國立政治大學 應用數學系

碩士 學位論文

卷積深度Q-學習之ETF自動交易系統

Convolutional Deep Q-learning for ETF Automated Trading System

碩士班學生：陳非霆　撰
指導教授：蔡炎龍 博士

中華民國 106 年 11 月 7 日

國 立 政 治 大 學 應 用 數 學 系

陳非霆君所撰之碩士學位論文

卷積深度Q-學習之 ETF 自動交易系統
Convolutional Deep Q-learning for ETF Automated Trading System

業經本委員會審議通過

論文考試委員會委員：

指導教授：

系主任：

中華民國 106 年 11 月 7 日

致謝

悄悄的，七年過去了，我在政大日子也不多了，這段時間以來，我非常感謝炎龍老師對我的指引以及教導，教會了我神經網路以及帶領我進入增強學習的領域，甚至能不斷給我各種方向以及糾正我的錯誤，讓我可以順利的完成碩士的學位。

感謝曾經幫助我的任何人，曾經教我寫過程式的人、曾經被幫我修改英文的人、曾經被我問過論文的人、以及教過我各種事情的人，我都非常非常感謝你們，沒有你們的幫助，我無法完成今天這篇論文，你們的幫助，才有今日的我，真的是萬分感謝。

我也非常感謝我的每一個家人，他們的支持以及諒解，使得我能安心以及順利的完成學業，讓我沒有後顧之憂讀書，因此感謝你們，你們是我努力的動力。

中文摘要

本篇文章使用了增強學習與捲積深度學習結合的 DQCN 模型製作交易系統，希望藉由此交易系統能自行判斷是否買賣 ETF，由於 ETF 屬於穩定性高且手續費高的衍生性金融商品，所以該系統不即時性的做買賣，採用每二十個開盤日進行一次買賣，並由這 20 個開盤日進行買賣的預測，希望該系統能最大化我們未來的報酬。

DQN 是一種增強學習的模型，並在其中使用深度學習進行動作價值的預測，利用增強學習的自我更新動作價值的機制，再用深度學習強大的學習能力成就了人工智慧，並在其取得良好的成效。

關鍵字：深度學習、增強學習、卷積神經網路、Q-learning、DQN、ETF

Abstract

In this paper, we used DCQN model, which is combined with reinforcement learning and CNN to train a trading system and hope the trading system could judge whether buy or sell ETFs. Since ETFs is a derivative financial good with high stability and related fee, the system does not perform real-time trading and it performs every 20 trading day. The system predicts value of action based on data in the last 20 opening days to maximize our future rewards.

DQN is a reinforcement learning model, using deep learning to predict value of actions in model. Combined with the RL's mechanism, which updates value of actions, and deep learning, which has a strong ability of learning, to finish an artificial intelligence. We got a perfect effect.

Keywords: Deep Learning, Neural Network, CNN, Q-leanring, DQN, ETF

Contents

List of Figures

Chapter 1

Introduction

Deep learning has gradually matured and completed, and it is widely applied in many field such as image recognition, generating sentences, play games and regression analysis. However it received a significant improvement. The research in this paper is composed of Convolutional and fully connect neural network to perform a prediction of future rewards. [1] [2] Reinforcement learning (RL) is a structure of machine learning, concerning with how agent ought to take action in an environment to maximize the cumulative reward in the end. The technique has been developed in many field for a period of time. First, because of computational complexity and restrict of tabular form, RL only processes discrete action and can not run big data. Along with the development of deep learning and progress of computer, original dilemma of reinforcement learning is solved. Deep learning replaced tabular form and not only improved accuracy of prediction but also solved the difficulty that RL could not apply in continuous action space. Therefore, reinforcement learning could be developed rapidly. Now it have been successfully applied in atari game by GoogleDeepMind and the result is perfect. The paper adopted DQN which combined traditional Q-learning with deep learning.

Then, the paper combined the above two techniques (RL+CNN) to construct an automatic trading system and, it could be applied in ETFs which expected the trading system to maximize future assets. The reasons, why chose ETFs, are stability of price, various underlying assets, and related costs, which includes fee, commission and tax. Cost

of ETFs is lower than similar mutual fund. Therefore it chooses ETFs as the underlying financial product and got good outcome.

Chapter 2

Deep Learning

Deep learning is a method of machine learning. It is a skill that has machine learn by itself. Actually, machine learning is to let machine automatically find useful function according to training data [9]. Let's take a look at the example of machine learning.

Using the skill of machine learning in image recognition, it just want machine to

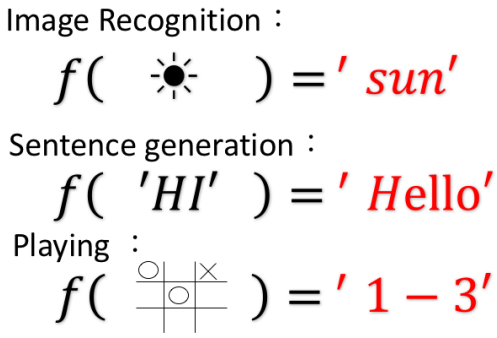

Figure 2.1: Example of function.

find corresponding object named according to object image. The function is the same as the first function in the midterm of figure 2.1. Actually, the input is "object" image and output is "object name". In sentence generation, the function is mapping from a text to exotic text as the second function in figure 2.1.

In third function of figure 2.1, it is to use machine to play game. The input of the function is the position of Circle or Fork on the board, and the output of the function is the position of the next step you should play. Now, the most famous skill is the application of AlphaGo developed by Google DeepMind ,which played the board game Go in London.

On 9th, 10th, 12th, 13th, and 15th March 2016, AlphaGo played with South Korean professional Go player Lee Sedol, ranked 9-dan, one of the best players at Go, with five games which were on-live.

The first three games were won by AlphaGo following resignations by Lee Sedol. However, Lee Sedol beat AlphaGo in the forth game, winning at move 180. In the fifth game, Alpha Go get the forth win, so Alpha Go beat Lee Sedol with 4-1 result and in the internet, artificial intelligence and deep learning were widely discussed and valued.

In the paper, we use neural network to find time series function. In [5], it introduces some various techniques to apply in time series. Conditional time series forecast based on the recent deep convolutional neural network in [1]. In 1997, Ramon Lawrence publish using neural network to forecast stock price. [8] [16] David Enke published that VAMA and EMV with neural network, which predicts future value, in stock trading have better than without neural network [3].

Now, deep learning can be divided into three steps to understand and the step are building model, loss function, training. The first step is like human's brain and it is a function set corresponding to the structure of network which is provided by human. Deciding loss function is the second step. After deciding the network structure, we can not identify which functions are good or bad. Therefore, we have to define the goodness of a function according to training data. Then, the next job is handed over to the machine. Machine could automatically find the best selection from the function set.

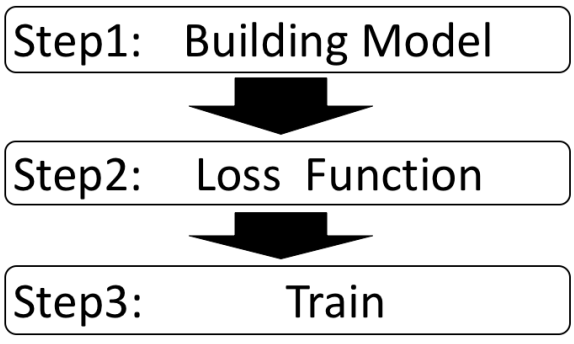

Figure 2.2: Three Step for Deep Learning.

2.1 Neural Network and Neuron

Neural networks were inspired by biological neural networks. Deep learning algorithms may approximate how learning occurs in the real brain. We know human brain is constructed by neuron and the brain of neural networks is also connected with neuron. In the following content, we will introduce the construction and operation of neurons in Neural network.

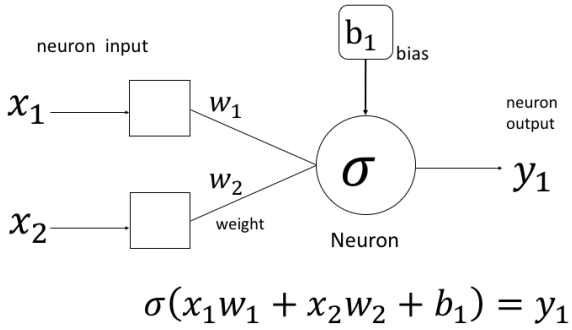

Figure 2.3: The construction of neuron.

In figure 2.3, this is one neuron of operation in neural network. Every neuron is a simple function. a_1, a_2, a_3 in the left side of the figure is input value of neuron and output is a_1 in the right side. w_1, w_2, w_3 acts weights corresponding to every input values and activation function is a nonlinear function defined in advanced ,and its input and output are one values. Well-known activation function has sigmoid function, hyperbolic tangent function, Rectified linear unit (ReLU) and Logistic function.

