



**TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS**

THESIS NO: 075/MSICE/009

“Design of Stock Trading Agent Using Deep Reinforcement Learning ”

by

Janak Kumar Lal

A THESIS REPORT

**SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND
COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF MASTER OF
SCIENCE IN INFORMATION AND COMMUNICATION
ENGINEERING**

**DEPARTMENT OF ELECTRONICS AND COMPUTER
ENGINEERING
LALITPUR, NEPAL**

September, 2022

Design of Stock Trading Agent Using Deep Reinforcement Learning

by

Janak Kumar Lal

075MSICE009

Thesis Supervisor

Arun K. Timalina Ph.D

Associate Professor, Department of Electronics and Computer Engineering,
Pulchowk Campus

A final thesis report submitted in partial fulfillment of the requirements for the
degree of Masters of Science in Information and Communication
Engineering

Department of Electronics and Computer Engineering

Institute of Engineering, Pulchowk Campus

Tribhuvan University

Lalitpur, Nepal

September, 2022

COPYRIGHT©

The author has agreed that the library, Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk Campus, may make this thesis freely available for inspection. Moreover the author has agreed that the permission for extensive copying of this thesis work for scholarly purpose may be granted by the professor(s), who supervised the thesis work recorded herein or, in their absence, by the Head of the Department, wherein this thesis was done. It is understood that the recognition will be given to the author of this thesis and to the Department of Electronics and Computer Engineering, Pulchowk Campus in any use of the material of this project. Copying of publication or other use of this thesis for financial gain without approval of the Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk Campus and author's written permission is prohibited.

Request for permission to copy or to make any use of the material in this thesis in whole or part should be addressed to:

Head of Department,
Department of Electronics and Computer Engineering,
Institute of Engineering, Pulchowk Campus
Pulchowk, Lalitpur, Nepal

DECLARATION

I declare that the work hereby submitted for Master of Science in Information and Communication Engineering (MSICE) at IOE, Pulchowk Campus entitled **“Design of Stock Trading Agent Using Deep Reinforcement Learning ”** is my own work and has not been previously submitted by me at any university for any academic award.

I authorize IOE, Pulchowk Campus to lend this thesis to other institution or individuals for the purpose of scholarly research.

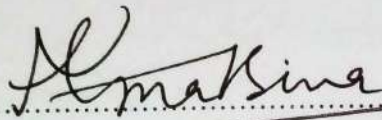
Janak Kumar Lal

075MSICE009

September, 2022

RECOMMENDATION

The undersigned certify that they have read and recommended to the Department of Electronics and Computer Engineering for acceptance, a thesis entitled “**Design of Stock Trading Agent Using Deep Reinforcement Learning**”, submitted by **Janak Kumar Lal** in partial fulfillment of the requirement for the award of the degree of “**Master of Science in Information and Communication Engineering**”.

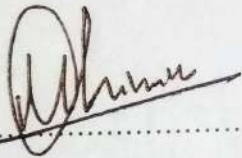


Supervisor: Arun K. Timalina, Ph.D.

Associate Professor,

Department of Electronics and Computer Engineering,

Pulchowk Campus, Institute of Engineering, Tribhuvan University.



External Examiner: Manoj Ghimire

Co-Founder/CEO, Raralabs, Lalitpur



Committee Chairperson: Babu R Dawadi, Ph.D.

Program Coordinator,

MSc in Information and Communication Engineering,

Department of Electronics and Computer Engineering,

Institute of Engineering, Tribhuvan University.

Date: September, 2022

त्रिभुवन विश्वविद्यालय
Tribhuvan University
इन्जिनियरिङ्ग अध्ययन संस्थान
Institute of Engineering

पुल्चोक क्याम्पस

Pulchowk Campus

इलेक्ट्रॉनिक्स तथा कम्प्युटर इन्जिनियरिङ्ग विभाग

Department of Electronics and Computer Engineering

553 4070
552 1260
Extn: 315

Fax: 977 1 555 3946

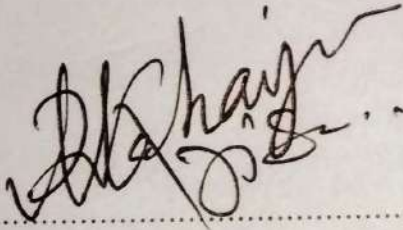
email: ece@ioe.edu.np

<http://doece.pcampus.edu.np>

Pulchowk, Lalitpur
P.O. box-1175

DEPARTMENTAL ACCEPTANCE

The thesis entitled "Design of Stock Trading Agent Using Deep Reinforcement Learning", submitted by Janak Kumar Lal in partial fulfillment of the requirement for the award of the degree of "Master of Science in Information and Communication Engineering" has been accepted as a bonafide record of work independently carried out by his/her in the department.



Prof. Dr. Ram Krishna Maharjan

Head of the Department

Department of Electronics and Computer Engineering,

Pulchowk Campus,

Institute of Engineering,

Tribhuvan University,

Nepal.

ACKNOWLEDGEMENT

I would like to thank **Arun K. Timalisina, Ph.D.** for his encouragement and precious guidance during this thesis. I would also like to pay my sincere thanks to our program coordinator **Babu R Dawadi, Ph.D.** and the Department of Electronics and Computer Engineering for guiding in this thesis.

Janak Kumar Lal

075MSICE009

ABSTRACT

This study adopts Double Deep Q learning algorithm to design trading strategies to trade stocks of four commercial banks listed in NEPSE. The reinforcement learning agent takes discrete actions and gets negative or positive reward from the environment. CNN is utilized to form the policy network. A target network is used to mitigate instability due to Deep Q Network. The concept of experience replay is used to randomly sample the batches of experience from the memory and train the network. The performance of Double Deep Q learning agent was compared with various baseline trading strategies in terms of annualised expected trade return. The maximum annualised expected trade return obtained with traditional baseline methods was 103% for testing data of NABIL, while for the same data the reinforcement learning agent using double deep Q learning algorithm obtained annualized expected trade return of 114.44%. The experiments showed that, Double Deep Q learning agent with experience replay had higher annualised expected trade return compared to baseline trading strategies.

Keywords: Reinforcement learning, Double Deep Q-Learning, CNN, NEPSE

TABLE OF CONTENTS

COPYRIGHT	iii
DECLARATION	iv
RECOMMENDATION	v
DEPARTMENTAL ACCEPTANCE	vi
ACKNOWLEDGEMENT	vii
ABSTRACT	viii
TABLE OF CONTENTS	ix
LIST OF FIGURES	xi
LIST OF TABLES	xiii
LIST OF ABBREVIATIONS	xv
1 INTRODUCTION	1
1.1 Background	1
1.2 Problem definition	2
1.3 Objectives	2
1.4 Scope and limitations	3
1.5 Organization of the project report	3
2 LITERATURE REVIEW	4
3 THEORETICAL BACKGROUND	7
3.1 Technical Indicators	7
3.1.1 Stochastic Oscillator	7
3.1.2 On Balance Volume	8
3.1.3 Moving Average Convergence Divergence	8
3.1.4 Average Directional Moving Index	9
3.1.5 Japanese Candlestick	10
3.2 Reinforcement Learning	12
3.2.1 Q-Learning	13
3.2.2 Deep Q-Learning	14
3.2.3 Double Deep Q-Networks	15
4 METHODOLOGY	17

4.1	Framework	17
4.2	Data samples preparation	19
4.3	Experimental tools and techniques	20
4.4	Stock Trading Environments	20
4.5	Double Deep Q Networks	22
4.5.1	Type I	22
4.5.2	Type II	23
4.5.3	Type III	23
5	RESULTS AND ANALYSIS	24
5.1	ADBL	24
5.2	CZBIL	28
5.3	LBL	31
5.4	NABIL	35
6	Conclusion and Future Work	39
6.1	Summary	39
6.2	Conclusion	40
6.3	Future Work	40
	REFERENCES	43

LIST OF FIGURES

3.1	Japanese Candlestick	10
4.1	Double Deep Q Learning Framework.	17
5.1	Closing price of ADBL stock for training data and test data respectively.	24
5.2	Net worth of deep Q learning agent for ADBL over 1000 episodes in L_CNN_5, L_CNN_25 and L_CNN_50 experiments for respectively	26
5.3	Net worth of deep Q learning agent for ADBL over 1000 episodes in M_CNN_5, M_CNN_25 and M_CNN_50 experiments respectively	26
5.4	Net worth of deep Q learning agent for ADBL over 1000 episodes in S_CNN_5, S_CNN_25 and S_CNN_50 experiments respectively	26
5.5	Closing price of CZBIL stock for training data and test data respectively.	28
5.6	Net worth of deep Q learning agent for CZBIL over 1000 episodes in L_CNN_5, L_CNN_25 and L_CNN_50 experiments for respectively	29
5.7	Net worth of deep Q learning agent for CZBIL over 1000 episodes in M_CNN_5, M_CNN_25 and M_CNN_50 experiments respectively	30
5.8	Net worth of deep Q learning agent for CZBIL over 1000 episodes in S_CNN_5, S_CNN_25 and S_CNN_50 experiments respectively	30
5.9	Closing price of LBL stock for training data and test data respectively.	31
5.10	Net worth of deep Q learning agent for LBL over 1000 episodes in L_CNN_5, L_CNN_25 and L_CNN_50 experiments for respectively	33
5.11	Net worth of deep Q learning agent for LBL over 1000 episodes in M_CNN_5, M_CNN_25 and M_CNN_50 experiments respectively	33
5.12	Net worth of deep Q learning agent for LBL over 1000 episodes in S_CNN_5, S_CNN_25 and S_CNN_50 experiments respectively	33
5.13	Closing price of NABIL stock for training data and test data respectively.	35

5.14	Net worth of deep Q learning agent for NABIL over 1000 episodes in L_CNN_5, L_CNN_25 and L_CNN_50 experiments for respectively	36
5.15	Net worth of deep Q learning agent for NABIL over 1000 episodes in M_CNN_5, M_CNN_25 and M_CNN_50 experiments respectively	37
5.16	Net worth of deep Q learning agent for NABIL over 1000 episodes in S_CNN_5, S_CNN_25 and S_CNN_50 experiments respectively	37

