

**REPUBLIC OF TURKEY
AKDENİZ UNIVERSITY**



REINFORCEMENT LEARNING FOR STOCK MARKETS

Uğur HAZİR

INSTITUTE OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

MASTER THESIS

JUNE 2021

ANTALYA

**REPUBLIC OF TURKEY
AKDENİZ UNIVERSITY**



REINFORCEMENT LEARNING FOR STOCK MARKETS

Uğur HAZİR

INSTITUTE OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

MASTER THESIS

JUNE 2021

ANTALYA

REPUBLIC OF TURKEY
AKDENİZ UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

REINFORCEMENT LEARNING FOR STOCK MARKETS

Uğur HAZİR

DEPARTMENT OF COMPUTER ENGINEERING

MASTER THESIS

This thesis unanimously accepted by the jury on 29/06/2021.

Asst. Prof. Dr. Taner DANIŞMAN (Supervisor)

Assoc. Prof. Dr. Gıyasettin ÖZCAN

Asst. Prof. Dr. Hüseyin Gökhan AKÇAY

ÖZET

HİSSE SENETLERİ İÇİN PEKİŞTİRMELİ ÖĞRENME

Uğur HAZIR

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Taner DANIŞMAN

Haziran 2021; 78 sayfa

“Finans sektöründe hangi hisse senedinin ne zaman alınıp satılabileceğine ve ne zaman hamle yapılması gerektiği hakkında karar vermeyi sağlayan pek çok model vardır” (Hazır ve Danışman 2020) . Bu tezde Derin-Q-Ağı ajanının Borsa İstanbul için performansı test edilmiş olup Borsa İstanbul’da yer alan BIST30 endeks hisselerine odaklanılmıştır. Ham veriler Mynet web sitesinden Google Chrome gezgini üzerinde JavaScript fonksiyonlarından yararlanılarak elde edilmiş olup operasyonlardan önce işlenmiştir.

İlk önce; her bir hisse için Stokastik osilatör ve MACD değerleri hesaplanır. Bu hesaplamalar esnasında başlangıç verisi hesaplamalar için harcanır ve bu nedenle az miktarda veri hesaplamalar uğruna kaybedilir.

İkinci olarak; elde edilen veri, eğitim verisi ve test verisi olarak iki parçaya bölünür. Eğitim periyodu 08.03.2000 tarihinden başlar ki bu işlenmiş verinin başkangıç tarihidir ve 31.12.2014 tarihinde sona erer. Test periyodu ise 01.01.2015 tarihinde başlar ve 21.12.2019 tarihinde sona erer. Verilen herhangi bir günde her bir hisse işlem görmez. Bir hisse verilen günde tüm işlemlere kapatılmış olabilir. Bu nedenle eğitim ve test periyod süreleri her bir hissede eşit değildir.

Pekiştirmeli Öğrenme ajanı yaratmak için Derin-Q-Ağı metodu seçilmiştir. Her bir hisse için Q değerlerini hesaplamak üzere Derin-Q-Ağı üretilmiştir.

Program, Tensorflow kütüphanesini arka uç olarak kullanan Keras kütüphanesi kullanılarak Python programlama dili ile üretilmiştir.

Her bir Derin-Q-Ağına, günlük parametreler, Stokastik ve MACD parametreleri girdi olarak verilmiş olup; bu parametreler şunlardır: Kapanış, Düşük, Yüksek, Hacim, Yıl, EMA12, EMA26, MACD, MACDsinyal9, StokastikMax14, StokastikMin14, Stokastik-K14, StokastikD14, StokastikMax5, StokastikMin5, StokastikK5 ve StokastikD5. 17 adet girdi parametresi Derin-Q-Ağına verilir; ki bu parametreler veri giriş katmanını oluşturur. İlk gizli katmanda 540 nöron bulunur. İkinci gizli katmanda 180, üçüncü gizli katmanda 64, dördüncü gizli katmanda 32, beşinci gizli katmanda 8 nöron bulunur. Bütün gizli katmanlarda ReLU aktivasyon fonksiyonu kullanılır. Bir Derin-Q-Ağı için eylem alanı bekle, al ve sat eylemlerinden oluşur. Bu nedenle çıktı katmanında 3 nöron bulunur. Çıktı katmanında lineer aktivasyon fonksiyonu kullanılır. Lineer aktivasyon fonksiyonu nörondaki değerlerin hiç bir matematiksel operasyona maruz bırakılmadan doğrudan kullanılmasını sağlar. Tekrar Hafızası geçmiş tecrübeleri kullanarak bir sonraki eylemi tahmin etmek için kullanılır.

Eğitim periyodu için eğitim verisi girdi olarak verilir ve program her bir Derin-Q-Ağı için 5000 bölüm kadar çalıştırılır. Ajan 1000 TL anapara ile başlar ve daha zorlu bir ortam yaratmak adına her bir alım satım eyleminde 1 TL işlem ücreti öder. Ajan yeterince parası varsa ve al sinyali geldi ise satın alma işlemini gerçekleştirir. Ajan elinde hisse varsa ve sat sinyali geldiyse ya da yüzde beş kârlı durumda ise satma işlemini gerçekleştirir. Diğer durumlarda, bekle sinyali geldiyse hisseyi elinde tutar ya da elinde yoksa satın almayı bekler. Q değerlerini tahmin etmek için epsilon açgözlü stratejisini kullanırız. Başlangıçta Derin-Q-Ağı, keşif için daha rastgele hareket edecektir. Sonunda daha az rastgele eylemler yapacaktır çünkü öğrendiği deneyimden faydalanacaktır. 5000 bölüm sonunda keras model kaydetme fonksiyonu ile mevcut durum ve değerleri kaydederiz.

Son olarak, test verisi algoritmanın başarısını sınamak için programa girdi olarak verilir. Bu safhada bütün Derin-Q-Ağları eş zamanlı olarak çalıştırılır ve aynı zamanda benim ajanı daha fazla hamle yapmaya zorlamak için icat ettiğim Birleştirilmiş Derin-Q-Ağı metodu hesaplanır. Birleştirilmiş Derin-Q-Ağı metodu, gerçek bir Derin-Q-Ağı metodu değildir. Bu metodda verilen koşullarda en iyi performans sergileyen Derin-Q-Ağına ait hamleler takip edilir. Eğer Birleştirilmiş Derin-Q-Ağı ajanı elinde hisse yok ise kendisini al sinyali veren (eylem olarak al eylemi seçilmiş ise) Derin-Q-Ağına bağlar. Birleştirilmiş Derin-Q-Ağı ajanı eğer bağlandığı Derin-Q-Ağı ajanı sat sinyali verirse (eylem olarak

sat eylemi seçilmiş ise) veya elindeki hisse belirlenen oranda kâr ya da zararda ise bağ-landığı Derin-Q-Ağı ajanını salar. Birleştirilmiş Derin-Q-Ağı ajanı bir sonraki al sinyali- ni bekler ve verilen gün için kendisini en iyi performans gösteren Derin-Q-Ağına bağlar. Birleştirilmiş Derin-Q-Ağı ajanı bu şekilde bağlanma ve salma işlemleri yaparak süreci tekrar eder. Test aşamasında epsilon açgözlü stratejisi uygulanmaz; çünkü ajan öğrenme sürecini tamamlamıştır ve Q değerleri artık hesaplanmıştır. Hesaplanan bu Q değerlerinin üzerine yazmak istenmeyen bir durumdur.

Sonuç olarak, açıkça görülmektedir ki; ajan öğrendikçe iflas sayısı azalmakta ve son bakiye artmaktadır. Birleştirilmiş Derin-Q-Ağı metodu en iyi metod değildir; fakat eylem sayısını ve aynı zamanda riski de arttırmaktadır.

ANAHTAR KELİMELER: Birleştirilmiş Derin-Q-Ağı Metodu, BIST30, Borsa İstanbul, Derin-Q-Ağı, MACD, Pekiştirmeli Öğrenme, ReLU, Stokastik osilatör.

JÜRİ: Dr. Öğr. Üyesi Taner DANIŞMAN

Doç. Dr. Gıyasettin ÖZCAN

Dr. Öğr. Üyesi Hüseyin Gökhan AKÇAY

ABSTRACT

REINFORCEMENT LEARNING FOR STOCK MARKETS

Uğur HAZİR

MSc Thesis in Computer Engineering

Supervisor: Asst. Prof. Dr. Taner DANIŞMAN

June 2021; 78 pages

“There are many models at Finance sector to decide which stock to buy or sell and when to act” (Hazır and Danişman 2020, 1) . In this thesis performance of the Deep-Q Network agent for Borsa Istanbul (Istanbul Stock Exchange) is tested. We focused on BIST30 index stocks Borsa Istanbul (Istanbul Stock Exchange) Top 30 index stocks. Pure data is gathered from Mynet website by the help of JavaScript functions using a Google Chrome browser and data is processed before operations.

First of all, Stochastic oscillator and MACD values are calculated for each stock. Some small amount of initial data is lost and spend during those calculations.

At second, data is splitted into training data and test data for a stock. Training period starts at 08.03.2000 which is the beginning of the processed data and ends at 31.12.2014. Test period starts at 01.01.2015 and ends at 21.12.2019. Not all stock may be open to process during given day. A stock might be banned for all processes for a given day. So that the training and test day period size is not equal for every stocks.

Deep-Q Network methodology is chosen to create a Reinforcement Learning agent. For each stock a DQN is generated and run to calculate the Q values.

Program is produced using Python programming language by using Keras library which uses Tensorflow library as backend.

We give daily parameters, Stochastic and MACD values as input parameters to our DQNs which are : Close, Low, High, Volume, Year, EMA12, EMA26, MACD, MACDsignal9, StochasticMax14, StochasticMin14, StochasticK14, StochasticD14, StochasticMax5, StochasticMin5, StochasticK5 and StochasticD5. We give 17 input parameters to DQN which is our input layer. At first hidden layer there are 540 neurons. At second hidden layer there are 180 neurons. At third hidden layer there are 64 neurons. At forth hidden

layer there are 32 neurons. At fifth hidden layer there are 8 neurons. For all hidden layers ReLU activation function is used. For a DQN action space is wait, buy and sell so that we have 3 neurons at output layer. Output layer uses linear as activation function which means the value of the neuron is used directly without any mathematical operation. The Replay Memory is used to predict next action from previous experience.

For the training phase we give training data as input and run the program for 5000 episodes for each DQNs. The agent starts with 1000 TL initial money and every transaction costs 1 TL to create a harsh environment. The agent buys if it has enough money and has buy signal. It sells if it has 5 percent profit or sell signal when it has stocks. Otherwise; it waits or holds if wait action is selected as next step. We apply epsilon greedy strategy to forecast Q values. At the beginning DQN will act more randomly for exploration. At the end it will make less random actions because it has learned so it exploits. After 5000 episodes we save the states and values of the DQN by saving keras model.

Finally, test data is given as input to calculate the success of the algorithm. We load the last saved state of the DQNs and give the test data as input. At this phase we run all DQNs simultaneously and at the same time The Combined DQN Method (Hazır and Danişman 2020, 5) is calculated which is a process that I invented to force the agent to make more actions. It is not a real DQN but it follows the acts of the best performing DQN during the execution by using given conditions. If The Combined DQN agent has no stock, it attaches itself to the best performing DQN which has a buy signal (which produces buy action). The Combined DQN agent detaches itself if the attached DQN gives sell signal (which produces sell action) or the stock is at the specified rate loss or profit. The Combined DQN agent waits for the next buy action and attaches itself to the best performing DQN for a given day and it repeats the processes of attaching and detaching. Epsilon greedy strategy is not applied at testing phase because the agent is learned and calculated Q values already. It is not desirable if we overwrite those values.

In conclusion, it can be clearly seen that as the agent learns the number of bankruptcy drops and end balance is increase. The Combined DQN method is not the best method but it increases the number of operations and also the risk.

KEYWORDS: BIST30, Borsa Istanbul, Deep-Q Network, MACD, Reinforcement Learning, ReLU, Stochastic oscillator, The Combined DQN Method.

COMMITTEE: Asst. Prof. Dr. Taner DANIŞMAN

Assoc. Prof. Dr. Gıyasettin ÖZCAN

Asst. Prof. Dr. Hüseyin Gökhan AKÇAY



ACKNOWLEDGEMENTS

Firstly, I would like to thank my thesis supervisor and my advisor Asst. Prof. Dr. Taner DANIŞMAN for his support, encourage and advises.

Secondly, I would like to thank Prof. Dr. Melih GÜNAY for Artificial Intelligence lecture and Asst. Prof. Dr. Hüseyin Gökhan AKÇAY for Machine Learning lecture that they helped me a lot about learning the required basic concepts and knowledge for this thesis which led me progress forward.

Finally, I would like to thank my wife and sons for their patience and support.



LIST OF CONTENTS

ÖZET	i
ABSTRACT	iv
ACKNOWLEDGEMENTS	vii
TEXT OF OATH	x
LIST OF SYMBOLS AND ABBREVIATIONS	xi
LIST OF FIGURES	xiii
LIST OF TABLES	xiv
1. INTRODUCTION	1
2. LITERATURE REVIEW	5
2.1. The Moving Average Convergence Divergence (MACD)	5
2.2. Stochastic Oscillator	6
2.3. Markov Property	6
2.4. Markov Decision Process (MDP)	8
2.5. Reinforcement Learning	8
2.6. Q-Learning	9
2.6.1. Epsilon Greedy Strategy	10
2.7. Artificial Neural Networks (ANN)	11
2.7.1. Rectified Linear Unit (ReLU)	12
2.7.2. Adam Optimizer	13
2.8. Deep-Q Network (DQN)	14
2.8.1. Replay Memory	14
3. MATERIAL AND METHOD	15
3.1. Assumptions for the Model	15
3.2. Gathering Data	16
3.3. Preprocessing Data to Calculate MACD and Stochastic Values	20
3.4. Programming Environment	26
3.5. Creating DQN	26
3.6. Training Method	28
3.7. Test Method	30
3.8. The Combined DQN Method	33
4. RESULTS AND DISCUSSION	35
4.1. Results for the Training Phase	36

4.1.1. Training Phase for ARCLK	36
4.1.2. Training Phase for ASELS	38
4.1.3. Training Phase for SAHOL	40
4.1.4. Training Phase for TUPRS	42
4.2. Results for the Test Phase	44
4.2.1. Buy and Hold Strategy Calculation for the Test Phase	45
4.2.2. 14 Day Period Stochastic Oscillator Calculation for the Test Phase	45
4.2.3. Test 1	46
4.2.4. Test 2	48
4.2.5. Test 3	50
4.2.6. Test 4	52
4.3. Discussions of the Results	53
5. CONCLUSION	55
6. REFERENCES	56
CURRICULUM VITAE	

TEXT OF OATH

I declare this study “Reinforcement Learning For Stock Markets” which I present as master thesis, is in accordance with academic rules and ethical conduct. I also declared that I cited and referenced all material and results that are not original to this work.

29/06/2021

Uğur HAZIR



LIST OF SYMBOLS AND ABBREVIATIONS

Symbols:

C	: Close value of a stock
H	: High value of a stock
L	: Low value of a stock
n	: Number of days
t	: Time period
$\%K$	: A Stochastic Oscillator Index
$\%D$	: A Stochastic Oscillator Index
K	: Stochastic K Index
S	: Set of possible states
A	: Set of possible actions
T	: A transaction model for MDP
R	: Reward
A	: Set of possible actions
γ	: Discount rate
Q	: Q value
s	: State
a	: Action
α	: Learning rate
r	: A random variable
ϵ	: Probability of choosing to explore
i	: Given day
X	: Numpy array
f	: Function

Abbreviations:

AI	: Artificial Intelligence
BH	: Buy and Hold
BIST30	: Borsa Istanbul (Istanbul Stock Exchange) Top 30 index stocks
CSV	: Comma Separated Value
DQN	: Deep Q-Network
EMA	: Exponential Moving Average
MA	: Moving Average
MACD	: The Moving Average Convergence Divergence
MDP	: Markov Decision Processes
ML	: Machine Learning
NASAA	: North American Securities Administrators Association
ReLU	: Rectified Linear Unit
RL	: Reinforcement Learning
RSI	: Relative Strength Index
SMA	: Simple Moving Average
TL	: Turkish Lira

LIST OF FIGURES

Figure 2.1. A stochastic process model (Tribello, G. 2015)	7
Figure 2.2. Markov Property	7
Figure 2.3. Reinforcement Learning Agent and Environment Interaction	8
Figure 2.4. An Artificial Neuron (Gershenson, C. 2003, 2)	11
Figure 2.5. A Feed-forward ANN using back propagation	12
Figure 2.6. Adam performance comparison with other methods (Kingma, D. P. and Ba, J. 2017, 7)	13
Figure 3.7. Screenshot of ARCLK Historical data from Mynet website (Anonymous 6)	16
Figure 3.8. Screenshot of JavaScript commands executed on Google Chrome Developer Tools Console	19
Figure 3.9. Screenshot of JavaScript results on Google Chrome Developer Tools Console	19
Figure 3.10. The Artificial Neural Network to approximate the Q-values	27
Figure 4.11. Epsilon decays according to the Epsilon Greedy Strategy	36
Figure 4.12. SMA 100 End Balance for ARCLK	37
Figure 4.13. SMA 100 Number of Operations for ARCLK	37
Figure 4.14. SMA 100 Bankruptcy and Loss Rate for ARCLK	38
Figure 4.15. SMA 100 End Balance for ASELS	38
Figure 4.16. SMA 100 Number of Operations for ASELS	39
Figure 4.17. SMA 100 Bankruptcy and Loss Rate for ASELS	39
Figure 4.18. SMA 100 End Balance for SAHOL	40
Figure 4.19. SMA 100 Number of Operations for SAHOL	40
Figure 4.20. SMA 100 Bankruptcy and Loss Rate for SAHOL	41
Figure 4.21. SMA 100 End Balance for TUPRS	42
Figure 4.22. SMA 100 Number of Operations for TUPRS	42
Figure 4.23. SMA 100 Bankruptcy and Loss Rate for TUPRS	43

LIST OF TABLES

Table 4.1. Close values of Stocks at the Start and at the End of the Test Period . . .	44
Table 4.2. End Balance of Stocks for the Test Period according to BH Strategy . . .	45
Table 4.3. End Balance of Stocks for the Test Phase according to 14 day period Stochastic Oscillator	45
Table 4.4. Test 1 - 1000 TL initial investment Results	46
Table 4.5. Test 2 - 10000 TL initial investment Results	48
Table 4.6. Test 3 - 100000 TL initial investment Results	50
Table 4.7. Test 4 - 100000 TL initial investment Results with 7 percent loss sell rate only for the Combined DQN agent and 3 percent profit sell rate for both the DQN agents and the Combined DQN agent	52

1. INTRODUCTION

When the computers emerged their effects in economics was inevitable that they made significant changes (Backhouse, R. and Cherrier, B. 2016, 1). As the computational power increases the computers opened a new era for algorithmic trading. The speed and accuracy of the computers enabled to solve new problems by developing new techniques such as machine learning (Backhouse, R. and Cherrier, B. 2016, 13-17).

