of the objective function is shown in Equation 2.33

$$J(\theta) = E \left[A^\pi(s_t, a_t) \ln \pi_\theta(a_t | s_t) \right] \quad (2.32)$$

$$\nabla J(\theta) = E \left[A^\pi(s_t, a_t) \nabla \ln \pi_\theta(a_t | s_t) \right] \quad (2.33)$$

2.2.8 Proximal Policy Optimisation (PPO)

Proximal Policy Optimisation is an algorithm that is improved based on A2C. This is because A2C faces a problem known as performance collapse caused by sudden large changes in the policy. Hence, PPO introduces gradient clipping to prevent the policy to be changed drastically. Equation 2.34 [19] shows the objective function of PPO.

$$J(\theta) = E \left[\min \left(r_t(\theta) A_t^{\pi^{old}}(s_t, a_t), \text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) A_t^{\pi^{old}}(s_t, a_t) \right) \right] \quad (2.34)$$

, where $r_t(\theta) = \frac{\pi^{new}(a_t | s_t)}{\pi^{old}(a_t | s_t)}$ is known as importance sampling weights, $A_t^{\pi^{old}}$ represents the advantage estimated by using old policy and ϵ is the clip parameter.

2.2.9 Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 is a successor algorithm of Deep Deterministic Policy Gradient (DDPG) proposed by [35]. One major difference between TD3 and DDPG as compared to A2C and PPO is that it outputs the action directly instead of the probability distribution for sample the action. There were a few key improvements made in TD3. The first improvement was that TD3 uses two critic networks instead of one to reduce the overestimation bias. Furthermore, TD3 uses a target network with a slow-moving

update rate parameterised by τ to produce the target output for training. TD3 also delayed the update of the policy network to prevent divergence from occurring.

2.2.10 Soft Actor Critic (SAC)

SAC is also a stochastic off-policy algorithm which aims to tackle two major challenges in reinforcement learning. The first challenge is high sample complexity, which refers to the number of steps required for training especially for on policy algorithms. The second challenge is that reinforcement learning algorithms can be very sensitive to hyperparameters and can be difficult to tune. In SAC, the entropy will be optimised along with the reward signal. SAC is also highly similar to TD3 and several concepts such as the use of two critic networks and a target network.

2.3 Related works

Two major types of reinforcement learning have been applied in financial trading which are critic-only algorithms and actor-critic algorithms. Examples of critic-only algorithms include Q-Learning and Deep Q-Network (DQN) while examples of actor-critic algorithms include Advantage Actor-Critic (A2C), Asynchronous Advantage Actor-Critic (A3C), and Proximal Policy Optimisation (PPO).

The performance of DQN and its variation Deep Recurrent Q-Network (DRQN) have been studied [36]. The results showed that DRQN performs slightly better than DQN. This was because a recurrent neural network allows the model to capture more past information that was not present in the latest time step. However, one problem of critic-only algorithms is that they cannot be applied to a problem with continuous action space.

Some researchers have also applied both critic-only algorithms and actor-critic algorithms to make a comparison among the performance of various algorithms. An improved version of Deep Q-Network (DQN) and Asynchronous advantage Actor-

CHAPTER 2

Critic (A3C) with discrete action space have been applied and proposed by [18]. The results showed that A3C performed better than DQN for both the basic version and the extended version. The effect of transaction cost on DQN and A2C algorithms has been studied by [7]. Although DQN has a higher Sharpe ratio in low transaction cost settings, the Sharpe ratio of DQN decreased much faster than A2C as the transaction cost increased. This might be because the action space of DQN was defined to be discrete that only allow neutral, maximally short position and maximally long position while the action space of A2C was defined to be continuous. Hence, this caused a large amount of transaction cost to be incurred when using DQN which degrade the performance. Using discrete actions might also lead to higher risk as the reinforcement learning agents can only choose to go long or short by using all of the cash available.

There was also a reinforcement learning framework that were proposed for financial trading, known as Recurrent Reinforcement Learning (RRL) proposed by [37]. The core idea of RRL is that the trading decision (action) of the previous time step will be passed alongside the state information of the current time step into the model, then output the next trading decision directly. This allowed the model to be optimised directly based on the objective function such as total return or Sharpe ratio. Since then, other researchers have also applied the RRL in their approaches. [37] has proposed a financial trading adopted approach using RRL framework and compare its performance with genetic programming (GP) algorithm. The author showed that the model with RRL framework performed significantly better than GP when applied to daily stock index data but underperform when applied to monthly data. [37] have also applied RRL framework in intraday trading. The authors proposed to maximise both average return and Sharpe ratio in the objective function. The results showed that the proposed approach gains a higher return than the standard RRL algorithm. One common problem of these works is that no feature learning or feature extraction is applied. [17] have proposed an approach called Fuzzy Deep Recurrent Neural Network (FDRNN). The authors used fuzzy learning to reduce the uncertainty of the

CHAPTER 2

financial market data. The authors also added a deep neural network for feature learning before the recurrent neural network. The results showed that the model achieved a high return even with high transaction costs.

Furthermore, many researchers have also applied various approaches and neural network architectures to improve the performance of reinforcement learning models. [39] have proposed a multi-ensemble approach to increase the robustness of the model. The authors first pre-processed the time series data into Gramian Angular Field images. This was to perform feature extraction by using a 2D-Convolutional Neural Network (CNN) instead of using the financial time series directly as the input. Multiple reinforcement learning agents were trained in an independent environment. The final trading decision was made by using the majority vote of the agents. However, the authors did not consider the position that each agent was holding when fusing the decisions from the agents. This was crucial because failing to take the position holding into account might incur a large number of transaction costs for changing the position frequently. [40] generated a set of features by using technical indicators and cointegration tests. The authors used an attention mechanism on the output of a Gated Recurrent Unit (GRU) layer to weigh the features extracted from the historical time steps. Furthermore, the authors also included future price prediction as an additional output to provide more feedback signals for faster convergence. [41] have applied DQN in trading stocks and crypto-currencies with daily intervals. The authors compared the performance of different feature extraction networks including Multi-Layered Perceptron (MLP), Convolutional Neural Network (CNN), GRU and CNN-GRU. The authors also compared the performance using only candlestick prices from the last time step and the performance of using a series of past candlesticks prices. Interestingly, the results showed that each model exhibited different performance when applied to different stocks or crypto-currencies. Although every model gained a larger profit compared to the buy and hold strategy, the authors did not consider transaction costs in the work. This caused the models' performance to be overly optimistic.

CHAPTER 3

Research Methodology

3.1 Methodologies and General Work Procedures

Figure 3.1 shows the research framework of this research.

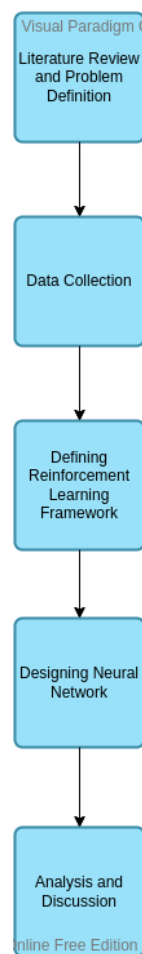


Figure 3.1 Research framework

3.1.1 Literature Review and Problem Definition

Literature review was first carried out in this research. Terminologies, formulas and concepts related to financial trading were studied. Previous works related to machine learning and financial trading were reviewed. Reinforcement learning was then selected to be applied and studied in this research. More literature review related to various reinforcement learning algorithms and their applications in financial trading were done. Furthermore, various deep learning models related to time-series were studied and identified. After identifying the research gap, the problem statements of the research were defined.

3.1.2 Data Collection

Datasets from Foreign Exchange markets (FOREX), equity indexes, and commodities were selected to be used in this research. The data were obtained from Dukascopy [42]. In this research, the interval of the data were chosen to be 1 minute. The collected data were then preprocessed such as removing flat values and computing the technical indicators. The details of data preprocessing will be explained in Chapter 4 Research Design.

3.1.3 Defining Reinforcement Learning Framework

After completing data collection and data preprocessing, a reinforcement learning framework was defined according to the problem statements and the research objectives. The reinforcement learning problem was defined as a Markov Decision Process (MDP). This include defining the state representation, the actions to be taken by the reinforcement learning agent, and the reward function. During this process, technical indicators such as Simple Moving Average (SMA) and Exponential Moving Average (EMA) were selected according to previous works to be included in the state representation. Various types of probability distributions were studied to identify the

most suitable distributions to sample the action. The reinforcement learning framework will be further discussed in Chapter 4 Research Design.

3.1.4 Designing Neural Network

A suitable neural network was designed according to the reinforcement learning framework that has been defined. This included selecting the suitable neural network modules such as InceptionTime [43], which is a type of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) for the task.

3.1.5 Analysis and Discussion

The reinforcement learning algorithms and neural network were then implemented to perform model training. The performance of the models were then evaluated by using performance metrics such as total return and Sharpe ratio. The model was also compared with other baseline methods including Buy and Hold, Sell and Hold, FDRNN [17] to further verify its performance.

CHAPTER 4

Research Design

4.1 Data Acquisition and Preprocessing

The datasets were acquired from Dukascopy through the official website [42]. The datasets of this research were 1-minute Bid price candlesticks starting from 1st Jan 2017 until 31st Dec 2020. The asset classes that were studied in this research include foreign exchange market (FOREX), commodities and equity indices. After data acquisition, flat values that are longer than 60 continuous time steps were removed. The filtered datasets were pre-processed to form datasets of 5-minute and 30-minute intervals. The candlesticks data were then used to calculate the Heikin-Ashi candlesticks. Technical indicators including Simple Moving Average (SMA), Exponential Moving Average (EMA), Bollinger Bands, Moving-Average Convergence Divergence (MACD), Average True Range (ATR), Rate of Change (ROC) and Commodity Channel Index (CCI) were calculated by using the Heikin-Ashi candlesticks price. Time data for each record was converted into minutes, and then normalised. The datasets were split into two parts which are training set, and testing set. The training sets used were between 1st Jan 2017 and 31st Dec 2019 while the test set starts from 1st Jan 2020 until 31st Dec 2020.

4.2 Markov Decision Process Formalisation

The trading problem was formulated by using Markov Decision Process (MDP). This section discussed the state space, action space and reward function for the MDP.

State:

The state space included both market variables, technical indicators, and private variables. Market variables consisted of the close price from normal candlesticks, close price, low price and high price from Heikin-Ashi candlesticks, volume. Technical indicators consisted of SMA, EMA, MACD, Bollinger Bands, ATR, ROC and CCI as referred from [44]. At each time step, the past 60 observations of 1-minute, 5-minute and 30-minute intervals of each market variable were obtained. Heikin-Ashi close price, low price, high price and overlay indicators including SMA, EMA and Bollinger Bands were grouped and normalised together while each oscillator indicators including MACD, ATR, ROC and CCI were normalised independently.