LIST OF TABLES

4.1	Programming language and libraries used in this study.	20
4.2	Architecture of Type I policy networks.	22
4.3	Architecture of Type II policy networks.	23
4.4	Architecture of Type II policy networks.	23
5.1	Summary of the training data of ADBL.	25
5.2	Summary of the test data of ADBL.	25
5.3	Comparison of performance of various trading methods on ADBL training data.	25
5.4	Comparison of performance of various trading methods on test data for ADBL.	27
5.5	Comparison of performance of various Double Deep-Q networks on test data for ADBL in terms of latency.	27
5.6	Summary of the training data of CZBIL.	28
5.7	Summary of the test data of CZBIL.	29
5.8	Comparison of performance of various trading methods on CZBIL training data.	29
5.9	Comparison of performance of various trading methods on test data for CZBIL.	30
5.10	Comparison of performance of various Double Deep-Q networks on test data for CZBIL in terms of latency.	31
5.11	Summary of the training data of LBL.	32
5.12	Summary of the test data of LBL.	32
5.13	Comparison of performance of various trading methods on LBL training data.	32
5.14	Comparison of performance of various trading methods on test data for LBL.	33
5.15	Comparison of performance of various Double Deep-Q networks on test data for LBL in terms of latency.	34

5.16	Summary of the training data of NABIL.	35
5.17	Summary of the test data of NABIL.	36
5.18	Comparison of performance of various trading methods on NABIL training data.	36
5.19	Comparison of performance of various trading methods on test data for NABIL.	37
5.20	Comparison of performance of various Double Deep-Q networks on test data for NABIL in terms of latency.	38

LIST OF ABBREVIATIONS

2D	:Two Dimensional
ADBL	:AGRICULTURE DEVELOPMENT BANK LIMITED
ADX	:Average Directional Index
CNN	:Convolutional Neural Network
CZBIL	:CITIZEN BANK INTERNATIONAL LIMITED
EMA	:Exponential Moving Average
GA	:Genetic Algorithm
LBL	:LAXMI BANK LIMITED
MA	:Moving Average
MACD	:Moving Average Convergence Divergence
ML	:Machine Learning
NABIL	:NABIL BANK LIMITED
NEPSE	:Nepal Stock Exchange
OBV	:On Balance Volume
RL	:Reinforcement Learning
SVM	:Support Vector Machine
TD	:Temporal Difference
VAR	:Value at Risk

CHAPTER 1

INTRODUCTION

1.1 Background

Malcom Gladwell in his book “Outliers: The Story of Success” [1] writes that it takes 10,000 hours of intensive practice to achieve mastery of complex skills. Can an amateur violinist be an expert by playing the same song for 10,000 hours? Does a person who has tossed an unbiased coin 10,000 times predict the next outcome more accurately than a person who has not tossed a coin? In case of tossing an unbiased coin no matter how many times a person has tossed a coin the outcome of the next toss will be random and unpredictable. In the case of a violinist, after playing the same song on the violin for 10,000 hours might give the violinist an edge over the amateur but won’t make him/her an expert. To be an expert violinist, he/she might need other factors such as regular feedback and deliberate practice. The regular feedback will help the violinist to know whether he/she is going in the right direction or not. While practicing different types of songs at different tempos will help to enhance the skill of a violinist. In general, a normal human being needs a valid environment, multiple repetitions, timely feedback from an expert and deliberate practice, to be an expert in a skill. The goal of this research is to close the knowledge gap between computer science and finance by evaluating if an RL system using a CNN can outperform standard trading methods that use a variety of technical indicators and charts. A policy function for each state in the space of trading parameters is approximated using Q-learning, a sort of temporal difference reinforcement learning.

A valid stock trading environment was created so that agent gets necessary information and feedback to learn trading strategy. This environment assisted the agent to learn the patterns embedded inside the data of the stock market. The agent uses CNN for learning the trading pattern. The agent takes the decisions regarding the buying, selling and holding stocks after observing the values provided by the environment. The environment also provides the agent with positive or negative feedback, so that agent will know whether its action was good or bad. The concepts of experience replay and target network was used in the experiments carried out in this work. The experience gained by the agent is stored in the memory buffer. After the buffer is full, batches of experience are randomly sampled from the experience and used to train the network. A target network has similar architecture and parameters as the policy network. The weights of the target network gets updated after certain number of episodes. This help to maintain stability in the training process.

1.2 Problem definition

Stock trading is the act of purchasing and selling a company's stock on a financial market. The goal is to maximize the return on capital by taking advantage of market volatility through a series of buy and sell orders. New design of trading mechanisms will be effective trading mechanisms, only when it generates positive returns in all market conditions.

1.3 Objectives

The objective of this thesis study is to design a deep reinforcement learning agent which will take proper actions to maximize portfolio and gain positive returns in all market conditions with the help of double deep Q-learning.

1.4 Scope and limitations

The data of four commercial banks listed in NEPSE are used in this study. Limitations of this study are as follows:

- The commission of the broker has not been considered while trading the stock.
- All the experiments are under the assumption that there is no delay in buying a stock and adding it in agent's portfolio.
- The dividends and stock split are not taken in account while conducting this experiment.

1.5 Organization of the project report

The report is organized as follow:

Chapter 2: Available literature on application of reinforcement learning across various domain.

Chapter 3: Theoretical background of technical indicators and reinforcement learning.

Chapter 4: The framework used to train the agent is discussed in this chapter. This chapter also includes the design of stock trading environment and double deep-Q networks.

Chapter 5: The results obtained from experiments conducted on given data set in stock trading environment designed in Chapter 4 with various double deep-Q networks are presented in this chapter.

Chapter 6: The last chapter concludes findings of this study and discusses extensions of this work.

CHAPTER 2

LITERATURE REVIEW

Signals generated by technical indicators are used traditionally by the stock traders to take trading decisions. With the advent of technology and the possibility of online trading several algorithms have been developed to make trading decisions as well as to design trading bots. Genetic algorithms have been used to formulate stock trading rules [2] [3] [4] . Support vector machines along with genetic algorithms were used [5] to optimize investment in foreign markets with high investments. The SVM categorizes the market into three different types. The GA optimizes an investment strategy with dynamic approaches.

A framework for stock market trading using data analytics and business intelligence was proposed by [6], analyzed the intraday trading strategy employing the concept of Bollinger Bands to identify stocks that generate maximum profit.

Reinforcement learning delivers decisions. By creating a simulation of an entire business or system, it becomes possible for an intelligent system to test new actions or approaches. The intelligent agent learns from its failures and successes. Much in the way human beings can develop a skill as they practice it, reinforcement learning only becomes more powerful when it's executed at scale.

One of the early breakthroughs in reinforcement learning was the development of an off policy TD control algorithm known as Q-learning [7], defined by

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (2.1)$$

Q-learning has been used in robotics [8] [9] [10]. Fluid dynamics is another field where Q-learning finds its applications [11].

Q-learning has been used by [12] for trading stocks. The value functions subjected to optimization at each time step employed interval profit, sharpe ratio, and derivative of sharpe ratio. Their findings imply that choosing the right value function is critical for achieving a robust learning algorithm in value iteration, especially when dealing with noisy data.

Multiple Q-learning agents are used by [13], allowing agents to successfully divide and conquer the stock trading problem by defining appropriate roles for jointly carrying out stock price and selection judgments. This work also proposes a representation scheme that may simply explain the history of price movements, in an attempt to overcome the complexity issue when examining a huge quantity of data to establish long-term dependence among stock prices. The suggested trading framework beats those educated by other alternative methodologies in terms of profit and risk management, according to experimental results on a Korean stock market.

One of the major developments in reinforcement learning is in the field of gaming. Using only raw pixels as input, [14] proposed a new deep learning model for reinforcement learning and demonstrated its capacity to master challenging control policies for Atari 2600 computer games. This research also demonstrated an online Q-learning variation that combines stochastic minibatch updates with experience replay memory to make deep network training for RL easier. With no changes to the architecture or hyperparameters, this method produced state-of-the-art outcomes in six of the seven games it was tested on.

A different target network Q' was used by [15] to mitigate the instability of deep Q-network where a small change in the Q-network might result in large change in present Q value. A trading system that can predict the number of shares to trade was proposed by [16]. An automated method was created that predicts the number of shares by merging reinforcement learning and a deep neural network (DNN) regressor to a deep Q-network.

Japanese candlestick images were fed to to CNN which was utilized within Double Deep Q-network [17]. This paper also demonstrated the use of feature map to learn what is input images caused the agent to take the particular action.

The advantage actor critic trading algorithm has been created by [18] which proves to be a profitable and attractive method for investment. This algorithm achieved the maximum profit of 110 % per annum without accounting for commission. The researchers of [18] concluded that the use of recurrent layer and dropout layer is fruitful while developing the deep reinforcement learning network.

Inspired by the idea proposed in [14] deep Q learning has also been used in wireless edge computing [19] and in energy management of hybrid electric vehicles [20] [21].

Various study has been conducted in NEPSE. Most of these studies focuses on forecasting. [22] used Arima model to predict NEPSE index, geometric brownian was used to forecast stock market [23], CNN-BGRU method was used to forecast the stock price of commercial banks listed in NEPSE [24]. The assets of NEPSE are used to form the portfolio in [25], and the mean-variance framework based on contemporary portfolio theory is used to determine the ideal weight to be given to each asset in the portfolio.

CHAPTER 3

THEORETICAL BACKGROUND

3.1 Technical Indicators

Technical indicators, which are utilized by traders who follow technical analysis, are heuristic or pattern-based indications generated by the price, volume, and/or open interest of an asset or contract. Technical analysts utilize indicators to forecast future price changes by examining historical data.

3.1.1 Stochastic Oscillator

The Stochastic Oscillator, created by George C. Lane in the late 1950s, is a momentum indicator that displays where the close falls in relation to the high-low range over a predetermined number of periods. The stochastic oscillator adheres to the rate or momentum of price. Typically, the momentum shifts before the price. The Stochastic Oscillator's bullish and bearish divergences can therefore be used to predict reversals. The first and most significant signal that Lane noticed was this one. Lane additionally used this oscillator to spot bullish and bearish setups in order to foresee a potential reversal. The Stochastic Oscillator is also helpful because it is range-bound.