Technical analysis and Fundamental analysis are two main methods for trading decisions and predicting stock prices (Beyaz, E., Tekiner, F., Zeng, X. and Keane, J. 2018). Fundamental analysis is generally used for long-term investment and it focuses on financial strength, working capital and profitability of a company (Thomset, M. C. 2015, xiii, 38). On the other hand, technical analysis is used for short term investment and it focuses on market trends (Kirkpatrick, C. D. and Dahlquist, J. R. 2011, 9) which is based on market time interval data like: real-time or daily parameters such as Close and Volume.

By using Technical Analysis models day trading concept was available to investors. Day trading is a trading activity that is based on repeatedly buying and selling the same financial assets throughout a trading day and it is known as an extremely short-term investment strategy (Ryu, D. 2012, 2). According to an Analysis of NASAA for day trading and short-term trading from 26 accounts; seventy percent of the accounts lost money; only three accounts was profitable at short-term trading and the most successful account in the study, didn't apply day trading but only used short-term trading (NASAA, 1999, 53). According to the article named "The Profitability of Day Traders" 64 percent of the day traders lost money and only the 20 percent of the day traders are more than marginally profitable; but being among 20 percent requires three to five month learning period to survive (Jordan, D. J. and Diltz, J. D. 2003, 10).

According to investor analysis methods it is obvious that there is high probability of losing money but what about Technical Analysis methods and how successful they are. According to the article named "Performance of technical trading rules: evidence from Southeast Asian stock markets" when Technical analysis methods like MACD, Stochastic-D and RSI is applied to Southeast Asian stock markets and compared with simple Buy and Hold strategy they observed that transaction costs can eliminate trading

profits in most markets, in terms of market timing they found out that using technical indicators are not useful and profitable strategies such as MACD and STOCH-D could not predict market directions reliably. (Tharavanij, P., Siraprapasiri, V. and Rajchamaha, K. 2015, 2, 39).

Knowledge-based algorithmic trading methods helped us to increase the calculation speeds; but maybe only Machine Learning based methods can unleash profitable patterns that are yet unknown to people (Wang, Y., Wang, D., Zhang, S., Feng, Y., Li, S. and Zhou, Q. 2017). Machine Learning is part of Artificial Intelligence (Anonymous 1) that learns from data and makes predictions and/or decisions and is usually categorized as supervised, unsupervised, and reinforcement learning (Li, Y. 2018, 7).

Reinforcement Learning is selected as one of the MIT Technology Review 10 Breakthrough Technologies in 2013 and 2017 respectively (Li, Y. 2018, 5). RL consists of an agent which has the goal to maximize the rewards in the long run is the learner and the decision maker that interacts with the environment by selecting an action and environment responding to those actions by giving rewards or penalties and presenting new situations to the agent (Sutton, R. S. and Barto, A. G. 2014, 2015, 53,54,57).

Markov property refers to the memoryless property of a stochastic process that there is no dependence on previous states, actions or rewards and future states of a process depends only upon the present state that process is said to be a Markov process (Anonymous 2; John, C. and Watkins, C. H. 1989, 39). Markov Decision Process refers to a process that consists of four tuples: S as State Space, A stands for possible actions for each state, T stands for a transition function and R stands for a reward function (John, C. and Watkins, C. H. 1989, 37; Anonymous 3). Q-learning is the one of the most commonly used reinforcement learning method that is used for any Finite Markov Decision Process by calculating $Q(s,a)$ (Q state-action pair, Q (Quality) values) function using Bellman Equation where Q stands for the long-term value (expected future reward) of an action (Hu, J. 2016; Anonymous 4).

Q-learning is a very good model for a Finite Markov Decision Process; however it doesn't fit with stock analysis. The real world of stock markets is not finite and many different things can affect the performance of a stock; such as good amount of rainfall throughout the season for a farming company may increase the value of the stock, or

an unexpected fire at an offshore oil platform may cause a drop of the value of an oil refinery company that may indirectly affect a plastic toy factory. Also, we don't know all the states; because we don't know all the accounting data of the companies. It is also impossible to know what other investors willing to act and their acts will change the performance of the stock market. So that we can say that stock market technical analysis is a Partially Observable Markov Decision Process (Raval, S. 2018).

We have to use an advance technique to overcome to create a method for a successful technical analysis method for stock markets that is called Deep-Q Network or Deep-Q Learning. Q-learning is memoryless but the technical analysis requires memory to benefit from previous experiences. Calculations of Q values requires too much computation so we have to benefit from an approximation function. To overcome those limitations Deep-Q Learning benefits from Artificial Neural Networks and Experienced Replay Memory. The ANNs are a black box having multiple input and multiple output which operates using a large number of mostly parallel connected simple arithmetic units (Zupan, J. 1994, 2) called neurons that DQNs use ANN functions to approximate the Q values (Choudhary, A. 2019). Experienced Replay Memory is used to store the trajectory of the Markov decision process (MDP) that at each iteration of DQN, a mini-batch of states, actions, rewards, and next states are sampled to approximate the action-value function and to break the temporal dependency among the observations in training the deep neural network (Fan, J., Wang, Z., Xie, Y. and Yang, Z. 2020, 1,2).

There are already some previous studies that apply DQN for technical analysis which use a DQN agent on a single stock or asset but the major limitation of this approach is that it is not practical for a large portfolio of stocks, since the prices are continuous (Yang, H., Liu, X., Zhong, S. and Walid, A. 2020).

The main purpose of this thesis is to create a DQN agent which tries to maximize the end balance. The agent starts with limited initial investment money and transaction costs are applied for creating a harsh environment where it is hard to survive in a long period of time to simulate the real stock market world. 4 DQN agents for ARCLK, ASELS, SAHOL and TUPRS, which are BIST-30 index stocks, are created and trained. The daily parameters : Close, Low, High, Volume and Year are given as input. Also, to benefit from technical analysis methods Stochastic Oscillator and MACD parameters are given

as input. Unlike previous works 4 DQN agents tested simultaneously and The Combined DQN Methodology which I invented is applied to increase the number of transactions during the test phase.



2. LITERATURE REVIEW

This thesis based on creating a DQN Reinforcement Learning agent, which accepts daily and technical analysis parameters as input, for stock markets.

To achieve this goal, at initial step the input technical analysis parameters must be calculated. In this thesis, MACD and Stochastic Oscillator techniques are introduced as technical analysis methods.

After learning the scientific principles to prepare the input, it is required to learn the basic principles of a DQN Reinforcement Learning agent. To understand how a DQN agent is generated, it should be better to follow the light of the scientific literature. At first, the mathematics and the logic behind the Reinforcement Learning is introduced by the concept of the Markov Property and Markov Decision Process (MDP). At second, Reinforcement Learning itself is introduced. At third, Q-Learning which is one of the most known technique used for Reinforcement Learning is introduced. At this step, Epsilon Greedy Strategy, which is used by the Q-Learning agent, and the Exploration vs. Exploitation concept is introduced. At forth, we will deep dive to the ANNs, understand the principles of an Artificial Neuron, ReLU activation function to create the network and the Adam Optimizer algorithm to update the weights of the neural network. Finally we will complete our journey by the deep power of Neural Networks and the Experienced Replay Memory to upgrade Q-Learning technique to Deep-Q Network. So that the Deep-Q Network (DQN) and Replay Memory concept is introduced.

2.1. The Moving Average Convergence Divergence (MACD)

The Moving Average Convergence Divergence (MACD) was introduced by Gerald Appel (Appel, G. 1979) and this technique is a simple subtraction of two moving averages, it's mean-reverting around zero so that it is an oscillator, and MACD is also an indicator that measures the strength and direction of a trend in a stock's price (Anghel, G. D. I. 2015, 1416).

$$MACD(t_1, t_2) = MA(C_i, t_1) - MA(C_i, t_2) \quad (2.1)$$

At Equation 2.1, $MACD(t_1, t_2)$ means the MACD indicator which is the subtraction

of two series of moving averages with different periods which are t_1 and t_2 .

2.2. Stochastic Oscillator

Stochastic Oscillator was presented by George Lane in 1950s that this indicator is the combination of $\%K$ and $\%D$ lines and this method compares the difference of the closing prices between highest and lowest prices in a short period of time (Chootong, C., and Sornil, O. 2012, 205).

$$\%K = \frac{100 * C - L_{TimePeriod}}{H_{TimePeriod} - L_{TimePeriod}} \quad (2.2)$$

At Equation 2.2, C is the Close price of the given day which is multiplied by 100 at first and minimum Low price for the given Time Period is subtracted at second. Afterwards it is divided by the subtraction of the maximum High value and minimum Low value for the given time period.

$$\%D = \frac{K_1 + K_2 + K_3}{3} \quad (2.3)$$

At Equation 2.3, K_1 , K_2 and K_3 represents last three $\%K$ indices. $\%D$ is the average of the last 3 days of $\%K$ indices.

2.3. Markov Property

Markov property refers to the memoryless property of a stochastic process that there is no dependence on previous states, actions or rewards and future states of a process depends only upon the present state that process is said to be a Markov process (Anonymous 2; John, C. and Watkins, C. H. 1989, 39).

A stochastic process model is a mathematical model that takes measurements from the past and the present to make predictions about the future (Tribello, G. 2015).

To become a Markov property the past measurements, states, actions or rewards must not affect the future predictions. So that the future predictions only depend on the present states, actions or rewards.

Stochastic processes

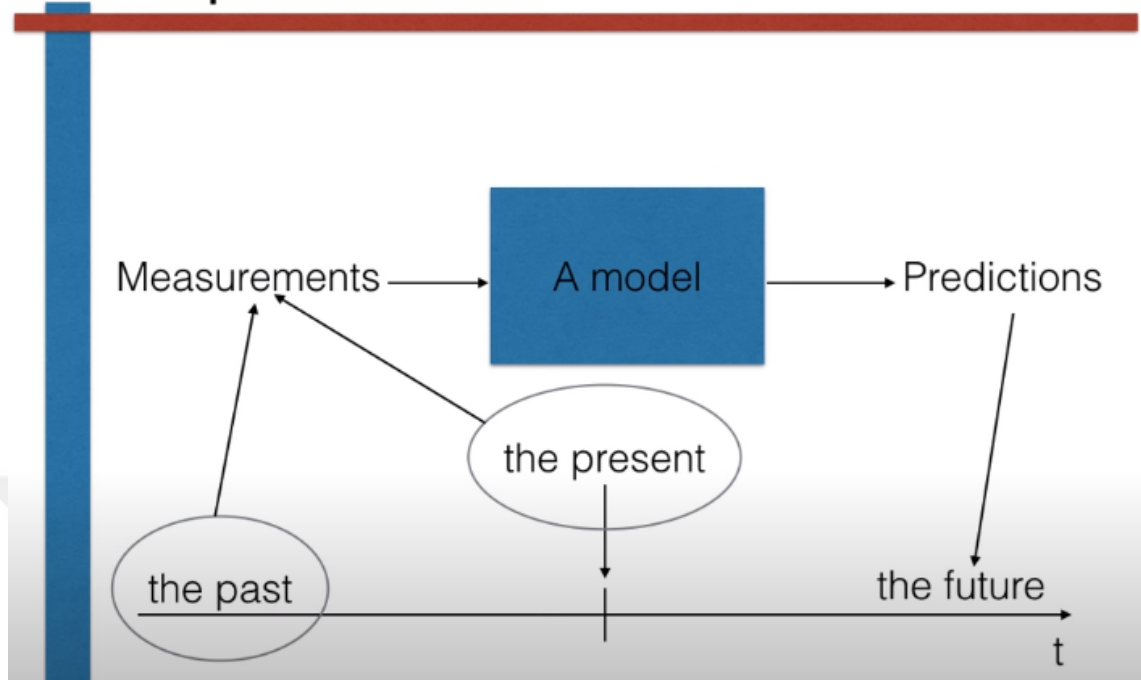


Figure 2.1. A stochastic process model (Tribello, G. 2015)

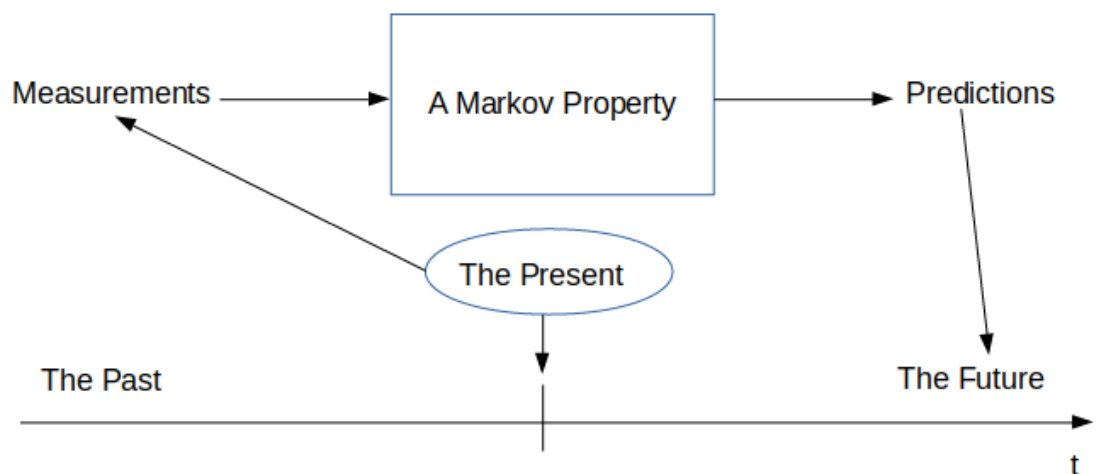


Figure 2.2. Markov Property

2.4. Markov Decision Process (MDP)

MDP is defined by S, A, T, R, γ where S is the set of possible states of the system and A is the set of possible actions, T represents the transition model, R corresponds to the reward structure while γ is the discount parameter which represents the relative value of future versus immediate rewards (LaMar, M. M. 2018).

2.5. Reinforcement Learning

Reinforcement Learning is a Machine Learning technique that it has an Agent which makes decisions and tries to maximize Rewards inside an Environment which responds to those actions by presenting new situations and Rewards to the Agent (Sutton, R. S. and Barto, A. G. 2014, 2015).

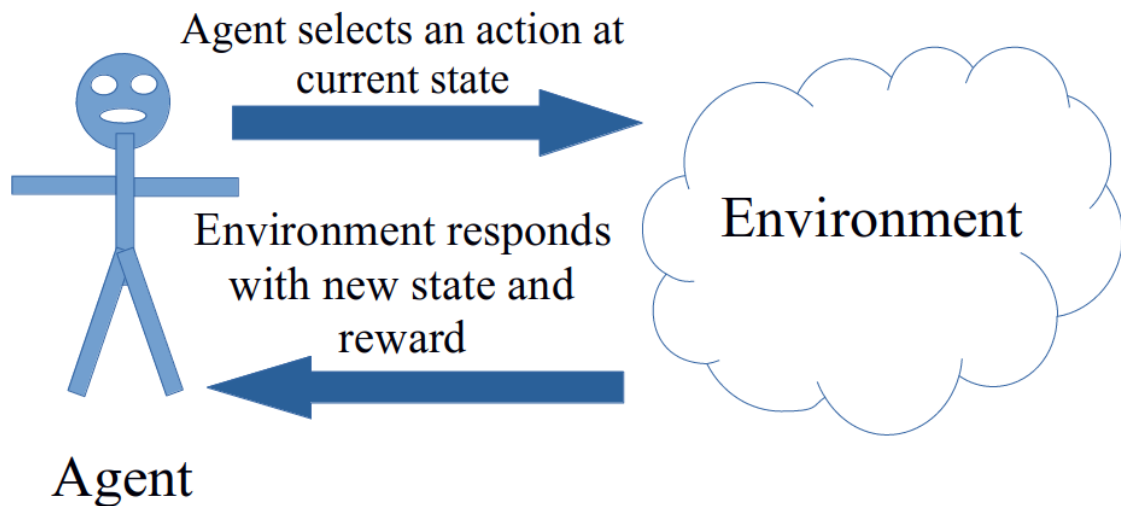


Figure 2.3. Reinforcement Learning Agent and Environment Interaction

2.6. Q-Learning

Q-learning is the one of the most commonly used reinforcement learning method that is used for any Finite Markov Decision Process by calculating $Q(s,a)$ (Q state-action pair, Q (Quality) values) function using Bellman equation where Q stands for the long-term value (expected future reward) of an action (Hu, J. 2016; Anonymous 4).