There were four private variables in total. The first private variable was $\frac{v_t - v_{t-1}}{v_{t-1}}$,

where v_t represents the account value at time t . This can provide the agent information regarding the change in account value. The second private variable was

defined to be $\frac{p_t \times C_t}{v_t}$ where p_t and C_t represents the position size and close price at

time t respectively. This was to provide information to the agent regarding its position at the time to prevent frequent changes of position that can lead to large amount of

transaction cost being incurred. The third private variable was defined to be $\frac{f_t}{v_t}$, where

f_t was the floating profit or loss at time t . This reinforcement learning agent would then be able to determine whether profit should be taken or to stop the loss. The

fourth private variable provided was the volatility of the past 60 time steps of 1-minute data. Volatility was included because after the market variables and technical indicators were normalised, the information of the exact price changes will be lost.

Action:

For Twin Delayed DDPG (TD3), the action space was a single value with the range of $[-1, 1]$ that represents the targeted position. Each action for Advantage Actor Critic (A2C), Proximal Policy Optimisation (PPO) and Soft Actor Critic (SAC) was a tuple of size 2 as shown in Table 4.1. The first action can take 3 different values. Value 0 represents short, value 1 represents long and value 2 represents that no trade should be performed for this time step and the p_t will remain the same as p_{t-1} . This was to prevent frequent changes in position that incur transaction cost. The second value represents the targeted ratio of used margin to account value. Discrete action space was used by all of them while continuous action space was used only by A2C and PPO.

Table 4.1 Action space for A2C, PPO and SAC

Action	Set of possible values
Direction	{0, 1, 2}
Targeted ratio of used margin to account value (Discrete)	{0, 0.25, 0.5, 0.75, 1}
Targeted ratio of used margin to account value (Continuous)	[0, 1]

Reward:

The reward function was defined as in Algorithm 1. All transaction costs were only incurred when positions are opened. As shown in Equation 2.24, return was calculated as the discounted sum of rewards. Hence, the reinforcement learning agent will not be able to obtain sufficient return from future steps to overcome the transaction costs that were charged when closing a position. However, this will cause the reinforcement learning agent to close the positions frequently due to the lack of transaction cost. Hence, as shown at line 10 of Algorithm 1, an additional reward or penalty has been given when the positions were closed. The reward was divided by the account balance

and scaled to a more suitable range to ensure that the reward does not get affected by the account balance.

Algorithm 1 Reward

Input:

Current close price C_t and new close price $C_{t'}$

Current position p_t and previous position p_{t-1}

Current account balance b_t and previous account balance b_{t-1}

Output:

```

1:    $R_t \leftarrow (C_{t'} - C_t) \times p_t$ 
2:
3:   // Incur transaction cost only when open position
4:    $\Delta p \leftarrow p_t - p_{t-1}$ 
5:   if  $p_{t-1} = 0$  or  $\text{sign}(\Delta p) = \text{sign}(p_{t-1})$  then
6:        $R_t \leftarrow R_t - |\Delta p| \times s$ 
7:   end if
8:
9:   // Provide additional reward and penalty according to changes in account
   balance. Occur when positions are closed
10:  if  $b_t - b_{t-1} > 0$  then
11:       $R_t \leftarrow R_t + (b_t - b_{t-1}) \times \text{coef}_1$ 
12:  else
13:       $R_t \leftarrow R_t + (b_t - b_{t-1}) \times \text{coef}_2$ 
14:  end if
15:
16:  // Scale reward to suitable range

```

```

17:    $R_t \leftarrow \frac{R_t}{b_{t-1}} \times \text{reward scaling}$ 
18:   return  $R_t$ 

```

Algorithm 2 shows the function of the trading environment that has been executed at each time step. In this research, a novel approach was proposed for the agent to interact with the trading environment. As shown from line 1 to line 5, the reinforcement learning agent was not allowed to go from long position to short position or vice versa directly. Instead, they can only go from long or short to neutral position. Afterwards, line 6 will determine the next time step, t' to obtain the state information. If the direction has been changed as evaluated at line 1, the time step t' will remain unchanged as t . By doing so, the agent can select an action again to go long or short position. Lastly, as shown at line 17, if the condition at line 1 was evaluated to be true, the episode will be terminated. The motivation of this approach was to prevent the discounted return from affecting by the performance of subsequent trade.

Algorithm 2 Step function for environment

Input:

Current time step , t

Action taken at time step t , a_t

Output:

```

1:   Direction changed  $\leftarrow \text{sign}(a_{t-1}) = -\text{sign}(a_t)$ 
2:   if Direction changed then
3:      $a_t \leftarrow 0$ 
4:   end if
5:   Update position according to  $a_t$ 

```

Chapter 4

```
6:   if Direction changed then
7:      $t' = t$ 
8:   else
9:     Find next time step,  $t'$  where  $t' > t$  and  $(C_t \neq C_{t'}, \text{ or } H_t \neq L_{t'})$ 
10:  end if
11:  Update account based on  $t'$ 
12:  Calculate  $r_t$ 
13:  Get  $s_{t'}$ 
14:  if  $s_{t'}$  is terminal state then
15:    Close all position
16:  end if
17:   $d_t \leftarrow$  Direction changed or  $s_{t'}$  is terminal state
18:  return  $(s_{t'}, r_t, d_t)$ 
```

Table 4.2 and 4.3 shows an example that illustrates the effect of using common approach and proposed approach. In this example, few assumptions have been made for the sake of simplicity. Assume that the only possible position size was -1, 0 and 1, the starting position was 0, the spread was 5.00, the reward for common approach have been calculated using Equation 4.1, *coef1* and *coef2* in the proposed reward function were set to 0, and $\gamma = 1$ when calculating return G_t .

$$R_t = (C_{t'} - C_t) \times p_t - \frac{|p_t - p_{t-1}|}{2} \times s \quad (4.1)$$

In Table 4.3, there were two separated episodes labelled as E1 and E2. This was because the direction of the position has changed from 1 to -1 at time step 6. As

mentioned in Algorithm 2, when the direction was changed, the episode will be terminated, and the agent will be required to select an action again at time step 6. The action has been set to 1 to compare with the common approach. In both tables, it shows that from time step 1 to 6, the price has increased by 10.00, from 2.00 to 12.00. As the spread was set to be 5.00, the agent has gained a total profit of 5.00. However, as shown in Table 4.2, all the return from time step 1 to 6 were negative. This was because the negative rewards due to a losing trade from time step 7 to 9 have also been summed when calculating the return. In comparison, Table 4.3 shows that all the return of the first episode was non-negative despite that time step 7 to 9 have made a losing trade. This shows that by terminating or splitting an episode when the direction has been changed can provide better feedback to the reinforcement learning agent.

Table 4.2 Example of common approach

Time step, t	1	2	3	4	5	6	7	8	9	10
Price, C_t	2.00	4.00	7.00	6.00	10.00	12.00	9.00	11.00	12.00	15.00
Position, p_t	1	1	1	1	1	-1	-1	-1	-1	-1
Reward, r_t	-0.5	3	-1	4	2	-2	-2	-1	-3	—
Return, G_t	-0.5	0	-3	-2	-6	-8	-2	-4	-3	—

Table 4.3 Example of proposed approach

Time step, t		1	2	3	4	5	6	7	8	9	10
Price, C_t		2.00	4.00	7.00	6.00	10.00	12.00	9.00	11.00	12.00	15.00
E1	Position, p_t	1	1	1	1	1	0	—	—	—	—
	Reward, r_t	-3	3	-1	4	2	0	—	—	—	—
	Return, G_t	5	8	5	6	2	0	—	—	—	—
E2	Position, p_t	—	—	—	—	—	-1	-1	-1	-1	-1
	Reward, r_t	—	—	—	—	—	-2	-2	-1	-3	—
	Return, G_t	—	—	—	—	—	-8	-6	-4	-3	—

4.3 Model Framework

Figure 4.1 shows the model framework for actor network and critic network. The neural network consists of three InceptionTime-LSTM encoders for feature extraction and a fully connected network. Each of the InceptionTime-LSTM encoders was trained on a dataset of different intervals, which were 1-minute, 5-minute and 30-minute. At each time step, the encoders extracted the features from the time series data of their respective interval. The extracted feature vectors of different intervals and private variables were then concatenated to form a single vector. The vector was then passed into a fully connected layer to generate the final output. For A2C, PPO and SAC, there were 2 additional heads at the final part of the neural network. Each of them outputs the parameters for a single probability distribution. For example, if the algorithm used was A2C with continuous action space, the first head will produce the logits of categorical distribution, while the second head will produce the alpha and beta for the beta distribution.

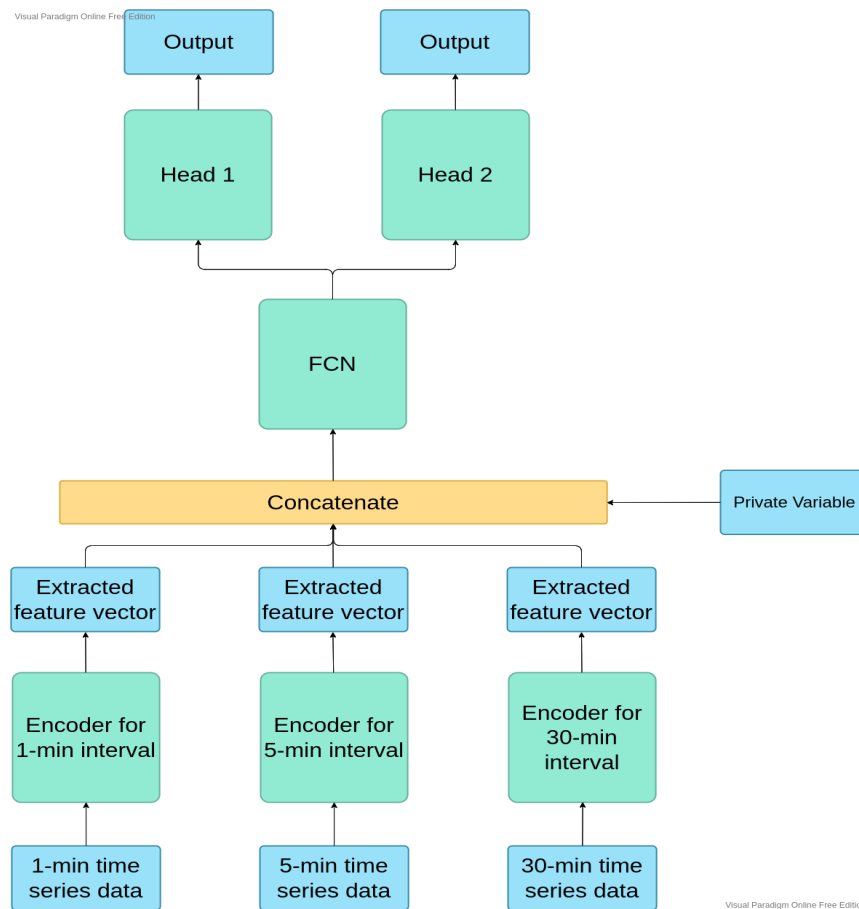


Figure 4.1 Model framework for actor network and critic network

Figure 4.2 shows the design of the InceptionTime-LSTM autoencoders. There are two major components in the proposed autoencoder, the encoder and the decoder. The encoder consists of an InceptionTime module and a LSTM module while the decoder only consists of a LSTM module. The input is first corrupted by adding random noise and applying dropout. The corrupted input was then passed into the encoder network and the final LSTM layer of the encoder generates an output vector. After that, the output vector is then repeated and passed into the decoder to reconstruct the input of the encoder. This is also known as repeat vector, which is shown in Figure 4.3 as referred from [45].

