A Framework of Hierarchical Deep Q-Network for Portfolio Management

Yuan Gao[1,*], Ziming Gao[1,*], Yi Hu[1], Sifan Song[1], Zhengyong Jiang[1] and Jionglong Su[2]
[1]*Department of Mathematical Sciences, Xi'an Jiaotong - Liverpool University, Suzhou, P. R. China*
[2]*School of AI and Advanced Computing, XJTLU Entrepreneur College (Taicang), Xi'an Jiaotong - Liverpool University,*

Keywords: Q-Learning, Hierarchical Reinforcement Learning, Convolutional Neural Network, Portfolio Management.

Abstract: Reinforcement Learning algorithms and Neural Networks have diverse applications in many domains, e.g., stock market prediction, facial recognition and automatic machine translation. The concept of modeling the portfolio management through a reinforcement learning formulation is novel, and the Deep Q-Network has been successfully applied to portfolio management recently. However, the model does not take into account of commission fee for transaction. This paper introduces a framework, based on the hierarchical Deep Q-Network, that addresses the issue of zero commission fee by reducing the number of assets assigned to each Deep Q-Network and dividing the total portfolio value into smaller parts. Furthermore, this framework is flexible enough to handle an arbitrary number of assets. In our experiments, the time series of four stocks for three different time periods are used to assess the efficacy of our model. It is found that our hierarchical Deep Q-Network based strategy outperforms ten other strategies, including nine traditional strategies and one reinforcement learning strategy, in profitability as measured by the Cumulative Rate of Return. Moreover, the Sharpe ratio and Max Drawdown metrics both demonstrate that the risk of policy associated with hierarchical Deep Q-Network is the lowest among all ten strategies.

1 INTRODUCTION

Profitable stock trading strategy is a process of making decisions based on optimizing allocation of capital into different stocks in order to maximize performance, such as expected return and Sharpe ratio (Sharpe, 1994). Traditionally, there exist portfolio trading strategies which may be broadly classified into four categories, namely *"Follow-the-Winner"*, *"Follow-the-Loser"*, *"Pattern-Matching"*, and *"Meta-Learning"* (Li and Hoi, 2014). However, in real financial environments with complex correlations between stocks as well as substantially noisy data, such traditional portfolio trading strategies tend to be limited in their usefulness.

To date, several deep machine-learning approaches have been applied to financial trading (Park et al., 2019) with varying degrees of success. Nevertheless, many of them tend to predict price movements by inputting historical asset prices to output a prediction of asset prices in next trading period via neural network, and the trading agent will take action based on these predictions (Heaton et al., 2016). The performance of these algorithms is highly dependent on the accuracy of future market prices, and it seems inappropriate to convert price predictions into actions because they are not part of the market actions. Therefore, these approaches are not fully machine learning based.

More recently, the applications of Reinforcement Learning (RL) methods in portfolio management are proposed, where these approaches are able to trade without predicting future prices (Dempster and Leemans, 2006). Most of these are related to policy-based RL such as Policy Gradient (Jiang et al., 2017), which are suitable for continuous actions in the stock scenario. However, with appropriate action discretization in stock trading, several valued-based RL methods such as Q-Learning have been applied as well. In (Gao et al., 2020), a Deep Q-Network (N-DQN) framework combining Q-Learning with a single deep neural network has been proposed. This framework allows the N-DQN agent to optimize trading strategies through learning from its experience in the finan-

* These authors contribute equally to this work.

Gao, Y., Gao, Z., Hu, Y., Song, S., Jiang, Z. and Su, J.
A Framework of Hierarchical Deep Q-Network for Portfolio Management.
DOI: 10.5220/0010233201320140
In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021) - Volume 2*, pages 132-140
ISBN: 978-989-758-484-8

cial environment, so that this agent may adapt strategies derived from historical data to actual trading. Nevertheless, the single DQN structure of the N-DQN is unable to handle extremely large action space, and large capital sectors will lead to higher commission fee during trading process. Therefore, the algorithm in (Gao et al., 2020) makes an unrealistic assumption of zero commission fee.

In this paper, we address this shortcoming by adopting hierarchical DQN structure (Kulkarni et al., 2016) to obtain a novel hierarchical Deep Q-Network (H-DQN) which takes into consideration of commission fee in transaction. It successfully decreases the number of assets assigned to each DQN and reduces the number of actions. Thus, the problem associated with high commission fee is solved and the functionality of the algorithm is greatly improved. To the best of our knowledge, the application of DQN to portfolio management with the consideration of commission fee is entirely novel.

Our key contributions are two-fold. first, we reduce the action space by proposing a H-DQN framework so that we no longer assume zero commission fee during transaction. Second, we construct the interacting environment between the framework and the market as well as the interacting environment inside the framework, thereby enabling the model to adapt to real-world trading process.

The rest of this paper is organized as follows. Section 2 defines the portfolio management problem in this research. All the assumptions made in this study are listed in Section 3. Section 4 gives the network architecture. Data processing and interacting process are given in Section 5. Section 6 presents the training process of the H-DQN model. Experiments and results are given in Section 7. Finally, Section 8 gives the conclusions and research directions for future work.