Before starting Q-learning process a matrix of states and actions need to be created which is called Q-table. Afterwards, the intersection values states vs actions, which is called Q-values, filled with a constant value. In general zero is selected as a constant value.

Because the Q-values initially set to zero, the Q-learning process starts by selecting a possible random action at initial state which returns new state and reward. Q-values are updated using Bellman equation at each step and the process run for a previously defined number of iterations.

$$Q_{new}(s_t, a_t) = Q(s_t, a_t) + \alpha [R(s_t, a_t) + \gamma \max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (2.4)$$

At Equation 2.4, which is known as Bellman equation, $Q_{new}(s_t, a_t)$ represents the updated Q value, $Q(s_t, a_t)$ represents current Q value, α is learning rate (which is generally set to 0.01 or 0.001 by programmers), $R(s_t, a_t)$ is the reward for the current state and action, γ is the discount rate that is used to decrease the effect of an expected future reward, $\max Q(s_{t+1}, a_{t+1})$ represents the maximum Q value expected to return from next state and possible next action.

Q-learning process ends when the current episode is equal to the number of iterations. So that selecting the number of iterations is critical. The number of iterations has to be chosen a high number to distinguish the Q-values clearly. So that the agent can act according to the Q-table directly after learning process. After learning process the agent will pick the action with highest Q-value for a given state.

2.6.1. Epsilon Greedy Strategy

Epsilon Greedy Strategy is a method in which ϵ value is assigned to 1, it decays through time and decaying process ends when it is close or equal to 0. A random variable r is selected and compared with ϵ and if $r > \epsilon$ then the reinforcement agent will choose its next action via exploitation otherwise it will choose its next action via exploration (Deeplizard 2018).

Because that ϵ is initially set to 1 the reinforcement agent will select a random possible action so that the agent will explore the environment. After the epsilon decaying period the reinforcement learning agent had enough knowledge through random actions the agent has experienced so that it will select the next action according to Q-table which is called exploitation.

In general, at DQN agents epsilon decays until 0.01. The advantage of this approach is because there is still with a one percent probability that there is a chance of moving randomly for the agent; the reinforcement agent keeps learning which makes the process a never-ending learning process.

2.7. Artificial Neural Networks (ANN)

According to McCulloch, W.S. and Pitts, W. (1943) it is possible to make a mathematical (logical) calculations of a set of neurons in a nervous system. The ANNs are a black box having multiple input and multiple output which operates using a large number of mostly parallel connected simple arithmetic units (Zupan, J. 1994, 2) called neurons. An artificial neuron mimics a natural neuron by receiving inputs with weights and applying an activation function which gives an output value (Gershenson, C. 2003, 2).

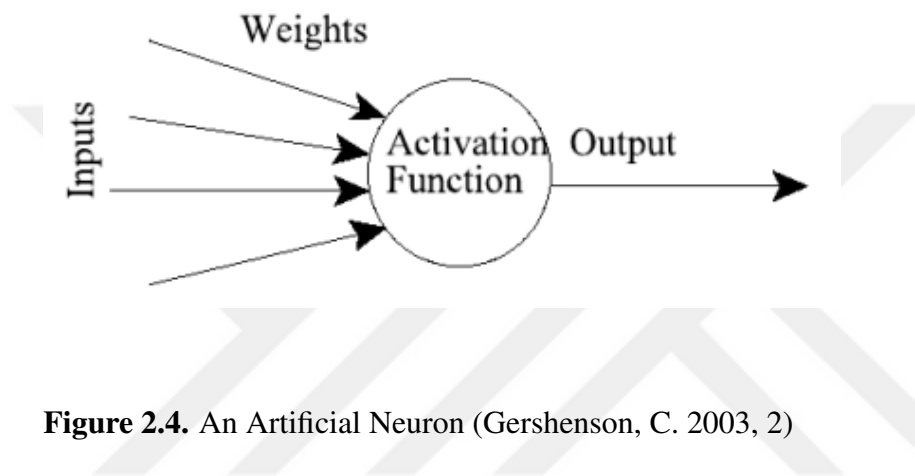


Figure 2.4. An Artificial Neuron (Gershenson, C. 2003, 2)

Feed-forward ANNs are neurons organized in layers in which network receives inputs by neurons in the input layer, send their signals to the forward layers that there may be one or more intermediate hidden layers, the output of the network is given by the neurons at output layer and then the errors are propagated backwards (Gershenson, C. 2003, 4). The weights of the neurons initially assigned to a random number and their value corrected during the back propagation phase. After the back propagation phase network calculates the forward process again. This forward calculation of the network and back propagation mechanism is repeated until ANN converges which means error is minimized and the system no longer makes it any better.

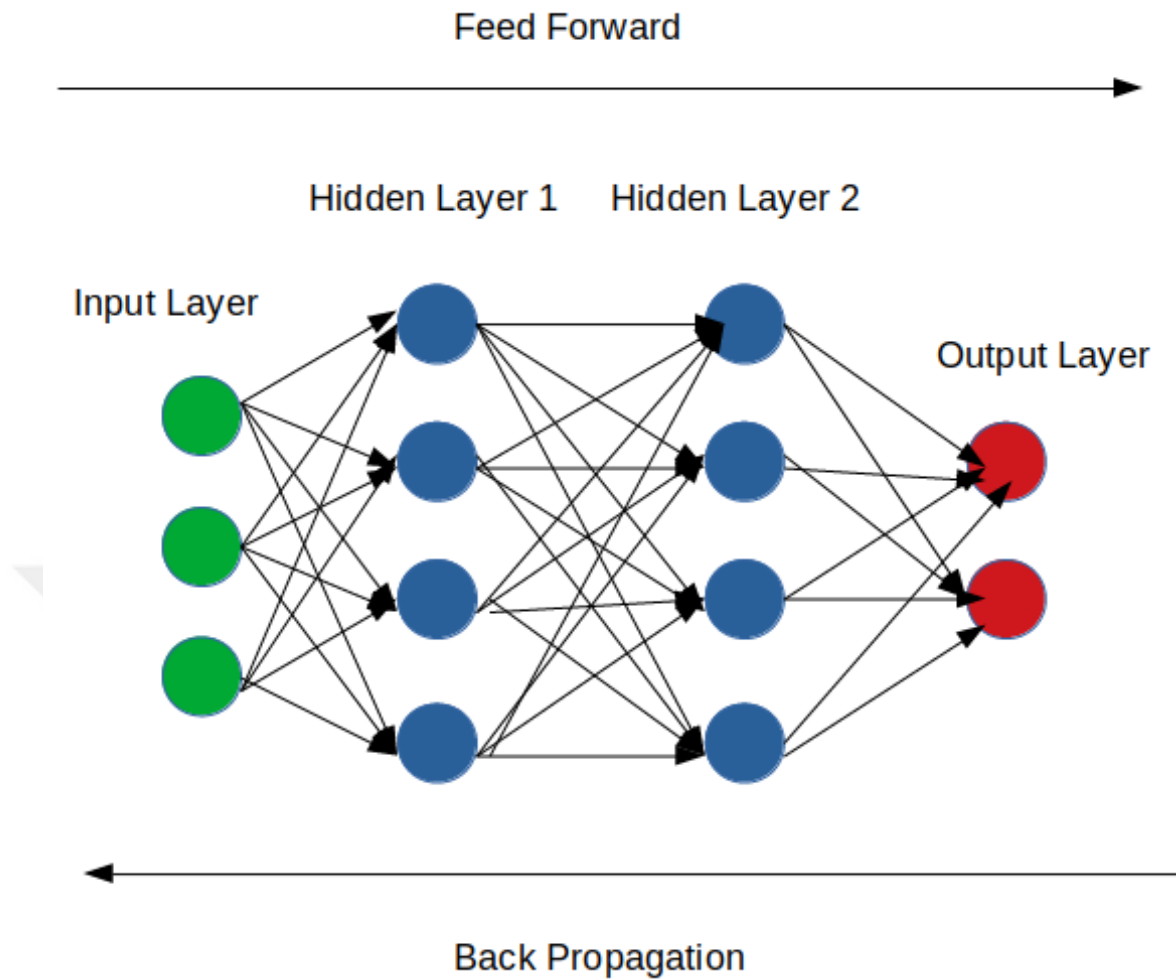


Figure 2.5. A Feed-forward ANN using back propagation

2.7.1. Rectified Linear Unit (ReLU)

ReLU is a non linear activation function that when it receives any negative input its output is zero, but when it receives any positive value its output is equal to value itself (Glorot, X., Bordes, A. and Bengio, Y. 2011).

$$f(x) = \max(0, x) \quad (2.5)$$

At Equation 2.5, which is known as ReLU function, x is given as input. It selects the maximum value of either zero or x itself. If the x is negative then the max function will select 0 as output. On the other hand when the x is positive x will be always greater than 0, so that the max function will select x itself as output result value.

2.7.2. Adam Optimizer

Adam, the name is derived from adaptive moment estimation, is a method which is a combination of AdaGrad (Adaptive Gradient Algorithm) and RMSProp (Root Mean Square Propagation) method that is designed for efficient stochastic optimization and it only requires first-order gradients with little memory requirement (Kingma, D. P. and Ba, J. 2017, 1).

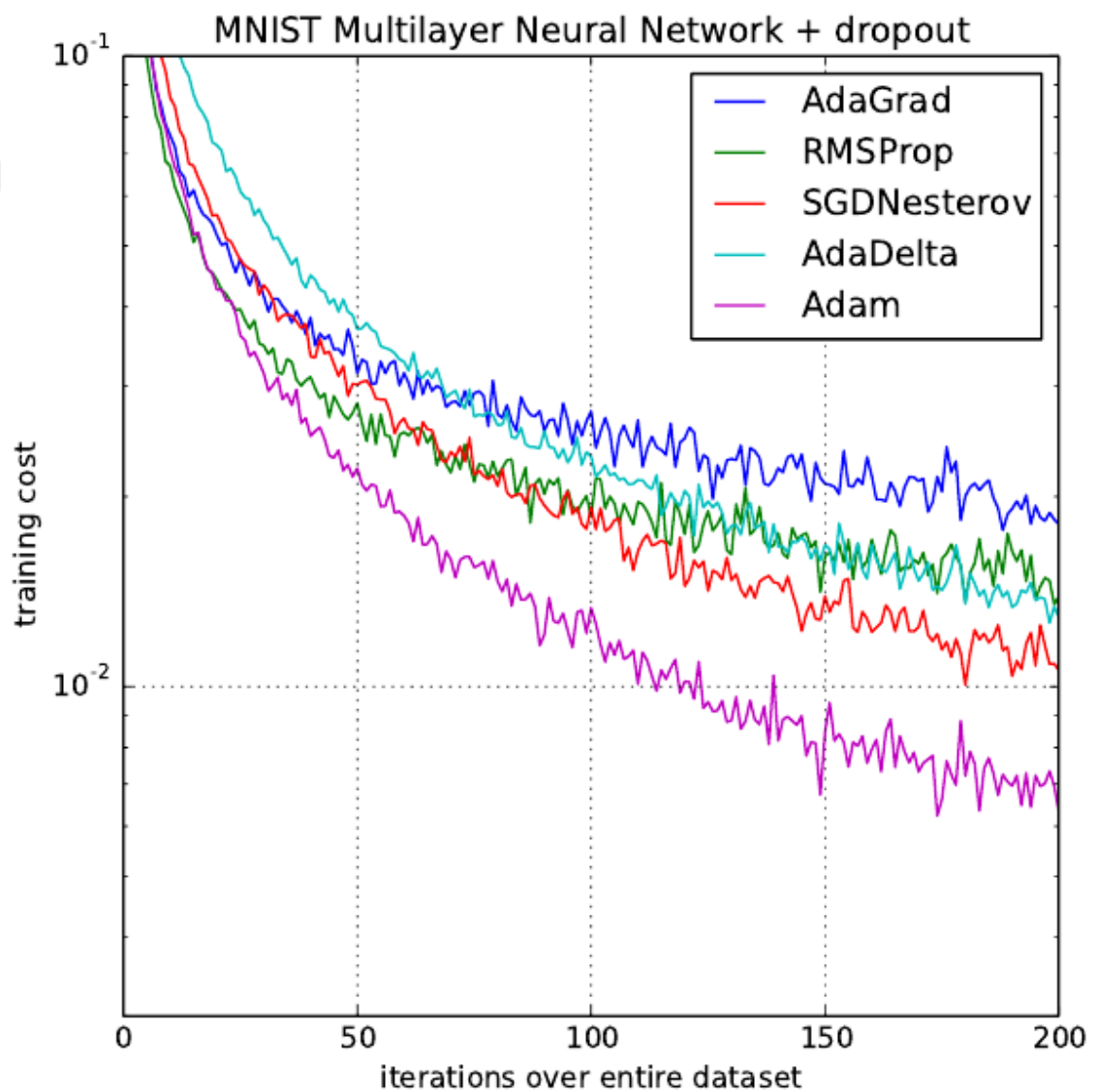


Figure 2.6. Adam performance comparison with other methods (Kingma, D. P. and Ba, J. 2017, 7)

2.8. Deep-Q Network (DQN)

DQN is designed to benefit from advancement at Deep Neural Networks which has a Reinforcement Q-Learning agent interacts with an environment but in this model optimal action-value function (Q-value) is approximated by a neural network and experience replay memory is introduced to store the trajectory of the Markov decision process (MDP) that at each iteration of DQN, a mini-batch of states, actions, rewards, and next states are sampled to approximate the action-value function and to break the temporal dependency among the observations in training the deep neural network (Mnih, V., Kavukcuoglu, K., Silver, D. et al 2015, 1), (Fan, J., Wang, Z., Xie, Y. and Yang, Z. 2020, 1,2).

2.8.1. Replay Memory

The experience replay memory is a fixed-size buffer that holds the most recent transitions (states, actions and rewards) collected by the policy which improves the sample efficiency of the algorithm and the stability of the network during training; because the data is used several times for training (Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., and Larochelle, H., Rowland, M. and Dabney, W. 2020, 2).

3. MATERIAL AND METHOD

In the scope of this thesis, a reinforcement learning agent which tries to maximize the end balance is designed using DQN technique. First of all, required data is collected from Mynet website by the help of JavaScript functions using a Google Chrome browser and preprocessed to create MACD and Stochastic parameters. Python programming language is used to create the program. Keras library which uses Tensorflow library as backend will be used to create the neural network. Then the model is trained. Finally 4 DQNs run simultaneously to apply the Combined DQN Method.

3.1. Assumptions for the Model

Because that the environment in this thesis is just a simulation of the real world; some assumptions has to be made to create a more realistic conditions.

First of all, the agent starts with a limited initial investment capital. To make conditions difficult for the agent the initial investment is selected as a small amount of money. This condition increase the possibility of bankrupting of the agent that the agent is likely to lose all the money at the beginning of the training period as it is observed in the real life conditions at the report of the NASAA (1999) and at the article named “The Profitability of Day Traders” (Jordan, D. J. and Diltz, J. D. 2003). For the train period the initial investment is assigned to 1000 TL. For the first test the initial investment is assigned to 1000 TL, for the second test to 10000 TL and for the third and forth test assigned to 100000 TL. If the end balance drops below 250 TL it is assumed that the agent is bankrupt. This value is determined through the observations at the training period if the agent’s end balance drops below 250 TL then it is unlikely to improve the financial situation.

At second, every transaction a cost is applied which is called transaction cost. For every buy or sell actions 1 TL transaction fee is applied for the training period.

At third, every buy and sell action price is equal to *Close* value of the operation day.