Chapter 4

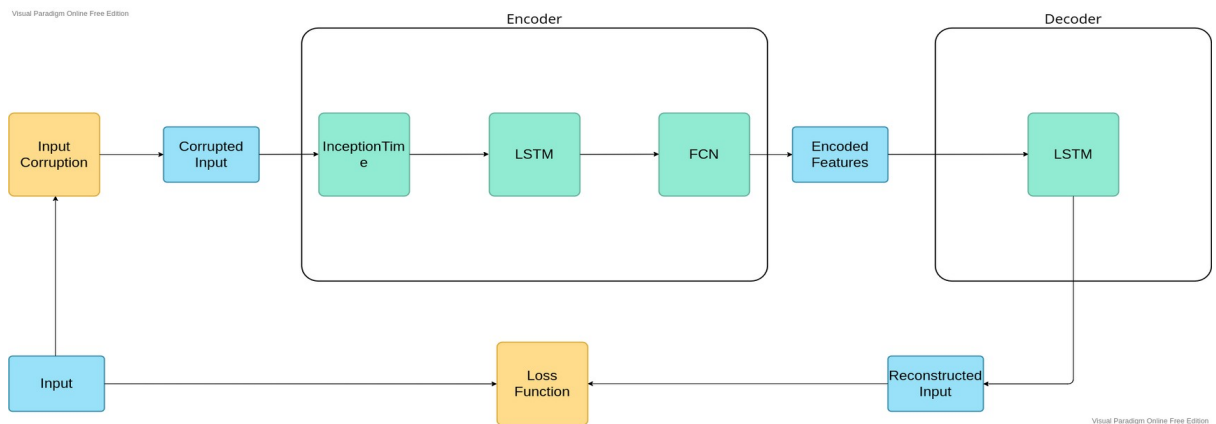


Figure 4.2 Design of InceptionTime-LSTM autoencoder

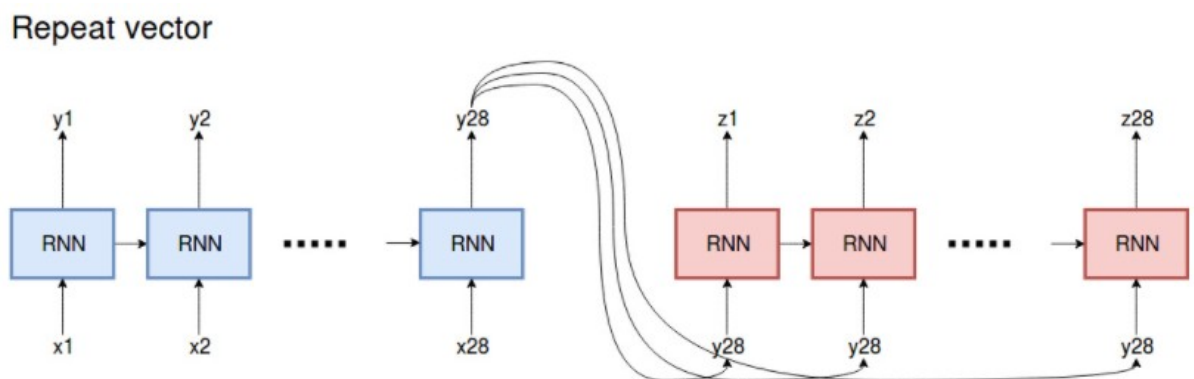


Figure 4.3 LSTM repeat vector [45]

Furthermore, causal convolution has been applied instead of standard convolution. As shown in Figure 4.4 and Figure 4.5 that were referred from [46] show an example of standard convolution block with two layers with kernel size 3 while Figure 4.5 shows an example of causal convolution block with two layers with kernel size 3. The figures show that the output of each convolution operation only consists of data from past time steps whereas standard convolution contains data from the future. Hence, causal convolution is a more suitable choice for time series data.

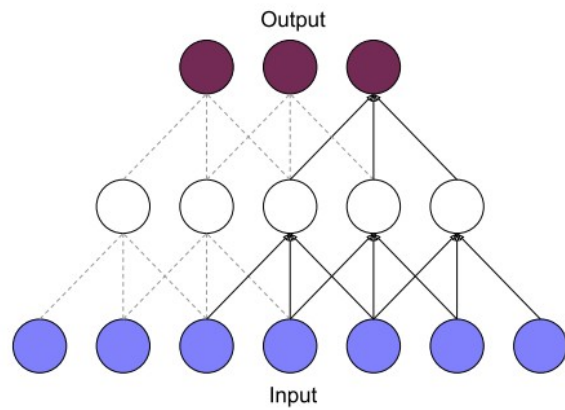


Figure 4.4 Example of standard convolution block with two layers with kernel size 3 [46]

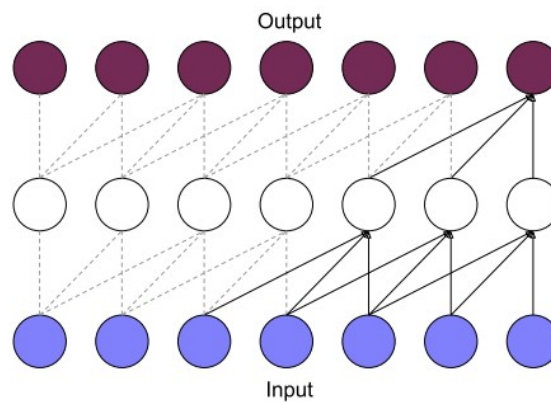


Figure 4.5 Example of causal convolution block with two layers with kernel size 3 [46]

4.4 Output Layers and Types of Distribution

Each combination of algorithms and action spaces have diverse probability distribution for the actors. Table 4.2 shows the distributions used by different algorithms and action space.

Table 4.4 Output layers and types of distribution for each algorithm and action space

Algorithms	A2C and PPO (Continuous)	A2C and PPO (Discrete)	SAC (Continuous)
Direction	Categorical	Categorical	Relaxed One-Hot Categorical
Targeted ratio of used margin to account value	Beta	Categorical	Beta

For continuous action, beta distribution was used as the values lies in the range of [0, 1], and thus it guaranteed that no invalid action would be sampled. The alpha and beta values have undergone a function as shown in Equation 4.2 to ensure the alpha and beta were at least 1.

$$f(x) = \text{ReLU}(x) + 1 \quad (4.2)$$

For SAC, the output size chosen was the same as A2C and PPO. However, relaxed bernoulli was used instead to allow backpropagation via the reparameterization trick.

4.5 Algorithms

This section shows the pseudocodes of the algorithms with the modifications. Algorithm 3 to Algorithm 6 shows the pseudocode for modified Advantage Actor Critic (A2C) [19], Proximal Policy OptimisatAion (PPO) [19], Twin Delayed DDPG (TD3) [21] and Soft Actor Critic (SAC) [20]. Apart from specifying the number of episodes to be sampled before updating the model parameters, at least K number of time steps must be sampled before updating the parameters. This was because each episode will be terminated once the direction

has changed, or when the position remains at 0. Hence, each episode will be of different length. N-step return has also been applied in TD3 and SAC to accelerate convergence [47].

Algorithm 3 Advantage Actor Critic (A2C)

Input:

Initialize critic network V_ψ and actor network π_θ with random parameters ψ, θ

Output:

- 1: **for** $o = 1$ to O **do**
 - 2: **while** total time steps $< T$
 - 3: Sample episode of transition tuple
 $E = (s_0, a_0, r_0, s_1, a_1, r_1 \dots s_I, a_I, r_I)$
 - 4: Calculate advantage $A(s, a)$ using Generalised Advantage Estimation
 - 5: Calculate target value $V_{tar}(s) = A(s, a) + V_\psi(s)$
 - 6: Calculate entropy $H(\pi_\theta(s))$
 - 7: Accumulate V-function loss:

$$J_V(\psi) = \frac{1}{|E|} \sum_{s \in E} (V_\psi(s) - V_{tar}(s))^2$$
 - 8: Accumulate policy loss:

$$J_\pi(\theta) = \frac{1}{|E|} \sum_{s, a \in E} (A(s, a) \log \pi_\theta(a|s) + \alpha H(\pi_\theta(s)))$$
 - 9: **end while**
 - 10: Update V-function:
 $\psi \leftarrow \psi - \lambda_V \nabla_\psi J_V(\psi)$
 - 11: Update policy:
 $\theta \leftarrow \theta + \lambda_\pi \nabla_\theta J_\pi(\theta)$
 - 12: **end for**
-

Algorithm 4 Proximal Policy Optimisation (PPO)

Input:

Initialize critic network V_ψ and actor network π_θ with random parameters ψ, θ

Output:

- 1: **for** $o = 1$ **to** O **do**
- 2: Initialise D as empty storage
- 3: **while** number of episodes $< K$ or total time steps $< T$
- 4: Sample episode of transition tuple $(s_0, a_0, r_0, s_1, a_1, r_1 \dots s_I, a_I, r_I)$
- 5: Store the transition tuple into D
- 6: Calculate advantage $A(s, a)$ using Generalised Advantage Estimation
- 7: Calculate target value $V_{tar}(s) = A(s, a) + V_\psi(s)$
- 8: **end while**
- 9: **for** $i = 1$ **to** I **do**
- 10: Split D randomly into mini batches of with maximum size M
- 11: **for** all mini batch B in D **do**
- 12: Calculate entropy $H(\pi_\theta(s))$
- 13: Calculate V-function loss:

$$J_V(\psi) = \frac{1}{|B|} \sum_{s \in B} (V_\psi(s) - V_{tar}(s))^2$$
- 14: Calculate policy loss:

$$J_\pi(\theta) = \min \left(\frac{\pi_\theta}{\pi_{\theta_k}} A, \text{clip} \left(\frac{\pi_\theta}{\pi_{\theta_k}}, 1 - \epsilon, 1 + \epsilon \right) A \right)$$
- 15: Update V-function:

$$\psi \leftarrow \psi - \lambda_V \nabla_\psi J_V(\psi)$$
- 16: Update policy:

