A Deep Reinforcement learning based Quantum Finance System

Abstract—**With the rapid development of computer science technology, quantitative trading has been more and more popular among traders and agency. Since the trading in financial markets is similar to the mechanism of reinforcement learning, reinforcement learning is used for automated trading in this project. Q-Learning is an effective algorithm in reinforcement learning but the memory it needs is unacceptable when the state is continuous. Therefore, we proposed a Deep Q-Learning model which accepts transaction data, financial indicators including Quantum Price Level, and account data as input and output the Q-value of each transaction behavior. To train and test the deep reinforcement learning model, 5 different financial products with 1024 trading days data are chosen: AUD/USD (Forex), Google (Stock), USD100M1(Future), Airbus (CFD), and XAU/USD (Spot Gold). During the process of training, the best agent of each product is recorded. The back test demonstrates that the model is able to have positive return rate in each product. In general, it has better performance in stock, futures, and CFD. The reason that it does not work well on forex and spot gold is the fluctuation are relatively small and the direction changes relatively sudden. The design of reward function and network structure are possible improvement direction.**

Index Terms — **Automated Trading, Deep Learning, Financial Market, Quantum Price Level, Reinforcement Learning**

I. Introduction

With the development of computer science technology, more and more traders and agency use program to make automated trading decision. The advantage is the program will strictly enforce transaction logic and avoid unstable impact of personal emotions on investment. The profit and risk can be controlled in a limited range. For different financial markets, there are numerous trading strategies. Most of the strategies are logical transaction, which means the computer monitors some indicators continuous or periodically. Once the conditions required to reach the starting point, the program will execute some transaction behaviors.

In recent years, machine learning is playing a more and more important role in all walks of life. Reinforcement learning is an important aspect of machine learning. Unlike supervised learning, the input data for reinforcement learning has no 'label', which means the reinforcement learning algorithm cannot know whether its behaviors are correct or wrong. There is only a delayed reward value given by the environment after the agent chose an action and execute it. Therefore, in the reinforcement learning algorithm, the agent must execute actions and interact with the environment enough time to learn the optimal actions in each different state to maximize the reward given by the environment each time. The working mechanism of reinforcement learning makes it quite suitable for financial product automated trading. Because similar to reinforcement learning, there is no correct nor wrong of each transaction behaviors until few days or few months later you know the price trend.

Q-learning is an effective algorithm in reinforcement since it is able to measure the value of each action in different environment. However, when the state of the environment becomes continuous, it is hard for computer to store a large Q-table in limited memory. Therefore, we use deep neural network to approximate the original value function. Transaction data such as open price and close price, financial indicators such as Moving Average and RSI, and account data such as balance and fund are used to build the environment that the agent in. It is worth mentioning that Quantum Price Level (QPL) is also introduced as a feature of environment. QPL is inspired by the energy levels in atoms. The theory assumes that there are multiple supports and resistances in the financial market similar to the energy levels in the atom.

We select five different financial products: forex, share, indexes future, CFD, and spot gold with corresponding 1024 trading days data to test the ability of our model.

II. Literature Review

The idea of reinforcement learning is to learn in practice. The biggest difference between it and supervised learning is that it does not have the output value of the training data that has been prepared by supervised learning, which means that it only has the delay given reward value. At the same time, each step of reinforcement learning is closely related to the time sequence.

A. Modelling of reinforcement learning

The modeling of reinforcement learning is as follows:

We need to operate an algorithm individual to make decision, which means select a suitable action A_t . There is an environment we are going to study which has its own state model. When we select the action A_t, the environment state will change into S_{t+1}, while the reward of action $A_t(R_{t+1})$ can be acquired.

According to this idea, the elements of reinforcement learning can be concluded:

First is the state of the environment S, at time t the state S_t of the environment is a state in its environment state set.

Second is the action of individual A , at time t an individual's action A_t is an action in its action set.

Third is the reward of environment R, at time t the reward R_{t+1} for the action at taken by the individual in state S_t will be obtained at time $t+1$.

Fourth is the policy of individual π, it represents that individual will follow the policy π to select action.