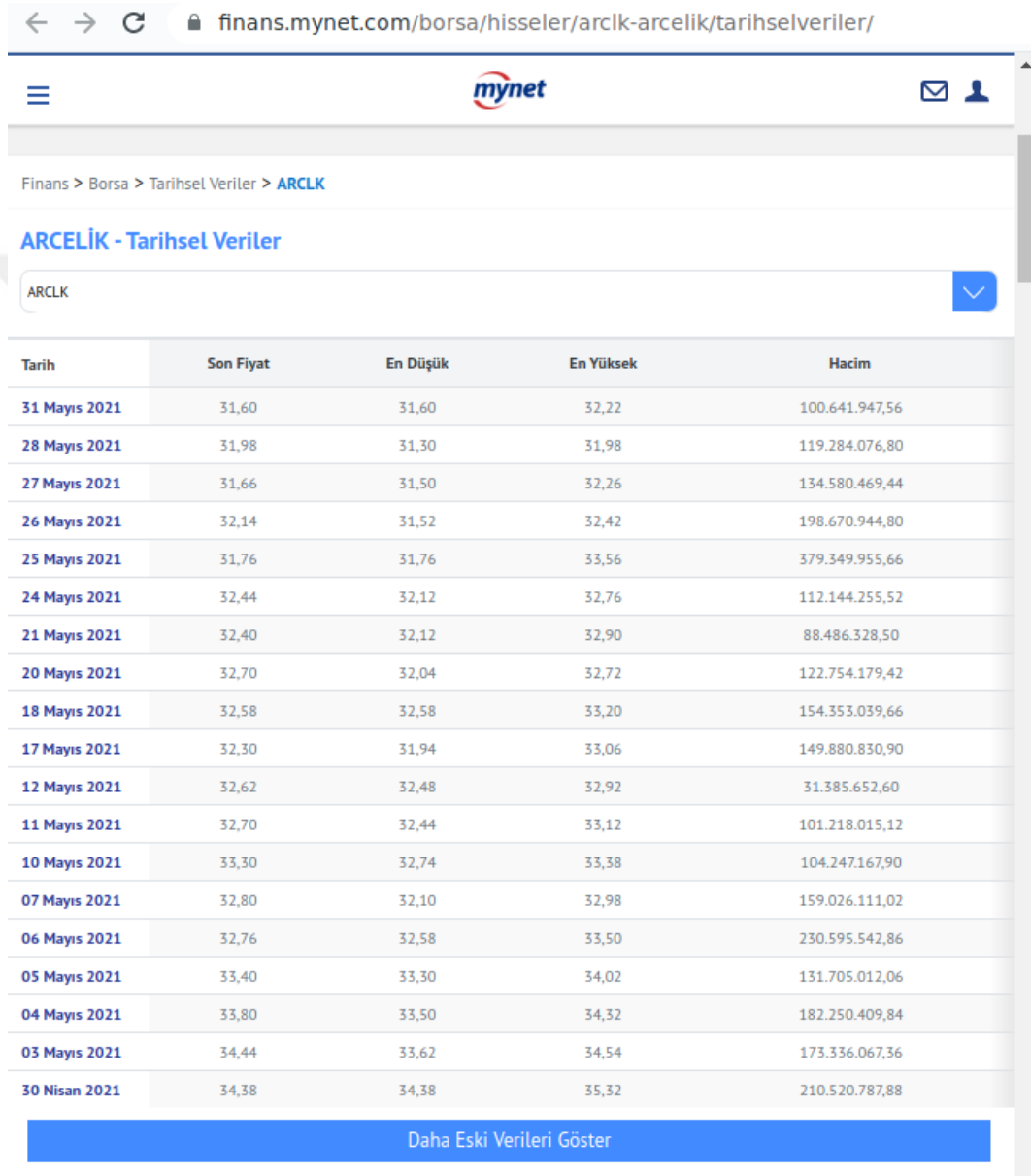
At forth, for the training period if the agent is at 5 percent profit it makes a sell action to benefit from the profit and to increase the number of transactions.

Finally, at the test period if the Combined DQN agent is at specified rate profit or loss then it makes a sell action to increase the number of transactions and increase to chance

of attaching to a better performing DQN for the given episode.

3.2. Gathering Data

Data is gathered from Mynet website for the stocks : ARCLK (Anonymous 6), ASELS (Anonymous 7), SAHOL (Anonymous 8) and TUPRS (Anonymous 9).



Tarih	Son Fiyat	En Düşük	En Yüksek	Hacim
31 Mayıs 2021	31,60	31,60	32,22	100.641.947,56
28 Mayıs 2021	31,98	31,30	31,98	119.284.076,80
27 Mayıs 2021	31,66	31,50	32,26	134.580.469,44
26 Mayıs 2021	32,14	31,52	32,42	198.670.944,80
25 Mayıs 2021	31,76	31,76	33,56	379.349.955,66
24 Mayıs 2021	32,44	32,12	32,76	112.144.255,52
21 Mayıs 2021	32,40	32,12	32,90	88.486.328,50
20 Mayıs 2021	32,70	32,04	32,72	122.754.179,42
18 Mayıs 2021	32,58	32,58	33,20	154.353.039,66
17 Mayıs 2021	32,30	31,94	33,06	149.880.830,90
12 Mayıs 2021	32,62	32,48	32,92	31.385.652,60
11 Mayıs 2021	32,70	32,44	33,12	101.218.015,12
10 Mayıs 2021	33,30	32,74	33,38	104.247.167,90
07 Mayıs 2021	32,80	32,10	32,98	159.026.111,02
06 Mayıs 2021	32,76	32,58	33,50	230.595.542,86
05 Mayıs 2021	33,40	33,30	34,02	131.705.012,06
04 Mayıs 2021	33,80	33,50	34,32	182.250.409,84
03 Mayıs 2021	34,44	33,62	34,54	173.336.067,36
30 Nisan 2021	34,38	34,38	35,32	210.520.787,88

Daha Eski Verileri Göster

Figure 3.7. Screenshot of ARCLK Historical data from Mynet website (Anonymous 6)

When using Google Chrome browser if F12 is pressed on a Windows or Linux computer Developer Tool opens at the right side of the window. It is possible to execute JavaScript commands at the Console tab inside the Developer Tool window.

At first, following JavaScript code is executed to correct Month values and make the values Turkish month values to make the data single language.

```
1 var replace_Months = $('body').html().replace(/January/g, 'Ocak').
    replace(/February/g, 'Şubat').replace(/March/g, 'Mart').replace(/
    April/g, 'Nisan').replace(/May\0131s/g, 'May').replace(/May/g, '
    Mayıs').replace(/June/g, 'Haziran').replace(/July/g, 'Temmuz').
    replace(/August/g, 'Ağustos').replace(/September/g, 'Eylül').
    replace(/October/g, 'Ekim').replace(/November/g, 'Kasım').replace
    (/December/g, 'Aralık');
2 $('body').html(replace_Months);
```

Algorithm 1. Replacing all Month values to Turkish by using JavaScript to make the data single language

At second, following JavaScript code is executed to collect all the historical data from the website source code and assigning it to an object named "data" (CertainPerformance 2020).

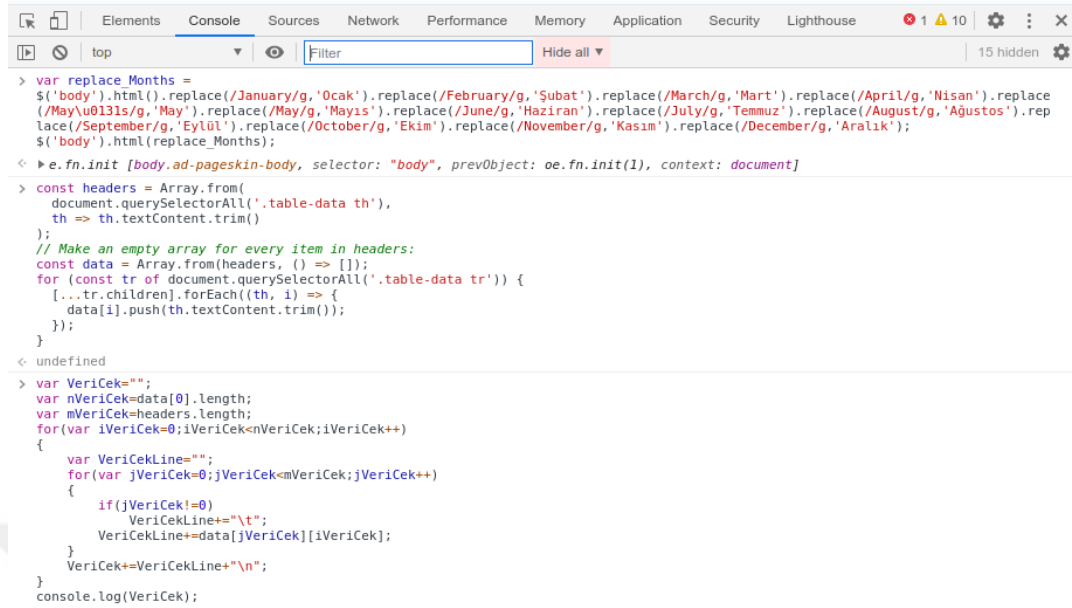
```
1 const headers = Array.from(
2   document.querySelectorAll('.table-data th'),
3   th => th.textContent.trim()
4 );
5 // Make an empty array for every item in headers:
6 const data = Array.from(headers, () => []);
7 for (const tr of document.querySelectorAll('.table-data tr')) {
8   [...tr.children].forEach((th, i) => {
9     data[i].push(th.textContent.trim());
10  });
11 }
```

Algorithm 2. Collecting all the historical data from the website source code and assigning it to an object named "data" (CertainPerformance 2020)

Finally, following JavaScript code is executed to print the data gathered from the historical data from the website source code. The printed result is copied and pasted to a text file. Afterwards it is converted by using LibreOffice Calc program to create a comma-separated values (.csv) file.

```
1 var VeriCek="";
2 var nVeriCek=data[0].length;
3 var mVeriCek=headers.length;
4 for(var iVeriCek=0;iVeriCek<nVeriCek;iVeriCek++)
5 {
6     var VeriCekLine="";
7     for(var jVeriCek=0;jVeriCek<mVeriCek;jVeriCek++)
8     {
9         if(jVeriCek!=0)
10             VeriCekLine+="\t";
11         VeriCekLine+=data[jVeriCek][iVeriCek];
12     }
13     VeriCek+=VeriCekLine+"\n";
14 }
15 console.log(VeriCek);
```

Algorithm 3. Printing the data gathered from the historical data from the website source code by using JavaScript



```

> var replace_Months =
  $('body').html().replace(/January/g, 'Ocak').replace(/February/g, 'Şubat').replace(/March/g, 'Mart').replace(/April/g, 'Nisan').replace(
  (/May\0131s/g, 'May').replace(/May/g, 'Mayıs').replace(/June/g, 'Haziran').replace(/July/g, 'Temmuz').replace(/August/g, 'Ağustos').rep
  lace(/September/g, 'Eylül').replace(/October/g, 'Ekim').replace(/November/g, 'Kasım').replace(/December/g, 'Aralık');
  $('body').html(replace_Months);

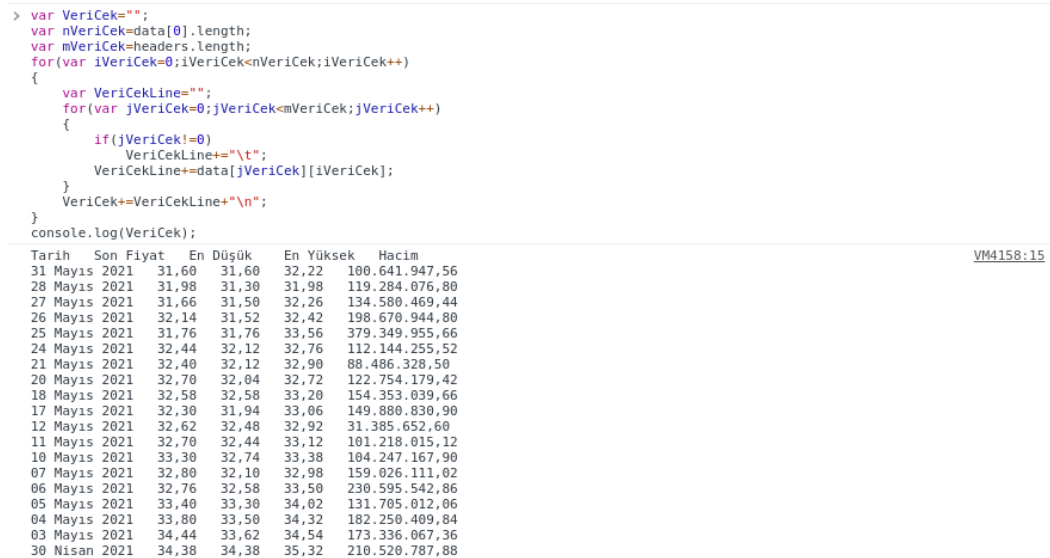
< ▶ e.fn.init [body.ad-pageskin-body, selector: "body", prevObject: oe.fn.init(1), context: document]

> const headers = Array.from(
  document.querySelectorAll('.table-data th'),
  th => th.textContent.trim()
);
// Make an empty array for every item in headers:
const data = Array.from(headers, () => []);
for (const tr of document.querySelectorAll('.table-data tr')) {
  [...tr.children].forEach((th, i) => {
    data[i].push(th.textContent.trim());
  });
}
< undefined

> var VeriCek="";
var nVeriCek=data[0].length;
var mVeriCek=headers.length;
for(var iVeriCek=0;iVeriCek<nVeriCek;iVeriCek++)
{
  var VeriCekLine="";
  for(var jVeriCek=0;jVeriCek<mVeriCek;jVeriCek++)
  {
    if(jVeriCek!=0)
      VeriCekLine+="\t";
    VeriCekLine+=data[jVeriCek][iVeriCek];
  }
  VeriCek+=VeriCekLine+"\n";
}
console.log(VeriCek);

```

Figure 3.8. Screenshot of JavaScript commands executed on Google Chrome Developer Tools Console



```

> var VeriCek="";
var nVeriCek=data[0].length;
var mVeriCek=headers.length;
for(var iVeriCek=0;iVeriCek<nVeriCek;iVeriCek++)
{
  var VeriCekLine="";
  for(var jVeriCek=0;jVeriCek<mVeriCek;jVeriCek++)
  {
    if(jVeriCek!=0)
      VeriCekLine+="\t";
    VeriCekLine+=data[jVeriCek][iVeriCek];
  }
  VeriCek+=VeriCekLine+"\n";
}
console.log(VeriCek);

```

Tarih	Son Fiyat	En Düşük	En Yüksek	Hacim
31 Mayıs 2021	31,60	31,60	32,22	100.641.947,56
28 Mayıs 2021	31,98	31,30	31,98	119.284.076,80
27 Mayıs 2021	31,66	31,50	32,26	134.580.469,44
26 Mayıs 2021	32,14	31,52	32,42	198.670.944,80
25 Mayıs 2021	31,76	31,76	33,56	379.349.955,66
24 Mayıs 2021	32,44	32,12	32,76	112.144.255,52
21 Mayıs 2021	32,40	32,12	32,90	88.486.328,50
20 Mayıs 2021	32,70	32,04	32,72	122.754.179,42
18 Mayıs 2021	32,58	32,58	33,20	154.353.039,66
17 Mayıs 2021	32,30	31,94	33,06	149.880.830,90
12 Mayıs 2021	32,62	32,48	32,92	31.385.652,60
11 Mayıs 2021	32,70	32,44	33,12	101.218.015,12
10 Mayıs 2021	33,30	32,74	33,38	104.247.167,90
07 Mayıs 2021	32,80	32,10	32,98	159.026.111,02
06 Mayıs 2021	32,76	32,58	33,50	230.595.542,86
05 Mayıs 2021	33,40	33,30	34,02	131.705.012,06
04 Mayıs 2021	33,80	33,50	34,32	182.250.409,84
03 Mayıs 2021	34,44	33,62	34,54	173.336.067,36
30 Nisan 2021	34,38	34,38	35,32	210.520.787,88

Figure 3.9. Screenshot of JavaScript results on Google Chrome Developer Tools Console

3.3. Preprocessing Data to Calculate MACD and Stochastic Values

MACD is the subtraction of 26 day period of EMA from 12 day period of EMA. To calculate MACD it is required to calculate the EMA values at first.

$$EMA_i = C_i \frac{2}{t+1} + EMA_{i-1} \left(1 - \frac{2}{t+1}\right) \quad (3.6)$$

At Equation 3.6 (Anonymous 5), t stands for time period interval, i is the given day, $i-1$ represents previous day and C_i is the Close price for the given day. By using Equation 3.6, EMA_{12} and EMA_{26} parameters are calculated by assigning 12 and 26 to t (time period) value.

$$MACD_i = EMA_{12_i} - EMA_{26_i} \quad (3.7)$$

At Equation 3.7, it can be easily understood that $MACD$ value of a given (i th) day is just the subtraction of EMA_{26} value from EMA_{12} value for the given day.

$$signal_i = MACD_i \frac{2}{t+1} + signal_{i-1} \left(1 - \frac{2}{t+1}\right) \quad (3.8)$$

At Equation 3.8 (Anonymous 5), $MACDsignal9$ indicator is calculated by setting Time Period t to 9 where i stands for the given day and $i-1$ represents previous day.