In the following, we would introduce how a neuron works. First, the input values products corresponding weight and then all add together with bias. Its sum is a value, which is also activation function of input value. Given an instance, a_1, a_2, a_3 is 1,2,3 , and w_1, w_2, w_3 is 1,2,-1, bias is 3, and activation function is ReLU function. The ReLU of input compute by $1 \cdot 1 + 2 \cdot 2 + 3 \cdot (-1) + 3 = 5$. Therefore, the neuron output is also 5, and we call weight and bias are the parameter which will been trained. This parameter decides the neuron of operation.

After understanding what is neuron, we learn neural network. The neural network is connected by many neurons, and we only need to decide how to link the neural network.

Machine would automatically get every neuron of parameter according to training data.

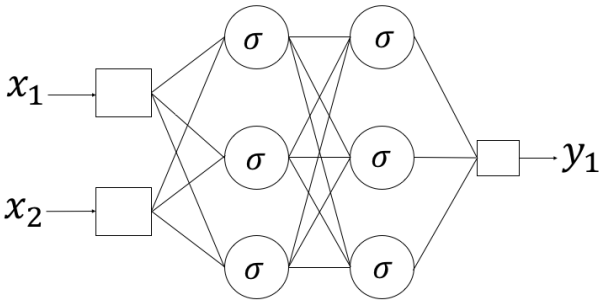

Figure 2.4: Fully Connect Feedforward Network.

As figure 2.4, we called the type of neural network structure is "fully connected feedforward network". The fully feedforward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, and from the input nodes to the output nodes, every neuron connects with all neuron of previous layer and next layer. There is no cycle or loop in the network. Neurons are arranged in rows and rows, each row is called a "layer." The first layer is called "input layer", the last layer is called "output layer", and the other layer are called "hidden layer".

For every neuron in network, forms is written as

$$a_j^l = \sigma(\sum_k w_{jk}^l a_k^{l-1} + b_j^l),$$

a_j^l is the activation of the j^{th} neuron in the l^{th} layer. w_{jk}^l is the weight from the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer. b_j^l is the activation of the j^{th} neuron in the l^{th} layer.

Now, given a neural network structure, the structure would define a function set. If the parameter was given, the neural network would define a complex function. Determin the neuron connection, and then train the network according to training data. In this way, which machine find out their own parameters equivalent to the situation that we

provide a set of functions first. The machine select useful function itself from the function set.

Neuron network is still necessary for human to decide beforehand in deep learning. Therefore if the structure of set is bad, i.e. the function set defined by this structure does not find a suitable function at all. And all is in vain. For every different type of task, there are different suitable neuron network structures. Although there are studies to let neuron network decides what the structure use by itself, the skill has fewer success cases so far.

2.1.1 Activation Function

If there are no activation functions in neuron, input of layer in neural network is linear combination of last layer's output, and it is still linear. Therefore, activation function is to let them have non-linear relationship.

Now, let us introduce some famous example of activation function:

1. Rectified linear unit (ReLU):

   Equation:

$$f(x) = \begin{cases} 0, \text{if } x < 0 \\ x, \text{if } x \geq 0 \end{cases}$$

   Range:$[0, \infty)$

   Graph:

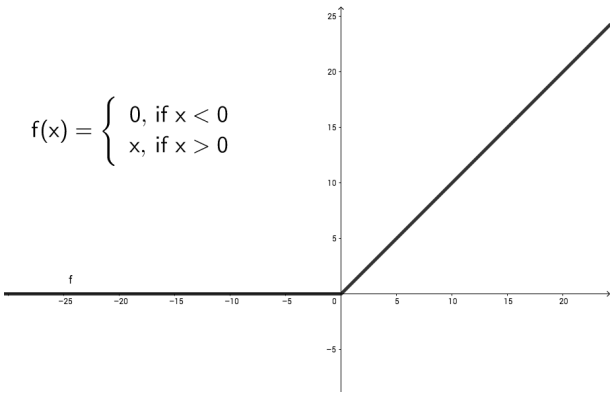

Figure 2.5: Relu function.

2. Logistic function:

   Equation:

$$f(x) = \frac{1}{1+e^{x}}$$

   Range:$(0,\ 1)$

   Graph:

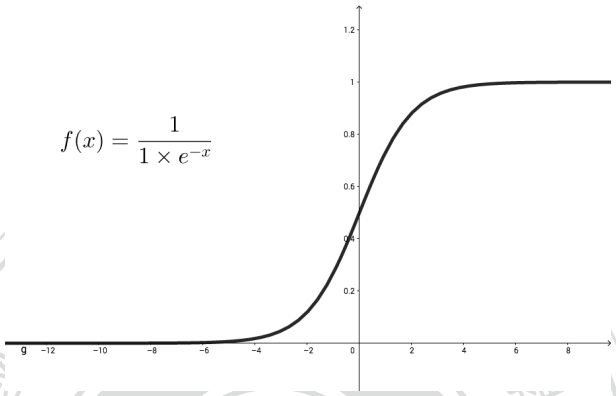

Figure 2.6: Logistic function.

3. hyperbolic tangent (tanh)

   Equation:

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{2}{1+e^{-2x}} - 1$$

   Range:$(-1,\ 1)$

   Graph:

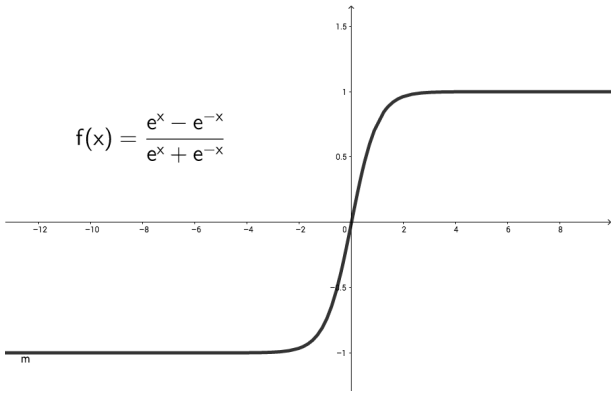

Figure 2.7: Hyperbolic tangent function.

2.1.2 Loss Function

When structure of neuron network and activation function are determined, adjustable parameters are just weights and biases. The sets of parameter are called θ. Every θ defines a function, and we see them as a set $\{F_\theta\}$. Therefore, we want to find an optimal function,which is signed F_{θ^*}. In the following, we go on to the second step and have to define what function is good for finding a best function.

We first introduce loss function, which is a function mapping from parameter space to real, and it is used to estimate the degree of inconsistency between the predicted value $F_\theta(x)$ and true value y. We think that a good function or good parameter is no distinction between predicted value and real value, so lose function is as small as possible, and vice versa. In the following, a widely used method called "mean squared error" or MSE is introduced.

Suppose training data (x_1, y_1), (x_2, y_2), ..., (x_k, y_k), x is k-dimension, θ is a parameter of neural network, i.e it is $\{(w_1, ..., w_n, b_1, ..., b_n)\}$, k is a number of training data. Defined mean squared error be a function by $L: R^k \to R$,

$$L(\theta) = \frac{1}{2}\sum_{i=0}^{n} ||y_i - F_\theta(x_i)||^2,$$

$(y_i - F_\theta(x_i)^2$ represents difference between true value y_i and true value $F_\theta(x_i)$. A good parameter should make the loss of all example as small as possible, so our objective function is

$$\min_\theta L(\theta) = \frac{1}{2}\sum_{i=0}^{n} ||y_i - F_\theta(x_i)||^2,$$

i.e, we hope to find a function in a function set that minimize total loss and find a network parameters θ^* that minimize total loss.

We already know that we have to find a function that minimize total loss, but how we get it. Next section, we would introduce optimization method to solve it.

2.1.3 Gradient Descent Method

Optimization is always the ultimate goal whether you are dealing with a real life problem or building a software product. Optimization basically means getting the optimal output for your problem. You would now with how optimization plays an important role in our real-life.

Optimization in machine learning has a slight difference. Generally, while optimizing, we know exactly how data looks like and what thing we could want to improve it. But in machine learning we have no clue how our "new data"looks like. In machine learning, we perform optimization on the training data and check its performance on a new validation data.

Next, we will look at a particular optimization technique called "Gradient Descent". It is the most commonly optimization technique used when dealing with machine learning and deep learning.

Suppose you are at the top of a mountain, and you have to reach a lake which is at the lowest point of the mountain. You are blindfolded and you have zero visibility to see where you are headed. So, what approach will you take to reach the lake? The best way is to check the ground near you and observe where the land tends to descend. This will give an idea in what direction you should take your first step. If you follow the descending path, it is very likely you would reach the lake.

Our objective target is to find the network parameter θ^* that minimize total loss $L(\theta)$. First, we pick an random initial value for w and then compute its first-order derivative $\frac{\partial L}{\partial w}$ to update the parameter, i.e $w \leftarrow w - \eta \cdot \frac{\partial L}{\partial w}$. Computing first-order derivative can find a steepest direction. η is a learning rate and it is just long step you deciding when you climbing. Next, repeat compute the derivate and update parameter until $\frac{\partial L}{\partial w}$ is approximately small, i.e when updating value is little. In deep learning, it updates receptively parameter w_i, b_i for every i until the parameter of norm is too little.