Fifth is value after taking action by individual in strategy π and state S, generally it can be represented by

$$v_\pi(s) = \mathbb{E}_\pi(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s)$$

Sixth is reward attenuation factor, which is between [0,1]. If it is 0, then it is greedy method, that is, the value is only determined by the current delay reward. If it is 1, then all subsequent status rewards and current rewards are treated equally.

Seventh is the state transition model of the environment, which can be understood as a probabilistic state machine. It can be expressed as a probabilistic model, that is, the probability of taking action A in state S and going to the next state S' is expressed as $P_{ss'}^a$.

The eighth is the exploration rate $\in$. This ratio is mainly used in the iterative process of reinforcement learning training, because we usually choose the actions that make the current round of iteration the most valuable, but this will cause some better actions that we have not performed to be missed.

B. Markov decision process (MDP)

These elements alone cannot be helpful to solve reinforcement learning problems. Markov decision process (MDP) is a key step in reinforcement learning that can directly solve simple problems with equations.

The seventh element is the state transition model of the environment. However, according to the real process of environmental transformation, the probability of transformation to the next state S' is related to every state before prediction. This leads to a very complex environment transformation model, so it needed to be simplified. The method is to assume the Markov property of state transformation, that is, the probability of the transformation to the next state S' is only related to the previous state S, and has nothing to do with the previous state. The formula is as follows:

$$P_{ss'}^a = \mathbb{E}(S_{t+1} = s' \mid S_t = s, A_t = a) \tag{1}$$

In addition to making Markov hypothesis for the state transition model of environment, an analysis of the fourth element of reinforcement learning is also made: individual policy π also made the Markov hypothesis. The probability of taking action a in state s is only related to the current state s, and has nothing to do with other elements:

$$\pi(a \mid s) = P(A_t = a \mid S_t = s) \tag{2}$$

For the fifth element, the value function $v_\pi(s)$ is same, since it only depends on the current state, now the value function $v_\pi(s)$ can be expressed as:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi(G_t \mid S_t = s) \\ &= \mathbb{E}_\pi(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s) \end{aligned} \tag{3}$$

G_t stands for return, which is the sum of all rewards in an MDP that are attenuated from the beginning of sampling in a state S_t to the end of the state.

If we need to consider the value impact of action a, except $v_\pi(s)$ there is also an action value function $q_\pi(s,a)$:

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi(G_t \mid S_t = s, A_t = a) \\ &= \mathbb{E}_\pi(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s, A_t = a) \end{aligned} \tag{4}$$

According to the expression of value function, we can deduce the recurrence relation of value function based on state, such as for state value function $v_\pi(s)$, we can find:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots . \mid S_t = s) \\ &= \mathbb{E}_\pi(R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots) \mid S_t = s) \\ &= \mathbb{E}_\pi(R_{t+1} + \gamma G_{t+1} \mid S_t = s) \\ &= \mathbb{E}_\pi(R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s) \end{aligned} \tag{5}$$

It means the state S_t at time t and the state S_{t+1} at time $t+1$ satisfy the recurrence relation. This is called Bellman Equation, and it represents the value of a state is composed of the reward of the state and the value of the subsequent states according to a certain attenuation ratio.

Similarly, we can get the Bellman Equation of action value function $q_\pi(s,a)$:

$$q_\pi(s,a) = \mathbb{E}_\pi(R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a) \tag{6}$$

To solve the problem of reinforcement learning means to find an optimal strategy to make the individual gain more than other strategies in the process of interaction with the environment $\pi*$ expressed. It can be defined that the optimal state value function is the largest of the many state value functions generated under all strategies:

$$v_*(s) = \max_\pi v_\pi(s) \tag{7}$$

Similarly, it can be defined that the optimal action value function is the largest of all the action state value functions generated under all strategies:

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{8}$$

For the optimal strategy, based on the action value function, it can be defined as:

$$\pi_*(a \mid s) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in A} q_*(s,a) \\ 0 & \text{else} \end{cases} \tag{9}$$