The pure data, which is obtained from Mynet website via JavaScript code, **is in reverse order**. The closest day is on the top and the past day is on the bottom of the csv file. So that to obtain the preprocessed data this condition should be taken into consideration. The pure data at csv file has *Date*, *Close*, *Low*, *High*, *Volume* and *Year* parameters.

```
1 import numpy as np # linear algebra
2 import pandas as pd #For reading data from file
3
4 df=pd.read_csv(stockFileName)
5 X=df.loc[:, "Date": "Year"].values
6
7 n=X.shape[0]
8 Xmacd=np.zeros((n,4)) #EMA12, EMA26, MACD, MACDsignal9
```

Algorithm 4. Start point for gathering pure data and make the code ready for calculating MACD parameters using Python language

At Algorithm 4, at first the csv file is read and assigned to *df* (dataframe) object and then the values are assigned to the *X* in a numpy array type. *n* stands for the number of days the data contains. *Xmacd* is the two dimensional numpy array that holds MACD parameters for each day. These MACD parameters are :*EMA12*, *EMA26*, *MACD* and *MACDsignal9*.

```

10 #EMA12
11 avgMacd12=0
12 for k in range(n-12):
13     index=n-13-k
14     if(k==12):
15         avgMacd12=(X[index+1,1]+X[index+2,1]+X[index+3,1]+X[index
            +4,1]+X[index+5,1]+X[index+6,1]+X[index+7,1]+X[index
            +8,1]+X[index+9,1]+X[index+10,1]+X[index+11,1]+X[index
            +12,1])/12
16         Xmacd[index,0]=(X[index,1]*2/13)+(avgMacd12*(1-(2/13)))
17     else:
18         Xmacd[index,0]=(X[index,1]*2/13)+(Xmacd[index
            -1,0]*(1-(2/13)))

```

Algorithm 5. Calculating EMA12 parameter

At Algorithm 5, at first an index parameter is generated because the data is in reverse order, at second because it is not possible to calculate values for the oldest 12 days are skipped, *avgMacd12* refers to the average of the skipped oldest 12 days *Close* parameter values to calculate the first *EMA12* value which is assigned to *Xmacd[index,0]* array value. After obtaining the first *EMA12* value it is easy to calculate next *EMA12* value by adding $X[index,1] * 2/13$ which is derived from the Equation 3.6.

```

20 #EMA26
21 avgMacd26=0
22 for k in range(n-26):
23     index=n-27-k
24     if(k==26):
25         avgMacd26=(X[index+1,1]+X[index+2,1]+X[index+3,1]+X[index
26             +4,1]+X[index+5,1]+X[index+6,1]+X[index+7,1]+X[index
27             +8,1]+X[index+9,1]+X[index+10,1]+
28                 X[index+11,1]+X[index+12,1]+X[index+13,1]+X[
29                     index+14,1]+X[index+15,1]+X[index+16,1]+X[
30                         index+17,1]+X[index+18,1]+X[index+19,1]+X[
31                             index+20,1]+
32                     X[index+21,1]+X[index+22,1]+X[index+23,1]+X[
33                         index+24,1]+X[index+25,1]+X[index+26,1])/26
34         Xmacd[index,1]=(X[index,1]*2/27)+(avgMacd26*(1-(2/27)))
35     else:
36         Xmacd[index,1]=(X[index,1]*2/27)+(Xmacd[index
37             -1,1]*(1-(2/27)))

```

Algorithm 6. Calculating EMA26 parameter

At Algorithm 6, the same process is applied to calculate the *EMA12*; but this time *t* (Time Period) set to 26 according to the Equation 3.6.

```

32 #MACD
33 for k in range(n-27):
34     index=n-28-k
35     Xmacd[index,2]=Xmacd[index,0]-Xmacd[index,1]

```

Algorithm 7. Calculating MACD parameter

At Algorithm 7, because *EMA12* and *EMA26* parameters are calculated it is easy to calculate the *MACD* parameter which is the subtraction of *EMA26* from *EMA12* according to the Equation 3.7. *Xmacd[index,0]* stands for *EMA12*, *Xmacd[index,1]* is *EMA26* and *Xmacd[index,2]* is the *MACD* parameter.

```

37 #MACDsignal9
38 avgMACD9=0
39 for k in range(n-36):
40     index=n-37-k
41     if(k==36):
42         avgMACD9=(Xmacd[index+1,2]+Xmacd[index+2,2]+Xmacd[index
43                     +3,2]+Xmacd[index+4,2]+Xmacd[index+5,2]+Xmacd[index
44                     +6,2]+Xmacd[index+7,2]+Xmacd[index+8,2]+Xmacd[index
45                     +9,2])/9
46         Xmacd[index,3]=(Xmacd[index,2]*2/10)+(avgMACD9*(1-(2/10)))
47     else:
48         Xmacd[index,3]=(Xmacd[index,2]*2/10)+(Xmacd[index
49                     -1,3]*(1-(2/10)))

```

Algorithm 8. Calculating MACDsignal9 parameter

At Algorithm 8, to calculate the first *MACDsignal9* parameter it is required to calculate the average *MACD* value of the skipped last additional 9 day parameters ($n - 36$ refers to that condition) and assigned to *avgMACD9* variable. After that *MACD* signal value is calculated using the Equation 3.8.

To calculate Stochastic parameters first max and min values for a given period has to be observed. *StockhasticMax14* is the maximum *High* value for the last 14 days and *StockhasticMin14* is the minimum *Low* value for the last 14 days. Likewise; *StockhasticMax5* is the maximum *High* value for the last 5 days and *StockhasticMin5* is the minimum *Low* value for the last 5 days.

StockhasticK14 and *StockhasticK5* indices calculated using Equation 2.2.

StockhasticD14 and *StockhasticD5* indices calculated using Equation 2.3.

```

47 %% Stockhastic Parameters
48 Xstoch=np.zeros((n,8)) #StockhasticMax14,StockhasticMin14,
    StockhasticK14,StockhasticD14,StockhasticMax5,StockhasticMin5,
    StockhasticK5,StockhasticD5

```

Algorithm 9. Calculating MACDsignal9 parameter

At Algorithm 9, *Xstoch* is the two dimensional numpy array that holds Stochastic parameters for each day. These Stochastic parameters are : *StockhasticMax14*, *StockhasticMin14*, *StockhasticK14*, *StockhasticD14*, *StockhasticMax5*,

StockhasticMin5, StockhasticK5 and StockhasticD5.

```

50 for k in range(n-14):
51     index=n-15-k
52     minStoch14=0
53     maxStoch14=0
54     for l in range(14):
55         if(l==0):
56             minStoch14=X[index+1,2]
57             maxStoch14=X[index+1,3]
58         else:
59             if(X[index+1,2]<minStoch14):
60                 minStoch14=X[index+1,2]
61             if(X[index+1,3]>maxStoch14):
62                 maxStoch14=X[index+1,3]
63     Xstoch[index,0]=maxStoch14
64     Xstoch[index,1]=minStoch14
65     if(maxStoch14-minStoch14!=0):
66         Xstoch[index,2]=100*(X[index,1]-minStoch14)/maxStoch14-
            minStoch14
67     if(k>1):
68         Xstoch[index,3]=(Xstoch[index,2]+Xstoch[index+1,2]+Xstoch[
            index+2,2])/3

```

Algorithm 10. Calculating StockhasticMax14, StockhasticMin14, StockhasticK14 and StockhasticD14 parameters

At Algorithm 10, an index variable is assigned because the data is in reverse order. $Xstoch[index, 0]$ stands for *StockhasticMin14* and $Xstoch[index, 1]$ stands for *StockhasticMax14*. $Xstoch[index, 2]$ stands for *StockhasticK14* and calculated according to Equation 2.2. $Xstoch[index, 3]$ stands for *StockhasticD14* and calculated according to Equation 2.3.

```

70 for k in range(n-5):
71     index=n-6-k
72     minStoch5=0
73     maxStoch5=0
74     for l in range(5):
75         if(l==0):
76             minStoch5=X[index+1,2]
77             maxStoch5=X[index+1,3]
78         else:
79             if(X[index+1,2]<minStoch5):
80                 minStoch5=X[index+1,2]
81             if(X[index+1,3]>maxStoch5):
82                 maxStoch5=X[index+1,3]
83     Xstoch[index,4]=maxStoch5
84     Xstoch[index,5]=minStoch5
85     if(maxStoch5-minStoch5!=0):
86         Xstoch[index,6]=100*(X[index,1]-minStoch5)/maxStoch5-
            minStoch5
87     if(k>1):
88         Xstoch[index,7]=(Xstoch[index,6]+Xstoch[index+1,6]+Xstoch[
            index+2,6])/3

```

Algorithm 11. Calculating StockhasticMax5, StockhasticMin5, StockhasticK5 and StockhasticD5 parameters

At Algorithm 11, an index variable is assigned because the data is in reverse order. $Xstoch[index, 4]$ stands for *StockhasticMin5* and $Xstoch[index, 5]$ stands for *StockhasticMax5*. $Xstoch[index, 6]$ stands for *StockhasticK5* and calculated according to Equation 2.2. $Xstoch[index, 7]$ stands for *StockhasticD5* and calculated according to Equation 2.3.

After calculating MACD and Stochastic parameters, all the data (daily parameters, MACD and Stochastic parameters) written into a new csv file in a correct order to generate input data. Input data consists of these parameters : Close, Low, High, Volume, Year, EMA12, EMA26, MACD, MACDsignal9, StochasticMax14, StochasticMin14, StochasticK14, StochasticD14, StochasticMax5, StochasticMin5, StochasticK5 and StochasticD5.

3.4. Programming Environment

JavaScript is a high-level, often just-in-time compiled, and multi-paradigm programming language that conforms to the ECMAScript specification which is the most popular programming language in the world (DeGroat, T.J. 2019), (Anonymous 10). In this thesis, JavaScript is used for gathering pure data from Mynet website using Google Chrome browser.

Python is an interpreted, high-level, object-oriented, multi-paradigm and a structured programming language which has many libraries for machine-learning (Mehta, R. 2019), (Anonymous 11). Python is used for creating the DQN reinforcement learning agent.

TensorFlow is a free and open-source software library for machine learning which is developed by the Google Brain team. (Yegulalp, S. 2019), (Anonymous 12). In this thesis, it is used for backend for the Keras library.

Keras is an open-source deep learning API written in Python which uses Tensorflow library as backend and its core structure is based on layers and models (Anonymous 13), (Anonymous 14). At this thesis, it is used for designing ANN to approximate Q-values which is an important part of the creating a DQN agent.

3.5. Creating DQN

Creating DQN mechanism is the goal of this thesis. At first, the preprocessed data is splitted into training data and test data for a stock. Training period starts at 08.03.2000 which is the beginning of the processed data and ends at 31.12.2014. Test period starts at 01.01.2015 and ends at 21.12.2019. Input data consists of 17 parameters which are : Close, Low, High, Volume, Year, EMA12, EMA26, MACD, MACDsignal9, StochasticMax14, StochasticMin14, StochasticK14, StochasticD14, StochasticMax5, StochasticMin5, StochasticK5 and StochasticD5.

Input parameters given to ANN of the DQN at input layer to approximate the Q-values. At first hidden layer there are 540 neurons. At second hidden layer there are 180 neurons. At third hidden layer there are 64 neurons. At forth hidden layer there are 32 neurons. At fifth hidden layer there are 8 neurons. For all hidden layers ReLU activation function is used. For a DQN action space is wait, buy and sell so that we have 3 neurons

at output layer. Output layer uses linear as activation function what means the value of the neuron is used directly without any mathematical operation. Adam optimizer with learning rate of 0.001 is used.

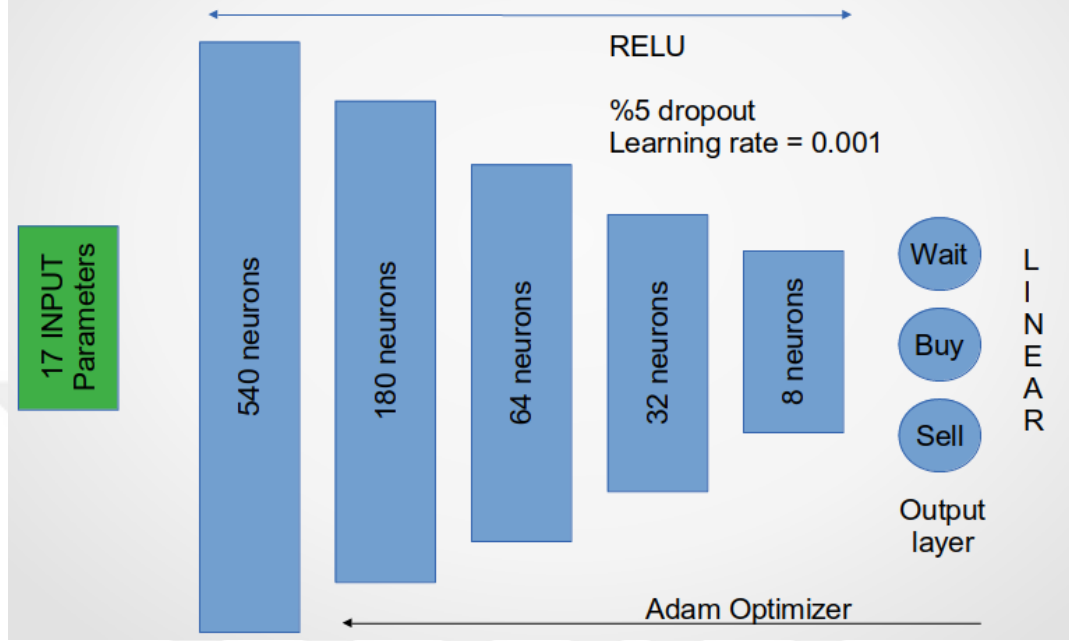


Figure 3.10. The Artificial Neural Network to approximate the Q-values

The Replay Memory is used to predict next action from previous experience which is set to 1000. Batch size is assigned as 32 which means random 32 experience will be selected from the memory. γ is set to 0.95 which is the discount rate and defined at Equation 2.4 which is known as Bellman equation.

3.6. Training Method

At training phase DQN agent starts with 1000 TL initial capital. The training phase consists of 5000 episodes.

```

1 def adaptiveEGreedy(self):
2     if self.epsilon > self.epsilon_min:
3         self.epsilon *= self.epsilon_decay

```

Algorithm 12. Decaying the ϵ according to the Epsilon Greedy Strategy

Epsilon Greedy Strategy is used at training phase. Epsilon (ϵ) is set to 1 at the start of the training phase decays by multiplying *epsilon_decay* variable which is assigned to 0.995; until 0.01 according to the Algorithm 12.

```

1 def act(self, state):
2     state = np.array(state)
3     if np.random.rand() <= self.epsilon:
4         return random.randrange(self.action_size)
5     act_values = self.model.predict(state)
6     return np.argmax(act_values[0])

```

Algorithm 13. Application of the Epsilon Greedy Strategy when selecting an action

The agent explores the environment by random actions at the beginning of the training phase if random variable $r \leq \epsilon$, after ϵ decays agent acts according to the Q-values which is approximated by ANN model according to the Algorithm 13.

```

1  for e in range(1,episode_count+1):
2      #initialize environment
3      state = env.reset()
4
5      total_reward=0
6
7      for time in range(numberOfDays):
8          action = agent.act(state) #select an action
9          next_state, reward, done = env.step(action)
10
11         #remember / storage
12         agent.remember(state, action, reward, next_state, done)
13
14         #update state
15         state = next_state
16
17         #replay
18         agent.replay(batch_size)
19
20         total_reward+=reward
21
22         if done:
23             break
24         #decay Epsilon until 0.01
25         agent.adaptiveEGreedy()

```

Algorithm 14. TrainingAlgorithm

According to the Algorithm 14, training phase starts with a for loop which is an iteration for 5000 episodes. For each episode, initially environment is reset to initial state and total_reward is set to 0.

Then for each possible work day in training period: at first we select an action. Action is selected according to Algorithm 13 so that Epsilon Greedy Strategy is applied. At second, we apply the selected action by step function which returns the tuple consist of *next_state*, *reward* and *done* values. *done* value is true if the agent bankrupts or it finishes the episode successfully. We store the current states and elements of the agent at memory. The state is updated to the next state which means moving agent to the next day. Then

according to replay memory ANN is trained to calculate the new q-values. Finally, if it is done we break end the episode. After the episode is end ϵ is decayed until the value of 0.01.

At training phase, a reward with the profit value is given after sell operation. Also if the agent bought stock but holding it -0.2 point penalty is applied and if the agent has no stock but also not buying -0.5 point penalty is applied to encourage the agent to make a transaction.

3.7. Test Method

There are 4 tests experimented during this thesis. At first three tests profit and loss sell rate is assigned to 5 percent for the Combined DQN Method and 5 percent profit sell rate is assigned for the DQN agents. At first test initial capital is set to 1000 TL with a constant 1 TL transaction cost. At second test initial capital is set to 10000 TL with a constant 1 TL transaction cost. At third test initial capital is set to 100000 TL with a 0.0018 transaction cost rate which is used to calculation of the real cost by multiplying the number of stocks bought with the Close price of the day. At forth test initial capital is set to 100000 TL with a 0.0018 transaction cost rate is which is the same as third test; however at this test loss sell rate is assigned to 7 percent for the Combined DQN Method and profit sell rate is assigned to 3 percent both for the DQN agents and the Combined DQN Method. Test phase took 100 episodes. During the test phase 4 DQNs run simultaneously to calculate the Combined DQN Method.