2 PROBLEM STATEMENT

In portfolio management, we seek the optimal investment policy that gives the maximum overall portfolio over a given period of time. In practice, adjusted weight of portfolio in different assets are based on the price of the assets and the previous weight of the portfolio. This process can be described as Markov Decision Process (MDP) (Neuneier, 1998). Essentially, the MDP is a mathematical model used to develop an optimal strategy, which consists of a tuple (S_t, a_t, P_t, R_t). The meaning of each element in the tuple is given as follows:

- S_t - the state at time t;
- a_t - the action taken at time t;
- P_t - the probability of state transiting from S_t to S_{t+1};
- R_t - the reward at time t.

In order to construct the MDP model for portfolio management, we define the state at time t, S_t, to be the price of investment products, and the action of time t as:

$$a_t \triangleq \boldsymbol{w}_t \tag{1}$$

where $\boldsymbol{w}_t$ is the weight vectors of portfolio at time t. Motivated by action discretized method (Gao et al., 2020), we divide the portfolio value equally into N parts, then consider these parts as the smallest trading units in portfolio management and allocate them to total assets $M+1$ (including cash). Therefore, we may discretize the action space, and then the total number of actions equals $\binom{M+N}{M}$, calculated by permutation, where M is the number of assets. In addition, we define the reward at time t as:

$$R_t \triangleq p_{t+1} - p_t \tag{2}$$

where p_t and p_{t+1} are respectively the portfolio value at time t and $t+1$.

In (2), we only focus on the reward at the current time t, whereas the state of time t caused by the given policy π affects all the states after time t, i.e., the value of S_t is not only current reward R_t, but also the rewards of subsequent time periods. Therefore, with policy π, the value function, G_π, of state S_t should be defined as:

$$G_\pi(S_t) \triangleq \sum_{k=t}^{T} \gamma^{k-t} R_k \tag{3}$$

where T denotes the last trading period and $\gamma \in (0,1]$ is the discount factor. In general, it is very complicated to calculate $G_\pi(S_t)$ using (3), so we need to compute the expectation of G_π to approximate its true value. Furthermore, since the policy π is defined solely by action a_t, we define the value function Q_π of S_t and a_t as:

$$Q_\pi(S_t, a_t) \triangleq E[G_\pi(S_t)] \tag{4}$$

which is the basic principle of Q Learning. Combining the basic principle and neuro network, we can build a Deep Q-Network (DQN) that can solve problems with infinite state space, e.g., portfolio management. However, in the above description, we find that the number of actions increases with the increment of number of parts that we divide the total portfolio into, and the number of assets. Moreover, if we decrease the number of parts N in order to reduce the number of actions, the trading unit will be larger, which may result in large commission fee as a result of frequent

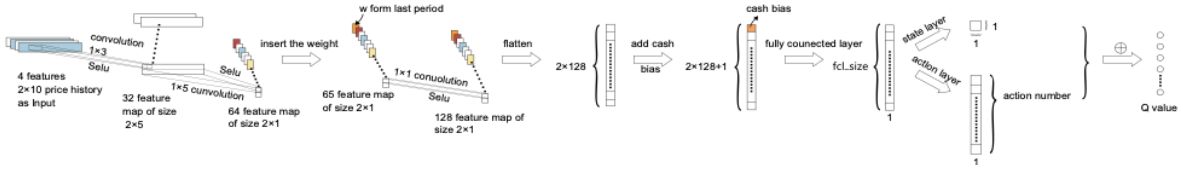

Figure 1: Structure of DQNs.

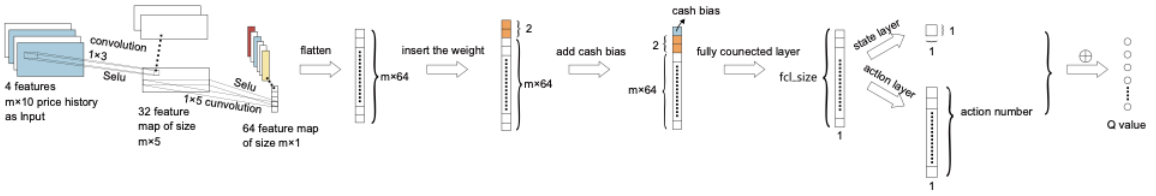

Figure 2: Structure of controllers.

trading. Therefore, we shall reduce the number of assets M handled by each DQN so that trading unit can be small enough to avoid the problem caused by large commission fee.

3 ASSUMPTIONS

Before introducing the model, we shall make the following simplifying assumptions:

1. The action taken by the agent will not affect the financial market such as the price and volume.
2. Other than the commission fee deducted during transition process, the portfolio value remains unchanged between the end of previous trading period and the beginning of next trading period.
3. The agent can not invest money into assets other than the selected ones.
4. The volume of each stock is large enough for the agent to buy or sell each of them at any trading day.
5. The transition process is short enough so that the time for this process may be ignored.

4 NETWORK ARCHITECTURE

In this section, we shall introduce the architecture of H-DQN (Fig.7) that can handle an arbitrary number of assets, e.g., stocks, cash etc. As mentioned in Section 2, to solve the problem associated with commission fee, the total portfolio value should be divided into smaller parts, which leads to large action space. Therefore, we shall decrease the number of assets assigned to the DQN. To achieve this, we consider several independent DQNs in the system. Each DQN has identical structure and is responsible for three assets (one cash and two stocks) so that the number of assets each DQN is smaller than that of traditional DQN which deals with large number of assets. Next, we define a controller which is actually a DQN whose structure is different from the DQNs interacting with market. Considering that controller is also a DQN, i.e., its action space should be limited as well, we assign two DQNs to the controller. Furthermore, for each controller, there is a controller of higher level to control it. Consequently, we obtain the general structure of H-DQN.