Chapter 4

$$\theta \leftarrow \theta + \lambda_{\pi} \nabla_{\theta} J_{\pi}(\theta)$$

17: **end for**

18: **end for**

19: **end for**

Algorithm 5: Soft Actor Critic (SAC)

Input :

Initialize critic networks Q_1, Q_2 and actor network actor network
with random parameters $\omega_1, \omega_2, \psi, \theta$

Initialize target network

Initialize replay buffer

Output:

1: **for** $t = 1$ to T **do**

2: **while** number of episodes $< E$ or total time steps $< T$

3: Sample episode of transition tuple

4: Store the transition tuple in replay buffer

5: **end while**

6: Sample a batch of N -step transition tuple

$B = \{(s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_N, a_N, r_N)\}$ from replay buffer D

7: Let $\omega = \{\omega_1, \omega_2\}$,

8:

9: for $i=1, 2$ and $\tilde{a} \sim \pi_{\theta}(\cdot|s)$

10: Calculate Q-functions loss:

$$J_Q(\omega) = \frac{1}{|B|} \sum_{s \in B} (Q_{\omega, i}(s, a) - Q_{tar}(s))^2, j=1,2$$

Chapter 4

11: Update Q-functions:

$$\omega \leftarrow \omega - \lambda_V \nabla_{\omega} J_V(\omega), j=1,2$$

12: Calculate V-function loss:

$$J_V(\psi) = \frac{1}{|B|} \sum_{s \in B} (V_{\psi}(s) - V_{tar}(s))^2$$

13: Update V-function:

$$\psi \leftarrow \psi - \lambda_V \nabla_{\psi} J_V(\psi)$$

14: $J_{\pi}(\theta) = \frac{1}{|B|} \sum_{s \in B} Q_{\omega,1}(s, \tilde{a}_{\theta} - \alpha \log \pi_{\theta}(\tilde{a}_{\theta}(s)|s))$, where $\tilde{a}_{\theta}(s)$ is a sample from $\pi_{\theta}(\cdot|s)$ which is differentiable w.r.t θ via reparameterization trick

15: Update target networks:

$$\psi'_j \leftarrow \tau \psi_j + (1 - \tau) \psi'_j, j=1,2$$

16: **end for**

Algorithm 6 Twin Delayed DDPG (TD3)

Input:

Initialize critic networks $Q_{\omega_1}, Q_{\omega_2}$ and actor network π_{θ} with random parameters $\omega_1, \omega_2, \theta$

Initialize target networks $\omega'_1 \leftarrow \omega_1, \omega'_2 \leftarrow \omega_2, \theta' \leftarrow \theta$

Initialize replay buffer D

Output:

1: **for** $o = 1$ **to** O **do**

2: **while** number of episodes $< K$ or total time steps $< T$

3: Sample episode of transition tuple $(s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_I, a_I, r_I)$
with exploration noise $a \sim \pi_{\theta}(s) + \epsilon, \epsilon \sim N(0, \sigma)$

4: Store the transition tuple in replay buffer D

5: **end while**

Bachelor of Computer Science (Honours)

Faculty of Information and Communication Technology (Kampar Campus), UTAR

Chapter 4

6: Sample a batch of N -step transition tuple

$$B = \left\{ (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_N, a_N, r_N) \right\} \text{ from replay buffer } D$$

7: Let $s \leftarrow s_0$

$$8: \tilde{a}_N \leftarrow \pi_{\theta'}(s_N) + \epsilon, \epsilon \sim \text{clip}(N(0, \tilde{\sigma}), -c, c)$$

$$9: Q_{tar}^{\pi}(r, s_N) = \sum_{n=0}^{N-1} \gamma^n r_n + \gamma^N \min_{j=1,2} Q_{\omega',j}(s_N, \tilde{a}_N)$$

10: Calculate Q-functions loss:

$$J_{Q,j}(\omega_j) = \frac{1}{|B|} \sum_{s \in B} (Q_j(s) - Q_{tar}(s))^2, j=1,2$$

11: Update Q-functions:

$$\omega_j \leftarrow \omega_j - \lambda_{Q,j} \nabla_{\omega,j} J_{Q,j}(\omega_j), j=1,2$$

12: **if** $o \bmod d$ **then**

13: Calculate policy loss:

$$J_{\pi}(\theta) = \frac{1}{|B|} \sum_{s \in B} Q_{\omega,1}(s, \pi_{\theta}(s))$$

14: Update policy:

$$\theta \leftarrow \theta + \lambda_{\pi} \nabla_{\theta} J_{\pi}(\theta)$$

15: Update target networks:

$$16: \omega'_j \leftarrow \tau \omega_j + (1 - \tau) \omega'_j, j=1,2$$

$$17: \theta' \leftarrow \tau \theta + (1 - \tau) \theta'$$

18: **end if**

19: **end for**

CHAPTER 5

Experiment Setup

This section discusses the hardware setup, software setup, and all the hyperparameter values of the trading environment and reinforcement learning algorithms.

5.1 Hardware Setup

Table 5.1 shows the hardware used in this research. The hardware used in this research are provided by Kaggle, the platform which the models were trained on.

Table 5.1 Hardware used in this research

Hardware	Description
CPU	Intel ® Xeon® CPU @ 2.20GHz
RAM	16 GB
Graphic Card	None

5.2 Software Setup

Table 5.2 shows the software used in this research. The experiment were conducted by using Python programming language and PyTorch as the library for deep learning. The models were trained on Kaggle with the hardware .

Table 5.2 Software used in this research

Software	Version	Description
Python	3.7.12	Programming languages
PyTorch	1.9.1	Library for implementing the neural network
Kaggle	—	Platform to perform the model training

5.3 Setting and Configuration

5.3.1 Trading Environment Setting

The models were trained and tested on EUR/USD, USA500.IDX/USD, and GAS.CMD/USD. Table 5.3 shows the settings for each dataset. The market trading hours were in Greenwich Mean Time (GMT). For EUR/USD and GAS.CMD/USD, every change in position size must be at least 1000 units.

Table 5.3 Settings for each dataset

	EUR/USD	USA500.IDX/USD	GAS.CMD/USD
Spread	0.0002	1	0.02
Smallest position size	1000	1	1000
Market trading hours	Sunday 21:00 to Friday 22:00	Monday to Friday 14:30 to 20:00	Sunday 23:00 to Friday 21:00 Not available from 21:01 to 22:59
Reward scaling	5000	2000	250

5.3.2 Trading Agent Setting

Table 5.4 shows the hyperparameter settings for the neural network and the reinforcement learning algorithms.

Table 5.4 Hyperparameters settings

Hyperparameter	A2C	PPO	SAC	TD3
Kernel size of CNN in InceptionTime	[10, 20, 40]			
Output channels of InceptionTime	128			
Number of LSTM layers	2			
Hidden size of LSTM	128			

CHAPTER 5

Units of Fully Connected Network	[256, 512, 256, 128]			
Hidden units for body	[512, 1024, 512, 128]			
Hidden units for each actor network head	[128, 64, 32]			
Initial balance	100000			
Window size for historical price data	60			
Loss Function	Mean Squared Error			
Actor network learning rate	0.00005			
Critic network learning rate	0.000025			
β_1 for Adam optimiser	0.9			
β_2 for Adam optimiser	0.999			
Discount factor, γ	0.999			
Maximum steps per episode, N	120			
<i>coef1</i> of reward function	0.001			
<i>coef2</i> of reward function	0.0002			
Minimum steps per loop, T	120	600	120	120
Number of episodes, K	—	5	3	3
Minibatch size, M	—	120	120	120
Parameter for Generalised Advantage Estimation, λ	0.99	0.99	—	—
Entropy regularisation coefficient, α	0.02	0.02	0.02	—
Clipping parameter, ϵ	—	0.2	—	—
Update Iteration, I	—	3	—	—
Replay buffer size	—	—	10000	10000
Initial replay buffer size	—	—	2000	2000
Target network update rate, τ	—	—	0.005	0.005
Act noise, σ	—	—	—	0.1

CHAPTER 5

Target noise, $\tilde{\sigma}$	—	—	—	0.2
Clip range, c	—	—	—	0.5
Policy delay, d	—	—	—	2

CHAPTER 6

Analysis and Discussion

6.1 Experimental Result

As stated in Chapter 4, the proposed model has been explained. In total, there were 6 different combinations of algorithms and action space. This include A2C with discrete and continuous action space, PPO with discrete and continuous action space, SAC with continuous action space and TD3 with continuous action space. This section will compare the performance of the proposed models and the baseline methods including Buy and Hold, Sell and Hold, and Fuzzy Deep Recurrent Neural Network (FDRNN). Three different datasets have been used to evaluate the performance, which were EUR/USD, USA500.IDX/USD, GAS.CMD/USD.

The performance of the models have been evaluated by using expected return, Sharpe ratio, Sortino ratio, Maximum Drawdown. The higher the value of these performance metrics, the better the performance. Table 6.1 shows the result of proposed models and compared with baseline method including Buy and Hold, Sell and Hold and Fuzzy Deep Recurrent Neural Network (FDRNN). Figure 6.1 to 6.3 shows the graph of the account value throughout the testing period. In Table 6.1, the entry that was green in colour highlights the best method for the performance metric and the dataset, while red highlights the method that performs the worst. The best proposed models were also identified by using yellow highlighting.

For the first dataset, it shows that the baseline method Buy and Hold performed the best in all aspect, which were annualised expected return, annualised Sharpe ratio, annualised Sortino ratio and maximum drawdown. On the other hand, FDRNN has produced the most negative results among all, achieving -0.472 annualised expected

return. Among the proposed models, PPO with continuous action space and SAC with continuous action space has obtained a considerable results. PPO obtained the highest annualised expected return and maximum drawdown, while SAC has the highest annualised Sharpe ratio and Sortino ratio. For the second dataset, A2C with continuous action space has the highest performance in terms of annualised expected return and maximum drawdown. It was also the best for all performance metrics among proposed models. On the other hand, Sell and Hold has achieve the best Sharpe ratio and Sortino ratio for the second dataset. For the third dataset, it can be seen that all of the proposed methods have 0 return due to the fact that no positions have been taken throughout the entire testing period. This explains the graph that is shown in Figure 6.3. All proposed models have maintained a horizontally flat account value. On the other hand, FDRNN experienced a huge drop right at the beginning.