At test phase, a reward with the profit value is given after sell operation. Also if the agent bought stock but holding it -0.2 point penalty is applied and if the agent has no stock but also not buying -0.1 point penalty is applied to encourage the agent to make a transaction.

```

1   for e in range(1,episode_count+1):
2       #initialize and reset environment
3       for i in range(stockSize):
4           states[i] = envs[i].reset()
5           total_rewards[i]=0
6           isDone[i]=False
7       c.reset()
8       DateCurrent=DateStart
9       while DateCurrent!=DateEnd:
10          isNoAction=True
11          for i in range(stockSize):
12              isDate[i]=False
13              date[i]=envs[i].data.iloc[envs[i].t,0]
14              isDate[i]=date[i]==DateStr
15              isNoAction=isNoAction and isDate[i]==False
16          for i in range(stockSize):
17              if isDate[i]==True:
18                  #select an action
19                  actions[i] = agents[i].act(states[i])
20                  next_states[i], rewards[i], done[i], date[i],
21                      prices[i], acts[i] = envs[i].step(actions[i]
22                      ])
23                  lastMoneyValues[i]=envs[i].money+(envs[i].
24                      numberOfStock*envs[i].finalPrice)
25                  c.acts[i]=acts[i]
26                  c.prices[i]=prices[i]
27                  c.lastMoneyValues[i]=lastMoneyValues[i]
28
29          ApplyCombinedDQN(isDate, c)
30          DateCurrent=DateCurrent+ timedelta(days=1)

```

Algorithm 15. Test Algorithm

According to the Algorithm 15, test phase requires some small modifications from the training phase. At test phase ϵ is set to 0.01 and it doesn't decays any more; so that Epsilon Greedy Strategy is not applied at test phase.

Test phase is executed for 100 episodes. Unlike the training phase, everything is stored in arrays; because 4 DQNs run simultaneously.

At first, 4 DQNs and the Combined DQN (*c*) are reset to initial states. Current Date is set to start date. After that, for each stock if it is an operation day we select a new action which is 99 percent determined from Q-values according to Algorithm 13. Then selected action is applied. After that the Combined DQN Method is applied. Finally, *DateCurrent* is set to next day. If the test period is finished the execution of the program starts the new episode.



3.8. The Combined DQN Method

The Combined DQN Method is created; to increase the chance of making a transaction.

```

1  if c.numberOfStock==0:#don't have stock
2      cBuySignal=False
3      for cBuySingalIndex in range(stockSize):
4          if isDate[cBuySingalIndex]:
5              if c.acts[cBuySingalIndex]==1:
6                  cBuySignal=True
7                  break
8  if cBuySignal:#buy signal
9      cPrice=0
10     cStockName=""
11     buyTargetIndex=-1
12     for buyTarget in range(stockSize):#find best performing
        with buy signal
13         if isDate[buyTarget]:
14             if buyTargetIndex==-1 and c.acts[buyTarget]==1:
15                 buyTargetIndex=buyTarget
16             elif buyTargetIndex>-1 and c.acts[buyTarget]==1:
17                 if c.lastMoneyValues[buyTarget]>c.
                    lastMoneyValues[buyTargetIndex]:
18                     buyTargetIndex=buyTarget
19     if buyTargetIndex!=-1:
20         cPrice=c.prices[buyTargetIndex]
21         c.currentStockIndex=buyTargetIndex
22         if cPrice>=c.money+actCost:#don't have enough money
23             c.ISbankrupt=True
24             c.done=True
25         else:
26             cStockName=c.stockName[c.currentStockIndex]
27             ncStock=math.floor((c.money-(actCost*2))/cPrice)
28             if c.money-(cPrice*ncStock+(actCost*2))==0:
29                 ncStock-=1

```

Continue at next page...

```

32 Continued from previous page...
33         if ncStock>0:
34             c.buyPrice=cPrice
35             c.numberOfStock=ncStock
36             c.expectedSellPrice=cPrice*expectedSellRate+(
                 actCost*2)/ncStock
37             c.lossSellPrice=cPrice*lossSellRate-(actCost*2)/
                 ncStock
38             c.initmoney=c.money
39             c.money-=(cPrice*c.numberOfStock+actCost)
40             c.finalDate=date[buyTargetIndex]
41             c.numberOfOperations+=1
42         else:
43             c.ISbankrupt=True
44             c.done=True
45 else:# has stock
46     #if we have sell signal for current stock
47     if isDate[c.currentStockIndex]:
48         if c.acts[c.currentStockIndex]==2:#sell signal
49             c.money+=c.prices[c.currentStockIndex]*c.numberOfStock-
                 actCost
50             c.finalDate=date[c.currentStockIndex]
51             c.numberOfOperations+=1
52             c.numberOfStock=0
53             c.buyPrice=0
54             c.currentStockIndex=-1
55     #we don't have sell signal but we are at profit
56     elif c.prices[c.currentStockIndex]>c.expectedSellPrice:#
         higher than expected sell price
57         c.money+=c.prices[c.currentStockIndex]*c.numberOfStock-
                 actCost
58         c.finalDate=date[c.currentStockIndex]
59         c.numberOfOperations+=1
60         c.numberOfStock=0
61         c.buyPrice=0
62         c.currentStockIndex=-1
63

```

Continue at next page...

```

64 Continued from previous page...
65     #we don't have sell signal but we are at loss
66     elif c.prices[c.currentStockIndex]<c.lossSellPrice:#lower
        than loss sell price
67         c.money+=c.prices[c.currentStockIndex]*c.numberOfStock-
            actCost
68         c.finalDate=date[c.currentStockIndex]
69         c.numberOfOperations+=1
70         c.numberOfStock=0
71         c.buyPrice=0
72         c.currentStockIndex=-1

```

Algorithm 16. The Combined DQN Algorithm

According to the Algorithm 16, if The Combined DQN agent has no stock, it attaches itself to the best performing DQN which has a buy signal (which produces buy action). The Combined DQN agent detaches itself if the attached DQN gives sell signal (which produces sell action) or the stock is at the specified rate loss or profit. The Combined DQN agent waits for the next buy action and attaches itself to the best performing DQN for a given day and it repeats the processes of attaching and detaching.

4. RESULTS AND DISCUSSION

After the methods designed, the training process took 5000 steps for each stocks. The calculation period for training phase took almost 45 days for a single stock. Four tests are made. At first three tests profit and loss sell rate is assigned to 5 percent for the Combined DQN Method and 5 percent profit sell rate is assigned for the DQN agents. At first test initial capital is set to 1000 TL with a constant 1 TL transaction cost, at second test initial capital is set to 10000 TL with a constant 1 TL transaction cost. At third and forth test initial capital is set to 100000 TL with a 0.0018 transaction cost rate which is used to calculation of the real cost by multiplying the number of stocks bought with the Close price of the day. At forth test loss sell rate is assigned to 7 percent for the Combined DQN Method and profit sell rate is assigned to 3 percent both for the DQN agents and the Combined DQN Method. Test phase took 100 episodes and each test calculation took more than 2 days. During the test phase 4 DQNs run simultaneously to calculate the

Combined DQN Method.

4.1. Results for the Training Phase

At the training phase Epsilon Greedy Strategy is applied. At first the agent explores the environment and at when the agent learns the environment, it starts to exploit its knowledge.

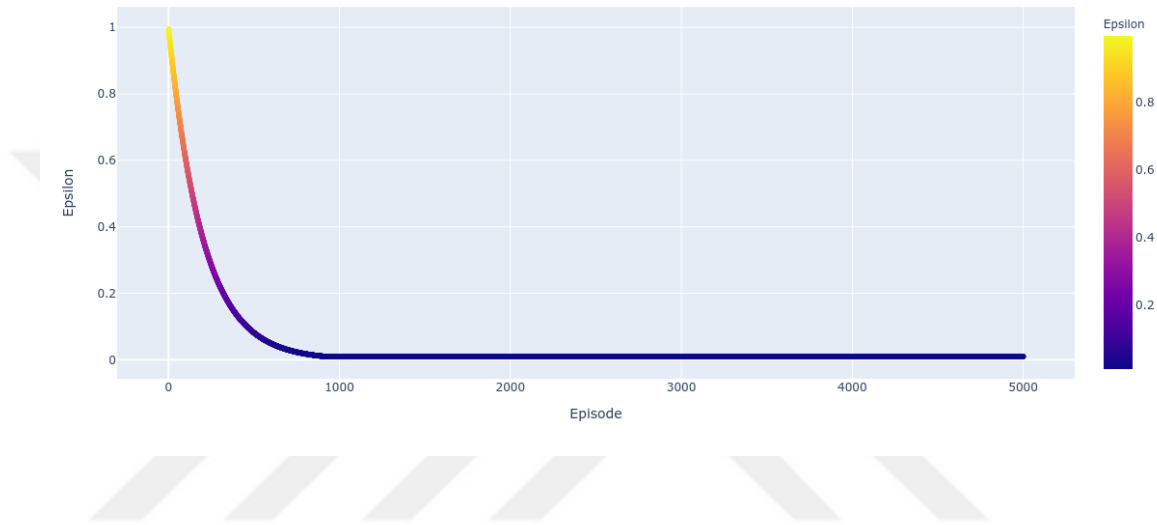


Figure 4.11. Epsilon decays according to the Epsilon Greedy Strategy

4.1.1. Training Phase for ARCLK

Figure 4.12 shows that 100 Episodes of Simple Moving Average End Balance for ARCLK is rising from 1781.28 TL at the 101st Episode to 5219.57 TL at the 5000th Episode.

Figure 4.13 shows that 100 Episodes of Simple Moving Average Number of Operations for ARCLK is decreasing from 895.44 at the 101st Episode to 39.38 at the 5000th Episode.

Figure 4.14 shows that 100 Episodes of Simple Moving Average for ARCLK loss rate is decreasing from 57 percent at the 101st Episode to 2 percent at the 5000th Episode and bankruptcy rate is decreasing from 47 percent at the 101st Episode to 2 percent at the 5000th Episode.



Figure 4.12. SMA 100 End Balance for ARCLK

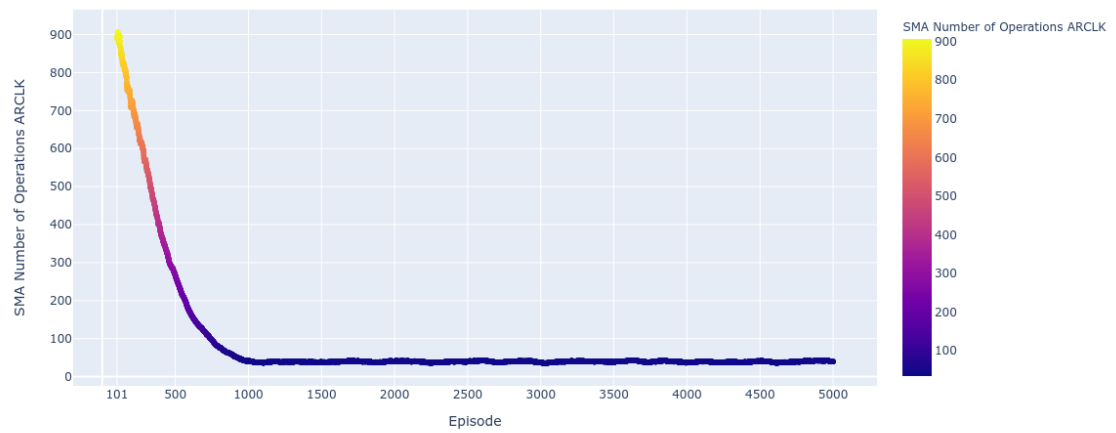


Figure 4.13. SMA 100 Number of Operations for ARCLK

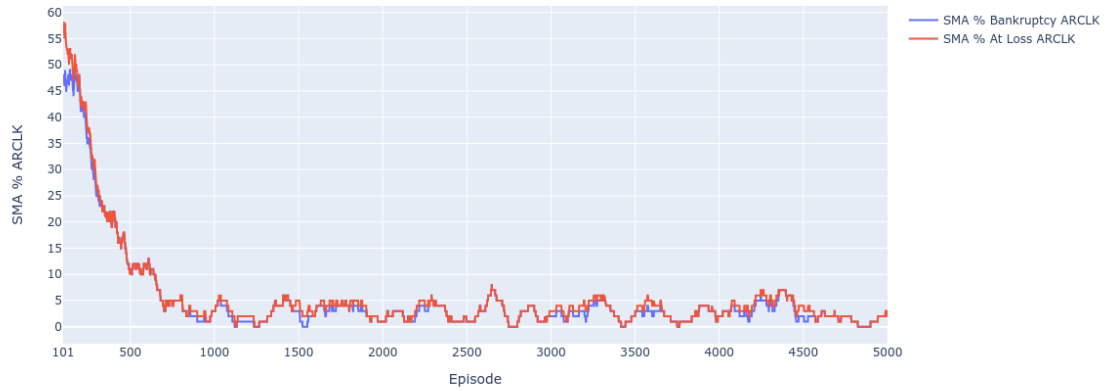


Figure 4.14. SMA 100 Bankruptcy and Loss Rate for ARCLK

4.1.2. Training Phase for ASELS

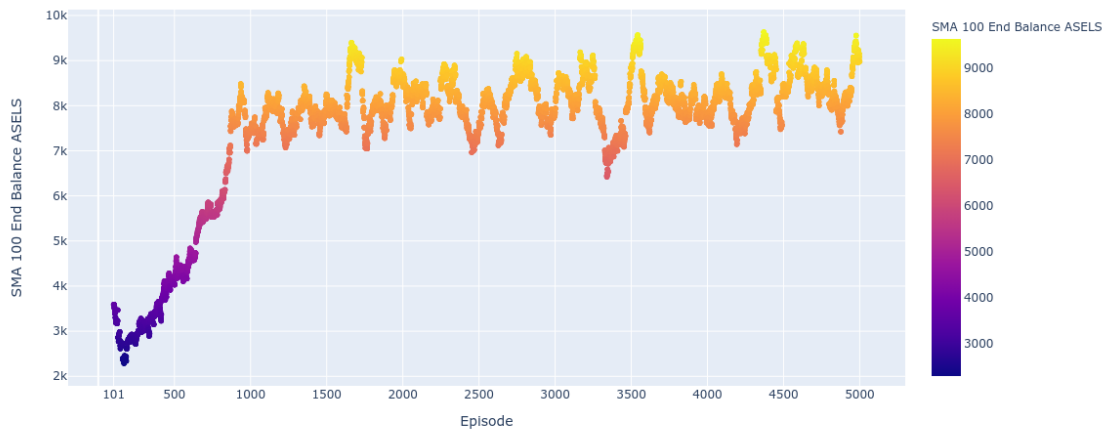


Figure 4.15. SMA 100 End Balance for ASELS

Figure 4.15 shows that 100 Episodes of Simple Moving Average End Balance for ASELS is rising from 3590.89 TL at the 101st Episode to 8978.65 TL at the 5000th Episode.

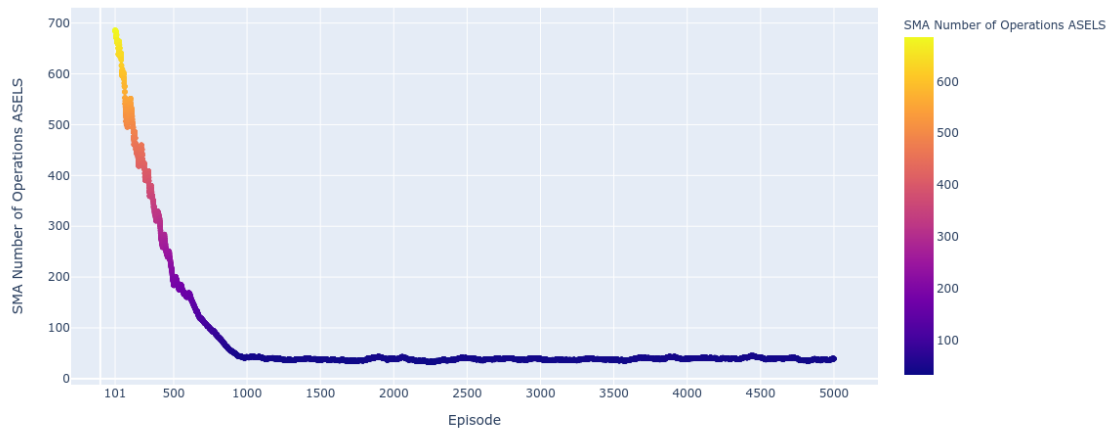


Figure 4.16. SMA 100 Number of Operations for ASELS

Figure 4.16 shows that 100 Episodes of Simple Moving Average Number of Operations for ASELS is decreasing from 685.81 at the 101st Episode to 39.32 at the 5000th Episode.