From now on, we shall focus on a structural unit of H-DQN (Fig.3) since the network topology and trading process of the general structure can be generalized from its structural unit. In Fig.3, we see that the controller divides the total portfolio into three parts, i.e., cash, the portfolio managed by DQN1 and the portfolio managed by DQN2. As such, the DQNs on the lower level receive the portfolio for investing in the market. The specific trading process will be introduced in section 6.

The network topologies of the DQNs and controllers are inspired by (Jiang et al., 2017). In our network, we change the structure of dense layers in (Jiang et al., 2017) and make it a dueling Q-net. We shall first describe the topology of the DQNs, as shown in Fig.1.

(1) Input and Output: The input of DQNs is the state S_t of last trading period, which is defined by

$$S_t \triangleq (\boldsymbol{P}_t, \boldsymbol{w}_{t-1})$$
$$s.t. \begin{cases} \boldsymbol{w}_{t-1} = (w_{t-1,0}, w_{t-1,1}, w_{t-1,2}) \\ \boldsymbol{P}_t = \left[\boldsymbol{P}_t^o, \boldsymbol{P}_t^c, \boldsymbol{P}_t^h, \boldsymbol{P}_t^l\right] \end{cases} \quad (5)$$

where $w_{t-1}, i = 0,1,2$ denotes the proportion of portfolio value assigned to this DQN that invested in the i^{th} stock at the beginning of previous trading period.

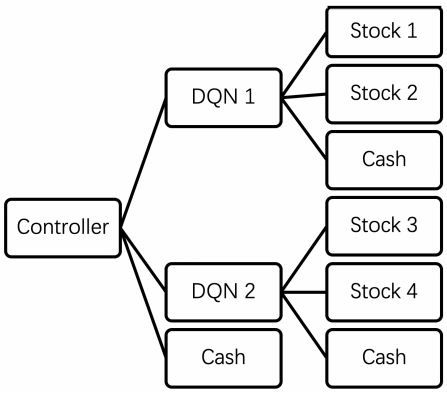

Figure 3: Architecture of structural unit in H-DQN.

Moreover, we initialize $\boldsymbol{w}_0$ (weight vector at the beginning of the episode) as

$$\boldsymbol{w}_0 = (1,0,0) \tag{6}$$

i.e., all the portfolio value is in form of cash. $\boldsymbol{P}_t$ is the price tensor of the previous N trading days (defined in Section 5.1). What should be mentioned is that the number of stocks M equals 2 because each DQN only holds 2 stocks in our hierarchical model shown in Fig. 1. The output of DQN is $\boldsymbol{w}_t$, the weight vector at the beginning of next trading period.

(2) 1st and 2nd CNN Layers: As shown in Fig. 1, the first CNN layer receives the price tensor P_t with dimension *(2, 10, 4)*. The filter of this layer is in size of 1×3, and the activation function we use here is *Selu* which is defined in (Klambauer et al., 2017). In this layer, we obtain 32 feature maps and each one is in size of 2×5, and these feature maps are received by the next CNN layer. In the second CNN layer, the filters are of size 1×5 and 64 feature maps are produced.

(3) Weight Insertion and 3rd CNN Layer: In the layers mentioned above, we only extract features from the price tensor, but the weight vector $\boldsymbol{w}_{t-1}$ has not been used. So after obtaining 64 feature maps from the second CNN layer, we insert the weight vector $\boldsymbol{w}_{t-1}$ (with the weight of cash removed) into these feature maps and produce a tensor with dimension *(2, 1, 65)*. In the third CNN layer, the size of filter is 1×1 and 128 feature maps are produced, following which the 128 features are flattened and a cash bias (Jiang et al., 2017) (a 1×1 tensor with value 1) is added onto the flattened feature map.

(4) Dense Layers: Every neuro in the first dense layer receives the flattened features and connects with each neuro in next dense layer. In the second dense layer, we use the structure called dueling Q-net (Wang et al., 2016), which consists of state layer and action layer. With this structure, the value of state and the

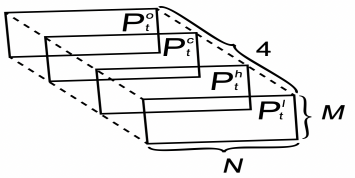

Figure 4: Original price tensor $\mathbf{P}_t^*$.

value of actions may be estimated separa(Wang et al., 2016)tely. After the Q-value of state Q_s and Q-value of each action Q_a are obtained, we can compute the final Q-value of each action by where $E[Q_a]$ is the expectation of Q-values of actions.

$$Q_{(S_t,a)} = Q_s + (Q_a - \mathrm{E}[Q_a]) \tag{7}$$

The structure of controller is similar to DQNs, which is given in Fig.2. The main difference lies in the convolutional layers. Since the price tensor received here is $M \times N \times 4$ but the weight is still a 2×1 (with first element removed) vector, i.e. the first dimension may not be equal, so we cannot insert the weight into the feature maps. Therefore, we flatten the feature maps after the first and second convolution layers and concatenate the weight to it. Finally, we input the flattened feature map into the remaining network which is also a dueling Q-net.