As shown in Table 6.1, the proposed models does not perform well as all of the algorithms show negative return. This is mainly due to frequent changes in position that leads to large transaction cost. At the rightmost column, it can be seen that most algorithms have around 10000 changes in direction throughout the testing period. This was because the entropy regularisation coefficient was set to be 0.1, which is a high value. The reason of setting a high entropy was to prevent the reinforcement learning agent from reaching a local minimum condition, in which it at a position for a long period. However, despite setting a high entropy, A2C with continuous action space in USA500.IDX/USD has only changed the direction for 323 times. Furthermore, Table 6.1 shows that all proposed models obtained 0 annualised expected return. This was because the transaction cost of GAS.CMD/USD was higher as compared to other dataset. This shows that it is difficult to identify a suitable set of the hyperparameters for the algorithm to prevent the reinforcement learning agent from reaching local minimum while performing well. Hence, it shows that the proposed models either changes too frequently or tends to hold. Buy and Hold, and Sell and Hold method has achieved slightly better annualised expected return compared to proposed models due

CHAPTER 6

to less changes in position. Note that the positions have been forced close at the end of market trading hours.

In Figure 6.1 and 6.2, it shows that the proposed models performed slightly better than the baseline method FDRNN. Among all of the algorithms that have been applied, PPO can be considered as the best as both continuous and discrete action space achieve a slightly better performance in both EUR/USD and USA500.IDX/USD datasets. In contrary, SAC has the lowest performance among all proposed models. This can be justified by the nature of SAC algorithm that maximizes the entropy which causes large number of direction changes that degraded the its performance. Surprisingly, despite A2C with discrete action space has higher number of direction changed, it showed a better result than SAC. One possible explanation might be that A2C is an on-policy algorithm while SAC is an off-policy algorithm. Off-policy algorithm tends to be more sample efficient but requires longer training iterations to converge to a desirable policy. Thus, it is possible for SAC to perform similar to A2C but it would require a much longer training time to obtain such performance.

Another interesting pattern that can be seen in Table 6.1 was that TD3 algorithm tends to have a lower number of direction changed as compared to other algorithms. This was because TD3 is a deterministic algorithm and the output value is a continuous value range between -1 and 1. However, the output of TD3 algorithm have saturated during the training time to avoid transaction costs, which lead to the output consistently being -1 or 1.

CHAPTER 7

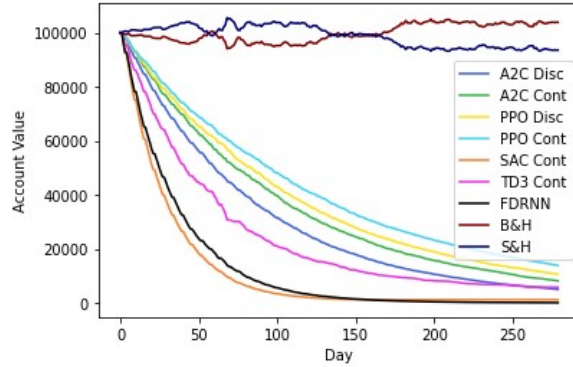


Figure 6.1 Account value over time for all methods for EUR/USD

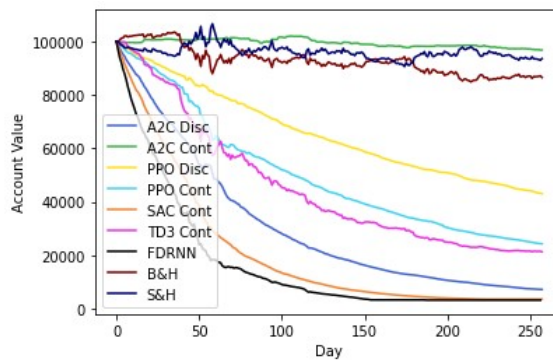


Figure 6.2 Account value over time for all methods for USA500.IDX/USD

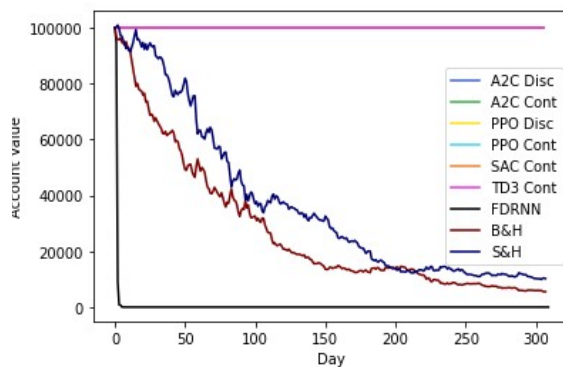


Figure 6.3 Account value over time for all methods for GAS.CMD/USD

CHAPTER 6

Table 6.1 Models performance measured by various performance metrics

	Annualised Expected Return	Annualised Sharpe Ratio	Annualised Sortino Ratio	Maximum Drawdown	Total Number of Direction Changed
Dataset 1: EUR/USD					
A2C Discrete	-0.180	-37.400	-37.97	-0.953	56184
A2C Continuous	-0.152	-33.309	-33.924	-0.928	38515
PPO Discrete	-0.137	-37.300	-37.300	-0.907	39661
PPO Continuous	-0.121	-35.842	-36.257	-0.878	32298
SAC Continuous	-0.261	-14.790	-16.440	-0.989	50306
TD3 Continuous	-0.165	-17.735	-18.619	-0.947	105
Buy and Hold	0.004	0.909	1.268	-0.065	103
Sell and Hold	-0.005	-1.247	-1.929	-0.142	103
FDRNN	-0.472	-39.831	-40.310	-0.9998	134397
Dataset 2: USA500.IDX/USD					
A2C Discrete	-0.163	-25.016	-27.353	-0.927	22792
A2C Continuous	-0.002	-1.133	-1.348	-0.051	323
PPO Discrete	-0.052	-20.211	-25.142	-0.568	8283
PPO Continuous	-0.148	-24.729	-28.604	-0.907	18627

CHAPTER 7

SAC Continuous	-0.206	-22.693	-25.249	-0.964	21174
TD3 Continuous	-0.094	-6.360	-7.871	-0.787	515
Buy and Hold	-0.008	-0.803	-1.016	-0.180	515
Sell and Hold	-0.003	-0.322	-0.486	-0.150	515
FDRNN	-0.210	-10.761	-10.969	-0.967	18064
	Dataset 3: GAS.CMD/USD				
A2C Discrete	0.000	—	—	0.000	0
A2C Continuous	0.000	—	—	0.000	0
PPO Discrete	0.000	—	—	0.000	0
PPO Continuous	0.000	—	—	0.000	0
SAC Continuous	0.000	—	—	0.000	0
TD3 Continuous	0.000	—	—	0.000	0
Buy and Hold	-0.156	-5.025	-7.420	-0.943	523
Sell and Hold	-0.121	-3.892	-5.193	-0.902	523
FDRNN	-0.270	-2.482	-0.765	-1.000	2268

CHAPTER 6

The models performance have also been tested when using a different transaction cost in the reward functions than the actual transaction cost. The transaction cost values were scaled by a constant factor to determine its effect. In this experiment, the values of the constant factor were set to be 1.25, 1, 0.75 and 0.5. Figure 6.4 to Figure 6.6 shows the comparison of using different transaction cost settings in reward function for all of the datasets. In Figure 6.4 and 6.5, it shows that setting the constant factor to be 1 or 1.25 perform better than 0.75 and 0.5 in most cases, except in SAC and TD3. This shows that using a constant factor of lower than 1 can affect the models performance negatively, as the reinforcement learning agent tends to underestimate the transaction cost that was incurred and thus leads to larger losses.

Furthermore, Figure 6.5 shows that setting constant factor to be 1 in continuous action space was better than 1.25 while underperforming slightly when applied in A2C with discrete action space. In Figure 6.6, it shows that the transaction cost of GAS.CMD/USD was too high. All transaction cost settings including the use of 0.5 constant factor did not help to encourage the exploration of the reinforcement learning agent. Therefore, it can be concluded that setting the constant factor of the transaction cost to be 1 in reward function is the most suitable as it demonstrated better performance than other settings. However, setting a constant factor slightly above 1 such as 1.25 might still be useful especially when the action space was discrete. Hence more tuning can be done to further improve the model performance.

CHAPTER 7

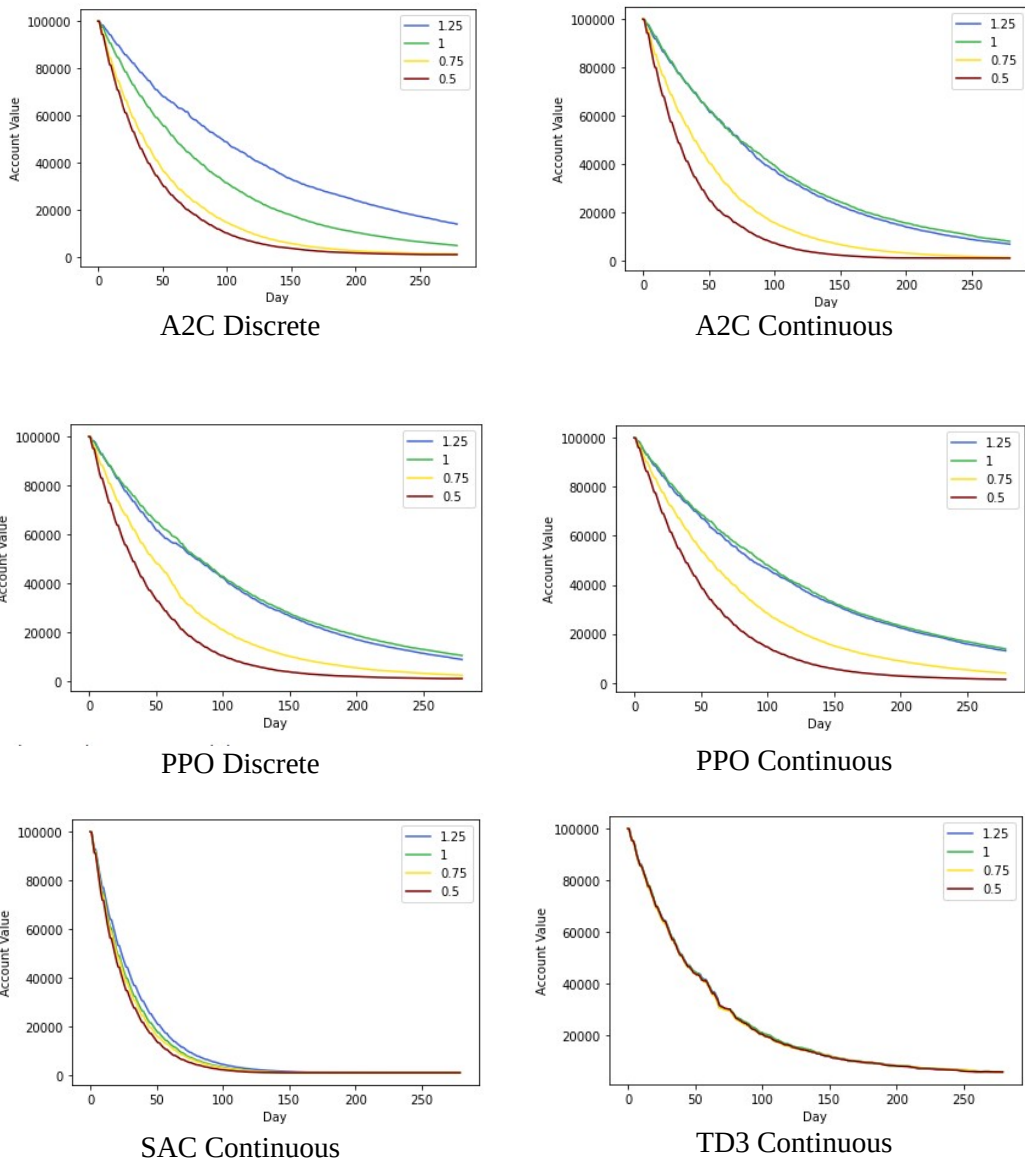


Figure 6.4 Comparison of different transaction cost settings in reward function for EUR/USD