By using the relationship between the state value function and the action value function, we can also get the following results:

$$v_*(s) = \max_a q_*(s,a) \tag{10}$$

$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a r_*(s') \tag{11}$$

$$v_*(s) = \max_a \left(R_s^a + \gamma \sum_{s' \in S} P_{ss}^a v_*(s') \right) \tag{12}$$

$$q_*(s,a) = R_s^a + \gamma \sum_{d' \in S} P_{ss'}^a \max_{a'} q_*(s', a') \tag{13}$$

C. Dynamic programming solution

Dynamic programming mainly uses Bellman equation to update state value iteratively and uses greedy method to update optimal strategy iteratively.

Dynamic programming algorithm uses full width backtracking mechanism to update state value. That is, whether synchronous or asynchronous dynamic programming, every time a state value is updated, it should be traced back to all possible subsequent states of the state, and the value of the state is updated by using the Bellman equation. This full width value update method is still effective for the reinforcement learning

problem with less states. However, when the problem is large, the dynamic programming algorithm will not be used due to the Bellman dimension disaster.

D. Monte Carlo method (MC)

Monte Carlo method can avoid too complex solution of dynamic programming and can also be used for massive data and complex models without knowing the environment transformation model in advance, it estimates the real value of a state by sampling several episodes.

For the Monte Carlo method, if the state value of a state is required, it is only necessary to calculate the harvest of the state in all complete sequences, and then take the average value to approximate the solution:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \gamma^{T-t-1} R_T \\ v_\pi(s) &\approx \text{ average } (G_t), s.t. S_t = s \end{aligned} \tag{14}$$

However, it needs a complete state sequence every time it samples. If there is no complete state sequence, or it is difficult to get more complete state sequences, Monte Carlo method is not easy to use.

E. Time series difference method (TD)

The time series difference method is similar to Monte Carlo method, and it is a model-based reinforcement learning problem solving method, but for the time series difference method, there is no complete state sequence, only partial state sequence. This can be done with $R_{t+1} + \gamma v(S_{t+1})$ to approximately replace the harvest G_t, generally $R_{t+1} + \gamma v(S_{t+1})$ is called TD target value. $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called TD error, and the process of using TD target value instead of harvesting G_t is called bootstrapping. In this way, only two continuous states and corresponding rewards are needed to try to solve the reinforcement learning problem.

F. Time series differential online control algorithm SARSA

SARSA algorithm is a method to solve reinforcement learning control problem by using time series difference. For its control problem solving, similar to Monte Carlo method, it is value iteration, that is, updating the current strategy by updating the value function, and then generating new states and immediate rewards through the new strategy, so as to update the value function. It goes on until the value function and strategy converge.

SARSA algorithm belongs to the category of online control, it always uses a strategy to update the value function and select new actions ϵ greedy law, it can be expressed as follows:

$$\pi_*(a \mid s) = \begin{cases} \dfrac{\epsilon}{m} + 1 - \epsilon & \text{if } a = \arg\max\limits_{a \in A} \mathrm{Q}(s, a) \\ \dfrac{\epsilon}{m} & \text{else} \end{cases} \tag{15}$$

When it iterates, we start with ϵ greedy method selects an action A in the current state S, so that the system will go to a new state S', and gives a real-time reward R. in the new state S', based on the ϵ greedy method selects an action A' in the state S', but notice that it doesn't execute this action A' at this time, just to update the value function. The update formula of value function is as follows:

$$Q(S, A) = Q(S, A) + \alpha\big(R + \gamma Q(S', A') - Q(S, A)\big) \tag{16}$$

SARSA algorithm also has a common problem of traditional reinforcement learning methods, which is that it cannot solve too complex problems. The value of $\mathrm{Q}(s, a)$ is stored by a large table. If the state and action reach millions or even tens of millions of levels, the large table that needs to be saved in memory will be super large, even overflowing.

G. Sequential differential offline control algorithm Q-Learning

Q-learning is a kind of reinforcement learning problem solving method, which does not need the state transition model of the environment and is not based on the model. For solving its control problem, the strategy is updated by updating the value function, and the new state and immediate reward are generated by the strategy, so as to update the value function. It goes on until the value function and strategy converge.