2.2 Convolutional Neural Network

Convolutional Neural Networks are a category of Neural Networks that have proven very effective in areas such as image recognition and classification [13]. If there are 1M pixel (1000×1000) image and one hidden layer which also have 1M hidden unit and we suppose to use fully connected Feedforward Network, every hidden unit connects with 1M input unit and so the total of connection is 10^{12} connection (1M input $\times$ 1M hidden unit) to learn. Obviously, this is not practical, but also unnecessary. Actually, image is highly local correlated. One pixel mostly only has high correlation with nearby pixels and basically have no correlation with remote pixels. If we use local connect neural network instead of full connect and a hidden unit only has 10×10 local connection, we could find that connection are significant reduction than fully connect because the connection or parameter space become 10^8 $(10 \times 10 \times 1M)$. Although so, the connection or parameter space are still too much.

As above, only local connection is not enough and the computation is still huge. Therefore, we need other skill to solve the problem and the skill is called Share Weight. Share Weight is just to suppose that hidden unit of weight are the same. Of course, this greatly simplifies training or learning. In local connection, a hidden layer has 1M neurons and every neuron has the same weight, so the parameter space only left with $10 \times 10 = 100$. Actually, we could see weight as the way of extracting features, and this way is supposes that the image signal is statistics stationary w.r.t location. The implication is that the statistical properties of a portion of the image are the same as those of the other parts.

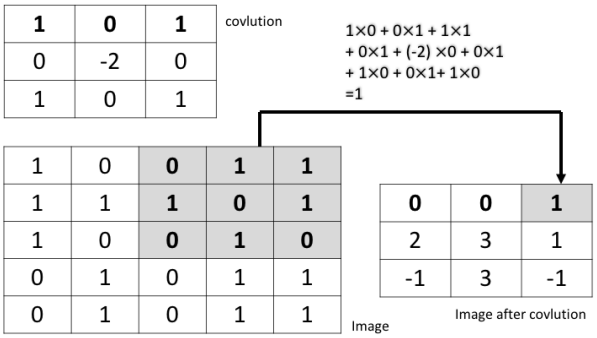

Figure 2.8: example of convolution.

Combined with local connection and share weight, we get a new method called "convolution" or "filter". First, we could decide a convolution or many convolutions and then use it to slide image. After sliding image by convolution, we can get a smaller and new image which is similar with original image. As figure 2.8, there are 5×5 image and 3×3 convolution and then use the filter to slide the image to get a new 3×3 image. Using convolution could not only extract feature but also reduce dimension. Actually, the parameter is still huge and we need other technique to avoid over-fitting.

Suppose input of a convolution is a matrix $o_{i,j}$ $(i = 1, ..., n) and (j = 1, ..., m)$, and convolution is $w_{i,j}$ where $i = 1, ..., I$ and $j = 1, 2, ..., J$. Convolution feature maps:

$$q_{i,j} = \sigma(\sum_{a=1}^{I} \sum_{b=1}^{J} o_{i+a-1,j+b-1} w_{a,b}),$$

where $i = 1, ..., n - I - 1$ and $j = 1, 2, ..., m - J - 1$.

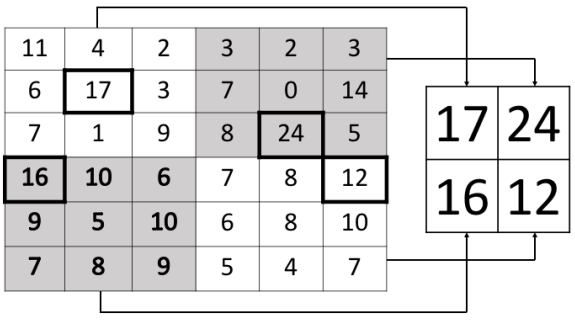

Figure 2.9: example of maxpooling.

Second, we use a skill called pooling to reduce dimension. As we said before, image has a special characteristic of invariant so we use convolution to detect other location whether or not have same feature. Pooling is just to statistically calculation on the feature of different locations. The most common pooling methods for maximum pooling and average pooling depend on how they are calculated. As figure 2.9, it is a example of max-pooling and we can find that it has a good effect in reducing dimension.

Suppose input of max-pooling is a matrix $q_{i,j}\ (i = 1, ..., n) and\ (j = 1, ..., m)$, and pooling size is $c \times d$. Output of max-pooling is:

$$p_{i,j} = \max_{1 \le a \le c,\ 1 \le b \le d} q_{(i-1)c+a,\ (j-1)d+b}$$

You could use more than once for first step and second step until the parameter space are enough small. Then we would flatten the image which were filtered as the vector and let it become as the input of fully connected neuron network as figure 2.10. You could find that using convolution and max-pooling greatly improve the speed and accuracy. So far, the CNN always a NO.1 method for image recognition. [7] Now, CNN is applied in various field such as sentence classification [6].

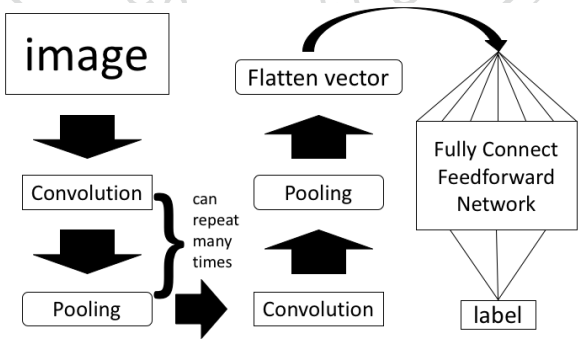

Figure 2.10: the process of CNN.

Chapter 3

Reinforcement Learning

Reinforcement learning is a learning method of machine learning, so the problems involve learning what to do—how to map situations to action—so as to maximize a numerical reward signal. Moreover, the learner is not taking an actions, as in many forms of machine learning, but instead it must find which actions can get the most reward by trying them out. Actually, reinforcement learning is not easy as you think, because actions may affect not only the immediate reward but also the next situation and, all subsequent rewards.

In the reinforcement learning problem, the most important is to face a learning agent interacting with its environment to get a goal. It's just like an agent must be able to judge what is the state of environment and must be able to take actions that affect the state. The agent also must has a goal for the state of the environment.

Reinforcement learning is different from supervised learning, because supervised learning is learning from a training set that has labeled provided by a knowledgable external supervisor for every data. Supervised learning is to generalize the function so its responses may be correct in the data non-existing in training test. This is an important learning in machine learning, but it has a problem that it could not deal with interaction issues. Actually, it is often impractical to obtain examples of labeled behavior and representative of all the situations in which the agent has to act.

Reinforcement learning is also different from one of the machine learning method of

unsupervised learning, which is typically about finding structure hidden in collections of unlabeled data. There are not only unsupervised learning field and supervised learning field in machine learning. Reinforcement learning is a kind of unsupervised learning because it does not rely on examples of correct behavior. Reinforcement learning is hope to a policy to maximize a reward signal instead of trying to find hidden structure.

Reinforcement learning takes the opposite task, starting with a complete, interactive, goal-seeking agent. All reinforcement learning agents have explicit goals, can sense aspects of their environments, and can choose actions to influence their environments.

One of the most exciting aspects of modern reinforcement learning is its substantive and fruitful interactions with other engineering and scientific disciplines. Reinforcement learning is part of a decades-long trend within artificial intelligence and machine learning toward greater integration with statistics, optimization, and other mathematical subjects.

Finally, reinforcement learning becomes a larger trend in artificial intelligence after ALPHA GO, so it hope to go back toward simple general principles.

3.1 Introduction

We consider tasks in which an agent represents one object with the ability to act interacts with an environment E in a sequence of actions a_t, observations and rewards r_t. The so-called reward is to interact the agent to implement the action with the environment. The environment will return the good and bad to use the reward to express the amount of action. [14]

The above figure 3.1 could see the whole process of interaction clearly. In fact, this is a model of interaction with human and environmental. In each time-step, the agent selects an a_t from the action set A that can be selected. The action set could be a continuous as robot control or discrete as several keys in the game.

$< A, S, R, P >$ are classical parameter space. A is an action space which collects all possible action. S is a state space that the agent can perceive in the world. R is reward

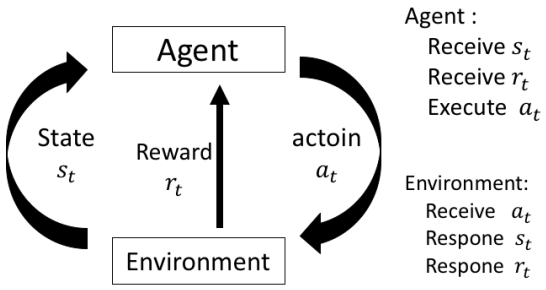

Figure 3.1: example of interaction with environment and agent.

which is real value representing reward or punishment. P is transition ($P : S \times A \to S$) i.e. it is a world which interacts with agent.

In the following, we would introduce some important concept and note.