The Formula for the Stochastic Oscillator is

$$\%k = \frac{C - L14}{H14 - L14} \times 100 \quad (3.1)$$

where: C = The most recent closing price

L14 = The lowest price traded of the 14 previous trading sessions

H14 = The highest price traded during the same 14-day period

%K = The current value of the stochastic indicator

Notably, %K is referred to sometimes as the fast stochastic indicator. The "slow" stochastic indicator is taken as %D = 3-period moving average of %K.

The overbought signal is generated when the difference between %k and %d is negative and the highest value of %k is greater than 80. This is the signal for the agent to sell the stock. The oversold signal is generated when the difference between %k and %d is positive and the lowest value of %k is less than 20. This is the signal for the agent to buy the stock.

3.1.2 On Balance Volume

Another metric, OBV [26] is considered as it explain the significant changes in market based on changes in volume. On-balance volume gives a running total of the trading activity in an asset and shows whether this activity is coming into or leaving a certain security or currency pair. The OBV is the sum of all volumes (positive and negative). There are three rules implemented when calculating the OBV.

They are:

1. If today's closing price is higher than yesterday's closing price, then: Current OBV = Previous OBV + today's volume
2. If today's closing price is lower than yesterday's closing price, then: Current OBV = Previous OBV - today's volume
3. If today's closing price equals yesterday's closing price, then: Current OBV = Previous OBV

The buy signal is generated when the current value of OBV is greater than the last value of OBV. The sell signal is generated when the current value of OBV is less than the last value of OBV

3.1.3 Moving Average Convergence Divergence

In the 1960s, Gerald Appel created this indication. The 26-period exponential moving average (EMA) is subtracted from the 12-period EMA to calculate moving average convergence divergence (MACD). The 12-period EMA (represented by the red line on the price chart) has a positive value (represented by the blue line in the bottom chart) if it is above the 26-period EMA (represented by the blue line in the price chart), and a negative value (represented by the red line) whenever it

is below. The MACD's greater range above or lower than its baseline denotes a widening gap between the two EMAs. By taking the MACD's nine-period moving average, a signal line, sometimes referred to as the trigger line, is produced. An agent gets a buy signal when the current difference between MACD and MACD signal is positive while the previous difference between MACD and MACD signal is negative. An agent gets a sell signal when the current difference between MACD and MACD signal is negative while the previous difference between MACD and MACD signal was positive.

3.1.4 Average Directional Moving Index

A set of directional movement indicators that make up a trading system created by Welles Wilder include the Average Directional Index (ADX), Minus Directional Indicator (-DI), and Plus Directional Indicator (+DI). Although Wilder's Directional Movement System was created with commodities and daily prices in mind, stocks can also benefit from similar indicators. The Directional Movement System's foundation is made up of positive and negative directional movement. By comparing the difference between two consecutive lows with the difference between their corresponding highs, Wilder was able to establish direction of movement. These disparities' smoothed averages are used to create the Plus Directional Indicator (+DI) and Minus Directional Indicator (-DI), which measure trend direction over time. The Directional Movement Indicator refers to these two indicators taken together (DMI). The Average Directional Index (ADX) is in turn derived from the smoothed averages of the difference between +DI and -DI; it measures the strength of the trend (regardless of direction) over time. A buy signal is generated when the current difference between +DI and -DI is positive and the previous difference between +DI and -DI was negative while the average is greater than 20. A sell signal is generated when the current difference between +DI and -DI is negative and the previous difference between +DI and -DI was positive while the average is greater than 20.

3.1.5 Japanese Candlestick

A Japanese candlestick is a type of price chart that shows the opening, closing, high and low price points for each given period. It was invented by Japanese rice merchants centuries ago, and popularized among Western traders by a broker called Steve Nison [27]. Today, Japanese candlestick charts are the most popular way to quickly analyze price action, particularly with technical traders. They offer much more information visually than traditional line charts, showing a market's highest point, lowest point, opening price and closing price at a glance.

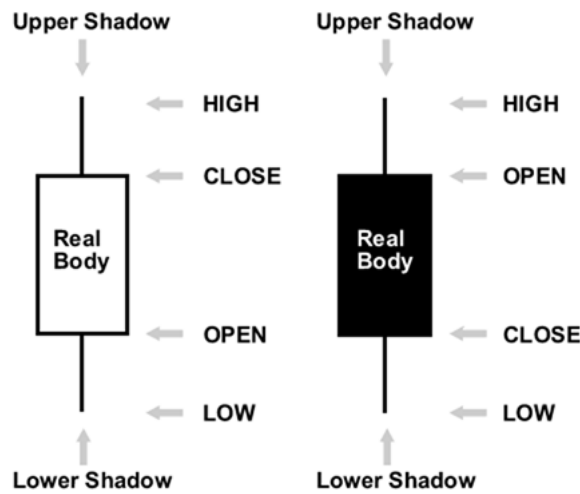


Figure 3.1: Japanese Candlestick

If the close is below the open, then a filled candlestick (usually displayed as black) is drawn. The hollow or filled section of the candlestick is called the “real body” or body. The thin lines poking above and below the body display the high/low range and are called shadows. The top of the upper shadow is the “high“. The bottom of the lower shadow is the “low“. An agent can get different types of signals by observing the Japanese candlestick chart. Eight such signals used in this study are described below.

1. Bullish swing : When yesterday's low is smaller than the lowest value of today and the lowest value of the day before yesterday.
2. Bearish swing : When yesterday's high is higher than the highest value of today and the highest value of the day before yesterday.

3. Bullish Pinbar : This signal appear in Japanese candlestick when the following three conditions are met:
 - The body of the candle is less than or equal to one-third of the entire candle range.
 - The minimum value among today's open or close is greater than the half of sum of the current maximum and minimum value.
 - The current minimum value is lower than the previous minimum value.
4. Bearish Pinbar: his signal appear in Japanese candlestick when the following three conditions are met:
 - The body of the candle is less than or equal to one-third of the entire candle range.
 - The maximum value among today's open or close is less than half of the sum of the current maximum and minimum value.
 - The current maximum value is greater than the previous maximum value.
5. Inside Bar: This signal appear in Japanese candlestick when the following two conditions are met:
 - Current maximum is less than the previous maximum.
 - Current minimum is greater than the previous minimum.
6. Outside Bar:This signal appear in Japanese candlestick when the following two conditions are met:
 - Current maximum is greater than the previous maximum.
 - Current minimum is smaller than the previous minimum.
7. Bearish Engulfing: This signal appear in Japanese candlestick when the following conditions are met:
 - There is an outside bar.
 - The real body of the candle is greater than eighty percent of the candle.

- Current closing price is greater than the current opening price.
8. Bullish Engulfing: This signal appears in Japanese candlestick when the following conditions are met:
- There is an outside bar.
 - The real body of the candle is greater than eighty percent of the candle.
 - Current closing price is smaller than the current opening price.

3.2 Reinforcement Learning

Reinforcement learning is a type of machine learning that determines the action within a specific environment in order to maximize a reward. The agent must continue interacting with the environment in order to find the best course of action through trial and error, which is one of the hallmarks of reinforcement learning. The agent can only receive a reward after completing the activity.

The trade-off between exploration and exploitation is one difficulty that reinforcement learning faces that other types of learning do not. A reinforcement learning agent must favor activities that it has previously attempted and found to be successful in creating reward in order to receive a lot of reward. However, it must try acts that it has never chosen before in order to find such activities. The agent must take advantage of its past experiences in order to profit, but it must also investigate in order to choose better future courses of action. The problem is that pursuing either exploration or exploitation solely would result in failure. The agent must try a variety of actions and progressively favor those that appear to be best. On a stochastic task, each action must be tried many times to gain a reliable estimate of its expected reward. The agent and the environment are RL's two primary building blocks. The agent makes decisions and provides a fix for issues. The environment is a representation of the problem. The agent and environment interact; the agent tries to affect the environment through actions, and the environment responds to the agent's activities.

This is one of the key differences between RL and other ML techniques.

There may or may not be a natural conclusion to the problem the agent is seeking to solve. Episodic tasks are those that have a predetermined conclusion, like a game. Continuous tasks are those that don't, like learning forward motion. An episode is the collection of time steps that make up an episodic task from start to finish. For agents to learn how to complete a task, it may take multiple time steps and episodes. A return is the total benefits accrued throughout a particular episode. Agents are frequently built to maximize return. Continuous jobs are frequently made into episodic tasks by adding a time step limit, allowing agents to optimize the return.

3.2.1 Q-Learning

Q-learning is a model-free reinforcement learning algorithm to learn the value of an action in a particular state. It does not require a model of the environment (hence "model-free"), and it can handle problems with stochastic transitions and rewards without requiring adaptations.

Q-learning identifies an optimal policy for any finite Markov decision process (FMDP) by maximizing the expected value of the total reward across any and all subsequent steps, beginning from the current state. Given limitless exploration time and a somewhat random policy, Q-learning can find the best action-selection strategy for any FMDP. The function "Q" that the algorithm computes is the anticipated rewards for a specific action in a particular state.

Q learning updates the Q function based on the following equation:

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a)) \quad (3.2)$$

Here, s' is the resulting state after taking action, a , in state s ; r is the associated reward; α is the learning rate; and γ is the discount factor. Also, $\max_{a'} Q(s', a')$ means that the behavior policy is greedy, where the highest Q-value among those in state s' is selected to generate learning data. In Q-learning, actions are taken according to the epsilon-greedy policy.

The amount that freshly learned knowledge supersedes previously learned information depends on the learning rate or step size. When the factor is 0, the agent learns nothing and solely uses prior knowledge, whereas when the factor is 1, the agent only takes into account the most recent information (ignoring prior knowledge to explore possibilities). An ideal learning rate in fully predictable situations is 1. In stochastic problems, the algorithm converges under certain technical constraints to the learning rate that calls for it to fall to zero. In reality, a constant learning rate, like 0.1, is frequently applied.

The significance of future benefits is determined by the discount factor. A number close to 1 will cause the agent to seek for a long-term high payoff, while a value close to 0 will cause it to be "myopic" (or short-sighted) by only considering immediate rewards. The action values may diverge if the discount factor is equal to or greater than 1. All environment histories become infinitely lengthy for a discount factor of 1 without a terminal state, or if the agent never achieves one, and utilities with additive, undiscounted rewards typically become infinite. When the value function is approximated by an artificial neural network, Q-function learning results in the propagation of mistakes and instabilities even with a discount factor that is only marginally less than 1. In that case, starting with a lower discount factor and increasing it towards its final value accelerates learning.