5 MATHEMATICAL FORMALISM

5.1 Data Processing

Considering that the raw price data cannot be received by the network directly, we need to process the data and transform it to a 'tensor' structure.

For the price tensor $\boldsymbol{P}_t$, it is converted from the original price tensor $\boldsymbol{P}_t^*$ consisting of $\boldsymbol{P}_t^o, \boldsymbol{P}_t^c, \boldsymbol{P}_t^h, \boldsymbol{P}_t^l$ which are the normalized price matrices of opening, closing, highest and lowest as denoted below:

$$\boldsymbol{P}_t^o = \left[\boldsymbol{p}_{t-n+1}^o \oslash \boldsymbol{p}_t^c \middle| \ldots \middle| \boldsymbol{p}_t^o \oslash \boldsymbol{p}_t^c\right] \tag{8}$$

$$\boldsymbol{P}_t^c = \left[\boldsymbol{p}_{t-n+1}^c \oslash \boldsymbol{p}_t^c \middle| \ldots \middle| \boldsymbol{p}_t^c \oslash \boldsymbol{p}_t^c\right] \tag{9}$$

$$\boldsymbol{P}_t^h = \left[\boldsymbol{p}_{t-n+1}^h \oslash \boldsymbol{p}_t^c \middle| \ldots \middle| \boldsymbol{p}_t^h \oslash \boldsymbol{p}_t^c\right] \tag{10}$$

$$\boldsymbol{P}_t^l = \left[\boldsymbol{p}_{t-n+1}^l \oslash \boldsymbol{p}_t^c \middle| \ldots \middle| \boldsymbol{p}_t^l \oslash \boldsymbol{p}_t^c\right] \tag{11}$$

where $\oslash$ is elementwise division. In addition, $\boldsymbol{P}_t^o, \boldsymbol{P}_t^c, \boldsymbol{P}_t^h, \boldsymbol{P}_t^l$ represent the price vectors of opening, closing, highest and lowest price for all assets in trade period t respectively. In other words, the i^{th} element

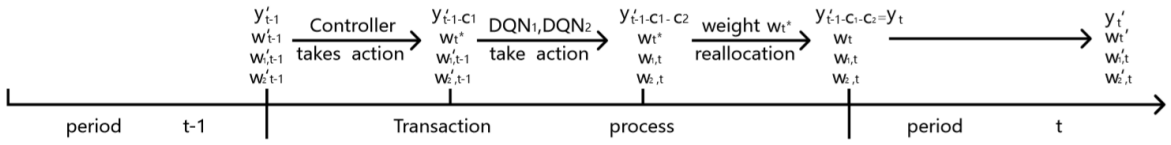

Figure 5: Transaction process.

of them, $p^o_{t,i}, p^c_{t,i}, p^h_{t,i}, p^l_{t,i}$, are relative technical indicators of i^{th} asset in the i^{th} period. Therefore, if there are M assets (except cash) in the portfolio, the original price tensor $\boldsymbol{P}^*_t$ is an $(M,N,4)$-dimensional tensor, as illustrated in Fig. 4

We note that simply normalizing the original tensor price may trigger some recognition problems, i.e., the features of normalized original price tensors are so similar that the network may not distinguish between them and determine which action should be taken. In view of this, each element in $\boldsymbol{P}^*_t$ should be reduced by 1 and multiplied by an expansion coefficient α to enhance the features of the price tensor. Therefore, the final price tensor is defined as:

$$\boldsymbol{P}_t \triangleq \alpha(\boldsymbol{P}^*_t - \mathbf{1}) \tag{12}$$

where $\mathbf{1}$ is a tensor of dimension of $(M,N,4)$, whose elements are all 1's.

Because the controllers and the DQNs (Fig.3) have similar structure, we may process the data received by DQNs using same method discussed in this section. The dimension of price tensor and weight vector of bottom DQNs are of size $(2,N,4)$ and 1×3 respectively, since each bottom DQN can only handle 2 stocks and cash.

5.2 Interaction with Environment

As mentioned in Section 4, the structure of the system is hierarchical which consists of controllers and bottom DQNs. Since the controllers have similar properties, we shall introduce the interacting process of one controller and the DQNs corresponding to it (Fig.3) instead of the interacting process of the whole system. The interaction process is shown in Fig. 5.

The interaction process starts with the controller receiving the state S_t as defined in (5), and giving the weight vector $\boldsymbol{w}^*_t$, where the weight vector represents the proportion of portfolio conserved as cash and assigned to the bottom DQNs. This relocation of portfolio incurs commission fee c_1 defined as:

$$c_1 = d y'_{t-1} \sum_{i=1}^{2} \left| w^*_{t,i} - w'_{t-1,i} \right| \tag{13}$$

where d is the commission rate. The notation y'_{t-1} and $\boldsymbol{w}'_{t-1}$ are respectively the portfolio value and weight at the end of the trading period $t-1$. Moreover, we omit the weight of cash here because the change of cash does not incur commission fee.