1. Policy:

   It is that agent selects an action in a state S, so we could call it "π" and it is the most important problem in reinforcement learning. Actually, policy is a mapping from a stats S to an action A. If a policy is stochastic, then the policy selects every action according to the probability ($\pi(a|s)$). If a policy is deterministic, then it selects an action according state S.

   stochastic policy: $\Sigma\pi(a|s) = 1$

   deterministic policy $\pi : S \to A$

2. Reward:

   Reward defines the learning target of agent. At each time that agent interacts with the environment, environment could return a reward to let agent know whether the action was good or not. We also could view it as reward and punishment for agent. The most important is that reward was not equal to goal and our goal is to optimal average cumulative return rather than current reward. In the following, it

is a sequence that agent interacts with the environment.

$$s_0,\ a_0,\ r_1,\ s_1,\ a_1,\ r_2,\ ...,\ s_{n-1},\ a_{n-1},\ r_n,\ s_n\ ,$$

s_t is state at time t, a_t is an action at time t and state s_t, r_t is reward at a_{t-1} and s_{t-1}.

3. Value function:

   Reward is to judge immediate sense return in one time interaction. Value function is defined to average long-term average return. $V_\pi(s)$ is defined that policy π long-term expected return at states. $Q_\pi(s,a)$ is defined that policy π get long-term expected return from at a state s and an action a.

   Define the long-term return G_t :

$$G_t = \sum_{n=0}^{N} \gamma^n r_{t+n}$$

   Define the state value function $V_\pi(s)$:

$$V_\pi(s) = E_\pi[G_t|S_t = s]$$

   Define the action value function $Q_\pi(s,a)$:

$$Q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a],$$

   $0 < \gamma < 1$ and γ is discount rate.

However, how we get the value function?

$$
\begin{aligned}
V^{\pi}(s) &= \mathbb{E}_{\pi}(G_t|s_t = s) \\
&= \mathbb{E}_{\pi}(\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s) \\
&= \mathbb{E}_{\pi}(r_{t+1} + \gamma\sum_{k=1}^{\infty}\gamma^k r_{t+k+1}|s_t = s) \\
&= \sum_a \pi(s,a)\sum_{s'} p^s_{ss'}(r^a_{ss'} + \gamma \cdot \mathbb{E}_{\pi}(\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s')) \\
&= \sum_a \pi(s,a)\sum_{s'} p^s_{ss'}(r^a_{ss'} + \gamma \cdot V^{\pi}(s'))
\end{aligned}
$$

This important recursive relation is called the Bellman equation for V^{π}. The value function is the solution to this Bellman equation.

Reinforcement learning is a method which agent interacts with an environment E. At the beginning of the training time, policy is not trained, so we could set some rule to select actions for first and exploring different field. “Linear annealing method” is a policy of selection action. In time of training data, there is a probability randomly extracting an action and the probability will reduce as the training goes on until training end.

3.2 Temporal-Difference Prediction

Temporal-Difference (TD) and Monte Carlo (MC) method are using experience to solve prediction problem. Given some experiences of policy π, two methods could update every value V_{π} of non-terminal state s_t in the experience. The main difference between the two method are updated time. MC method has to decide incremental of V_{π} after its episode ends and TD method just wait until the next time step ends.

MC method sets V_{π} as target value after the return was known, its update method

is as follows,

$$V(s_t) \leftarrow V(s_t) + \alpha[G_t - V(s_t)]$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[G_t - Q(s_t, a_t)]$$

G_t is the sum of true value after time t and α is a fixed step-size parameter.

At time t+1, TD method immediately update estimated $V(s_{t+1})$ by using observed reward and we call this simplest TD method as TD(0). The updated method is as followed:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Compared with the above mathematical formula, TD method and Monte Carlo method are based on existing estimates to update it, so TD method is also a bootstrapping.

Roughly speaking, the target value of MC method is G_t, and the target value of TD(0) method is $V_\pi(s) = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s]$. Because $V_\pi(S_{t+1})$ was unknown, we used current estimated $V_\pi(S_{t+1})$ to update. TD(0)'s presudocode is as follow:

Algotithm 1: Tabular TD for estimating V^π

Initialize V(s) arbitrarily, π to the policy to be evaluated
Repeat (for each episode) :
    Initialize s
    Repeat (for each step of the episode)
        a $\rightarrow$ action given by π for s.
        Take action a_t; observe reward r_{t+1}, and next state s_{t+1}
        $V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$
        s $\rightarrow$ s'
    until s is ternimal

3.3 Q-Learning

Q-learning is also one kind of TD method and is also a kind of Value-based method. Value-based method is just to evaluate every Q-value of action, and then it choose optimal policy ($\pi(a|s)$) according to Q-values. The core of the Q-learning algorithm is famous Bellman optimality equation, i.e.

$$Q_*(s,a) = E[R_{t+1} + \gamma \cdot \max_{a'} Q(s_{t+1}, a') | S_t = s, A_t = a]$$

Q-learning proposed a way to update the Q-value:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

It computes target Q-value according to vale iteration but it does not update directly Q-value by the Q-value, just be calculated. It is similar to stochastic gradient descent and finally it could converge to optimal Q-values. In the following, it is the pseudocode of specific algorithm:

Algotithm 1: One - step Q - learning

Initialize Q(s,a) arbitrarily, π to the policy to be evaluated

Repeat (for each episode) :

- Initialize s
- Repeat (for each step of the episode)
  - Choose a_t from s_t using derived from Q(e.g., ϵ-greedy)
  - Take action a_t; observe reward r_{t+1}, and next state s_{t+1}
  - $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$
  - $s \rightarrow s_{t+1}$
- until s is ternimal

3.4 Deep Q-Learning

Q-learning is a classic algorithm, but there still exists a problem that it is a tabular form. That is to say that is statistic and iterate Q-value according to past experience and state. The form would cause the state and action space of Q-learning are very small. On the other hand, there is a never seen state and Q-learning could not predict and it. That is to say Q-learning is not the ability to predict at all, that is no generalization.

To let it have predict ability, we could easy think that it is actually a linear regression. Use function to fit it:

$$Q(s, a : \theta) \approx Q(s, a),$$

θ is model parameter.

There are many kinds model as linear and non-linear model. In recent year, with deep learning developing great success in the field of supervised learning, it could use deep learning to fit Q-value. It is called "DQN". In 2013 year, deepmind publish [?] that DQN is to understand classic model of DRL (Deep Reinforcement learning). At network structure design, some input are original states and an action, and its output is Q-value of the action. After DQN was developed, the network structure was changed. Its input is a state, and output are Q-value of all action in the state.

The left of figure3.2 is before DQN and the right is after DQN. It uses DQN to solve

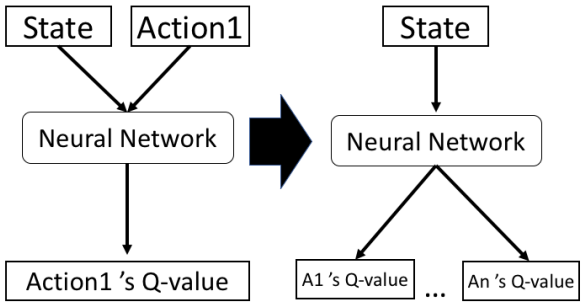

Figure 3.2: two kinds network structure of reinforcement learning.

problem of atari game in the paper. Its input (state s) are the original pixel of the game screen and output (action space) are direction of rocker. The biggest advantage is that it

saved the trouble of doing the feature engineering and greatly improved performance.

Because DQN was actually a regression, the optimization target of model was to minimize the loss of 1-step TD error and its gradient was also directly as shown below.

1. Here, we use simple squared error

$$L(w) = \mathbb{E}[(r + \gamma \cdot \max_{a'} Q(s', a' : \theta)) - Q(s, a : \theta)]$$

2. Leading to the following Q-learning gradient

$$\frac{\partial L(\theta)}{\partial \theta} = \mathbb{E}[((r + \gamma \cdot \max_{a'} Q(s', a' : \theta)) - Q(s, a : \theta)) \frac{\partial Q(s, a : \theta)}{\partial \theta}]$$

3. Optimize objective end-to-end by SGD

The main reason of DQN's success are that it uses NN to fit function of Q-values and trains model from end-to-end. However, there are two skill to improve effectiveness and stability [?]:

1. ***Experience Replay*:**

   In the deep learning of supervised learning, the samples are independent and identically distributed, but the sample of reinforcement learning are highly correlated and non-stationary and it could lead to train result had difficulty in convergence. The idea of Experience Replay is really simple, and it is just to build a store to store in itself. Each of these experiences are stored as a tuple of <state, action, reward, next state>. Then, sample are randomly sample from the store to remove the correlation. [4]

2. ***Separate Target Network***

   The second major addition skill to the DQN that is the utilization of a second network during the training procedure. This second network is used to generate the target-Q values network that will be used to compute the Q-value for every action, rather than using pre-updated Q-network directly. The issue is that at every step of

training, the Q-network's values shift, and if we are using a constantly shifting set of values to adjust our network values, then the value estimations can easily spiral out of control. If the target Q and estimation Q-values are the same, the network can become destabilized by falling into feedback loops between them. To mitigate that possibility, the target network's weights are fixed, and only periodically or slowly updated to the primary Q-networks values. So the loss function could change into the following form:

$$L(w) = (r + \gamma \max_{a'} Q(s', a' : \theta^-)) - Q(s, a : \theta)$$

As shown in the above loss function, the weights used to calculate the target Q value are w-, not w.