3.2.2 Deep Q-Learning

Q-learning can be combined with function approximation. This makes it possible to apply the algorithm to larger problems, even when the state space is continuous. One solution is to use an (adapted) artificial neural network as a function approximator. Deep Q-Networks (DQNs) are very similar to function approximation with neural networks, but they use neural networks to map the states to action values directly instead of using a set of generated features as media.

In Deep Q-learning, a neural network is trained to output the appropriate $Q(s,a)$ values for each action given the input state, s . The action, a , of the agent is chosen based on the output $Q(s,a)$ values following the epsilon-greedy policy. The deep Q-network learns to minimize the following error term.

$$\delta = r + \gamma \max_{a'} Q(s', a') - Q(s, a) \quad (3.3)$$

Finding the optimum network model to approximate the state-value function, $Q(s,a)$, for each potential action is the main objective in deep Q-learning. The mean squared error between the actual value and the estimated value serves as the loss function in this scenario, which is similar to that in a regression problem. It is not very stable to approximate Q-values using neural networks and one sample at a time. During episodes of a training session, the agent’s experiences—which are made up of an old state, a new state, an action, and a reward—are recorded in a memory queue. Batches of experiences are randomly selected from the memory and utilized to train the neural network once the agent has accumulated enough experience. Gaining experience and updating models based on randomly chosen past experiences are the two steps of learning using experience replay. Otherwise, the neural network model can become stuck in a local minimum since it will continue to learn from the most recent experience.

3.2.3 Double Deep Q-Networks

When the same neural network is used to calculate the predicted values and the target values, divergence occurs as the target values keep on changing and the prediction has to chase it. In double DQNs, the prediction network is not used to estimate the target; instead, a different network is used. The structure of the separate network is identical to that of the prediction network. For each T-episode, this network’s weights are fixed (T is a hyperparameter that can be tuned). The update is completed by simply copying the prediction network’s weights. This approach fixes the goal function for a while, making the training process more reliable.

Mathematically, double DQNs are trained to minimize the following error term:

$$\delta = r + \gamma \max_{a'} Q_T(s') - Q(s) \quad (3.4)$$

Here, s' is the resulting state after taking action, a , in state s ; r is the associated reward; α is the learning rate; and γ is the discount factor. Also, Q_T is the function for the target network, and Q is the function for the prediction network.

CHAPTER 4

METHODOLOGY

4.1 Framework

The basic framework of the methodology used in this study is depicted in the following figure.

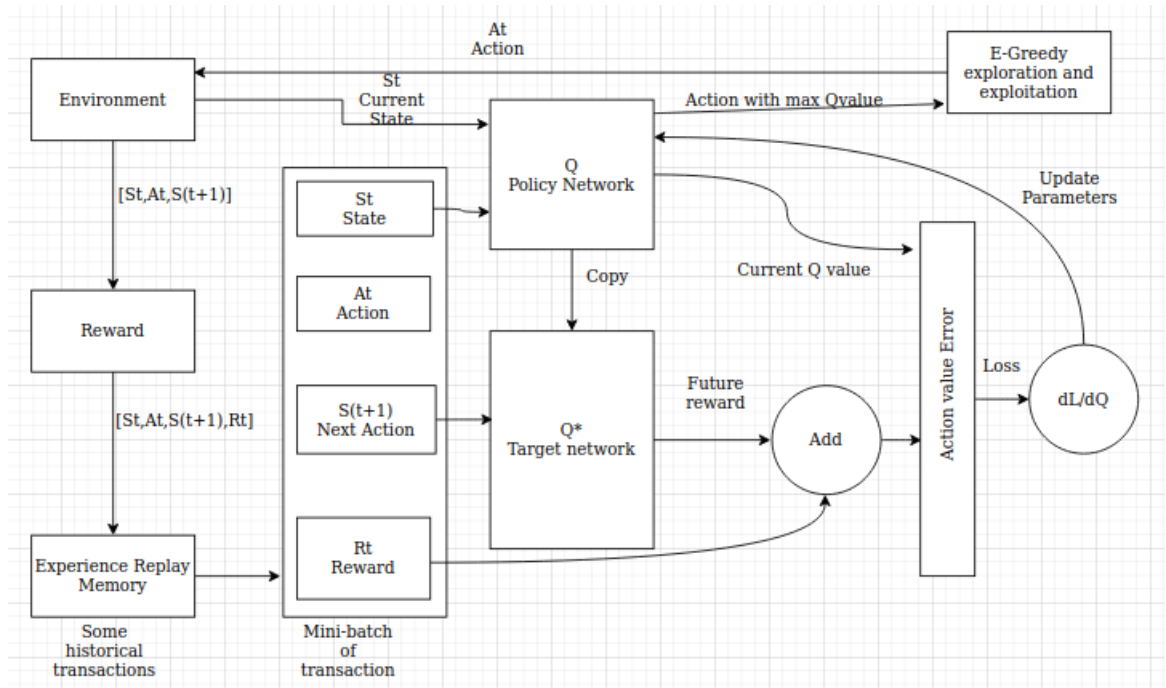


Figure 4.1: Double Deep Q Learning Framework.

The working mechanism of Double Deep Q learning framework described in Figure 4.1 is as follows.

1. Initialize the replay memory capacity.
2. Initialize the policy network with random weights.
3. Clone the policy network with random weights.

4. For each episode:
 - (a) Initialize the starting state.
 - (b) For each time step.
 - i. Select an action.
 - Via exploration and exploitation.
 - ii. Execute selected action in an emulator.
 - iii. Observe reward and next state.
 - iv. Store experience in replay memory.
 - v. Sample random batch from replay.
 - vi. Preprocess states from batch.
 - vii. Pass batch of preprocessed states to policy network.
 - viii. Calculate loss between output Q-values and target Q-values.
 - Requires a pass to the target network for the next state.
 - ix. Gradient descent update weights in the policy network to minimize loss.
 - After 'x' time steps, weights in the target network are updated to the weights in the policy network.

4.2 Data samples preparation

Data of four commercial Banks (ADBL, CZBIL, LBL and NABIL) from 2012-01-01 to 2022-07-15 were scrapped from Nepalipaisa.com. This data includes the daily, open, close, volume, maximum, minimum and number of transactions. The data was cleaned to delete empty values and values that were other than numbers. This data was normalized using z-score normalization. The cleaned data was further processed to obtain the following technical indicator.

1. MACD
2. OBV
3. ADX
4. Stochastic oscillator

The cleaned data is also used to obtain the following signals of the Japanese candlestick chart.

1. Bullish swing
2. Bearish swing
3. Bullish pinbar
4. Bearish pinbar
5. Inside bar
6. Outside bar
7. Bullish engulfing
8. Bearish engulfing

4.3 Experimental tools and techniques

Different hyperparameters that were tuned during the experiments are as follows.

- Replay size
- Target update
- Discount Factor
- Learning Rate

Together with these four hyperparameters, the dynamic exploration rate was also used. At the beginning of the experiment the exploration rate was set to 0.1, which means there was a 10 percent chance that the agent will explore. This exploration rate was decayed in each factor with the decaying factor of 0.99. The exploration rate was allowed to decay till it reached 0.01. All Double Deep Q networks were trained with Adam optimizer [28]. The state of the neural network along with the epsilon and rewards in each episode were saved after every 5 episodes. All the experiments were conducted using a professional version of Google Colaboratory. The programming language, libraries and their versions used in this project are listed below:

Table 4.1: Programming language and libraries used in this study.

SN	Programming Language/Library	Version	Usage
1	Python	3.7.13	To carry out experiment
2	Pytorch	1.12.0+cu113	Modeling neural network
3	Numpy	1.21.6	For mathematical calculation
4	Gym	0.17.3	For creating reinforcement learning environment
5	Pandas	1.3.5	For processing data
6	Scipy	1.7.3	For z-score normalization
7	Talib	0.1.24	Used for calculation of technical indicators

4.4 Stock Trading Environments

This stock trading environment is inherited from the reinforcement learning environment provided by the openai gym. Observation space of this environment is 10x6, which includes following information:

- 10 days opening price of the stock

- 10 days closing price of the stock
- 10 days maximum stock price
- 10 days minimum stock price
- 10 days volume of stock
- 10 days balance of the trading agent

The action space of this environment is 3, which includes holding stock, buying stock and selling stock. The agent gets reward of +1 points if the net worth of the agent is greater than the maximum net worth among baseline methods. The agent gets reward of -1 points if the net worth of the agent is smaller than the maximum net worth among baseline methods. Resetting this environment sets the following parameters to following values:

- Balance = 100,000
- Net worth = 100,000
- Net worth old = 100,000
- Maximum net worth = 100,000
- Shares held = 0
- Cost basis = 0
- Total shares sold = 0
- Total sales value = 0

Rendering this environment prints the value of the following output:

- Current step
- Current balance
- Shares held
- Total shares sold
- Cost basis
- Total sales value

- Net worth old
- Current net worth
- Maximum net worth
- Profit

A single episode in this environment is defined by the number of steps which is equivalent to the number of days in the processed data of the stock.

4.5 Double Deep Q Networks

Deep Q Networks used in this study consists of two networks: policy network and target network. The target network has same parameters as that of policy network. The weights of the target network are updated with the weights of the policy network after certain number of episodes. The discount factor was set to 0.99, the replay size was set to 5 and learning rate was set to 0.001. Three types of policy networks used in this study are mentioned below.

4.5.1 Type I

Table 4.2: Architecture of Type I policy networks.

NN Function	Kernel Size	Stride	Padding
2D CNN	3x3	1	1
MaxPool	2x2	1	1
2D CNN	3x3	1	1
MaxPool	2x2	1	0
2D CNN	3x3	1	1
MaxPool	2x2	1	0
	In Features	Out Features	Activation Function
Dense	45	50	ReLU
Dense	50	50	ReLU
Dense	50	50	ReLU
Dense	50	3	Softmax

This model was trained with three different values of target update: 5, 25 and 50 and saved with the name L_CNN_5 ('L' indicates Large. As its largest among all three types of models. CNN: as CNN architecture was used, 5 indicate value of target update.), L_CNN_25 and L_CNN_50 for each set of training data.

4.5.2 Type II

Table 4.3: Architecture of Type II policy networks.