CHAPTER 6

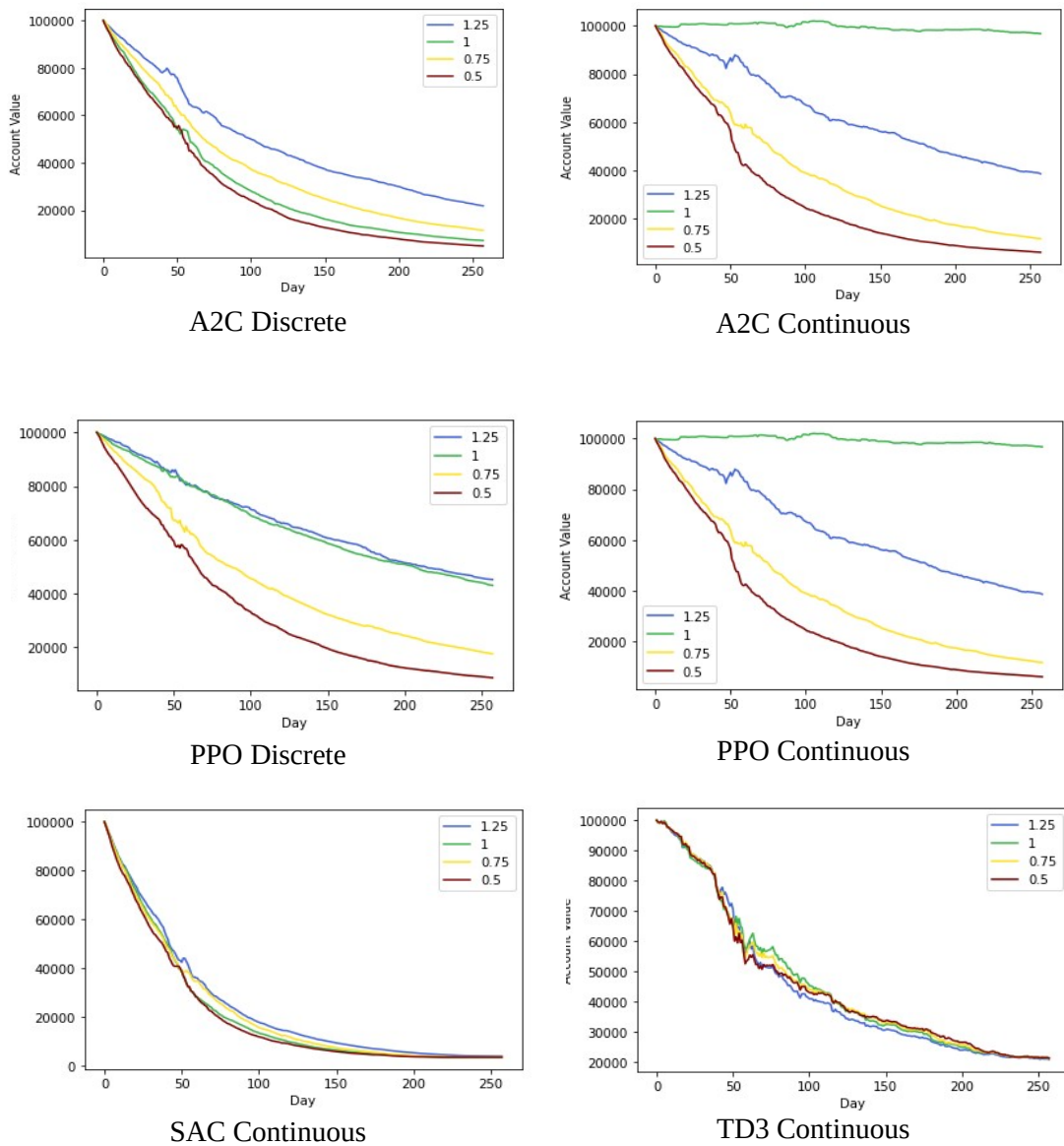


Figure 6.5 Comparison of different transaction cost settings in reward function for USA500.IDX/USD

CHAPTER 7

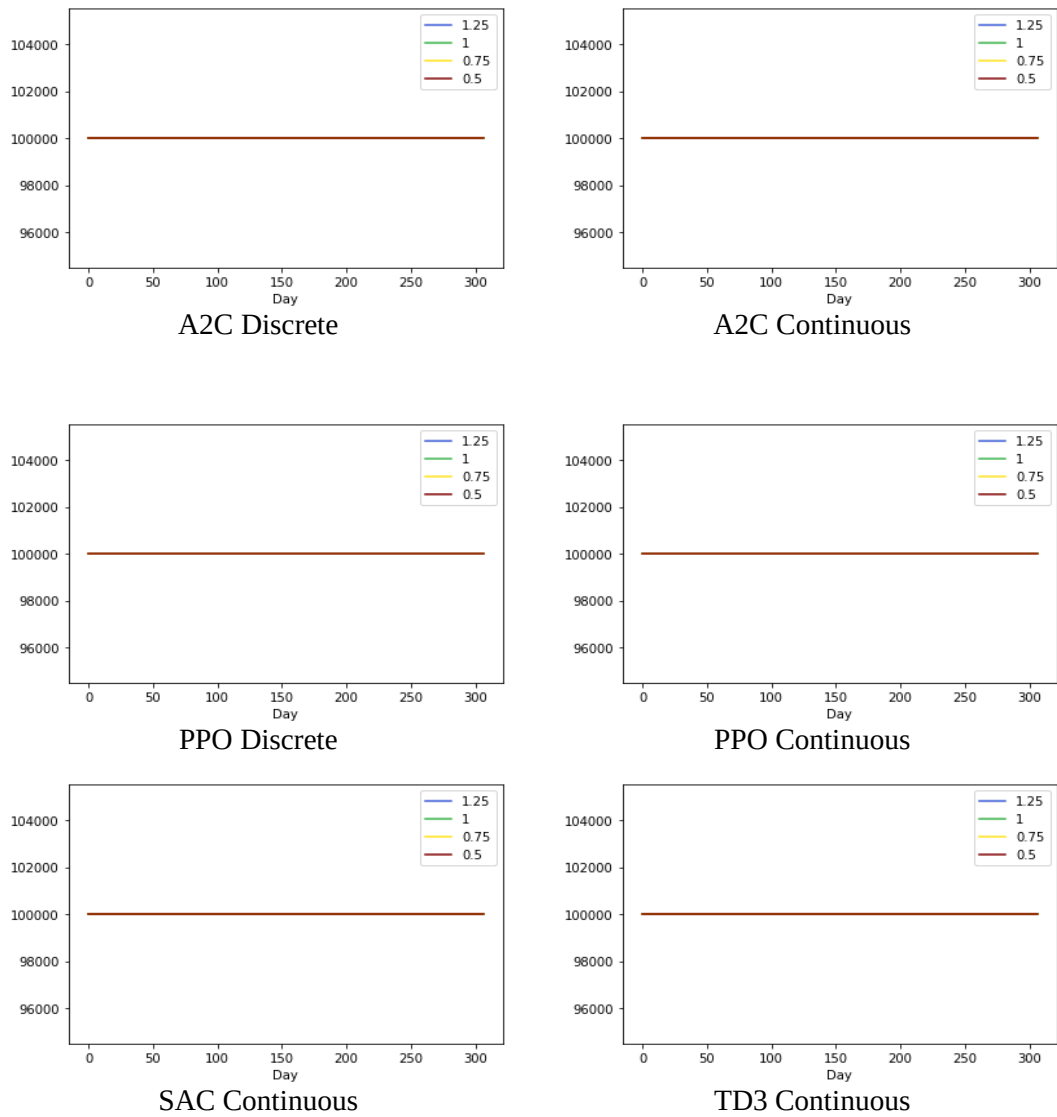


Figure 6.6 Comparison of different transaction cost settings in reward function for GAS.CMD/USD

6.2 Research Challenges

There were several challenges faced during this research. The first challenge was that the reinforcement learning agent tended to achieve local minima of not taking any position or not changing the position for a long period. The second challenge was the lack of Graphical Processing Unit (GPU) to try out a deeper neural network. The third challenge was that different datasets will produce various outcome. Hence, it was difficult to achieve consistent performance across different datasets.

Chapter 7

Conclusion

In conclusion, this research has applied various reinforcement learning algorithm including Advantage Actor Critic (A2C), Proximal Policy Optimisation (PPO), Soft Actor Critic (SAC) and Twin Delayed DDPG (TD3). A reinforcement learning model for financial trading has also been proposed. Furthermore, the effect of transaction cost in the reward function was also studied. However, the proposed model did not achieve desirable performance as the algorithms produce negative return. The main challenging that was facing is that transaction cost can cause the reinforcement learning agent to achieve a local minimum in which no trading or changing of positions occur. Furthermore, the hyperparameter tuning can be very difficult to identify as suitable range of values. It was suggested that future research can be continued to focus on resolving the local minima issues follow by improving the model performance. More research can also be done to identify a suitable framework that can achieve consistent performance for different datasets. This can greatly reduce the hyperparameter tuning to obtained the targeted performance.