There are three main method to improve the efficiency of DQN in atari game, like Double DQN [?], Prioritised Replay and Dueling Network [15]. David Silver introduced the Tutorial in ICML 2016: Deep Enhancement Learning Tutorial. The following introduces it:

1. ***DoubleDQN*** **:** Remove upwards bias caused by $\max_a Q(s, a : \theta)$

   1.Current Q-network w is used to select actions

   2.Older Q-network w^- is used to evaluate to actions

$$L = (r + \gamma Q(s', \operatorname{argmax}_{a'} Q(s', a', \theta) : \theta^-) - Q(s, a : \theta))^2$$

2. ***Prioritised Replay*** **:**

   Store experience in priority according to DQN error.

$$|r + \gamma \max_{a'} Q(s', a' : \theta^-) - Q(s, a : \theta)|$$

3. ***Dueling Network*** **:** Split Q-network into two channels.

   1.Action-independent value function $V(s)$.

2.Action-independent advantage function $A(s, a : \theta)$

$$Q(s, a : \theta) = V(s) + A(s, a : \theta)$$

Double Q-network has two networks as mentioned in the table above, and its main purpose is to reduce computational error due to maximum Q-value, or it is called over-estimated problem. You could think it that if there are double-check in your work, the error would significantly reduce. ??

Prioritized replay is actually as the name implies. Original mechanism of replay is to random sample and prioritized replay sample according to priority experience and replay weight. The calculation of the priority is used by difference of target Q-value and Q-value. If an object of prior is high, the probability sampled is high.

Dueling Network separate Q-network into two channel, one output V, the other output A. In other words, Q(s,a) decomposed into two more fundamental notions of value, V(s) and A(s,a). The V-function is independent of the action, and it says simple how good it is to be in any given state. The A-function is dependent of the action. A(s,a) tells how much better taking a certain action would be compared to the others and is to solve problem of reward-bias, so we also call it advantage function A(s,a). The goal of Dueling DQN is to have a network that separately computes the advantage and value functions, and combines them back into a single Q-function only at the final layer. [15]

3.5 Policy-Based Method

We have analyzed the DQN algorithm in detail and it is an algorithm based on value. We analyze another algorithm in deep reinforcement learning, that is an algorithm based on the policy. The detail is introduced in [?] The value method is to compute a value for

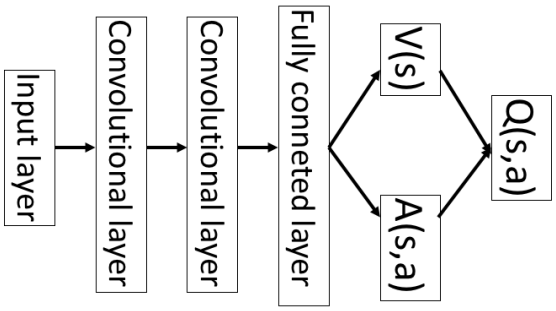

Figure 3.3: The dueling network.

every action and state and then select the maximum value. This is an indirect approach.

Now, we introduce a direct method to update policy network. Policy network is actually a neural network. Its input is also states and its output are actions. I represent policy by deep network with weights θ.

$$a = \pi(a|s,u) \quad or \quad a = \pi(s,u)$$

or output is probability of action: $a = \pi(a|s,u)$. There are two advantage when policy-based is better than value-based. One is that its output is probability rather than Q-value is that person could not always choose same behavior. None could always act same action but DQN could not output probability, so the using policy network is a better way. Two is that DQN need to know every q-value of action. If action is continuous, DQN will be inappropriate.

In policy network, we define its objective function as total discounted reward

$$J(u) = \mathbb{E}[r_1 + \gamma r_2 + \gamma^2 r_3 + ...|\pi(\cdot,u)]$$

This objective function is the cumulative expectation of all declining rewards and we want to maximize it. The policy network would optimize objective end-to-end by SGD. Actually, it is to adjust policy weight u to achieve more reward. How to make adjust action

to enhance the value:

Theorem 3.5.1. For any differentiable policy $\pi_u(s,a)$, for any of the policy objective functions $J(u)$, the policy gradient is

$$\nabla_u J(u) = \mathbb{E}_{\pi_u}[\nabla_u log\ \pi_u(s,a) Q^{\pi_u}(s,a)]$$

The theorem is published by [?]. Actually, the idea of policy gradient is easy. If an action could achieve a good result, we enhance its probability. If an action could achieve a bad result, we reduce the possibility of being drawn. In the following, we will introduce that the famous policy gradient method is REINFORCE and its persudocode.

Algotithm 3: REINFORCE

Initialize θ arbitrarily
for each episode $s_1, a_1, r_2, ..., s_{t-1}, a_{t-1}, r_t\ \pi_\theta$ do:
  for $t = 1$ to $T - 1$ do:
    $\theta \leftarrow \theta + \alpha \nabla_\theta\ log\ \pi_\theta(s_t, a_t) v_t$
  end for
end for
return θ

3.6 Actor-Critic

Actor-Critic is also a TD method and it combines the value-based and policy-based methods. Policy network is actor and it is to output action (action-selection). Value network is critic and it actually be used to evaluate the action which been selected by actor network is good or bad (action value estimated).

Then, the value network would generate TD-error, and it guide to update the actor

network and critic network. The following figure shows a structure of the actor-critic algorithm, DDPG is this type of famous algorithm.

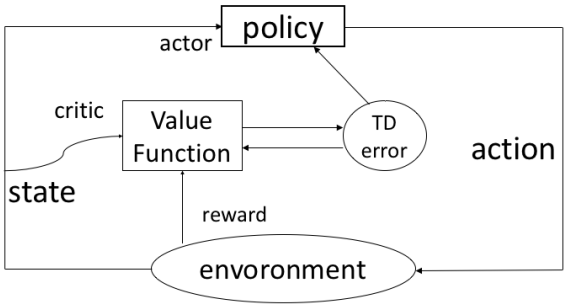

Figure 3.4: The structure of actor-critic algorithm.

Google's paper (DDPG) [10] successfully combined some skill in DQN and DPG, and it make deep reinforcement learning was push to continuous control. In DDPG, the input of actor network is state and output is action. We use DNN to fit function of actor network. If action is continuous, the output could be tanh or sigmod. If action is discrete, it could use softmax layer to output probability. The input of critic network is state and action, and its output is Q-value.

1. DPG

   In section 3.5, it supply the formula and proof of policy gradient .

2. DQN

   The critic network in DDPG use the skills which are experience replay and target network in DQN. The two ways are also to stable the train model.

3. Noise sample

   If now action is continuous, reinforcement learning will encounter a exploring trouble. DDPG uses add noise on action.

$$a = \pi'(s_t) = \pi(s_t : u_t^{\pi}) + \varepsilon,$$

   and ε is noise.

3.7 Asynchronous Advantage Actor-Critic (A3C)

We will introduce the asynchronous and advantage.

1. Asynchronous

   In 2015, google's Gorila framework publish paper [12] and it say about Asynchronous Distributed RL Framework. Gorila adopted separate machines and a parameter server, and A3C is similarly to it. There is a little difference between the method by Gorila and A3C, it is multiple CPU threads on a single machine. Why we abandon to use distributed framework of many machine? The reason is that using a single machine could save communication costs of sending gradients and parameters. In [11], it has verified that the iteration is significantly faster. That is the first main advantage of the A3C.

   Now, we will introduce second main advantage of A3C published in 2016 year. [11] It uses multiple actors-learners running in parallel, and then we will explicitly use different exploration policies in each actor-learner to maximize this diversity. By running different exploration policies in different threads, multiple actor-learners applying online updates in parallel are likely to be less correlated than a single agent. Hence, we do not need to use experience replay mechanism to train in DQN.

   In addition, the A3C uses the CPU to train rather than GPU, because the training process batch in RL is very small, GPU has many time in waiting for new data.