NN Function	Kernel Size	Stride	Padding
2D CNN	3x3	1	1
MaxPool	2x2	1	1
2D CNN	3x3	1	1
MaxPool	2x2	1	0
	In Features	Out Features	Activation Function
Dense	60	90	ReLU
Dense	90	90	ReLU
Linear	90	3	Softmax

This model was trained with three different values of target update: 5, 25 and 50 and saved with the name M_CNN_5 ('M' indicates Medium. CNN: as CNN architecture was used, 5 indicate value of target update), M_CNN_25 and M_CNN_50 for each set of training data.

4.5.3 Type III

Table 4.4: Architecture of Type II policy networks.

NN Function	Kernel Size	Stride	Padding
2D CNN	3x3	1	1
MaxPool	2x2	1	1
	In Features	Out Features	Activation Function
Dense	77	90	ReLU
Dense	90	90	ReLU
Linear	90	3	Softmax

This model was trained with three different values of target update: 5, 25 and 50 and saved with the name S_CNN_5 ('S' indicates Small. As its smallest among all three types of models. CNN: as CNN architecture was used, 5 indicate value of target update.), S_CNN_25 and S_CNN_50 for each set of training data.

Total nine Deep Q models were trained and tested for data of each commercial bank stocks. The performance of these models were compared with each other and baseline trading strategies in terms of annualised expected trade return described by the following equation.

$$E(R) = \left(\frac{\text{Net worth at beginning of investment}}{\text{Net worth at end of investment}} \right)^{\frac{1}{\text{No. of years}}} - 1 \quad (4.1)$$

CHAPTER 5

RESULTS AND ANALYSIS

Experiments with the stocks of four commercial banks listed in NEPSE has been conducted in this study. The details of these experiments and results obtained from them are described in the following section.

5.1 ADBL

The data of ADBL from 2012-01-01 to 2022-07-15 was scraped from Nepalipaisa.com. The data was divided into training and test data. Data from 2012-01-01 to 2020-01-01 (1844 trading days) was used to train the models while the data from 2020-01-02 to 2022-07-15 (548 trading days). The figures showing the fluctuation in close price of stock in training data and test data and the table describing the training data and testing data are as follows.



Figure 5.1: Closing price of ADBL stock for training data and test data respectively.

Table 5.1: Summary of the training data of ADBL.

	Close	Volume	Low	High	Open
Count	1844	1844	1844	1844	1843
Mean	407.31	11885.23	400.39	412.73	407.338
Std	170.46	16549.98	166.715	173.22	170.51
Min	101.00	41	101	103	101.00
Max	1082	196356	1048	1113	1082

Table 5.2: Summary of the test data of ADBL.

	Close	Volume	Low	High	Open
Count	548	548	548	548	547
Mean	438.97	63370.88	432.87	446.03	439.16
Std	63.186459	68231.1016	62.108	64.68	63.074
Min	305	90	298	310	305
Max	597	800592	590	620	597

The following table summarizes the performance of various methods on training data. Here, the starting capital in each case is Nrs. 1,00,000 and the various methods are compared based on Final Net Worth and Annualized Expected Trade Return in percentage.

Table 5.3: Comparison of performance of various trading methods on ADBL training data.

SN	Methods	Final Net Worth (Nrs)	Annualized Expected Trade Return (%)
1	JCS	6,26,560	25.78
2	OBV	9,11,775	31.82
3	Stochastic Oscillator	1,24,986	2.8
4	ADX	2,38,710	11.49
5	MACD	2,46,680	11.95
6	L_CNN_5	9,00,502	31.61
7	L_CNN_25	9,46,059	32.43
8	L_CNN_50	9,43,897	32.4
9	M_CNN_5	8,84,556	31.32
10	M_CNN_25	9,48,879	32.48
11	M_CNN_50	9,48,679	32.47
12	S_CNN_5	9,17,303	31.92
13	S_CNN_25	9,49,342	32.48
14	S_CNN_50	8,97,597	31.56

Each deep-Q network were trained for 1000 episodes. The final net-worth for 1000 episodes for each set of experiments is shown by the following figures.

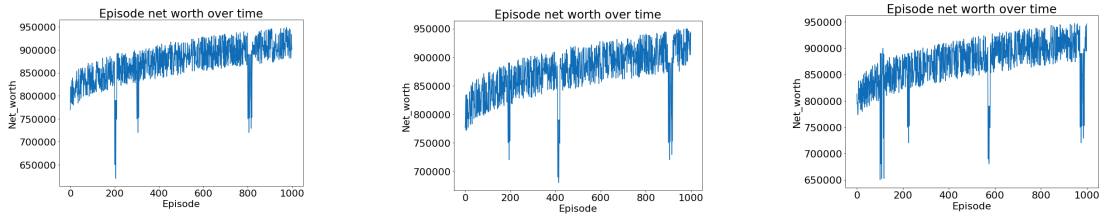


Figure 5.2: Net worth of deep Q learning agent for ADBL over 1000 episodes in L_CNN_5, L_CNN_25 and L_CNN_50 experiments for respectively

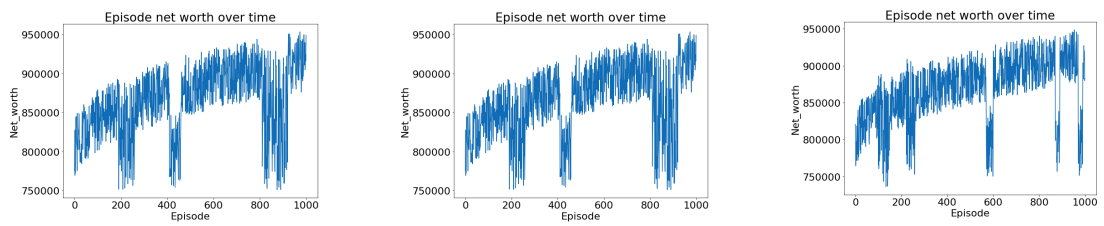


Figure 5.3: Net worth of deep Q learning agent for ADBL over 1000 episodes in M_CNN_5, M_CNN_25 and M_CNN_50 experiments respectively

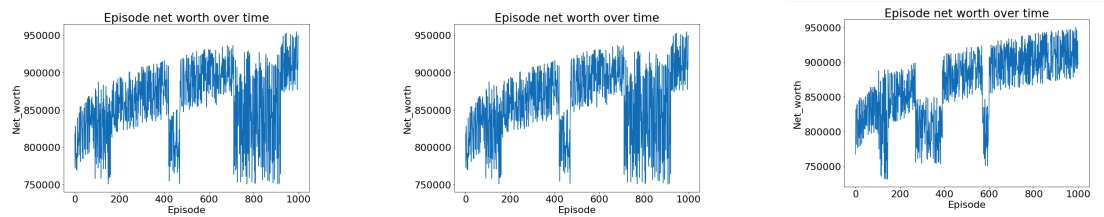


Figure 5.4: Net worth of deep Q learning agent for ADBL over 1000 episodes in S_CNN_5, S_CNN_25 and S_CNN_50 experiments respectively

The following table summarizes the performance of various methods on test data. Here, the starting capital in each case is Nrs. 100000 and the various methods are compared based on Final Net Worth and Annualized Expected Trade Return in percentage.

Table 5.4: Comparison of performance of various trading methods on test data for ADBL.

SN	Methods	Final Net Worth (Nrs)	Annualized Expected Trade Return (%)
1	JCS	64,348	-25.4
2	OBV	2,28,846.5	73.65
3	Stochastic Oscillator	96,192.0	-2.55
4	ADX	86,726	-9.05
5	MACD	98,447	-1.04
6	L_CNN_5	2,49,829	84.41
7	L_CNN_25	2,28,903	73.68
8	L_CNN_50	2,29,159	73.81
9	M_CNN_5	2,50,914	84.65
10	M_CNN_25	2,29,132	73.80
11	M_CNN_50	2,28,980	73.72
12	S_CNN_5	2,44,972	81.72
13	S_CNN_25	2,23,937	71.165
14	S_CNN_50	2,24,092	71.244

The following table summarizes the performance of various double deep-Q networks on test data in terms of latency. The latency is calculated by averaging the performance over 100 episodes.

Table 5.5: Comparison of performance of various Double Deep-Q networks on test data for ADBL in terms of latency.

SN	Methods	Latency (millisecond/step)
1	L_CNN_5	1.31
2	L_CNN_25	1.37
3	L_CNN_50	1.36
4	M_CNN_5	0.67225
5	M_CNN_25	0.668
6	M_CNN_50	0.67
7	S_CNN_5	1.034
8	S_CNN_25	1.1337
9	S_CNN_50	1.097

5.2 CZBIL

The data of CZBIL from 2012-01-01 to 2022-07-15 was scraped from Nepali-paisa.com. The data was divided into training and test data. Data from 2012-01-01 to 2020-01-01 (1840 trading days) was used to train the models while the data from 2020-01-02 to 2022-07-15 (547 trading days). The figures showing the fluctuation in close price of stock in training data and test data and the table describing the training data and testing data are as follows.

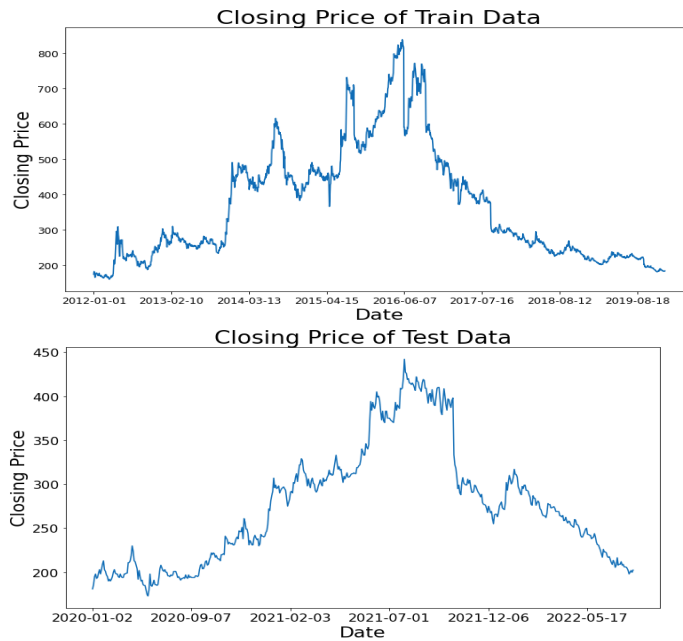


Figure 5.5: Closing price of CZBIL stock for training data and test data respectively.