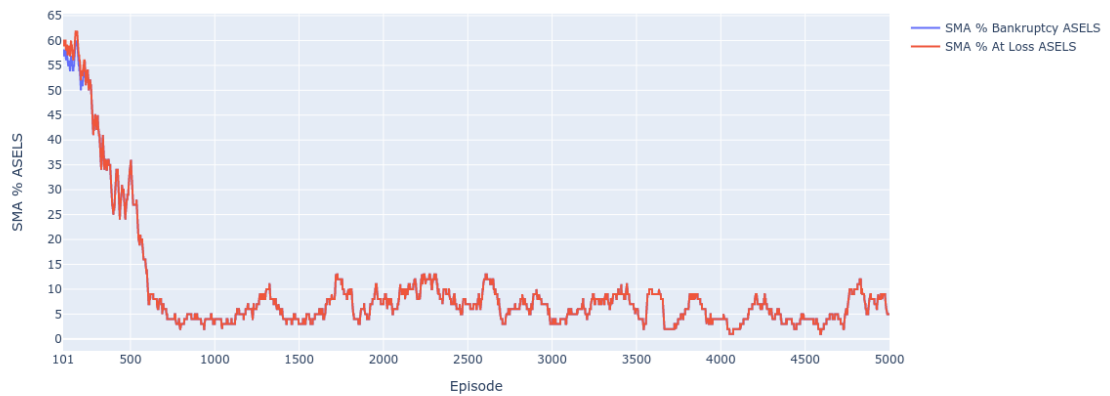


Figure 4.17. SMA 100 Bankruptcy and Loss Rate for ASELS

Figure 4.17 shows that 100 Episodes of Simple Moving Average for ASELS loss rate is decreasing from 59 percent at the 101st Episode to 5 percent at the 5000th Episode and bankruptcy rate is decreasing from 57 percent at the 101st Episode to 5 percent at the

5000th Episode.

4.1.3. Training Phase for SAHOL

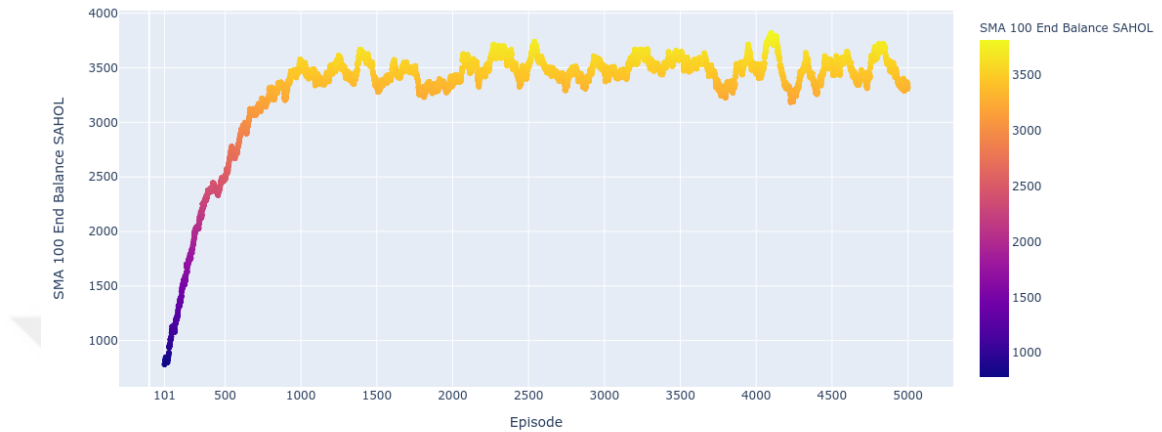


Figure 4.18. SMA 100 End Balance for SAHOL

Figure 4.18 shows that 100 Episodes of Simple Moving Average End Balance for SAHOL is rising from 781.98 TL at the 101st Episode to 3318.71 TL at the 5000th Episode.

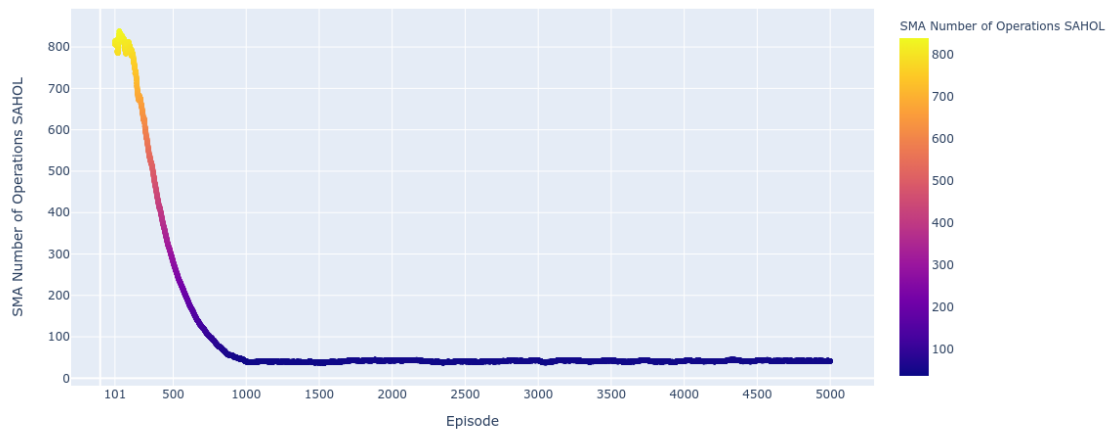


Figure 4.19. SMA 100 Number of Operations for SAHOL

Figure 4.19 shows that 100 Episodes of Simple Moving Average Number of Operations for SAHOL is decreasing from 809.57 at the 101st Episode to 41.65 at the 5000th Episode.

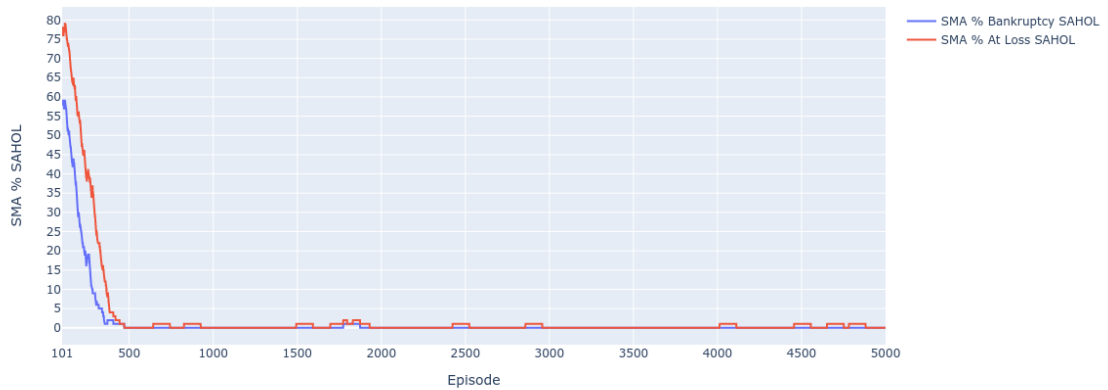


Figure 4.20. SMA 100 Bankruptcy and Loss Rate for SAHOL

Figure 4.20 shows that 100 Episodes of Simple Moving Average for SAHOL loss rate is decreasing from 78 percent at the 101st Episode to 0 percent at the 5000th Episode and bankruptcy rate is decreasing from 59 percent at the 101st Episode to 0 percent at the 5000th Episode.

4.1.4. Training Phase for TUPRS



Figure 4.21. SMA 100 End Balance for TUPRS

Figure 4.21 shows that 100 Episodes of Simple Moving Average End Balance for TUPRS is rising from 2347.22 TL at the 101st Episode to 5503.66 TL at the 5000th Episode.

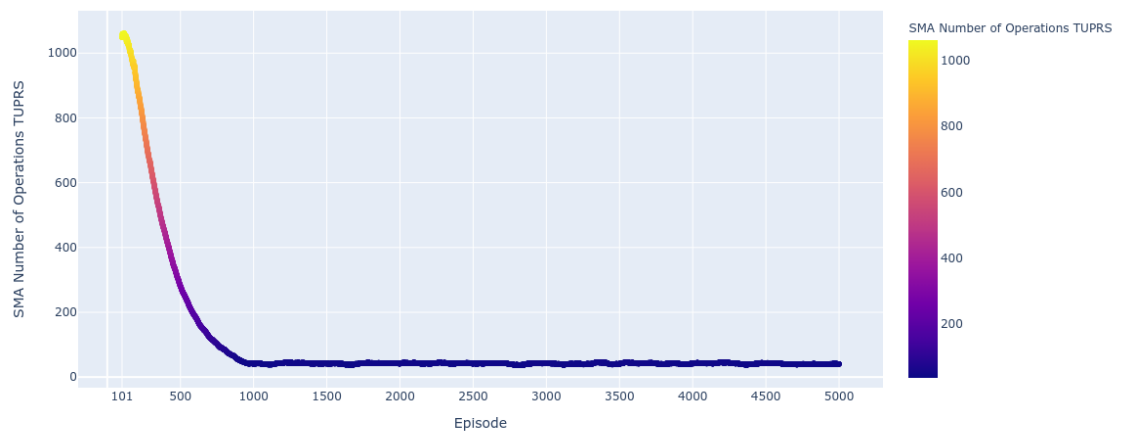


Figure 4.22. SMA 100 Number of Operations for TUPRS

Figure 4.22 shows that 100 Episodes of Simple Moving Average Number of Operations for TUPRS is decreasing from 1048.4 at the 101st Episode to 40.43 at the 5000th Episode.

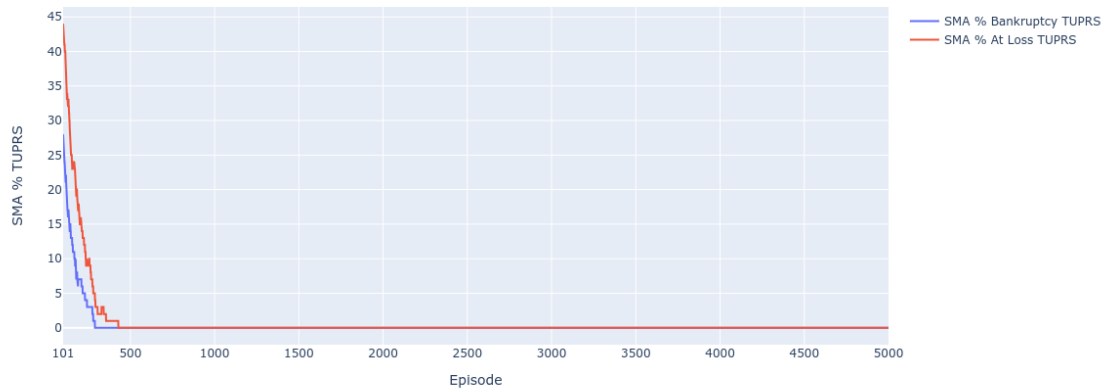


Figure 4.23. SMA 100 Bankruptcy and Loss Rate for TUPRS

Figure 4.23 shows that 100 Episodes of Simple Moving Average for TUPRS loss rate is decreasing from 44 percent at the 101st Episode to 0 percent at the 5000th Episode and bankruptcy rate is decreasing from 28 percent at the 101st Episode to 0 percent at the 5000th Episode.

4.2. Results for the Test Phase

Test period starts at 01.01.2015 and ends at 21.12.2019. 01.01.2015 is holiday so the next operation day is 02.01.2015. At first, BH strategy is calculated for the test phase. There is no buy and sell signal for the test period; because MACD is greater than MACDsignal9 for the input data. There has to be a transition between MACD and MACDsignal9 values to create a buy or sell signal. Therefore; it was not possible to analyze the MACD technique for the test period. At second Stochastic Oscillator for 14 day period is calculated. Finally, result of the tests shown and analyzed.

Table 4.1. Close values of Stocks at the Start and at the End of the Test Period

Stocks	Close Value	
	02.01.2015	21.12.2019
ARCLK	13.35	20.66
ASELS	5.85	19.31
SAHOL	9.01	9.23
TUPRS	37.87	125.5

For the Test1 initial investment is 1000 TL, for Test2 it is 10000 TL. For Test1 and Test2 1 TL transaction fee is applied. For Test3 and Test 4 initial investment is 100000 TL and 0.0018 cost rate is applied for every transaction.

4.2.1. Buy and Hold Strategy Calculation for the Test Phase

Considering the cost values following results can be calculated.

Table 4.2. End Balance of Stocks for the Test Period according to BH Strategy

	End Balance		
Stocks	Test1	Test2	Test3 = Test4
ARCLK	1538.94	15465.88	154199.15
ASELS	3286.2	33001.14	328895.23
SAHOL	1022.2	10243.98	102073.45
TUPRS	3276.38	33132.2	330130.18

4.2.2. 14 Day Period Stochastic Oscillator Calculation for the Test Phase

According to Stochastic Oscillator when there is a transition that if it becomes $\%K$ is greater than $\%D$ then we buy and if it becomes $\%K$ is less than $\%D$ then we sell. Considering the cost values following results can be calculated for the 14 day period Stochastic Oscillator.

Table 4.3. End Balance of Stocks for the Test Phase according to 14 day period Stochastic Oscillator

	End Balance		
Stocks	Test1	Test2	Test3 = Test4
ARCLK	586.04	10764.42	47054.03
ASELS	1620.89	21788.16	97539.15
SAHOL	360.92	7570.13	33525.15
TUPRS	835.15	12490.14	57841.47

4.2.3. Test 1

Test 1, consists of 100 episodes. For Test 1 initial investment is 1000 TL and for every transaction there is a constant fee which is equal to 1 TL.

Table 4.4. Test 1 - 1000 TL initial investment Results

	ARCLK	ASELS	SAHOL	TUPRS	Combined
Average End Balance	1203.46	1331.57	1000.77	1178.52	1058.73
BH Strategy End Balance	1538.94	3286.20	1022.20	3276.38	X
Number of Episodes Better than BH	8	4	34	0	X
14 Day Period Stochastic Oscillator End Balance	586.04	1620.89	360.92	835.15	X
Number of Episodes Better than Stochastic Oscillator	100	15	100	100	X
Maximum End Balance	2298.24	4435.99	1136.67	3121.48	2030.45
Minimum End Balance	697.34	563.00	807.17	857.46	306.79
Average Num. of Operations	14.89	12.66	9.47	13.98	168.44
% At Loss	24	24	44	18	55

According to Table 4.4 the most successful DQN for the Test1 is the one that is trained with the data of ASELS. ASELS has the maximum average end balance and the maximum end balance. Combined DQN method is one of the worst performing method. Combined DQN method has the minimum end balance and the 55 percent of the episodes ends at loss. DQN agent for ARCLK has 8, ASELS has 4, SAHOL has 34 episodes out of 100 episodes perform better than BH strategy. DQN agent for TUPRS is not able to perform any better than BH strategy for all episodes. Only 14 day period of Stochastic Oscillator value for ASELS is greater than Average End Balance of the DQN agent for ASELS. Still 15 episodes out of 100 for ASELS DQN agent performs better than Stochastic Oscillator. At Test1 Stochastic Oscillator is only profitable for ASELS.

4.2.4. Test 2

Test 2, consists of 100 episodes. For Test 2 initial investment is 10000 TL and for every transaction there is a constant fee which is equal to 1 TL.

Table 4.5. Test 2 - 10000 TL initial investment Results

	ARCLK	ASELS	SAHOL	TUPRS	Combined
Average End Balance	12062.13	15687.91	10222.14	12257.79	13472.07
BH Strategy End Balance	15465.88	33001.14	10243.98	33132.20	X
Number of Episodes Better than BH	11	10	45	0	X
14 Day Period Stochastic Oscillator End Balance	10764.42	21788.16	7570.13	12490.14	X
Number of Episodes Better than Stochastic Oscillator	64	21	100	20	X
Maximum End Balance	17635.85	41864.28	12214.13	30951.26	36375.93
Minimum End Balance	6636.31	4973.61	7938.01	8125.59	5662.74
Average Num. of Operations	11.65	12.34	9.11	11.61	170.91
% At Loss	20	25	38	13	30

According to Table 4.5 the most successful DQN for Test2 is the one that is trained with the data of ASELS. ASELS has the maximum average end balance and the maximum end balance. SAHOL is the worst performing method. SAHOL has the minimum average end balance and the 38 percent of the episodes ends at loss. Combined DQN method performs a little bit better that loss rate drops to 30 percent and it has the second largest average end balance value. DQN agent for ARCLK has 11, ASELS has 10, SAHOL has 45 episodes out of 100 episodes perform better than BH strategy. DQN agent for TUPRS is not able to perform any better than BH strategy for all episodes. DQN agents for ARCLK and SAHOL has better average value than 14 day period Stochastic Oscillator method. Still 21 episodes for ASELS and 20 episodes for TUPRS out of 100 episodes DQN agents perform better than Stochastic Oscillator. At Test2, Stochastic Oscillator is at profit for ARCLK, ASELS and TUPRS; because that the cost is small and initial investment is increased to 10000 TL.

4.2.5. Test 3

Test 3, consists of 100 episodes. For Test 3 initial investment is 100000 TL and for every transaction there is a rate fee for the price which is equal to 0.0018.