2. Advantage Actor-Critic

   In section 3.5, we have mentioned that standard REINFORCE updates the policy parameters θ in the direction $\nabla_\theta \ log \ \pi_\theta(s_t, a_t)V_t$, which is an unbiased estimate of $\nabla_\theta \mathbb{E}[R_t]$. We could use a new learned function of state $b_t(s_t)$, known as a baseline, and make the unbiased estimate subtract it. The skill could reduce the variance of this estimate. The resulting gradient is $\nabla_\theta \ log \ \pi_\theta(a_t, s_t)(R_t - b_t(s_t))$, and then

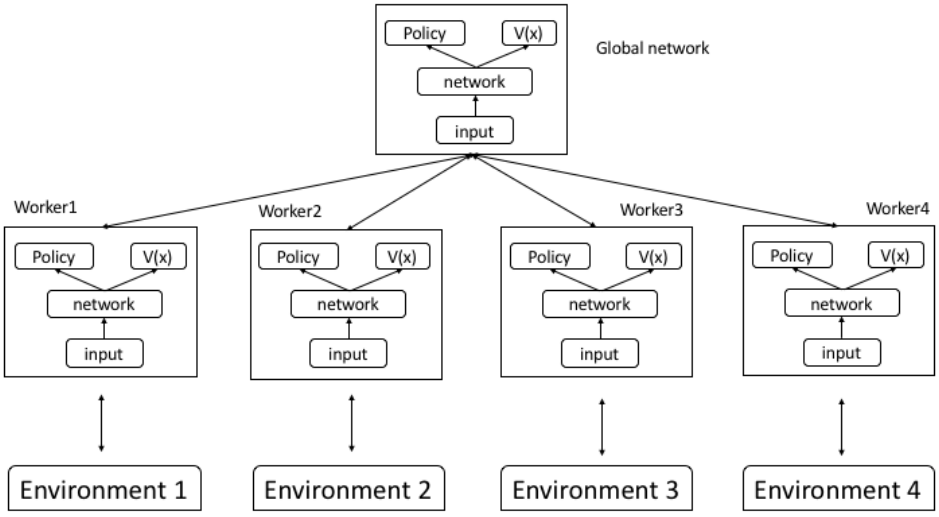

Figure 3.5: The abstract structure of asynchronous method.

using state function to estimate the baseline. In addition, we also use value function $Q^{\pi}(a_t, s_t)$ to estimate the reward R_t. When an approximate state function uses to estimated as the baseline and R_t is an estimation of $Q^{\pi}(a_t, s_t)$, the quantity $R_t - b_t(s_t)$ can be seen as an estimate of the advantage of action a_t in state s_t i.e, $A(a_t, s_t) = Q(a_t, s_t) - V(s_t)$. Therefore, advantage function $A(a_t, s_t)$ could evaluate how good and bad are action a_t in state s_t.

Chapter 4

Exchange-Traded Fund

ETF stands for Exchange-Traded Fund. An ETF trades like a stock on a stock exchange and looks like mutual fund. Now, we could spilt ETF into three parts, as following figure, to introduce it.

First, one part is index because its performance tracks an underlying index, which

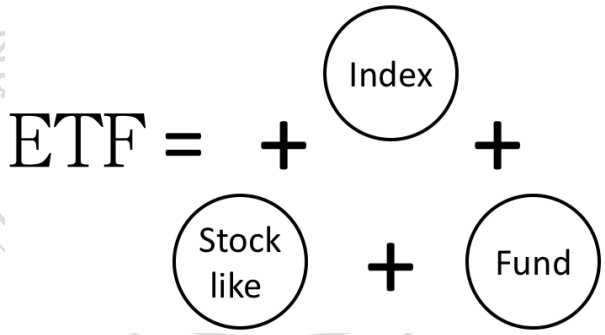

Figure 4.1: the structure of ETF.

the ETF is designed to replicate. Another is stock-like. ETF has a unique trading architecture design such that it can trade on Stock Exchange Market like stock. The remaining part is fund part because ETF has a lot of places like mutual fund. ETF is delivered by Taiwan Stock Exchange Corporation. It also like a mutual fund, because it also has a prospectus. An ETF delivers a formal legal document to the retail purchaser or provides investors with a document, which summarizes key information about the ETF.

In conclusion, ETFs are designed to track specific market indexes, they combine the broad diversification of a mutual fund with the trading flexibility of a stock. Actually,

ETFs are just what their name implies: baskets of securities that are traded, like individual stocks, on an exchange (primarity the Taiwan Stock Exchange Corporation).

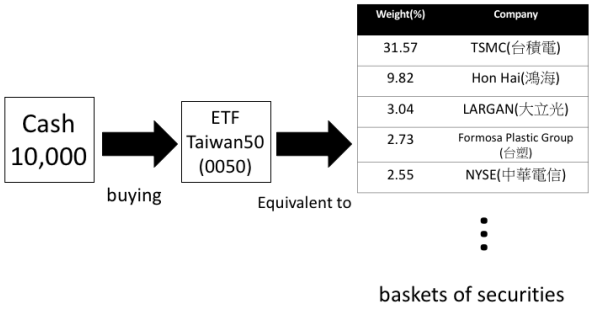

Figure 4.2: ETF illustration : Taiwan50 as an example.

4.1 Exchange Traded Funds

1. The big difference with ETF and mutual fund:

   The big difference between them is the way of trading. Investor are only selling and buying fund with fund company and you could not sell umber units of on-hand fund for other investors. Because ETFs are listed in stock market, you could sell and buy ETF with whom is willing to.

   Their trading mode are different, therefore fund only has a final price at one day and it is actually the net closing price in that day. ETFs are trading at market price and you can trade in opening time.

2. Investment in ETF:

   ETF is listed trading on stock market, so the way how invest in ETF is to trade ETFs by their securities trading accounts with a brokerage house. The NAV of the fund is calculated and will be disclosed by the TSEC in every 15 seconds during the trading hours, exactly the same frequency as the updates on the Index. In Taiwan, you can open an account in taiwan securities firm and then you could buy all Taiwan

ETF(example: 0050, 0056, ...).

3. The Lowest Investment Amount:

   The amount is affected by two main factor, one is the lowest number of shares that you can buy or sell in the stock market, and the other is a unit price os ETF. The lowest number is different according to local conditions. For instance, trading unit is a lot which is 1000 shares in Taiwan stock market, so you buy a 1000 share at least. Supposed that one ETF price is 50$, then you need at least 50,000$ to invest the ETF. But in America, minimal trading unit is a share so you only need ETF price to buy the ETF.

4.2 Advantage of ETF

1. Easy of Trading

   The trading of ETF is same as stocks so in the stock trading hours, investors can order at any time through the securities broker.

2. Fees and Commission

   ETFs charge are lower than most comparable index mutual funds, because unlike mutual funds, ETFs do not need require the support of a research team for each individual stock. ETFs are passive Investing so managers only passively adjust the constituents in the portfolio by tracking and monitoring the pulse of the market.

3. Tax Efficiency

   In transaction tax, ETFs is only 1 percentage and it is lower than stock trading and mutual fund.

4. Diversification

   ETF is comprised of a basket of securities, so it could been tiny affected by down of individual stocks. Therefore we can see it as trend and reduce investment risk.

5. Completing Portfolio

	Mutual Fund	ETF
Management and Marketing	1~3%	0.3%~0.8%
Transaction Tax	0.3%	0.1%
Total	1.3%~3.3%	0.4~0.9%

Figure 4.3: ETF illustration : Taiwan50 as an example.

ETFs are designed to tracking index and the information of the constituent stocks of the Benchmark Index, performance comparison between the Benchmark Index and other important data and information may be downloaded from TSEC and fund house websites. In open market, ETF can update its latest net value every fifteen seconds in trading time, so investors can grasp the change of ETF price at any time and trade in close to the net value of the fund.

4.3 Example of ETF

The first ETF was the S&P 500 index fund, which began trading on the American Stock Exchange (AMEX) in 1993. Today there are many types of ETF like sector-specific, country-specific and broad-market indexes and hundreds of them are trading in open market. Most ETFs are passively managed, meaning investors could save big on management fees. Below we will introduce some popular ETFs:

1. Nasdaq-100 Index Tracking Stock (Nasdaq: QQQ)

   This ETF represents the Nasdaq-100 Index, which consists of the 100 largest and most actively traded non-financial stocks on the Nasdaq. The QQQ is a great ETF for who want to invest in future of the technology industry, because it can avoid risk that investing in individual stocks. The diversification which it offers can be a huge advantage when there's volatility in the markets.

2. SPDRs

SPDR is a family of exchange-traded funds (ETFs) traded in the United States, Europe, and Asia-Pacific and managed by State Street Global Advisors (SSGA). It is designed to track the S&P 500 stock market index and give you ownership in the index. This is good because it saves trouble and expenses involved in trying to buy all 500 stocks in the S&P 500.

3. iShares

   iShares are a family of exchange-traded funds (ETFs) managed by BlackRock. Each iShares ETF also tracks a stock market index on many of the major indexes around the world including the Nasdaq, NYSE, Dow Jones, and Standard & Poor's. In the U.S, iShares is the largest issuer of ETFs and trade on the major exchanges.

4. Taiwan50

   The full name of Taiwan50 is Yuanta Taiwan Top 50 ETF, and its code is 0050. As the name implies, it is ETF tracking index on the top 50 largest and most representative stocks in Taiwan. The ETF is just a basket of stock and it is to diversify risk. The detail list of company in Taiwan could be found in TWSE.

Chapter 5

Automated Trading System

It has introduced reinforcement learning and ETF. Now, it would use DQCN model to learn how to buy ETF. Through the AI (=RL + DL) model design, i want an agent to learn the trend of stock market for maximize our profit.

5.1 ETF data

We randomly select 40 different ETF which can be gotten from the internet. Furthermore, each ETF is extracted from the recent five-year data from the database. After extracting the data, it was divided into the two parts. The first 20 are training part and the rest is testing datasets. Every ETF has the 1260 days of transaction in five years and six feature in a day, so every size of data is 1260×6.