Table 5.6: Summary of the training data of CZBIL.

	Close	Volume	Low	High	Open
Count	1840	1840	1840	1840	1839
Mean	367.25	15513.04	361.40	372.62	367
Std	162.46	19045.96	159.90	165.14	162.44
Min	159.00	10.0	156.0	159.0	159.0
Max	838	324872	832	870	838

Table 5.7: Summary of the test data of CZBIL.

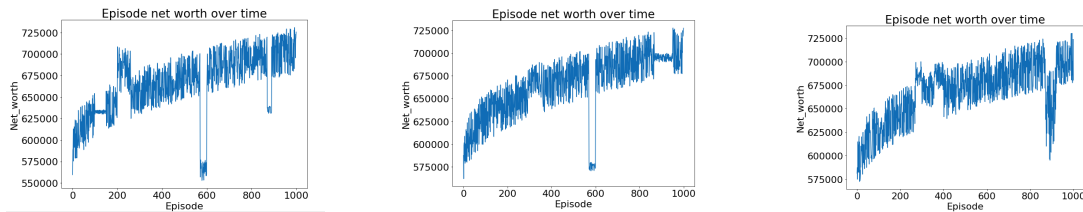
	Close	Volume	Low	High	Open
Count	547	547	547	547	546
Mean	275.13	112045.45	270.917	280.36	275.2
Std	67.29	107730.767	66.30	68.743	67.28
Min	173	23.00	164	173	173.0
Max	442	741066	425	444	442

The following table summarizes the performance of various methods on training data. Here, the starting capital in each case is Nrs. 1,00,000 and the various methods are compared based on Final Net Worth and Annualized Expected Trade Return in percentage.

Table 5.8: Comparison of performance of various trading methods on CZBIL training data.

SN	Methods	Final Net Worth (Nrs)	Annualized Expected Trade Return (%)
1	JCS	5,28,169	23.125
2	OBV	7,15,538	27.887
3	Stochastic Oscillator	71,301	-4.14
4	ADX	88,257	-1.154
5	MACD	1,32,886	3.61
6	L_CNN_5	7,25,550	28.110
7	L_CNN_25	7,27,236	28.147
8	L_CNN_50	7,30,137	28.21
9	M_CNN_5	7,19,431	27.97
10	M_CNN_25	7,29,524	28.19
11	M_CNN_50	7,15,477	27.886
12	S_CNN_5	6,91,825	27.35
13	S_CNN_25	7,04,968	27.65
14	S_CNN_50	6,95,477	27.434

Each deep-Q network were trained for 1000 episodes. The final net-worth for 1000 episodes for each set of experiments is shown by the following figures.

**Figure 5.6:** Net worth of deep Q learning agent for CZBIL over 1000 episodes in L_CNN_5, L_CNN_25 and L_CNN_50 experiments for respectively

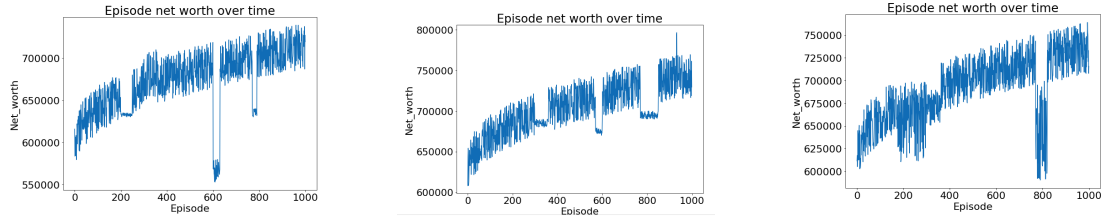


Figure 5.7: Net worth of deep Q learning agent for CZBIL over 1000 episodes in M_CNN_5, M_CNN_25 and M_CNN_50 experiments respectively

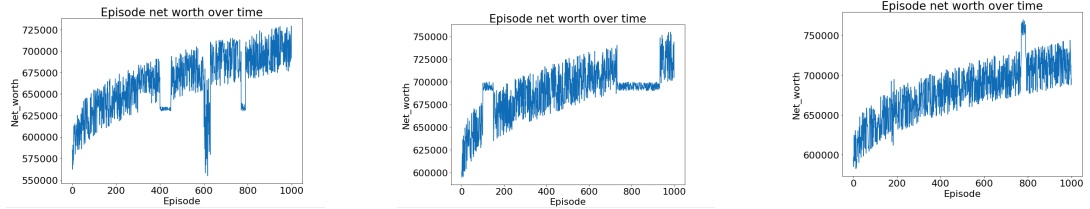


Figure 5.8: Net worth of deep Q learning agent for CZBIL over 1000 episodes in S_CNN_5, S_CNN_25 and S_CNN_50 experiments respectively

The following table summarizes the performance of various methods on test data. Here, the starting capital in each case is Nrs. 100000 and the various methods are compared based on Final Net Worth and Annualized Expected Trade Return in percentage.

Table 5.9: Comparison of performance of various trading methods on test data for CZBIL.

SN	Methods	Final Net Worth (Nrs)	Annualized Expected Trade Return (%)
1	JCS	1,02,224.5	1.47
2	OBV	1,09,282.9	6.09
3	Stochastic Oscillator	1,04,773.2	3.15
4	ADX	76,265	-16.5
5	MACD	83,324	-11.45
6	L_CNN_5	1,12,564	8.209
7	L_CNN_25	1,12,466	8.14
8	L_CNN_50	1,19,982	12.9
9	M_CNN_5	1,12,370	8.08
10	M_CNN_25	1,07,737	5.09
11	M_CNN_50	1,17,589	11.4
12	S_CNN_5	1,09,995	6.5
13	S_CNN_25	1,07,723	5.08
14	S_CNN_50	1,07,507	4.94

The following table summarizes the performance of various double deep-Q networks on test data in terms of latency. The latency is calculated by averaging the performance over 100 episodes.

Table 5.10: Comparison of performance of various Double Deep-Q networks on test data for CZBIL in terms of latency.

SN	Methods	Latency (millisecond/step)
1	L_CNN_5	1.155
2	L_CNN_25	1.144
3	L_CNN_50	1.144
4	M_CNN_5	1.21
5	M_CNN_25	1.11
6	M_CNN_50	1.102
7	S_CNN_5	1.02
8	S_CNN_25	1.049
9	S_CNN_50	1.042

5.3 LBL

The data of LBL from 2012-01-01 to 2022-07-15 was scraped from Nepalipaisa.com. The data was divided into training and test data. Data from 2012-01-01 to 2020-01-01 (1816 trading days) was used to train the models while the data from 2020-01-02 to 2022-07-15 (547 trading days). The figures showing the fluctuation in close price of stock in training data and test data and the table describing the training data and testing data are as follows.

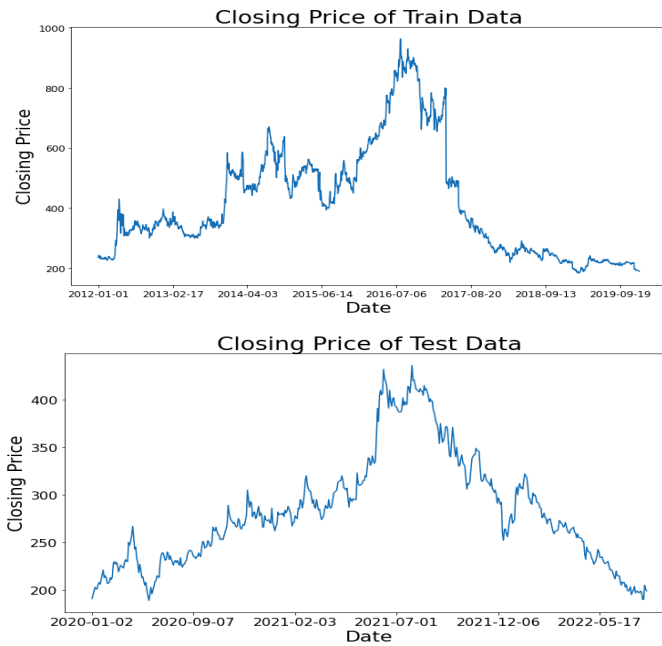


Figure 5.9: Closing price of LBL stock for training data and test data respectively.

Table 5.11: Summary of the training data of LBL.

	Close	Volume	Low	High	Open
Count	1816.0	1816.0	1816.0	1816.0	1815.
Mean	415.0	10630.0	410.0	421.0	415
Std	184.0	36906.0	181.0	186.0	184.0
Min	183.0	12.0	182.0	185.0	183.0
Max	963.0	1402678.0	931.0	972.0	963.0

Table 5.12: Summary of the test data of LBL.

	Close	Volume	Low	High	Open
Count	547.0	547.0	547.0	547.0	546.0
Mean	282.0	70541.0	278.0	288.0	283.0
Std	158.0	86890.0	57.0	60.0	58.0
Min	189.0	130.0	184.0	191.0	189.0
Max	436.0	826295.0	422.0	447.0	436.0

The following table summarizes the performance of various methods on training data. Here, the starting capital in each case is Nrs. 1,00,000 and the various methods are compared based on Final Net Worth and Annualized Expected Trade Return in percentage.

Table 5.13: Comparison of performance of various trading methods on LBL training data.

SN	Methods	Final Net Worth (Nrs)	Annualized Expected Trade Return (%)
1	JCS	1,66,980	6.61
2	OBV	1,05,290	0.64
3	Stochastic Oscillator	73,389	-3.7
4	ADX	50,508	-8.1
5	MACD	82,882	-2.3
6	L_CNN_5	1,63,902	6.3
7	L_CNN_25	1,65,728	6.5
8	L_CNN_50	1,70,819	6.9
9	M_CNN_5	1,70,229	6.8
10	M_CNN_25	1,67,188	6.6
11	M_CNN_50	1,58,251	5.9
12	S_CNN_5	1,61,738	6.1
13	S_CNN_25	1,52,530	5.4
14	S_CNN_50	1,47,658	4.9

Each deep-Q network were trained for 1000 episodes. The final net-worth for 1000 episodes for each set of experiments is shown by the following figures.