Table 4.6. Test 3 - 100000 TL initial investment Results

	ARCLK	ASELS	SAHOL	TUPRS	Combined
Average End Balance	111596.89	140891.26	102153.06	129841.22	124792.45
BH Strategy End Balance	154199.15	328895.23	102073.45	330130.18	X
Number of Episodes Better than BH	6	6	49	5	X
14 Day Period Stochastic Oscillator End Balance	47054.03	97539.15	33525.15	57841.47	X
Number of Episodes Better than Stochastic Oscillator	100	83	100	100	X
Maximum End Balance	219865.71	671088.97	126417.73	365082.81	527718.87
Minimum End Balance	63917.92	59573.04	85245.87	86989.54	58005.91
Average Num. of Operations	10.35	9.97	9.07	10.10	165.57
% At Loss	32	32	36	23	41

According to Table 4.6 the most successful DQN for Test3 is the one that is trained with the data of ASELS. ASELS has the maximum average end balance and the maximum end balance. SAHOL is the worst performing method. SAHOL has the minimum average end balance and the 36 percent of the episodes ends at loss. Combined DQN method is not performing better but has the second maximum end balance. All DQNs and Combined DQN Method loss rate increased due to the effect of the cost rate. DQN agent for ARCLK has 6, ASELS has 6, SAHOL has 49 and TUPRS has 5 episodes out of 100 episodes perform better than BH strategy. Only 17 episodes out of 100 episodes for the DQN for ASELS performs poorly than 14 day period Stochastic Oscillator. For the other DQN agents their performance surpass the 14 day period Stochastic Oscillator. At Test3, Stochastic Oscillator is in loss due to the increase of the costs; because of using a cost rate rather than a fix cost.

4.2.6. Test 4

Test 4, consists of 100 episodes. For Test 4 initial investment is 100000 TL and for every transaction there is a rate fee for the price which is equal to 0.0018. Loss sell rate is assigned to 7 percent and profit sell rate is assigned to 3 percent for the Combined DQN Method. Also, profit sell rate is assigned to 3 percent for the DQNs.

Table 4.7. Test 4 - 100000 TL initial investment Results with 7 percent loss sell rate only for the Combined DQN agent and 3 percent profit sell rate for both the DQN agents and the Combined DQN agent

	ARCLK	ASELS	SAHOL	TUPRS	Combined
Average End Balance	116511.96	131467.37	101189.90	121271.15	140166.67
BH Strategy End Balance	154199.15	328895.23	102073.45	330130.18	X
Number of Episodes Better than BH	10	5	45	2	X
14 Day Period Stochastic Oscillator End Balance	47054.03	97539.15	33525.15	57841.47	X
Number of Episodes Better than Stochastic Oscillator	100	83	100	100	X
Maximum End Balance	185227.24	531387.10	115306.15	366168.75	317398.56
Minimum End Balance	60371.95	69248.86	86450.04	87921.54	60564.61
Average Num. of Operations	11.10	9.41	8.36	9.26	185.23
% At Loss	19	33	37	27	15

According to Table 4.7 the most successful method for Test4 is the Combined DQN Method which has the maximum average end balance and the minimum loss rate. SAHOL is the worst performing method. SAHOL has the minimum average end balance and the 37 percent of the episodes ends at loss. The change of the profit and loss sell rates affected the performance of the Combined DQN Method significantly. DQN agent for ARCLK has 10, ASELS has 5, SAHOL has 45 and TUPRS has 2 episodes out of 100 episodes perform better than BH strategy. At Test4, again as it is for the Test3, Stochastic Oscillator is in loss due to the increase of the costs; because of using a cost rate rather than a fix cost.

4.3. Discussions of the Results

Results of the training period shows that for all DQNs according to 100 episode of SMA average end balance increases, number of operations significantly drops, loss rate and bankruptcy rate decreases. The most successful DQN agent is that is used for training with the data of ASELS which has the SMA 100 end balance of 8978.65 TL at the end of the training period. The second successful DQN agent for the training period is used for training with the data of TUPRS, which has the SMA 100 end balance of 5503.66 TL. The third successful is the ARCLK DQN agent which has the SMA 100 end balance of 5219.57 TL. The least successful is the SAHOL DQN agent which has the SMA 100 end balance of 3318.71 TL at the end of the training phase. The bankruptcy rate and the loss rate drops below 5 percent at the end of the training period. The SMA 100 number of operations (transactions) drops from above 500 to around 40.

At the test period 4 DQNs run simultaneously for 100 episodes to calculate the Combined DQN. There are 4 tests experimented at this thesis. At the first test the initial investment is 1000 TL, at second test it is 10000 TL, at third and forth test it is 100000 TL. At the first and second tests 1 TL transaction cost is applied. However, for the third and forth test 0.0018 cost rate fee is applied.

If the Test 1 is analyzed its results are parallel with the training period. DQN for ASELS has the best average end balance and the TUPRS has the lowest at lost percentage. The Combined DQN performs worst that it is at loss for 55 episodes from 100 episodes.

At Test 2 DQN for the ASELS has the highest end balance and the Combined DQN method has the second. Although the Combined DQN method has the second best end balance 30 episode ends at loss.

At Test 3 DQN for the ASELS has the highest end balance and the maximum return of all tests with 671 percent profit return at maximum end balance. Using cost rate of 0.0018 affected the performance of the DQNs and the Combined Method that loss rate is increased for all.

At Test 4 Combined DQN Method is the most successful method. It has approximately 40.17 percent average profit return. Also, the loss rate drops to 15 percent due to the change of the profit and loss sell rates.

For 14 day period Stochastic Oscillator, SAHOL DQN agent performs better in all episodes for all tests. Test2 is the best performance for Stock Oscillator. However; still DQN for ARCLK has 64, ASELS has 20, SAHOL has 100 and TUPRS has 20 episodes out of 100 episodes performs better than 14 day period Stochastic Oscillator. For Test3 and Test4 because the costs increased and calculated according to cost rate rather than fix cost; Stochastic Oscillator showed poor performance and lost money.

Result of the test period shows that the success of the test period is directly proportional to the training period. ASELS was the best performing at both training and test periods. The affect of the cost is decisive for the success of the agents as well as the success of the Stochastic Oscillator. DQN agents tend to decrease the number of operation (transactions) that it seems that they want to hold the stocks within a long period. The Combined DQN method is not the best for all the test methods except the forth test, but it successfully increase the number of operations. BH strategy has all the advantages and performs as best technique.

5. CONCLUSION

Within the scope of this thesis, a Reinforcement Learning agent with DQN method and a new test method called the Combined DQN method has been developed. Daily parameters as well as technical analysis methods of Stochastic and MACD parameters are given as input to the DQN agent. The DQN agent is trained for 5000 episodes. 4 DQN agents tested simultaneously to calculate the Combined DQN method for 100 episodes in three different experiment with different conditions.

The results of the training phase shows that a DQN agent can learn and perform better at the end of the training phase. As the agent learns, the number of transactions significantly drops; that means the DQN agent tends to hold the stock at long term.

The results of the test phase shows that the performance of the DQN agent at the test phase is directly proportional the performance of the DQN at the training phase. The costs affect the performance of the DQN agent. The Combined DQN Method is useful to increase the number transactions. By the help of the increase of the costs at Test3 and Test4 DQN agents surpass the 14 day Stochastic Oscillator. BH strategy has all the advantages and performs as best technique.

This thesis is also useful in understanding some controversial issues in the scientific world between the fundamentalists and technical trading analysts. The DQN agent tends to hold stock longer means that the agent founds out that long term investment is better which is something like the fundamental analysts do; but the agent performs this by daily and technical parameters. However, the DQN agent is able to survive in a harsh realistic environment even if the Combined DQN method is used. This shows that the agent can survive by using technical analysis methods. This thesis also shows that at least the guidance of the AI is inevitable for stock market trading and it will be vital.

As future studies, the number of DQN agents can be increased to analyze how the Combined DQN method performs with more than 4 DQN agents. Different technical analysis methods can be given as input to the agent like RSI or Chaikin Money Flow. The attaching and detaching mechanism can be improved for the Combined DQN Method.

6. REFERENCES

- Anghel, G. D. I. 2015. *Stock Market Efficiency and the MACD. Evidence from Countries around the World, Procedia Economics and Finance*, Volume 32, Pages 1414-1431, ISSN 2212-5671, [https://doi.org/10.1016/S2212-5671\(15\)01518-X](https://doi.org/10.1016/S2212-5671(15)01518-X) : 1416.
- Anonymous 1 : Machine Learning, https://en.wikipedia.org/wiki/Machine_learning [Last access date: 25.05.2021].
- Anonymous 2 : Markov property, https://en.wikipedia.org/wiki/Markov_property [Last access date: 25.05.2021].
- Anonymous 3 : Markov decision process, https://en.wikipedia.org/wiki/Markov_decision_process [Last access date: 25.05.2021].
- Anonymous 4 : Q-learning, <https://en.wikipedia.org/wiki/Q-learning> [Last access date: 26.05.2021].
- Anonymous 5 : How to Calculate MACD in Excel, <http://investexcel.net/how-to-calculate-macd-in-excel/> [Last access date: 27.05.2021].
- Anonymous 6 : ARCELİK, ARCLK TarihselVeriler | MynetFinans, <http://finans.mynet.com/borsa/hisseler/arclk-arcelik/tarihselveriler/> [Last access date: 2021/05/31].
- Anonymous 7 : ASELSAN, ASELS TarihselVeriler | MynetFinans, <http://finans.mynet.com/borsa/hisseler/asels-aselsan/tarihselveriler/> [Last access date: 2021/05/31].
- Anonymous 8 : SABANCI-HOLDING, SAHOL TarihselVeriler | MynetFinans, <http://finans.mynet.com/borsa/hisseler/sahol-sabanci-holding/tarihselveriler/> [Last access date: 2021/05/31].
- Anonymous 9 : TUPRAS, TUPRS TarihselVeriler | MynetFinans, <http://finans.mynet.com/borsa/hisseler/tuprs-tupras/tarihselveriler/> [Last access date: 2021/05/31].

- Anonymous 10 : JavaScript, <https://en.wikipedia.org/wiki/JavaScript> [Last access date: 02.06.2021].
- Anonymous 11 : Python (programming language), [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)) [Last access date: 02.06.2021].
- Anonymous 12 : TensorFlow, <https://en.wikipedia.org/wiki/TensorFlow> [Last access date: 02.06.2021].
- Anonymous 13 : Keras, <https://en.wikipedia.org/wiki/Keras> [Last access date: 02.06.2021].
- Anonymous 14 : About Keras, <https://keras.io/about/> [Last access date: 02.06.2021].
- Appel, G. 1979. *The Moving Average Convergence Divergence Trading Method*, Signalert Corp, 150 Great Neck Rd, Great Neck, NY 11021
- Backhouse, R. and Cherrier, B. 2016. *'It's Computerization, Stupid!'* *The Spread of Computers and the Changing Roles of Theoretical and Applied Economics*, <https://ssrn.com/abstract=2781253>, <http://dx.doi.org/10.2139/ssrn.2781253> [Last access date: 23.05.2021] : 1,13-17.
- Beyaz, E., Tekiner, F., Zeng, X. and Keane, J. 2018. *Comparing Technical and Fundamental Indicators in Stock Price Forecasting*, 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 1607-1613, doi: 10.1109/HPCC/SmartCity/DSS.2018.00262.
- CertainPerformance 2020. <https://codereview.stackexchange.com/questions/242660/javascript-extract-data-from-html-table> [Last access date: 01.06.2021].
- Chootong, C., and Sornil, O. 2012. *Trading Signal Generation Using A Combination of Chart Patterns and Indicators*, International Journal of Computer Science Issues (IJCSI), 9(6), 202 : 205.

- Choudhary, A. 2019. A Hands-On Introduction to Deep Q-Learning using OpenAI Gym in Python, <https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/> [Last access date: 26.05.2021].
- Deeplizard 2018. Exploration vs. Exploitation - Learning the Optimal Reinforcement Learning Policy, <https://www.youtube.com/watch?v=mo96Nqlo1L8> [Last access date: 30.05.2021].
- DeGroat, T.J. 2019. The History of JavaScript: Everything You Need to Know | Springboard Blog, <https://www.springboard.com/blog/data-science/history-of-javascript/> [Last access date: 02.06.2021]
- Fan, J., Wang, Z., Xie, Y. and Yang, Z. 2020. *A Theoretical Analysis of Deep Q-Learning*. Proceedings of the 2nd Conference on Learning for Dynamics and Control, in PMLR 120:486-489 : 1,2.
- Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., and Larochelle, H., Rowland, M. and Dabney, W. 2020. *Revisiting Fundamentals of Experience Replay*, arXiv:2007.06700 [cs.LG] : 2.
- Gershenson, C. 2003. *Artificial Neural Networks for Beginners*, arXiv:cs/0308031 : 2,4.
- Glorot, X., Bordes, A. and Bengio, Y. 2011. *Deep sparse rectifier neural networks*, In Proceedings of the fourteenth international conference on artificial intelligence and statistics (pp. 315-323), JMLR Workshop and Conference Proceedings.
- Hazır, U. and Danişman, T. 2020. *Deep Q-Learning For Stock Future Value Prediction*, ICAIAME 2020, International Conference on Artificial Intelligence and Applied Mathematics in Engineering, Seminar Submission 139 : 1,5.
- Hu, J. 2016. Reinforcement learning explained – O'Reilly. <https://www.oreilly.com/radar/reinforcement-learning-explained/> [Last access date: 26.05.2021].
- John, C. and Watkins, C. H. 1989. *Learning from delayed rewards*, PhD thesis, King's College, https://www.academia.edu/3294050/Learning_from_delayed_rewards [Last access date: 25.05.2021] : 37, 39.

- Jordan, D. J. and Diltz, J. D. 2003. *The Profitability of Day Traders*, *Financial Analysts Journal*, 59:6, 85-94, DOI: 10.2469/faj.v59.n6.2578 : 10.
- Kingma, D. P. and Ba, J. 2017. *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [cs.LG] : 1,7.
- Kirkpatrick, C. D. and Dahlquist, J. R. 2011. *Technical Analysis: The Complete Resource for Financial Market Technicians*, FT Press : 9.
- LaMar, M. M. 2018. *Markov Decision Process Measurement Model*, *Psychometrika*, 83(67-88) : 68.
- Li, Y. 2018. *Deep Reinforcement Learning: An Overview*, arXiv:1701.07274 [cs.LG] : 7.
- McCulloch, W.S. and Pitts, W. 1943. *A logical calculus of the ideas immanent in nervous activity*. *Bulletin of Mathematical Biophysics* 5, 115–133. <https://doi.org/10.1007/BF02478259> .
- Mehta, R. 2019. *A Brief History of Python Programming Language*, <https://www.digitalnest.in/blog/a-brief-history-of-python-programming-language/> [Last access date: 02.06.2021].
- Mnih, V., Kavukcuoglu, K., Silver, D. et al 2015. *Human-level control through deep reinforcement learning*, *Nature* 518, 529–533. <https://doi.org/10.1038/nature14236> : 1.
- NASAA, 1999. *Report of the Day Trading Project Group*, https://www.nasaa.org/wp-content/uploads/2011/08/NASAA_Day_Trading_Report.pdf, [Last access date: 23.05.2021] : 53.
- Raval, S. 2018. *Q Learning for Trading*, <https://www.youtube.com/watch?v=rRssY6FrTvU&t=68s> [Last access date: 26.05.2021].
- Ryu, D. 2012. *The profitability of day trading: An empirical study using high-quality data*, *Investment Analysts Journal*, 41:75, 43-54, DOI: 10.1080/10293523.2012.11082543 : 2.

- Sutton, R. S. and Barto, A. G. 2014, 2015. *Reinforcement Learning: An Introduction*, The MIT Press, Second edition, in progress : 53,54,57.
- Tharavanij, P., Siraprapasiri, V. and Rajchamaha, K. 2015. *Performance of technical trading rules: evidence from Southeast Asian stock markets*. *SpringerPlus* 4, 552, <https://doi.org/10.1186/s40064-015-1334-7> : 2,39.
- Thomset, M. C. 2015. *Getting Started in Stock Analysis Illustrated Edition*, Wiley, ISBN 978-1-119-01951-0, 978-1-119-01951-7 : xiii,38.
- Tribello, G. 2015, markov property, <https://www.youtube.com/watch?v=1meaW5GxUbY> [Last access date: 29.05.2021].
- Wang, Y., Wang, D., Zhang, S., Feng, Y., Li, S. and Zhou, Q. 2017. Deep Q-trading, CSLT TECHNICAL REPORT-20160036, <http://cslt.riit.tsinghua.edu.cn/mediawiki/images/5/5f/Dtq.pdf> [Last access date: 23.05.2021].
- Yang, H., Liu, X., Zhong, S. and Walid, A. 2020. *Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy*, SSRN: <https://ssrn.com/abstract=3690996> or <http://dx.doi.org/10.2139/ssrn.3690996>.
- Yegulalp, S. 2019. What is TensorFlow? The machine learning library explained, <https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html> [Last access date: 02.06.2021].
- Zupan, J. 1994. *Introduction to Artificial Neural Network (ANN) Methods: What They Are and How to Use Them*, *Acta Chimica Slovenica*, 41 : 2.

CURRICULUM VITAE

UĞUR HAZİR

EDUCATION

Master of Science 2018-2021	Akdeniz University Graduate School of Natural and Applied Sciences, Department of Computer Engineering, Antalya
Bachelor of Science 2001-2007	Yeditepe University Engineering and Architecture Faculty, Computer Engineering Department, Istanbul

WORK EXPERIENCE

Computer Engineer 2014-Present	Antalya Police Department, Cybercrime Prevention Department, Antalya
Computer Engineer 2009-2014	Turkish National Police Academy, Ankara
Programmer 2007-2008	Istanbul Public Registration and Citizenship Office, Istanbul