REFERENCES

REFERENCES

- [1] J. D. Schwager and M. Etzkorn, *A complete guide to the futures Market, Second edition: Fundamental ANALYSIS, technical Analysis, Trading, spreads and options*. John Wiley & Sons, Inc, 2017.
- [2] P. Treleaven, M. Galas, and V. Lalchand, "Algorithmic trading review," *Communications of the ACM*, vol. 56, no. 11, pp. 76–85, 2013.
- [3] M. M. Rounaghi and F. Nassir Zadeh, "Investigation of market efficiency and financial stability between S&P 500 and London Stock Exchange: Monthly and Yearly forecasting of time Series stock returns Using ARMA model," *Physica A: Statistical Mechanics and its Applications*, vol. 456, pp. 10–21, 2016.
- [4] T. B. Trafalis and H. Ince, "Support vector machine for regression and applications to financial forecasting," *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, 2000.
- [5] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.
- [6] Researchr.org. 2022. *Stock Market Prediction with Deep Learning: A Character-based Neural Language Model for Event-based Trading - researchr publication*. [online] Available at: <<https://researchr.org/publication/PinheiroD17>> [Accessed 22 April 2022].
- [7] Z. Zhang, S. Zohren, and S. Roberts, "Deep reinforcement learning for trading," *The Journal of Financial Data Science*, vol. 2, no. 2, pp. 25–40, 2020.

REFERENCES

- [8] M. M. Rounaghi and F. Nassir Zadeh, "Investigation of market efficiency and financial stability between S&P 500 and London Stock Exchange: Monthly and Yearly forecasting of time Series stock returns Using ARMA model," *Physica A: Statistical Mechanics and its Applications*, vol. 456, pp. 10–21, 2016.
- [9] B. A. S. Reddy SK, "Exchange rate forecasting using Arima, neural network and Fuzzy Neuron," *Journal of Stock & Forex Trading*, vol. 04, no. 03, 2015.
- [10] A. Hossain, F. Zaman, M. Nasser, and M. M. Islam, "Comparison of GARCH, neural network and support vector machine in financial time Series prediction," *Lecture Notes in Computer Science*, pp. 597–602, 2009.
- [11] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.
- [12] J. Peña, P. Gutiérrez, C. Hervás-Martínez, J. Six, R. Plant, and F. López-Granados, "Object-based image classification of summer crops with machine learning methods," *Remote Sensing*, vol. 6, no. 6, pp. 5019–5041, 2014.
- [13] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.
- [14] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K.

REFERENCES

- Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, “Grandmaster level in StarCraft II Using Multi-agent reinforcement learning,” *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.
- [15] R. Liu, F. Nageotte, P. Zanne, M. de Mathelin, and B. Dresch-Langley, “Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review,” *Robotics*, vol. 10, no. 1, p. 22, 2021.
- [16] H. S. Ansari and G. R, “Autonomous driving using deep reinforcement learning in urban environment,” *International Journal of Trend in Scientific Research and Development*, vol. Volume-3, no. Issue-3, pp. 1573–1575, 2019.
- [17] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, “Deep direct reinforcement learning for financial signal representation and trading,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 653–664, 2017.
- [18] Y. Li, W. Zheng, and Z. Zheng, “Deep robust reinforcement learning for practical algorithmic trading,” *IEEE Access*, vol. 7, pp. 108014–108022, 2019.
- [19] L. Graesser and W. L. Keng, *Foundations of deep reinforcement learning: Theory and practice in python*. Boston: Addison-Wesley, 2020.
- [20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” *arXiv [cs.LG]*, 2018.
- [21] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” *arXiv [cs.AI]*, 2018.
- [22] BabyPips.com, “Heikin ashi candlestick CHART vs. Traditional Japanese candlestick chart,” *BabyPips*, 20-Apr-2021. [Online]. Available: <https://www.babypips.com/learn/forex/heikin-ashi-vs-traditional-japanese-candlestick>. [Accessed: 24-Aug-2021].

REFERENCES

- [23] J. Chen, "Technical indicator definition," *Investopedia*, 05-Aug-2021. [Online]. Available: <https://www.investopedia.com/terms/t/technicalindicator.asp>. [Accessed: 24-Aug-2021].
- [24] R. W. Colby, *The encyclopedia of technical market indicators*, 2nd ed. New York: McGraw Hill, 2003.
- [25] A. Hayes, "Bollinger Band," *Investopedia*, 28-Jul-2021. [Online]. Available: <https://www.investopedia.com/terms/b/bollingerbands.asp>. [Accessed: 24-Aug-2021].
- [26] A. Hayes, "Average true range (atr)," *Investopedia*, 28-Jul-2021. [Online]. Available: <https://www.investopedia.com/terms/a/atr.asp>. [Accessed: 24-Aug-2021].
- [27] J. Chen, "Rate of change (roc)," *Investopedia*, 19-May-2021. [Online]. Available: <https://www.investopedia.com/terms/r/rateofchange.asp>. [Accessed: 24-Aug-2021].
- [28] J. C. Francis and D. Kim, *Modern portfolio theory: Foundations, analysis, and new developments*. Hoboken, NJ: J. Wiley & Sons, 2013.
- [29] W. Kenton, "Understanding the sortino ratio," *Investopedia*, 19-May-2021. [Online]. Available: <https://www.investopedia.com/terms/s/sortinoratio.asp#:~:text=The%20Sortino%20ratio%20is%20a,standard%20deviation%20of%20portfolio%20returns>. [Accessed: 24-Aug-2021].
- [30] A. Hayes, "Maximum drawdown (MDD) Definition," *Investopedia*, 19-May-2021. [Online]. Available: <https://www.investopedia.com/terms/m/maximum-drawdown-mdd.asp>. [Accessed: 24-Aug-2021].

REFERENCES

- [31] M. Wiering and M. Otterlo, *Reinforcement learning state-of-the-art*. Berlin: Springer Berlin, 2014.
- [32] R. S. Sutton, F. Bach, and A. G. Barto, *Reinforcement learning: An introduction*. Massachusetts: MIT Press Ltd, 2018.
- [33] N. Kanwar, “Deep reinforcement learning-based portfolio management,” *Uta.edu*. [Online]. Available: <https://rc.library.uta.edu/uta-ir/bitstream/handle/10106/28108/KANWAR-THESIS-2019.pdf?sequence=1&isAllowed=y>. [Accessed: 22-Apr-2022].
- [34] Blackburn, “Reinforcement learning : Markov-decision process (part 2),” *Medium*, 25-Nov-2020. [Online]. Available: <https://towardsdatascience.com/reinforcement-learning-markov-decision-process-part-2-96837c936ec3>. [Accessed: 24-Aug-2021].
- [35] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” *arXiv [cs.AI]*, pp. 1587–1596, 10--15 Jul 2018.
- [36] L. Chen and Q. Gao, “Application of deep reinforcement learning on automated stock trading,” *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, 2019.
- [37] J. Moody and M. Saffell, “Reinforcement learning for trading,” *Advances in Neural Information Processing Systems*, vol. 11, 1998.
- [38] D. Gorse, “Application of stochastic recurrent reinforcement learning to index trading,” in *ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, ESANN, 2011.

REFERENCES

- [39] S. Carta, A. Corrigan, A. Ferreira, A. S. Podda, and D. R. Recupero, "A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning," *Applied Intelligence*, vol. 51, no. 2, pp. 889–905, 2020.
- [40] K. Lei, B. Zhang, Y. Li, M. Yang, and Y. Shen, "Time-driven feature-aware jointly deep reinforcement learning for financial signal representation and algorithmic trading," *Expert Systems with Applications*, vol. 140, p. 112872, 2020.
- [41] M. Taghian, A. Asadi, and R. Safabakhsh, "A reinforcement learning based encoder-decoder framework for learning stock trading rules," *arXiv [q-fin.ST]*, 2021.
- [42] "Historical data Feed :: Dukascopy Bank sa: Swiss Forex BANK: Ecn BROKER: Managed ACCOUNTS: Swiss FX trading platform," *Dukascopy Bank SA | Swiss Forex Bank | ECN Broker | Managed accounts | Swiss FX trading platform*. [Online]. Available: <https://www.dukascopy.com/swiss/english/marketwatch/historical/>. [Accessed: 26-Aug-2021].
- [43] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLOS ONE*, vol. 12, no. 7, 2017.
- [44] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLoS One*, vol. 12, no. 7, p. e0180944, 2017.
- [45] R. Shinde, "Understanding SEQUENTIAL/TIMESERIES data for lstm....," *Medium*, 24-Sep-2019. [Online]. Available: <https://medium.com/@raman.shinde15/understanding-sequential-timeseries-data-for-lstm-4da78021ecd7>. [Accessed: 25-Aug-2021].

REFERENCES

- [46] P. Lara-Benítez, M. Carranza-García, J. M. Luna-Romera, and J. C. Riquelme, “Temporal convolutional networks applied to energy-related time series forecasting,” *Applied Sciences*, vol. 10, no. 7, p. 2322, 2020.
- [47] D.-Y. Park and K.-H. Lee, “Practical algorithmic trading using state representation learning and imitative reinforcement learning,” *IEEE Access*, vol. 9, pp. 152310–152321, 2021.

PLAGIARISM CHECK RESULT

APPENDIX A

A.1 Biweekly Report

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3, Year 3	Study week no.: 1
Student Name & ID: Tan Li Xue 18ACB01119	
Supervisor: Ts Dr Lim Seng Poh	
Project Title: Financial Trading using Learning-based Approach	

1. WORK DONE Study more previous works to identify possible solutions
2. WORK TO BE DONE Identify a baseline method.
3. PROBLEMS ENCOUNTERED
4. SELF EVALUATION OF THE PROGRESS Currently having no issues or difficulties in programming part and can develop model without problems.



Supervisor's signature




Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3, Year 3	Study week no.: 3
Student Name & ID: Tan Li Xue 18ACB01119	
Supervisor: Ts Dr Lim Seng Poh	
Project Title: Financial Trading using Learning-based Approach	

1. WORK DONE Implemented a baseline method
2. WORK TO BE DONE Coding for proposed reinforcement learning algorithms. Identify or design a neural network model to be trained.
3. PROBLEMS ENCOUNTERED
4. SELF EVALUATION OF THE PROGRESS Currently having no issues or difficulties in programming part and can develop model without problems.



Supervisor's signature




Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3, Year 3	Study week no.: 5
Student Name & ID: Tan Li Xue 18ACB01119	
Supervisor: Ts Dr Lim Seng Poh	
Project Title: Financial Trading using Learning-based Approach	

1. WORK DONE Study existing time series neural network architecture. InceptionTime has been selected
2. WORK TO BE DONE Implement InceptionTime
3. PROBLEMS ENCOUNTERED
4. SELF EVALUATION OF THE PROGRESS Currently having no issues or difficulties in programming part and can develop model without problems.



Supervisor's signature




Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3, Year 3	Study week no.: 7
Student Name & ID: Tan Li Xue 18ACB01119	
Supervisor: Ts Dr Lim Seng Poh	
Project Title: Financial Trading using Learning-based Approach	

1. WORK DONE Coding for InceptionTime autoencoder has completed Started training reinforcement learning agent
2. WORK TO BE DONE A new algorithm Soft Actor Critic (SAC) will be implemented.
3. PROBLEMS ENCOUNTERED
4. SELF EVALUATION OF THE PROGRESS Slightly slow due to midterm test.