For Q-learning, it is based on state S' and is not used ϵ greedy method chooses A', but uses the greedy method to choose A',, that is to say, chooses a which makes $Q(S', a)$maximum as a' to update the value function. It can be expressed by mathematical formula as:

$$Q(S, A) = Q(S, A) + \alpha\left(R + \gamma \max_a Q(S', \mathrm{a}) - Q(S, A)\right) \tag{17}$$

In the era of big data, the Q table to be maintained by Q-learning is abnormally large, even far beyond the memory, due to the extremely complex states and optional actions, which limits the application scenarios of sequential differential algorithm.

III. Methodology

Dynamic Programming, Monte Carlo method and Timing Difference method are all using Discrete finite set of states $\mathbb{S}$. When the scale of the problem is relatively small, the computer can easily solve the solutions. However, when the set of states becomes much more complex, for example, the state is continuous instead of discreate. The set of states will be large enough even though the continuous state is discretized. It is difficult to maintain a huge Q-table in the RAM.

In the financial market, numerous features such as Moving Averaging, RSI and close price are continuous. If we use the typical Q-learning method to simulate the environment where the agent in, there will be almost infinite different environment in the situation. The model can hardly learn useful behavior information and earn profit in the market.

Thus, the deep Q-learning model is used in the project. Since the scale of the set of states is large, a possible solution is to use a function to approximately represent the original value function. A state value function $\hat{v}$ which is described by parameter $\widehat{w}$ and accept state s is introduced. Then, the value of state s can be calculated:

$$\hat{v}(s, w) \approx v_\pi(s) \tag{18}$$

Similarly, an action value function $\hat{q}$ which is also described by parameter w and takes inputs of state s and action a is introduced to calculate the real value of the current action:

$$\hat{q}(s, a, w) \approx q_\pi(s, a) \tag{19}$$

There are multiple models to represent the approximated state value function such as decision tree and Fourier transformation. In our project, we proposed the artificial neural network as our model to represent the approximated state value function since its unique non-linear function can effectively fit the target function in limited iterations.

Table I demonstrates the pseudo-code of Deep Q-Learning algorithm.

Table I
THE PSEUDO-CODE OF DEEP Q-LEARNING

Algorithm 1 Deep Q-Learning

Input: Numbers of Iterations T, Dimensions of Features n, Set of Actions A, Step α, Attenuation Factor γ, Exploration Rate ϵ, Network Q, Batch Size m

Output: Parameters of Q

Initialize all parameters w of Q;
Initialize Q-value of all states and actions based on w;
Empty The collection of experience D;

for $i =$ i to T **do**

a)Initialize S as the first state of current sequence, compute its feature vector $\phi(S)$;

b)Uses $\phi(S)$ as the input of Q and get the $Q-alue$ of all actions. Choose action based on current $Qvalue$ with the probability of $1-\epsilon$ or choose a random action with the probability of ϵ;

c)Execute Action A in State S. Get new state S' and compute its feature vector $\phi(S')$,Reward R, and Termination State is_end;

d)Append $\phi(S), A, R, \phi(S'), is\_end$ into D;

e)Set $S = S'$

f)Samples m data $\phi(S_j), A_j, R_j, \phi(S'_j), is\_end_j,\ j = 1, 2, ..., m$.computes current target $Qvalue\ y_j$ base on:

$$y_j = \begin{cases} R_j & is\_end_j istrue \\ R_j + \gamma \max_{a'} Q\left(\phi\left(S'_j\right), A'_j, w\right) & is\_end_j isfalse \end{cases}$$

g) Used MSE loss function $\frac{1}{m}\sum_{j=1}^{m}\left(y_j - Q\left(\phi\left(S_j\right), A_j, w\right)\right)^2$ to calculate the loss value. Uses Backpropagation algorihtm to update w of Q;

h) Continue to next iteration if S' is termination state. Else go to step b);

end for

A. Dataset

To build the deep reinforcement learning model and test its effectiveness, As Table II demonstrates, we choose transaction data of 5 different kinds of financial products in the past 1024 trading days (UTC+08:00 2021/05/14 is the last day). The selected features are daily open price, daily close price, daily highest price, daily lowest price, and daily trading volume.