Next, the DQNs of next level receive the state $S_{t,1}, S_{t,2}$ respectively and output $\boldsymbol{w}_{1,t}, \boldsymbol{w}_{2,t}$. The commission fee of this relocation of portfolio is given by

$$\begin{aligned} c_2 = d\left(y'_{t-1} - c_1\right)\left(w^*_{t,1} \sum_{i=1}^{2} \left| w_{1,t,i} - w'_{1,t-1,i} \right| + \right. \\ \left. w^*_{t.2} \sum_{i=1}^{2} \left| w_{2,t,i} - w'_{2,t-1,i} \right| \right) \end{aligned} \tag{14}$$

So far, we obtain the portfolio value at the beginning of next trading period as:

$$y_t = y'_{t-1} - c_1 - c_2 \tag{15}$$

The portfolio value held by DQN1 and DQN2 are given respectively by

$$\begin{aligned} y_{1,t} &= w^*_{t,1}\left(y'_{t-1} - c_1\right)\left(1 - d\sum_{i=1}^{2}\left| w_{1,t,i} - w'_{1,t-1,i} \right|\right) \\ y_{2,t} &= w^*_{t,2}\left(y'_{t-1} - c_1\right)\left(1 - d\sum_{i=1}^{2}\left| w_{2,t,i} - w'_{2,t-1,i} \right|\right) \end{aligned} \tag{16}$$

and weight vectors of DQNs at the beginning of next trading period are $\boldsymbol{w}_{1,t}$ and $\boldsymbol{w}_{2,t}$.

However, the weight vector of the controller at the beginning of the next trading period is not simply given by $\boldsymbol{w}^*_t$, since the portfolio value assigned to DQNs is changed after interacting with market, i.e., commission fee is deducted. Therefore, the weight vector of controller at the beginning of next trading period is the proportion of portfolio value after the transition, given by

$$\boldsymbol{w}_t = \left(\frac{w^*_{t,0}\left(y'_{t-1} - c_1\right)}{y_t}, \frac{y_{1,t}}{y_t}, \frac{y_{2,t}}{y_t} \right) \tag{17}$$

Then, during the next trading period, the price relative vector of assets assigned to DQN1 and DQN2 are given by

$$\begin{aligned} \boldsymbol{\mu}^*_{1,t} &= \boldsymbol{p}^o_{1,t+1} \oslash \boldsymbol{p}^o_{1,t} = \left(\frac{p^o_{1,t+1,1}}{p^o_{1,t,1}}, \frac{p^o_{1,t+1,2}}{p^o_{1,t,2}}, \ldots, \frac{p^o_{1,t+1,m}}{p^o_{1,t,m}} \right) \\ \boldsymbol{\mu}^*_{2,t} &= \boldsymbol{p}^o_{2,t+1} \oslash \boldsymbol{p}^o_{2,t} = \left(\frac{p^o_{2,t+1,1}}{p^o_{2,t,1}}, \frac{p^o_{2,t+1,2}}{p^o_{2,t,2}}, \ldots, \frac{p^o_{2,t+1,m}}{p^o_{2,t,m}} \right) \end{aligned} \tag{18}$$

Here, we define the time of opening quotation as the demarcation point of previous and next trading period, i.e., the previous trading period is before the point and next trading period is after the point, and by Section 3 Assumption 5, all the transition activities take place in a very short period of time after the opening quotation. Moreover, since the total portfolio value includes cash, we need to add cash price to $\boldsymbol{\mu}^*_{1,t}$ and $\boldsymbol{\mu}^*_{2,t}$. Considering the fact that the cash price remains unchanged, $\boldsymbol{\mu}_{1,t}$ and $\boldsymbol{\mu}_{2,t}$ would take the following form

$$\begin{aligned}\boldsymbol{\mu}_{1,t} &= \left(1, \frac{p^o_{1,t+1,1}}{p^o_{1,t,1}}, \frac{p^o_{1,t+1,2}}{p^o_{1,t,2}}, \ldots, \frac{p^o_{1,t+1,m}}{p^o_{1,t,m}}\right) \\ \boldsymbol{\mu}_{2,t} &= \left(1, \frac{p^o_{2,t+1,1}}{p^o_{2,t,1}}, \frac{p^o_{2,t+1,2}}{p^o_{2,t,2}}, \ldots, \frac{p^o_{2,t+1,m}}{p^o_{2,t,m}}\right)\end{aligned} \tag{19}$$

and the portfolio value of DQN1 and DQN2 at the end of next trading period are given by

$$\begin{aligned}y'_{1,t} &= y_{1,t}\boldsymbol{w}_{1,t}\boldsymbol{\mu}^{\mathrm{T}}_{1,t} \\ y'_{2,t} &= y_{2,t}\boldsymbol{w}_{2,t}\boldsymbol{\mu}^{\mathrm{T}}_{2,t}\end{aligned} \tag{20}$$

Therefore, the total portfolio value is

$$y'_t = w^*_{t,0}\left(y'_{t-1} - c_1\right) + y'_{1,t} + y'_{2,t} \tag{21}$$

Using the portfolio value at the end of next and previous trading period, we may calculate the reward of controller and reward of DQNs as

$$\begin{aligned}r_t &= \log\left(\frac{y'_t}{y'_{t-1}}\right) \\ r_{1,t} &= \log\left(\frac{y'_{1,t}}{y'_{1,t-1}}\right) \\ r_{2,t} &= \log\left(\frac{y'_{2,t}}{y'_{2,t-1}}\right)\end{aligned} \tag{22}$$

6 TRAINING PROCESS

Since the controllers and DQNs are all Deep Q-networks essentially, their parameters may be updated by general training process of DQN. For this model, we applied the training process that is widely used in Double Deep Q-network (Van Hasselt et al., 2015).