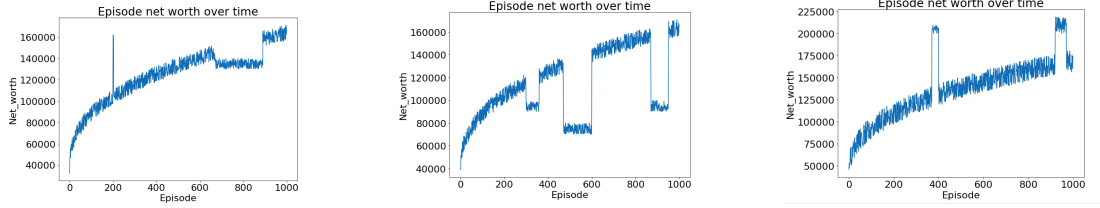


Figure 5.10: Net worth of deep Q learning agent for LBL over 1000 episodes in L_CNN_5, L_CNN_25 and L_CNN_50 experiments for respectively

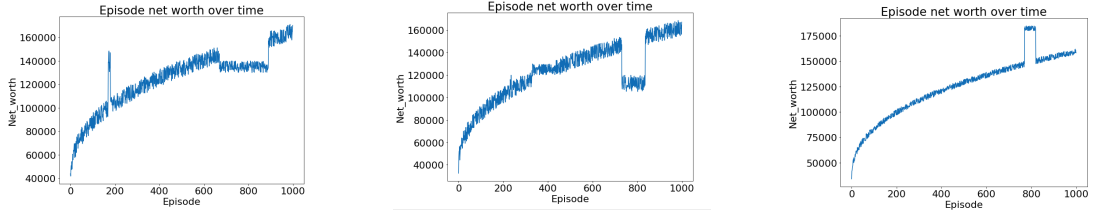


Figure 5.11: Net worth of deep Q learning agent for LBL over 1000 episodes in M_CNN_5, M_CNN_25 and M_CNN_50 experiments respectively

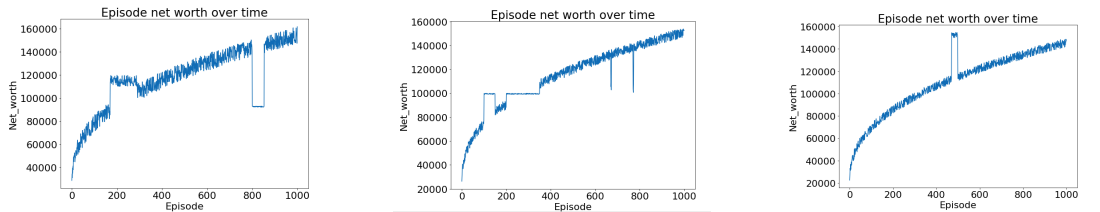


Figure 5.12: Net worth of deep Q learning agent for LBL over 1000 episodes in S_CNN_5, S_CNN_25 and S_CNN_50 experiments respectively

The following table summarizes the performance of various methods on test data. Here, the starting capital in each case is Nrs. 100000 and the various methods are compared based on Final Net Worth and Annualized Expected Trade Return in percentage.

Table 5.14: Comparison of performance of various trading methods on test data for LBL.

SN	Methods	Final Net Worth (Nrs)	Annualized Expected Trade Return (%)
1	JCS	45,904	-40.5
2	OBV	1,16,498	10.71
3	Stochastic Oscillator	90,730	-6.2
4	ADX	1,15,571	10.12
5	MACD	71,270	-20.21
6	L_CNN_5	1,07,694	5.06
7	L_CNN_25	1,17,535	11.37
8	L_CNN_50	1,22,791	14.668
9	M_CNN_5	1,15,078	9.81
10	M_CNN_25	1,12,394	8.1
11	M_CNN_50	1,17,613	11.42
12	S_CNN_5	96,289	-2.4
13	S_CNN_25	92,756	-4.8
14	S_CNN_50	87,475	-8.5

The following table summarizes the performance of various double deep-Q networks on test data in terms of latency. The latency is calculated by averaging the performance over 100 episodes.

Table 5.15: Comparison of performance of various Double Deep-Q networks on test data for LBL in terms of latency.

SN	Methods	Latency (millisecond/step)
1	L_CNN_5	1.25
2	L_CNN_25	1.12
3	L_CNN_50	1.23
4	M_CNN_5	1.11
5	M_CNN_25	1.11
6	M_CNN_50	1.13
7	S_CNN_5	1.06
8	S_CNN_25	1.05
9	S_CNN_50	1.05

5.4 NABIL

The data of NABIL from 2012-01-01 to 2022-07-15 was scraped from Nepali-paisa.com. The data was divided into training and test data. Data from 2012-01-01 to 2020-01-01 (1835 trading days) was used to train the models while the data from 2020-01-02 to 2022-07-15 (548 trading days). The figures showing the fluctuation in close price of stock in training data and test data and the table describing the training data and testing data are as follows.

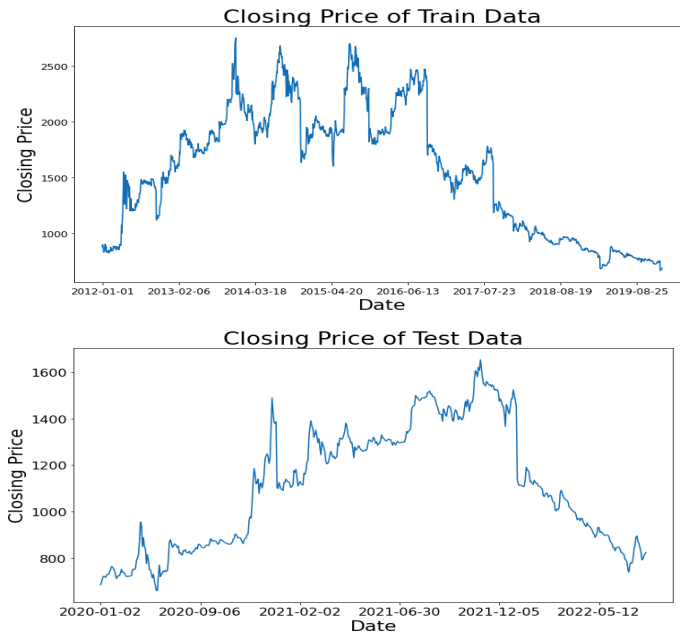


Figure 5.13: Closing price of NABIL stock for training data and test data respectively.

Table 5.16: Summary of the training data of NABIL.

	Close	Volume	Low	High	Open
Count	1835.0	1835.0	1835.0	1835.0	1834.0
Mean	1560.0	5697.0	1543.0	1576.0	1561.0
Std	554.0	6256.0	546.0	561.0	554.0
Min	667.0	26.0	660.0	669.0	667.0
Max	2750.0	75157.0	2700.0	2800.0	2750.0

Table 5.17: Summary of the test data of NABIL.

	Close	Volume	Low	High	Open
Count	548.0	548.0	548.0	548.0	547.0
Mean	1107.0	86509.0	1092.0	1126.0	1107.0
Std	263.0	73814.0	260.0	268.0	263.0
Min	660.0	435.0	594.0	667.0	660.0
Max	1653.0	705430.0	1640.0	1740.0	1653.0

The following table summarizes the performance of various methods on training data. Here, the starting capital in each case is Nrs. 1,00,000 and the various methods are compared based on Final Net Worth and Annualized Expected Trade Return in percentage.

Table 5.18: Comparison of performance of various trading methods on NABIL training data.

SN	Methods	Final Net Worth (Nrs)	Annualized Expected Trade Return (%)
1	JCS	5,65,897	24.19
2	OBV	7,88,252	29.44
3	Stochastic Oscillator	73,106	-3.84
4	ADX	51,600	-7.9
5	MACD	84,672	-2.0
6	L_CNN_5	8,03,902	29.76
7	L_CNN_25	8,05,728	29.79
8	L_CNN_50	8,10,819	29.9
9	M_CNN_5	7,40,229	28.43
10	M_CNN_25	7,69,188	29.04
11	M_CNN_50	7,28,251	28.16
12	S_CNN_5	7,71,738	29.10
13	S_CNN_25	7,82,681	29.30
14	S_CNN_50	7,83,701	29.35

Each deep-Q network were trained for 1000 episodes. The final net-worth for 1000 episodes for each set of experiments is shown by the following figures.

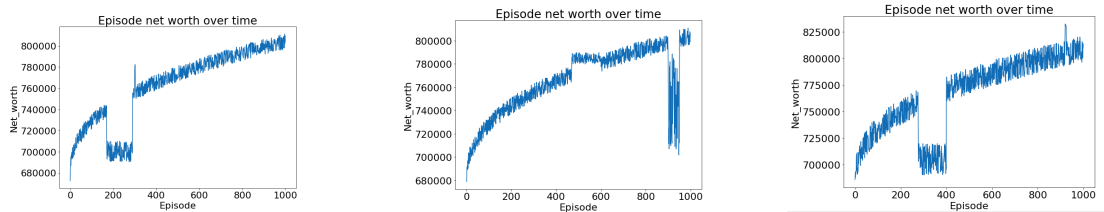


Figure 5.14: Net worth of deep Q learning agent for NABIL over 1000 episodes in L_CNN_5, L_CNN_25 and L_CNN_50 experiments for respectively

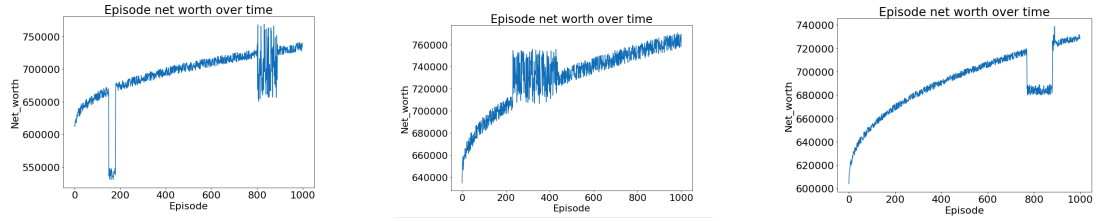


Figure 5.15: Net worth of deep Q learning agent for NABIL over 1000 episodes in M_CNN_5, M_CNN_25 and M_CNN_50 experiments respectively

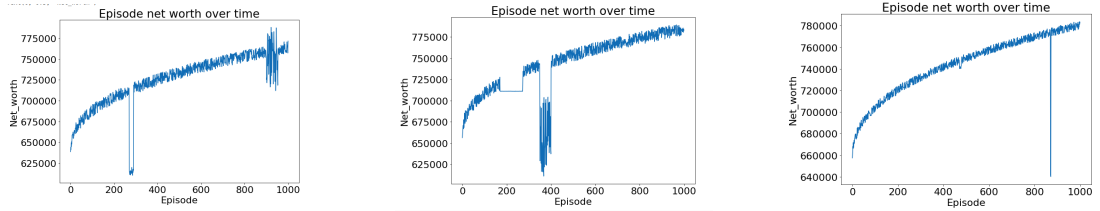


Figure 5.16: Net worth of deep Q learning agent for NABIL over 1000 episodes in S_CNN_5, S_CNN_25 and S_CNN_50 experiments respectively

The following table summarizes the performance of various methods on test data. Here, the starting capital in each case is Nrs. 1,00,000 and the various methods are compared based on Final Net Worth and Annualized Expected Trade Return in percentage.