Table II
THE COMPOSITION OF DATASET

Product Type	Product Name
Forex	AUD/USD
US Share	Google
Index Futures	US100_M1
CFD	Airbus
Spot Gold	XAU/USD

B. Data Preprocessing and Feature Engineering

The scale of features can significantly affect the convergence speed and accuracy of deep neural networks. Performing normalization of data before taking them as the raw input of the network is a common way to speed up the training of network. In the dataset, the average scale of trading volume ($10^5 \sim 10^7$) is about 1,000 times larger than other features. Thus, a Min-Max standardization is performed:

$$x^* = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{20}$$

The Min-Max standardization linearly reshapes the scale of volume to [0, 1].

In the financial market, plenty of artificial financial indicators can help traders predict the development trend of the market. Moving average indicator, oscillatory indicator and orbital indicator are all common indicators people use. In order to make the agent understand the current state better, Moving Average (MA), Relative Strength Index (RSI), and Quantum Price Level (QPL) indicators are introduced. The formular of Moving Average is defined as:

$$\hat{y}_{t+1} = M_t = \frac{y_t + y_{t-1} + \cdots + y_{t-N+1}}{N} = \frac{1}{N}\sum_{i=0}^{N-1} y_{t-i} \tag{21}$$

Where y_t is the close price in day t, N is average number of items used. In the project, $N = 5$ is used as the short-term moving average indicator and $N = 21$ is used as the long-term moving average indicator.

RSI is a momentum indicator used in technical analysis that measures the magnitude of recent price changes to evaluate overbought or oversold conditions in the price of a stock or other asset. It is defined as the formula. A large RSI may indicate that the market is overheated, and the price will probably drop in the future. A small RSI may indicate that the market is not active, and the price will probably increase in the future. We took n = 14 in the project.

$$\begin{cases} AUn = \dfrac{\sum the\ rising\ points\ in\ n\ days}{n} \\ ADn = \dfrac{\sum the\ droping\ points\ in\ n\ days}{n} \\ RSI = 100 \times \dfrac{AUn}{AUn + ADn} \end{cases} \tag{22}$$

QPL, raised by Dr. Raymond Lee, is a hybrid concept, originating from physics and economic research area.
Inspired by energy levels in atoms, QPLs can be considered as invisible energy levels that exist as supports and resistances of product price. Though Fibonacci analysis plays similar function as QPLs, compare with its simple logic, QPLs seem to be more credible with its scientific basis and calculation process.
First, since we see the distribution of product prices as a wave-function and apply wave-particle dualism on it, Schrodinger Function should be recalled:

$$\hat{H}\psi(r) = E\psi(r) \tag{23}$$

In quantum finance, $\hat{H}$ is the Hamiltonian operator, $\psi(r)$ is the wave-function of products price return and E is the quantum finance energy level. According to the definition of Hamiltonian operator:

$$\hat{H} = \frac{-\hbar}{2m}\frac{\partial^2}{\partial r^2} + V(r) \tag{24}$$

By the modeling of the key market players, such as market maker, investors, speculators, hedgers and arbitrageurs, the potential energy part ($V(r)$) can be replaced:

$$V(r) = \frac{\gamma\eta\delta}{2}r^2 - \frac{\gamma\eta\upsilon}{4}r^4 \tag{25}$$

And the final Quantum Finance Schrodinger Function should be:

$$\left[\frac{-\hbar}{2m}\frac{\partial^2}{\partial r^2}+\left(\frac{\gamma\eta\delta}{2}r^2-\frac{\gamma\eta\upsilon}{4}r^4\right)\right]\psi(r)=E\psi(r) \quad (26)$$

With knowing the Hamiltonian operator and the wave-function of product price return, the Quantum Finance Energy Levels E can be figure out and final QPLs can be calculated. Coordinating with RSI, more reference on price fluctuation can help trader making better decisions. We took QPL_o, QPL_{-1}, and QPL_{+1} as the features of the trading environment.