Considering that the basic principle of DQN is to approximate the real Q-function, there should be two Deep Q-Networks, i.e., the evaluation network Q_{eval} and the target network Q_{target}, which have exactly same structure but different parameters. The parameters of Q_{eval} are continuously updated, while the parameters of Q_{target} are fixed until they are replaced by the parameters of Q_{eval}.

Algorithm 1: Training process.

Input: Batch size N, Target network Q_{θ^*}
Input: Estimation network Q_θ, Target vector $\mathbf{v}_{tar}$
Input: Estimation vector $\mathbf{v}_{est}$, Real value vector $\mathbf{v}_{real}$

for $i = 1 \rightarrow N$ **do**
  Take sample $(S_{t_i}, a_{t_i}, r_{t_i}, S_{t_i+1})$
  $\mathbf{v}_{tar}(i) = Q_{\theta^*}(S_{t_i+1}, \mathrm{argmax}_a(Q_\theta(S_{t_i+1}, a)))$
  $\mathbf{v}_{est}(i) = Q_\theta(S_{t_i}, \mathrm{argmax}_a(Q_\theta(S_{t_i}, a)))$
  if S_{t_i+1} is terminal **then**
    $\mathbf{v}_{real}(i) = r_{t_i}$
  else
    $\mathbf{v}_{real}(i) = r_{t_i} + \gamma\mathbf{v}_{tar}(i), \gamma \in (0, 1]$
  end if
end for
Do a gradient decent step with $\|\mathbf{v}_{real} - \mathbf{v}_{est}\|^2$
Replace the parameter of target net $\theta^* \leftarrow \theta$

7 EXPERIMENT

7.1 Experimental Setting

In our experiment, four low-correlation stocks in Chinese A-share market, codes of which are 600260, 600261, 600262, 600266 respectively, are chosen as risk assets (downloaded from tushare). Combined with the cash as risk-free asset, there are a total of five investment assets to be managed. In order to increase the difference between price tensors, we set the trading period as two days. Meanwhile, we set 2011/1/1-2012/12/31, 2014/1/1-2015/12/31, 2015/1/1-2016/12/31 as the period of training set and 2013/1/14-2013/12/19, 2016/1/14-2016/12/19, 2017/1/13-2017/12/18 as the period of back-test intervals. The same hyperparameters were used in the above three experiments.

7.2 Performance Metrics

This section presents three different financial metrics for evaluating the performance of trading strategies. The first metric is the cumulative rate of return (CRR) (Jiang et al., 2017), defined as:

$$\mathrm{CRR} = \exp\left(\sum_{t=1}^{T} r_t\right) - 1 \tag{23}$$

where T is the total number of trading periods and r_t is the reward in t^{th} period as defined in (22). This metric may be used to evaluate the profitability of strategies directly.

The second metric, the Sharpe ratio (SR), is mainly used to assess the risk-adjusted return of strategies (Sharpe, 1994). It is defined as:

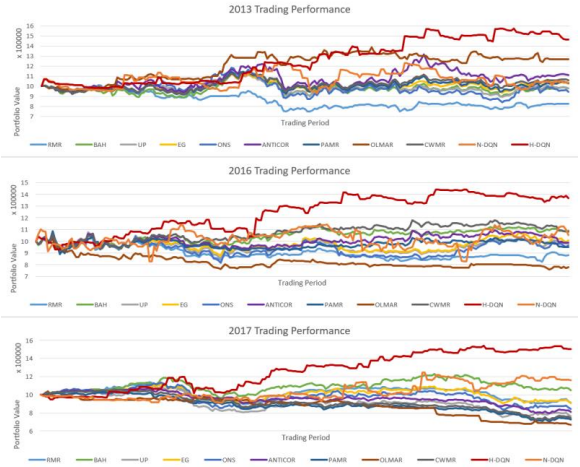

Figure 6: Trading performance in three back-test intervals.

$$\mathrm{SR} = \frac{\mathrm{E}\left[\rho_t - \rho_{RF}\right]}{\sqrt{\mathrm{Var}\left(\rho_t - \rho_{RF}\right)}} \tag{24}$$

where E and Var are the expectation and variance respectively. The notation ρ_t is the rate of return defined as:

$$\rho_t \triangleq \frac{y'_t}{y'_{t-1}} - 1 \tag{25}$$

Here, y'_t, y'_{t-1} are portfolio value at the end of t^{th} period and $t-1^{th}$ period. The parameter ρ_{RF} represents the rate of return of risk-free asset.

To evaluate an investment strategy's risk tolerance, we introduce Maximum Drawdown (Magdon-Ismail and Atiya, 2004) as the third metric. The formula of Maximum Drawdown (MDD) is

$$\mathrm{MDD} = \max_{\beta > t} \frac{y_t - y_\beta}{y_t} \tag{26}$$

This metric denotes the maximum portfolio value loss from a global maximum to global minimum so that we can measure the largest possible loss using it.