Six feature are Open, High, Low, Close, Volume, and Adjust Close respectively. Open represents the opening price that is the current selling price for a ETF at the time that the exchange opens each trading day. High is the highest price for a ETF in a trading day. Low is relative to the high and it is the lowest price. Close is the closing price that is the final price at which a security is traded on a given trading day. Volume is the numbers of ETF traded in an entire market during a given period of time. An adjusted closing price is a closing price on any day of trading that has been revised to include any distributions and corporate actions that occurred at any time before next day's open.

However, the selected datas have different scale of range, so they use normalization to

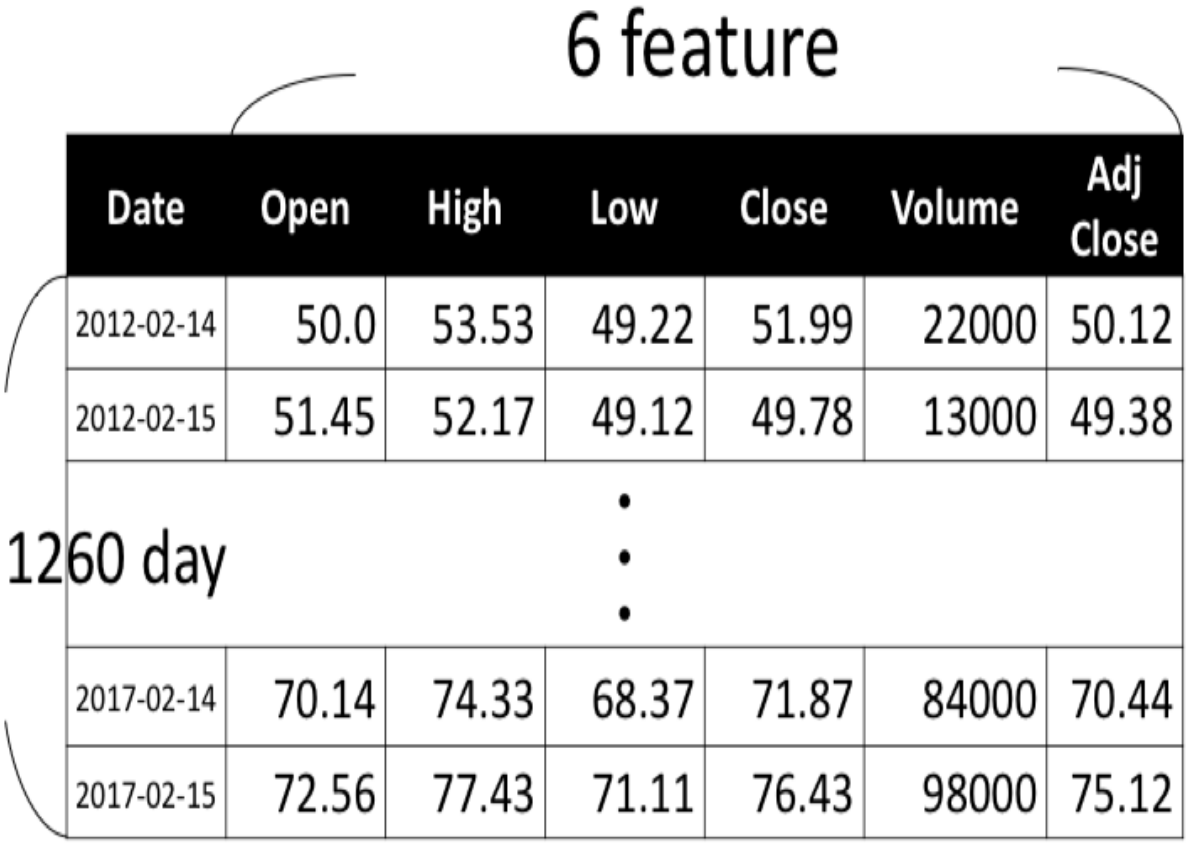

Figure 5.1: The example of ETF data.

adjusting value measured on different scale to a notionally common scale. Normalization could make different scale data be meaningful comparison. There are many different methods of normalization, and in following these normalization method will be introduced.

1. Standard score

   Formula:

   $$\frac{x_i - \mu}{\sigma} ,$$

   x_i is one value which would be adjusted in data, μ and σ are the populations mean and the population standard deviation. The adjusted value range is real. This method work well when population parameters are known and the population are normally distributed.

2. Student's t-statistic

   Formula:

$$\frac{x_i - \bar{X}}{s},$$

   the skill is similar to the method above, but it is usd when the populations parameter is unknown. $\bar{X}$ and s are the sample mean and standard deviation.

3. Feature scaling

   Formula:

$$\frac{x_i - X_{min}}{X_{min} - X_{max}},$$

   X_{max} and X_{min} are minimum value and maximum value in the set. Feature scaling is able to transform all values into $[0, 1]$.

So, it selects feature scaling to eliminate differences in scale and the five price are seen as a set to normalize. The trading volumes are dependent and special. The max is maximum value in set of volume, and the min is set to 0. The data has been processed and then it will use them in reinforcement learning. My training data are DVY, DIA, DBO, XLB, VV, ADRA, EWV, GII, FWDI, FXG, DON, DFJ, DWX, DTD, FEX, DLS, FAB, VTWV, XPH, EDC.

5.2 Automated Trading System

5.2.1 Introduction

The goal is to construct an automated trading system to maximize the asset in the end. The system judges a trading action through data of nearly 20 trading days during 4 years. Because of trading cost and commission, it does an action every 20 trading day. We supposed that the system use 20,000 dollars as initial asset to operate one ETF every 20 trading days in finical market and then sell all on-hand ETF at the end.

In training data, we sampled randomly 20 various ETFs to train first four-year information and detect variety of reward. In testing, the recent one-year data were extracted

from the remaining 20 ETFs. We use the trading system to run them and view the reward rate.

5.2.2 Definition

1. Environment:

   The environment represents the entire financial market. We could do an action in the environment and it can return a reward. Now, the paper simplify the environment to 40 various ETF and agent will interact with them.

2. State

   State is actually the input of neural network. Because we hope that agent observes every 20 day to judge a action. ETFs have six feature such as open, close,..., and adj close. The state is size 20×6 of matrix The data are normalized as following.

Open	High	Low	Close	Volume	Adj Close
0.8325	0.8412	0.8319	0.8386	0.8685	0.8321
0.8278	0.8337	0.824	0.8337	0.5113	0.8273
0.82	0.8251	0.8174	0.8179	0.2835	0.8116
0.8214	0.8231	0.8185	0.8197	0.2331	0.8134
0.8277	0.8278	0.8197	0.8212	0.2687	0.8149
0.8408	0.842	0.8263	0.8294	0.3175	0.823
0.8358	0.8425	0.8357	0.8405	0.2451	0.834
0.8341	0.8397	0.8307	0.8345	0.1863	0.8281
0.8386	0.8394	0.8311	0.8329	0.1811	0.8265
0.8366	0.8403	0.8344	0.8377	0.1325	0.8312
0.8376	0.8398	0.8365	0.8384	0.1457	0.8319
0.8393	0.842	0.8355	0.8373	0.2127	0.8308
0.8345	0.8378	0.8324	0.8364	0.2779	0.8299
0.8405	0.842	0.8364	0.8382	0.3073	0.8317
0.8413	0.8444	0.8384	0.8438	0.145	0.8373
0.8435	0.845	0.8382	0.841	0.1217	0.8345
0.8384	0.8408	0.8369	0.8371	0.2	0.8306
0.841	0.8451	0.8382	0.8382	0.1279	0.8317
0.8353	0.8418	0.8318	0.8387	0.3173	0.8322
0.8365	0.8418	0.8353	0.8399	0.1466	0.8334

3. Action

We suppose that agent act only five actions due to DQN which is learning discrete control. These actions are buying 20 units ETF, buying 10 units ETF, holding, selling 10 units ETF, and selling 20 units ETF, except for the last day. In the last day, we hope the agent empty all hand-on ETF and then confirm whether agent can earn money.

Action	Purchase Quantity
0	20 units
1	10 units
2	hold
3	-10 units
4	-20 units

4. Reward

   There are some reward forms, and different form has different effect. It will example two form: First,

$$\text{reward} = \text{this time total asset} - \text{last time total asset},$$

   total asset is the sum of cash and the value of hand-on ETF. Its meaning is actually change in ETF, because cash is fixed. Second,

$$\text{reward} = log(\frac{\text{this time total asset}}{\text{last time total asset}}) ,$$

   its meaning is the reward rate and the paper adopt this reward.

5.2.3 Initial Parameter Settlement

We supposed the agent has 20,000 and no ETF on hand at the begining, so the parameter of "cash" and "number of ETF" are 20,000 and 0. Because the fee of ETFs are high and traded every 20 trading day, we set the parameter "day" is 20. The parameter of "training step" is 10,000,000, and it means the number of training. The system trains different ETF at every training. We hope that the system maximized the assets in the end, so we do not consider discount rate and set gamma = 1.