C. Setting of Agent and Environment

In the project, an automated trading agent is simulated. The agent has three possible trading behaviors (actions) in the market: buy, sell, and keep. Buy means that the agent buys in the financial products in current price with certain amount if the current fund is larger than or equal to a stipulated amount. If the fund is less than this fixed amount, the agent does nothing even though the model tells to buy in. Sell means that the agent sells all the positions it has to the market in current price. If the position is zero, the agent does nothing. Keep means that the agent does nothing in that trading day.

To simplify the situation, we set the initial capital of the agent is 10,000,000 USD. If the agent should buy in the financial product, the transaction amount each time must be 7,000,000 USD. The handling fee rate is 0.15% of the transaction amount. There is no take profit nor stop loss strategy during the trading days. Each trading day the agent will execute the transaction instruction given by the model with the probability of $1-\epsilon$ an execute a random action with the probability of ϵ.

The environment is a 1*14 vector. The first five numbers are the daily transaction data of the product. The next six numbers are the financial indicators: QPL, QPL-, QPL+, MA5, MA21, and RSI. The last three numbers are the current account profit, account fund, and account's position.

Table III demonstrates the setting of reward function which is logical. When the price change tomorrow is zero, no matter what behavior the agent toke or the position the daily profit is always zero. Thus, reward should be zero either. If the action the agent toke is to buy in, the reward should be positively related to the change of price tomorrow since earning profit is desirable. Due to the same reason, reward should be negatively related to the price change if the action is to sell. Because we do not hope the agent sell too early if the price keeps increasing or sell too late if the price is going to decrease. And if the action is to keep, case-by-case discussion is needed. When the position is empty, the reward should be negatively related to the price change since we hope the agent can buy in before the price rises. When the position is not empty, the reward should be positively related to the change of price due to same reason. The numerical value of the reward is the natural base logarithm of the absolute value of product of trade amount and price change tomorrow.

Table III
THE PSEUDO-CODE OF REWARD FUNCTION

Algorithm 1 Reward Function

Def Get_Reward(Action A, Order Amount O,Position P, Close Price C) {
PC = Tommorow's Price Change;
if $PC = 0$ **then**
  $reward = 0$;
else if $A = Buy$ **then**
  $reward = signof(PC) * \ln(|O * PC|)$;
else if $A = Sell$ **then**
  $reward = -5 * signof(PC) * \ln(|O * PC|)$;
else
  if $P = 0$ **then**
    $reward = -1 * signof(PC) * \ln(|O * PC|)$;
  else
    $reward = signof(PC) * \ln(|P * C * PC|)$;
  end if
end if
Return $reward$;
}

D. Neural Network

To approximate the value function, we proposed a typical neural network to fit it. Fig.1. demonstrates the structure of deep neural network to approximate the value function. It accepts the vector of environment as the input. A dense layer with 256 nodes with Relu6 activation function follows the input layer. Before the next dense layer, a batch normalization layer is added to speed up the training speed and stabilize the network. Between the output layer and the last dense layer, dropout layer with 0.25 probability is used to regularize the network. The output layer will output three floating numbers corresponds to the value of each action in current state. The agent will choose the action with the highest value.