7.3 Result and Analysis

The performance of trading strategy is compared with several strategies as listed below (each strategy is tested with commission rate of 0.25%), and these strategies can be categorized into two types:

Type I. Traditional trading strategies

- Robust Median Reversion (RMR) (Huang et al., 2012)
- Uniform Buy and Hold (BAH) (Li and Hoi, 2014)
- Universal Portfolios (UP) (Cover, 2011)
- Exponential Gradient (EG) (Helmbold et al., 1998)
- Online Newton Step (ONS) (Agarwal et al., 2006)
- Aniticor (ANTICOR) (Borodin et al., 2004)
- Passive Aggressive Mean Reversion (PAMR) (Li et al., 2012)
- Online Moving Average Reversion (OLMAR) (Li et al., 2015)
- Confidence Weighted Mean Reversion (CWMR) (Li et al., 2013)

Type II. Reinforcement learning trading strategy

- Single DQN (N-DQN), an approach combining Q-learning with a single deep neural network with commission fee taken into consideration (commission rate of 0.25%) (Gao et al., 2020)

The first dataset gives the cumulative return over the investment horizon of the test period as learning continuous from 2013/01/14 to 2013/12/19. Overall, the H-DQN strategy outperforms all other strategies in contrast to N-DQN strategy which does not perform as well as benchmarks such as OLMAR and ANTICOR. Although the advantage of H-DQN strategy is not apparent at the beginning, and the disparity between H-DQN strategy and other strategies becomes obvious especially in the last 1/3 of the trading period. Compared to N-DQN strategy which shows fluctuations along the time periods, H-DQN strategy tends to increase over the test trading period in general and the cumulative return is almost always above the initial cashflow.

Similarly, in the second dataset test which is from 2016/01/14 to 2016/12/19, the cumulative return shows that the H-DQN strategy has the best performance among all the strategies in the scenario that N-DQN shows no outstanding performance. In addition, the H-DQN strategy outperforms other strategies for the most of the time periods, and the difference between H-DQN strategy and other benchmarks is significant in the latter half of the time periods. Compared to the total portfolio value which remains at a low level as demonstrated by other strategies, the total portfolio value of H-DQN strategy tends to increase over the test trading period in general and the cumulative return is obviously higher than others for most of the time periods.

The third dataset gives the cumulative return over the test periods from 2017/01/05 to 2017/11/17. Again, the H-DQN strategy outperforms benchmark strategies for the majority of the test trading period, and begins to perform very well halfway through the trading period. Compared to N-DQN strategy which shows moderate increase, the H-DQN strategy tend to increase over the test trading period in general and the overall increase is much higher than N-DQN.

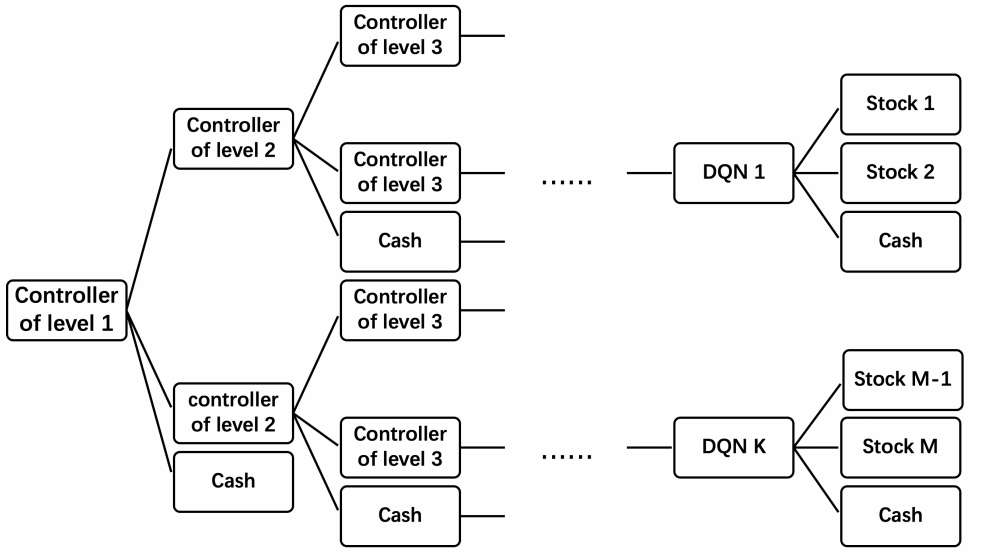

Figure 7: General architecture of H-DQN.

Table 1: Average Performance of Eleven Strategies.

	CRR	SR	MDD
RMR	-12.84%	-4.00%	22.98%
BAH	3.97%	1.97%	17.78%
UP	-9.12%	-2.35%	24.38%
EG	-1.93%	-0.22%	17.43%
ONS	-8.16%	-1.48%	26.16%
ANTICOR	-2.98%	-2.17%	19.89%
PAMR	-9.83%	-4.11%	23.04%
OLMAR	-9.55%	-5.26%	24.34%
CWMR	-3.84%	-2.32%	20.36%
N-DQN	8.19%	2.72%	22.72%
H-DQN	**44.37%**	**10.33%**	**11.69%**

Table 1 gives the average performance of hierarchical DQN strategy and benchmarks over three test sets based on the metrics CRR, SR and MDD. It is obvious that the numerical results of H-DQN strategy are the best in all aspects. In the case of CRR, the result of H-DQN strategy (44.37%) exceeds the second highest benchmark N-DQN (8.19%) by 36%. As for the risk measure, the H-DQN strategy still gives the best performance with minimum MDD (11.69%), as compared to the EG benchmark (17.43%). For SR, H-DQN strategy (10.33%) outperforms the next best performing strategy N-DQN benchmark (2.72%). It should be noted that N-DQN is also a reinforcement learning algorithm, but without the hierarchical architecture, its performance is much worse than H-DQN.