Table 5.19: Comparison of performance of various trading methods on test data for NABIL.

SN	Methods	Final Net Worth (Nrs)	Annualized Expected Trade Return (%)
1	JCS	1,08,785.9	5.7
2	OBV	2,89,670	103.2
3	Stochastic Oscillator	1,89,520	53.14
4	ADX	76,934	-16.038
5	MACD	1,02,812	1.86
6	L_CNN_5	1,65,435	39.87
7	L_CNN_25	3,07,470	111.44
8	L_CNN_50	3,05,055	110.33
9	M_CNN_5	2,65,724	91.846
10	M_CNN_25	2,79,890	98.605
11	M_CNN_50	2,93,614	105.046
12	S_CNN_5	2,83,840	100.4
13	S_CNN_25	2,59,109	88.64
14	S_CNN_50	2,64,030	91.03

The following table summarizes the performance of various double deep-Q networks on test data in terms of latency. The latency is calculated by averaging the performance over 100 episodes.

Table 5.20: Comparison of performance of various Double Deep-Q networks on test data for NABIL in terms of latency.

SN	Methods	Latency (millisecond/step)
1	L_CNN_5	1.45
2	L_CNN_25	1.43
3	L_CNN_50	1.33
4	M_CNN_5	1.21
5	M_CNN_25	1.21
6	M_CNN_50	1.23
7	S_CNN_5	1.09
8	S_CNN_25	1.08
9	S_CNN_50	1.02

CHAPTER 6

Conclusion and Future Work

6.1 Summary

The data of four different commercial banks listed in NEPSE were used in this study to develop the trading strategy using double deep Q-learning where the CNN was used to build a policy Q network and target Q network. By observing the plot of closing price of stocks of four banks used in this study, a rise in price of stocks in year 2016 can be seen. This was due to low interest rate, high liquidity and implementation of paperless transaction system. In case of the ADBL training data the model S_CNN_25 had highest annualized expected trade return of 32.48 %. Various models were trained with this data and later was tested with ADBL test data. M_CNN_25 model had highest annualized expected trade return of 84.65% for test data of ADBL. In case of the CZBIL training data the model L_CNN_50 had highest annualized expected trade return of 28.21%. Various models were trained with this data and later was tested with CZBIL test data. L_CNN_50 model had highest annualized expected trade return of 12.9% for test data of CZBIL.

In case of the LBL training data the model L_CNN_50 had highest annualized expected trade return of 6.9%. Various models were trained with this data and later was tested with LBL test data. L_CNN_50 model had highest annualized expected trade return of 14.68% for test data of LBL. L_CNN_50 had highest annualized expected trade return of 29.9% and for training data of NABIL. Various models were trained with this data and later was tested with NABIL test data. L_CNN_25 model had highest annualized expected trade return of 111.44% for test data of NABIL.

6.2 Conclusion

In this study Double Deep Q learning agent was designed. This agent was trained in the environment where it is fed basic stocks data and its learns by observing the performance of various baseline models. In all eight cases, double deep Q learning agent was able to outperform the baseline models. The CNN used in the policy network learns patterns inside the 10 days data of stock prices in each sample. Based on these patterns the agent takes trading decisions and get either positive or negative reward. Unlike supervised learning, there is no fixed target to shoot at in case of reinforcement learning, this is the main reason of getting noisy output while training the agent.

6.3 Future Work

This work was conducted under the assumption listed in section 1.4. With the data from diverse source regarding stock split, dividend, broker's commission etc. a double deep Q learning agent can be trained which can learn trading strategies and gain positive returns. The deep Q learning agent in this study deals with only one stock at a time, this can be upgraded to the agent who deals with multiple stocks at a time. Using Advantage Actor Critic method [29] an reinforcement learning agent can be trained to take continuous action so that it can buy or sell certain percentage of stocks. The noise and instability faced during the training of Double Deep Q learning can be reduced using weight averaging [30] and prioritized experience replay [31].

REFERENCES

- [1] M. Gladwell, *Outliers: The Story of Success*. New York, NY: Back Bay Books, 2009.
- [2] F. Allen and R. Karjalainen, “Using genetic algorithms to find technical trading rules,” *Journal of financial Economics*, vol. 51, no. 2, pp. 245–271, 1999.
- [3] J. Straßburg, C. González-Martel, and V. Alexandrov, “Parallel genetic algorithms for stock market trading rules,” *Procedia Computer Science*, vol. 9, pp. 1306–1313, 2012.
- [4] C.-H. Chen, Y.-H. Chen, J. C.-W. Lin, and M.-E. Wu, “An effective approach for obtaining a group trading strategy portfolio using grouping genetic algorithm,” *IEEE Access*, vol. 7, pp. 7313–7325, 2019.
- [5] B. J. de Almeida, R. F. Neves, and N. Horta, “Combining support vector machine with genetic algorithms to optimize investments in forex markets with high leverage,” *Applied Soft Computing*, vol. 64, pp. 596–613, 2018.
- [6] B. AlArmouty and S. Fraihat, “Data analytics and business intelligence framework for stock market trading,” in *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*. IEEE, 2019, pp. 1–6.
- [7] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.
- [8] C. Gaskett *et al.*, “Q-learning for robot control,” 2002.
- [9] E. S. Low, P. Ong, and K. C. Cheah, “Solving the optimal path planning of a mobile robot using improved q-learning,” *Robotics and Autonomous Systems*, vol. 115, pp. 143–161, 2019.
- [10] K.-H. Park, Y.-J. Kim, and J.-H. Kim, “Modular q-learning based multi-agent cooperation for robot soccer,” *Robotics and Autonomous systems*, vol. 35, no. 2, pp. 109–122, 2001.

- [11] R. Vinuesa and S. L. Brunton, “The potential of machine learning to enhance computational fluid dynamics,” *arXiv preprint arXiv:2110.02085*, 2021.
- [12] X. Du, J. Zhai, and K. Lv, “Algorithm trading using q-learning and recurrent reinforcement learning,” *positions*, vol. 1, no. 1, 2016.
- [13] J. W. Lee, J. Park, O. Jangmin, J. Lee, and E. Hong, “A multiagent approach to q-learning for daily stock trading,” *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 37, no. 6, pp. 864–877, 2007.
- [14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.
- [15] Y. Wang, D. Wang, S. Zhang, Y. Feng, S. Li, and Q. Zhou, “Deep q-trading,” *csl.t. riit. tsinghua. edu. cn*, 2017.
- [16] G. Jeong and H. Y. Kim, “Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning,” *Expert Systems with Applications*, vol. 117, pp. 125–138, 2019.
- [17] A. Brim and N. S. Flann, “Deep reinforcement learning stock market trading, utilizing a cnn with candlestick images,” *Plos one*, vol. 17, no. 2, p. e0263181, 2022.
- [18] E. Ponomarev, I. V. Oseledets, and A. Cichocki, “Using reinforcement learning in the algorithmic trading problem,” *Journal of Communications Technology and Electronics*, vol. 64, no. 12, pp. 1450–1457, 2019.
- [19] M. S. Elbamby, C. Perfecto, C.-F. Liu, J. Park, S. Samarakoon, X. Chen, and M. Bennis, “Wireless edge computing with latency and reliability guarantees,” *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1717–1737, 2019.
- [20] Y. Hu, W. Li, K. Xu, T. Zahid, F. Qin, and C. Li, “Energy management strategy for a hybrid electric vehicle based on deep reinforcement learning,” *Applied Sciences*, vol. 8, no. 2, p. 187, 2018.
- [21] W. Li, H. Cui, T. Nemeth, J. Jansen, C. Ünlübayır, Z. Wei, L. Zhang, Z. Wang,

- J. Ruan, H. Dai *et al.*, “Deep reinforcement learning-based energy management of hybrid battery systems in electric vehicles,” *Journal of Energy Storage*, vol. 36, p. 102355, 2021.
- [22] A. Maskey, “Predicting neps index using arima model,” 2022.
- [23] P. Thapa and B. Aryal, “Use of geometric brownian motion to forecast stock market scenario using post covid-19 neps index,” *BIBECHANA*, vol. 18, no. 2, pp. 50–60, 2021.
- [24] J. K. Lal and A. K. Timalina, “A CNN-BGRU Method for Stock Price Prediction,” in *Proceedings of 11th IOE Graduate Conference*, vol. 11. Institute of Engineering, Tribhuvan University, Nepal, March 2022, pp. 28 – 35.
- [25] P. C. Prasad, A. Jaiswal, S. Shakya, and S. Singh, “Portfolio optimization: A study of nepal stock exchange,” in *Proceedings of International Conference on Sustainable Expert Systems*. Springer, 2021, pp. 659–672.
- [26] J. E. Granville, *Granville’s New Key to Stock Market Profits*. Pickle Partners Publishing, 2018.
- [27] S. Nison, *Japanese Candlestick Charting Techniques*. Penguin, 2001.
- [28] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [29] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.
- [30] E. Nikishin, P. Izmailov, B. Athiwaratkun, D. Podoprikin, T. Garipov, P. Shvechikov, D. Vetrov, and A. G. Wilson, “Improving stability in deep reinforcement learning with weight averaging,” in *Uncertainty in artificial intelligence workshop on uncertainty in Deep learning*, 2018.
- [31] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” *arXiv preprint arXiv:1511.05952*, 2015.