Supervisor's signature




Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3, Year 3	Study week no.: 9
Student Name & ID: Tan Li Xue 18ACB01119	
Supervisor: Ts Dr Lim Seng Poh	
Project Title: Financial Trading using Learning-based Approach	

1. WORK DONE Algorithm SAC has been implemented.
2. WORK TO BE DONE Tuning hyperparameter to prevent reinforcement learning agent from achieveing local minima. Report writing will be started.
3. PROBLEMS ENCOUNTERED Current models do not show good performance.
4. SELF EVALUATION OF THE PROGRESS



Supervisor's signature




Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3, Year 3	Study week no.: 11
Student Name & ID: Tan Li Xue 18ACB01119	
Supervisor: Ts Dr Lim Seng Poh	
Project Title: Financial Trading using Learning-based Approach	

1. WORK DONE Completing report writing
2. WORK TO BE DONE Complete the final report.
3. PROBLEMS ENCOUNTERED Model does not perform well
4. SELF EVALUATION OF THE PROGRESS



Supervisor's signature



Student's signature

A.2 Poster

**Financial Trading
Using Learning-Based
Approach**



By: Tan Li Xue

Introduction
Financial trading has always been a hot topic. Recent advancement in reinforcement learning opens up new opportunity and method to trade automatically

Methods

- Use state-of-the-art reinforcement learning algorithm
- CNN-LSTM Autoencoders for time series feature extraction
- Use data of multiple time intervals

Discussion

- Transaction cost can affect the models' performance
- Using different algorithms or action space show significantly different performance

Conclusion

Reinforcement learning can be applied in financial trading but transaction cost poses a big challenge

thank you!

PLAGIARISM CHECK RESULT

Financial Trading Using Learning-based Approach

ORIGINALITY REPORT

8%

SIMILARITY INDEX

5%

INTERNET SOURCES

5%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1

www.rijadeja.com

Internet Source

<1%

2

Submitted to Imperial College of Science,
Technology and Medicine

Student Paper

<1%

3

Pedro Lara-Benítez, Manuel Carranza-García,
José M. Luna-Romera, José C. Riquelme.
"Temporal Convolutional Networks Applied to
Energy-Related Time Series Forecasting",
Applied Sciences, 2020

Publication

<1%

4

arxiv.org

Internet Source

<1%

5

Li Zhu, F. Richard Yu, Bin Ning, Tao Tang.
"Optimal Charging Control for Plug-in Electric
Vehicles in Smart Microgrids Fueled by
Renewable Energy Sources", International
Journal of Green Energy, 2013

Publication

<1%

6

repository.president.ac.id

Internet Source

<1%

PLAGIARISM CHECK RESULT

7	Huang, S.I.. "Using temporal-difference learning for multi-agent bargaining", Electronic Commerce Research and Applications, 200824 Publication	<1 %
8	dokumen.pub Internet Source	<1 %
9	edu.icourban.com Internet Source	<1 %
10	www.mdpi.com Internet Source	<1 %
11	mtc-m21d.sid.inpe.br Internet Source	<1 %
12	Jupiter Bakakeu, Franziska Schafer, Joerg Franke, Schirin Baer, Hans-Henning Klos, Joern Peschke. "Reasoning over OPC UA Information Models using Graph Embedding and Reinforcement Learning", 2020 Third International Conference on Artificial Intelligence for Industries (AI4I), 2020 Publication	<1 %
13	Submitted to University of Westminster Student Paper	<1 %
14	www.ijrte.org Internet Source	<1 %
	deepai.org	

PLAGIARISM CHECK RESULT

15	Internet Source	<1 %
16	inechain.com Internet Source	<1 %
17	Submitted to National University of Ireland, Galway Student Paper	<1 %
18	Jiahang Li, Junhui Zhao, Xiaoke Sun. "Deep Reinforcement Learning Based Wireless Resource Allocation for V2X Communications", 2021 13th International Conference on Wireless Communications and Signal Processing (WCSP), 2021 Publication	<1 %
19	P Gomide, R L Milidiu. "Assessing Stock Market Time Series Predictors Quality through a Pairs Trading System", 2010 Eleventh Brazilian Symposium on Neural Networks, 2010 Publication	<1 %
20	Submitted to National University of Singapore Student Paper	<1 %
21	R. Bisoi, P. K. Dash, V. Padhee, M. H. Naeem. "Mining of electricity prices in energy markets using a computationally efficient neural network", 2011 International Conference on Energy, Automation and Signal, 2011	<1 %

PLAGIARISM CHECK RESULT

Publication		
22	Submitted to University of Edinburgh Student Paper	<1 %
23	aclweb.org Internet Source	<1 %
24	ocglycre.no-ip.org Internet Source	<1 %
25	vtext.valdosta.edu Internet Source	<1 %
26	K.R. Williams, R.S. Muller. "Etch rates for micromachining processing", Journal of Microelectromechanical Systems, 1996 Publication	<1 %
27	Submitted to University of Melbourne Student Paper	<1 %
28	downloads.hindawi.com Internet Source	<1 %
29	unrealengine.com Internet Source	<1 %
30	www.forexlive.com Internet Source	<1 %
31	Jagadheesh Samala, Harshvardhan Takawale, Yash Chokhani, P. Veda Bhanu, Soumya J. "Fault-Tolerant Routing Algorithm for Mesh based NoC using Reinforcement Learning",	<1 %

PLAGIARISM CHECK RESULT

2020 24th International Symposium on VLSI
Design and Test (VDATE), 2020

Publication

32	Submitted to Oregon State University Student Paper	<1 %
33	pypi.org Internet Source	<1 %
34	Submitted to Emirates College of Technology Student Paper	<1 %
35	Ishan Budhiraja, Neeraj Kumar, Sudhanshu Tyagi. "Deep Reinforcement Learning Based Proportional Fair Scheduling Control Scheme for Underlay D2D Communication", IEEE Internet of Things Journal, 2020 Publication	<1 %
36	Ngoc Duy Nguyen, Thanh Nguyen, Saeid Nahavandi. "System Design Perspective for Human-Level Agents Using Deep Reinforcement Learning: A Survey", IEEE Access, 2017 Publication	<1 %
37	Yang Cao, Shao-Yu Lien, Ying-Chang Liang, Kwang-Cheng Chen, Xuemin Shen. "User Access Control in Open Radio Access Networks: A Federated Deep Reinforcement Learning Approach", IEEE Transactions on Wireless Communications, 2021 Publication	<1 %

PLAGIARISM CHECK RESULT

38	Yang Li, Wanshan Zheng, Zibin Zheng. "Deep Robust Reinforcement Learning for Practical Algorithmic Trading", IEEE Access, 2019 Publication	<1 %
39	d-scholarship.pitt.edu Internet Source	<1 %
40	diginole.lib.fsu.edu Internet Source	<1 %
41	doaj.org Internet Source	<1 %
42	mobt3ath.com Internet Source	<1 %
43	threadreaderapp.com Internet Source	<1 %
44	www.om.evaf.vu.lt Internet Source	<1 %
45	Submitted to University of South Africa Student Paper	<1 %
46	baadalsg.inflibnet.ac.in Internet Source	<1 %
47	docs.microsoft.com Internet Source	<1 %
48	"Machine Learning in Medical Imaging", Springer Science and Business Media LLC,	<1 %

PLAGIARISM CHECK RESULT

2019

Publication

49 "Recent Advances in Reinforcement Learning", Springer Science and Business Media LLC, 1996 <1 %

Publication

50 Borja Fernandez-Gauna, Igor Ansoategui, Ismael Etxeberria-Agiriano, Manuel Graña. "Chapter 46 An Empirical Study of Actor-Critic Methods for Feedback Controllers of Ball-Screw Drivers", Springer Science and Business Media LLC, 2013 <1 %

Publication

51 Elisabeth Roesch, Christopher Rackauckas, Michael P. H. Stumpf. "Collocation based training of neural ordinary differential equations", Statistical Applications in Genetics and Molecular Biology, 2021 <1 %

Publication

52 Jianyu Chen, Bodi Yuan, Masayoshi Tomizuka. "Model-free Deep Reinforcement Learning for Urban Autonomous Driving", 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019 <1 %

Publication

53 Lecture Notes in Computer Science, 2013. <1 %

Publication

PLAGIARISM CHECK RESULT

54	Submitted to Leiden University Student Paper	<1 %
55	Yitao Qiu, Rongkai Liu, Raymond S T Lee. "The Design and Implementation of Quantum Finance-based Hybrid Deep Reinforcement Learning Portfolio Investment System", Journal of Physics: Conference Series, 2021 Publication	<1 %
56	archive.org Internet Source	<1 %
57	bspace.buid.ac.ae Internet Source	<1 %
58	d-nb.info Internet Source	<1 %
59	digitalcommons.fiu.edu Internet Source	<1 %
60	hal.archives-ouvertes.fr Internet Source	<1 %
61	it.talend.com Internet Source	<1 %
62	journals.ut.ac.ir Internet Source	<1 %
63	levelup.gitconnected.com Internet Source	<1 %

PLAGIARISM CHECK RESULT

64	onlineresource.ucsy.edu.mm Internet Source	<1 %
65	repository.ntu.edu.sg Internet Source	<1 %
66	vtechworks.lib.vt.edu Internet Source	<1 %
67	whats-binary-options.logdown.com Internet Source	<1 %
68	worldwidescience.org Internet Source	<1 %
69	www.econstor.eu Internet Source	<1 %
70	www.packtpub.com Internet Source	<1 %
71	www.science.gov Internet Source	<1 %

PLAGIARISM CHECK RESULT

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

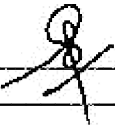
Full Name(s) of Candidate(s)	Tan Li Xue
ID Number(s)	18ACB01119
Programme / Course	Bachelor of Computer Science
Title of Final Year Project	Financial Trading Using Learning-based Approach

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: 8 % Similarity by source Internet Sources: 5% Publications: 5% Student Papers: 2%	Ok.
Number of individual sources listed of more than 3% similarity: 0	Ok.
Parameters of originality required and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

PLAGIARISM CHECK RESULT

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above



Signature of Supervisor

Name: Ts Dr Lim Seng Poh

Date: 22/04/2022

Signature of Co-Supervisor

Name: _____

Date: 22/04/2022

PLAGIARISM CHECK RESULT



UNIVERSITI TUNKU ABDUL RAHMAN

**FACULTY OF INFORMATION & COMMUNICATION
TECHNOLOGY (KAMPAR CAMPUS)**

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	18ACB01119
Student Name	Tan Li Xue
Supervisor Name	Ts Dr Lim Seng Poh

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Plastic Cover (for hardcopy)
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract

PLAGIARISM CHECK RESULT

√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.



(Signature of Student)

Date: 22/04/2022

PLAGIARISM CHECK RESULT