Overall, the results in all three back-test intervals demonstrate the good profitability and adaptability of H-DQN framework in comparison to N-DQN and all other traditional strategies.

8 CONCLUSIONS AND FUTURE WORK

In this paper, we construct a hierarchical reinforcement learning framework that can handle an arbitrary number of assets for portfolio management which takes commission fee into consideration. Four stocks are selected as our experimental data, and Cumulative Rate of Return, Sharpe ratio, Maximum Drawdown are used to compare the profitability and risk of our model in the back-test intervals against nine traditional strategies as well as the single DQN strategy. The results show that this hierarchical reinforcement learning algorithm outperforms all the other ten strategies, and it is also the least risky investment method our back-test intervals.

However, there are three major limitations. First, since the controller on higher level needs to manage more controllers, it is more difficult to train. Second, we assume that the volumes of stocks are large enough so each stock is available on any trading day. However, the stock might not be available sometimes, which will therefore impact the profit. Finally, for generalization, our strategies will be vulnerable due to a small mismatch between the learning environment and the testing environment.

For future work, considering that deep reinforcement learning algorithm is highly sensitive to the noise in data, we may use traditional approaches to reduce financial data noise, e.g., wavelet analysis (Rua and Nunes, 2009) and the Kalman Filter (Faragher, 2012). Moreover, as the number of assets increases, there will be more controllers and DQNs in the hierar-

chical structure (shown in Fig.7), which may require a long training period. To address this, we will look into proposing new training methods that may improve the efficiency in training the network.

REFERENCES

Agarwal, A., Hazan, E., Kale, S., and Schapire, R. E. (2006). Algorithms for portfolio management based on the newton method. In *Proceedings of the 23rd international conference on Machine learning*, pages 9–16.

Borodin, A., El-Yaniv, R., and Gogan, V. (2004). Can we learn to beat the best stock. In *Advances in Neural Information Processing Systems*, pages 345–352.

Cover, T. M. (2011). Universal portfolios. In *The Kelly Capital Growth Investment Criterion: Theory and Practice*, pages 181–209. World Scientific.

Dempster, M. A. and Leemans, V. (2006). An automated fx trading system using adaptive reinforcement learning. *Expert Systems with Applications*, 30(3):543–552.

Faragher, R. (2012). Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]. *IEEE Signal processing magazine*, 29(5):128–132.

Gao, Z., Gao, Y., Hu, Y., Jiang, Z., and Su, J. (2020). Application of deep q-network in portfolio management. In *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*, pages 268–275. IEEE.

Heaton, J., Polson, N. G., and Witte, J. H. (2016). Deep learning in finance. *arXiv preprint arXiv:1602.06561*.

Helmbold, D. P., Schapire, R. E., Singer, Y., and Warmuth, M. K. (1998). On-line portfolio selection using multiplicative updates. *Mathematical Finance*, 8(4):325–347.

Huang, D., Zhou, J., Li, B., HOI, S., and Zhou, S. (2012). Robust median reversion strategy for on-line portfolio selection.(2013). In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence: IJCAI 2013: Beijing, 3-9 August 2013*.

Jiang, Z., Xu, D., and Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683.

Li, B. and Hoi, S. C. (2014). Online portfolio selection: A survey. *ACM Computing Surveys (CSUR)*, 46(3):1–36.

Li, B., Hoi, S. C., Sahoo, D., and Liu, Z.-Y. (2015). Moving average reversion strategy for on-line portfolio selection. *Artificial Intelligence*, 222:104–123.

Li, B., Hoi, S. C., Zhao, P., and Gopalkrishnan, V. (2013). Confidence weighted mean reversion strategy for online portfolio selection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(1):1–38.

Li, B., Zhao, P., Hoi, S. C., and Gopalkrishnan, V. (2012). Pamr: Passive aggressive mean reversion strategy for portfolio selection. *Machine learning*, 87(2):221–258.

Magdon-Ismail, M. and Atiya, A. F. (2004). Maximum drawdown. *Risk Magazine*, 17(10):99–102.

Neuneier, R. (1998). Enhancing q-learning for optimal asset allocation. In *Advances in neural information processing systems*, pages 936–942.

Park, S., Song, H., and Lee, S. (2019). Linear programing models for portfolio optimization using a benchmark. *The European Journal of Finance*, 25(5):435–457.

Rua, A. and Nunes, L. C. (2009). International comovement of stock market returns: A wavelet analysis. *Journal of Empirical Finance*, 16(4):632–639.

Sharpe, W. F. (1994). The sharpe ratio. *Journal of portfolio management*, 21(1):49–58.

Van Hasselt, H., Guez, A., and Silver, D. (2015). Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461*.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003.