At the section 3.3, the skill ,"separate target network", describes that target weight

which compute Q-values of action, are different from updating weight. The target weight is postponed of updating weight and the parameter of " target update step" representing a fixed number of steps which update target weight by updating weight. It was set 100, i.e. target weight are equal to updating every 100 training step. We choose "Linear annealing method" introduced at section 3.1 as policy.

cash	20,000
number of ETF	0
day	20
training step	10,000,000
gamma	1
target update step	100
policy	Linear annealing method

5.2.4 Neural Network

The architecture of our network is summarized in figure 5.2. It contains five learned layers —two convolutional and two fully-connected. Input of the neural network is a 20×6 matrix, which is a state.

First layer is convolution. We set its kernel size and activation function to 2×2 vector and ReLU function. The parameter "filter" are 64. Second layer is also a convolution, and its activation function and filter are also ReLU function and 64. The only difference between the two layers is kernel size 3×3. Because the output of convolution is many matrix and we want to connect fully connected layer, so it first connect a flatten layer, which flatten output to a vector row by row.

Third layer and fourth layer are all fully connected layer, and their activation function are also ReLU function. The distinction between them are "units", which is a number of how many neuron in a layer, and we respectively set them as 300 and 100. Then, the output is a 5-dimension vector which its entry represents the Q value of an action, respectively.

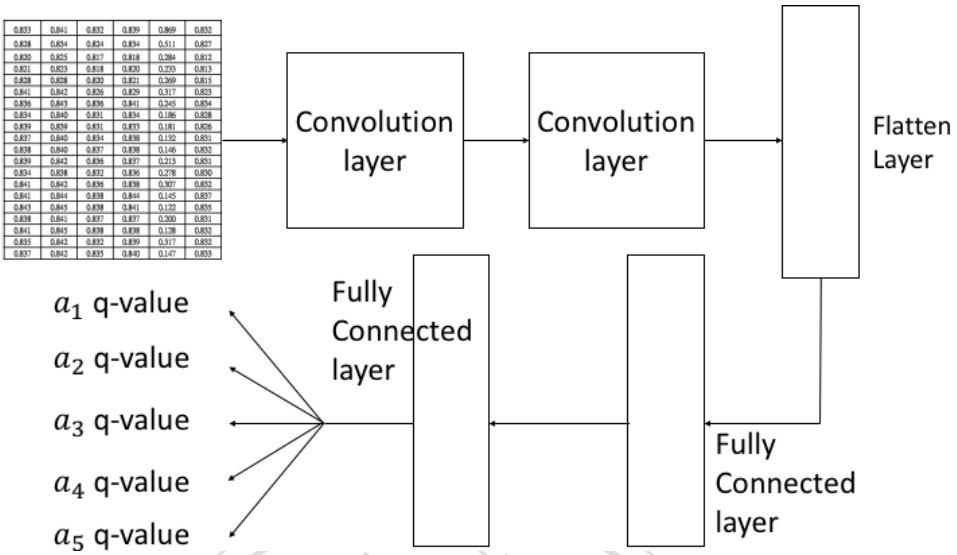

Figure 5.2: The model of CNN.

5.2.5 DCQN

The model, which connects Q-learning and CNN is called DCQN. It uses CNN to predict longterm return and uses q-learning find optimal policy. The process of DCQN is in figure 5.3.

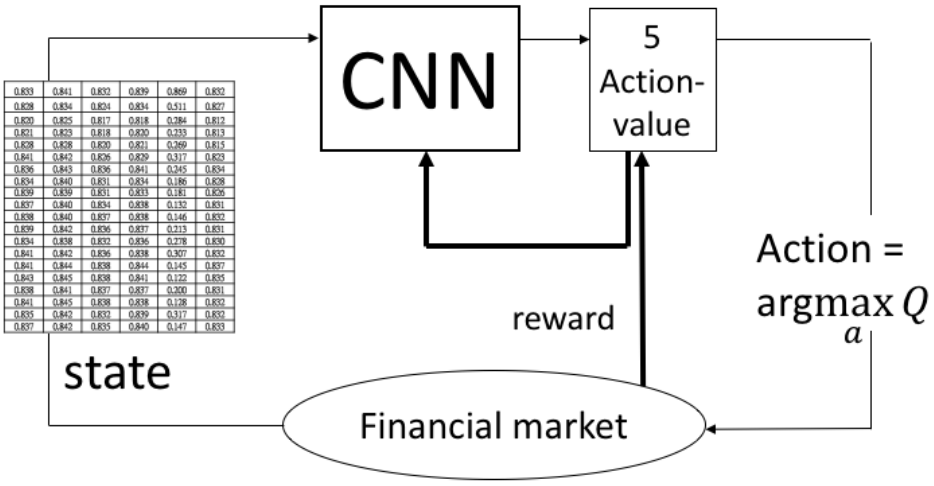

Figure 5.3: The model of DCQN.

5.3 Result

Is the following table, there are 20 results of test data, which are randomly sampled, and we can find that three ETFs of DQCN result are almost more than the result the results suppose to take all money to buy ETF in the beginning and sell all on-hand ETFs at the end. Among all the ETFs, four of them have reward rate of DQCN higher than basic reward rate for over fifteen percents. Five reward rate of DQCN are greater than basic reward rate for over five percent. Ten of them get higher reward rate of DQCN than basic reward rate. There are only two of them having worse reward rate than basic reward rate, and only one ETF achieve negative reward rate of DQCN. So, most ETFs in test data have better results than basic reward rate.

Dataname	Reward Rate of DQCN	Basic Reward Rate
XLG	17.71%	11.97%
VQT	16.3%	1.32%
GMF	11.09%	13.7%
VSS	10.76%	9.73%
FXA	9.3%	2.18%
CORP	7.8%	-1.08%
VOX	6.97%	3.91%
FXC	6.9%	0.38%
BWV	6.69%	11.57%
VPU	6.23%	0.46%
BIV	4.64%	-3.83%
DEF	3.96%	10.12%
EDV	3.94%	-13.77%
XRT	3.94%	-1.92%
EMIF	2.78%	4.76%
EFAV	2.55%	-2.43%
WREI	2.17%	-0.2%
WEAT	0.39%	-15.61%
DTUL	-0.26%	-2.84%
DGP	-1.27%	-7.84%

Chapter 6

Conclusion

Trading ETF by DQCN is a new attempt, and we get a perfect effect. The original ideas are that making use of neural network to forecast market trend and to evaluate the merits of each action. And then utilize reinforcement learning to select optimal actions. In the paper, we utilized different skill to increase effect, and it includes Q-learning combined with CNN. We use CNN to predict future excepted reward as time series, and it is different from traditional method using RNN. Q-learning is used to find an optimal action-selection policy for any given environment. It works by learning an action-value function in a given state and following the optimal policy thereafter. However CNN is used to fit action-value function.

The results created by the trading system were generated by self-learning of computer. Furthermore, the results are satisfying. The trading system was born through computer learn self, and it has a good result. In section 5.3, it shows that eighteen ETFs have better reward rate by DQCN than basic reward rate, and reward rate of two ETFs are worse than average reward rate. Only one ETFs is negative reward rate, so we could say that the trading system could earn money for most ETFs, and the effects created by the trading system are better than basic reward rate. Particular three ETFs have Significantly greater reward by DQCN, so automated trading system could achieve great outcome.

Although some rewards of trading in small parts of ETFs are not ideal, most ETFs could get well outcome. Therefore, we hope that reward rate could be better, and negative reward rate would be not exist. Hence there are two point which can be improved. One point is to improve neural network model. In this paper, it is using CNN as time series model to forecast Q- values and could use RNN or other neural network to observe whether the reward rate is better. The other point is to change reinforcement learning. A3C is the fastest and best reinforcement learning model so far. Therefore we also want to try it.

Bibliography

[1] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. arXiv preprint arXiv:1703.04691, 2017.

[2] Guglielmo Maria Caporale, Juncal Cuñado, and Luis A Gil-Alana. Modelling long-run trends and cycles in financial time series data. Journal of Time Series Analysis, 34(3):405–421, 2013.

[3] Thira Chavarnakul and David Enke. Intelligent technical analysis based equivolume charting for stock trading using neural networks. Expert Systems with Applications, 34(2):1004–1017, 2008.

[4] Tim de Bruin, Jens Kober, Karl Tuyls, and Robert Babuška. The importance of experience replay database composition in deep reinforcement learning. In Deep Reinforcement Learning Workshop, NIPS, 2015.

[5] John Cristian Borges Gamboa. Deep learning for time-series analysis. arXiv preprint arXiv:1701.01887, 2017.

[6] Yoon Kim. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882, 2014.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[8] Ramon Lawrence. Using neural networks to forecast stock market prices. University of Manitoba, 1997.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553): 436–444, 2015.

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

[11] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International Conference on Machine Learning, pages 1928–1937, 2016.

[12] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. arXiv preprint arXiv:1507.04296, 2015.

[13] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229, 2013.

[14] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction, volume 1. MIT press Cambridge, 1998.

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

[16] Yudong Zhang and Lenan Wu. Stock market prediction of s&p 500 via combination of improved bco approach and bp neural network. Expert systems with applications, 36(5):8849–8854, 2009.