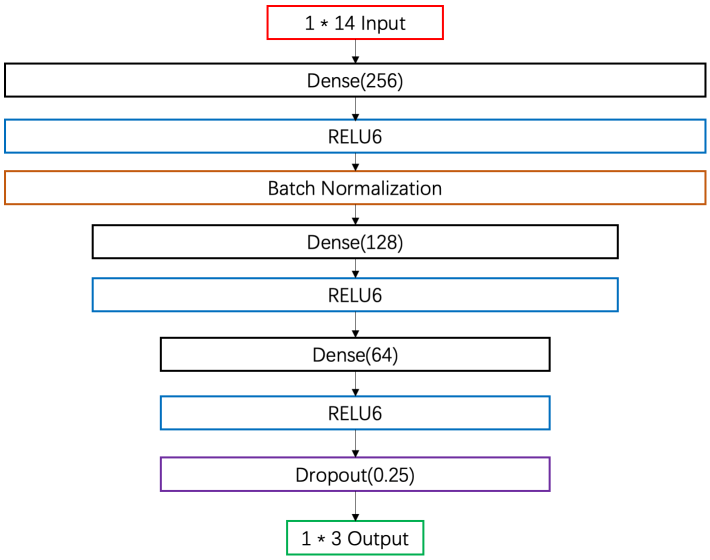

Fig. 1. Structure of deep neural network to approximate the value function

E. Training

Mean Square Error (MSE) loss is used to measure the distance between the evaluated Q-value and real Q-value. It is defined as:

$$MSE = \frac{SSE}{n} = \frac{1}{n}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \quad (27)$$

In the process of Deep Q-learning, arguments learning rate, attenuation factor, and batch size are all pre-defined parameters. Different combination of these parameters may lead to different result. To make the overall profit larger, a grid search of parameters is conducted. The goal is to find out the combination of (learning rate, attenuation factor, batch size) with the largest average total profit in the last 10 epochs. The candidates of learning rate are 0.01, 0.001, and 0.0001. The candidates of attenuation factor are 0.1, 0.3, 0.5, 0.7, and 0.9. The candidates of batch size are 64, 128, and 256. The epochs are 256 for each product. The optimizer used is Adam. Table IV is the final combination of parameters after grid searching.

Table IV
FINAL COMBINATION OF PARAMETERS

Product	AUD/USD	Airbus	Google	USD100 M1	XAU/USD
α	0.001	0.001	0.001	0.001	0.001
γ	0.3	0.5	0.7	0.9	0.5
M	128	64	128	128	64

IV. Result Analysis

Fig. 2. Shows the loss curve of each financial product during 256 epochs. In general, the change of the loss of each product are unstable. The magnitude of the change of near epochs is larger than common supervised learning. The tendency of the loss of product Google, USD100_M1, and XAUUSD is decreasing concussively and become relatively stable in last 100 epochs. The loss of AUD/USD increases steadily in the first epochs and reached the peak of loss then drops significantly in epochs 50 to epochs 100. It also becomes stable in the last 100epochs. The change of loss of Airbus is opposite. It first rises to the peak in the first 10 epochs then drops down to the valley in epoch 50 to epoch 150. However, the loss suddenly rises significantly from epoch 150. The average loss in the last 10 epochs is approximate to the average loss in the first 10 epochs. It means that the Q-Learning mechanism search into the new space that never been observed.

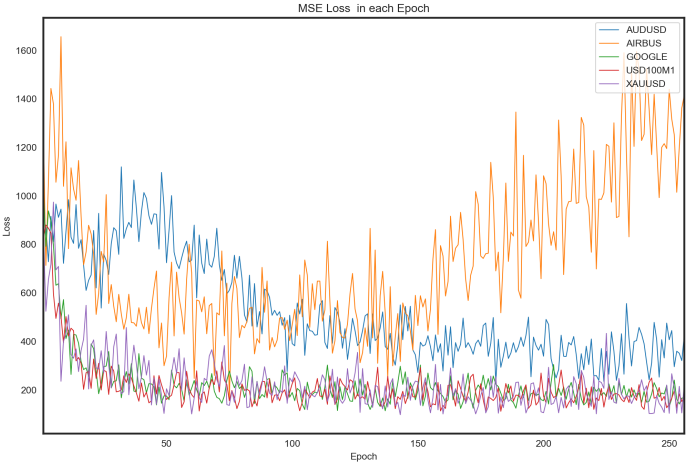

Fig. 2. Loss Curve of each Financial Product

Fig.3. illustrates the accumulative return rate of each financial product. The line in brown labeled 'Principle' represents the initial principle. For each product, the agent with minimum loss is recorded in the process of training and a simulated transaction was done by this agent. The curve records the change of assets during 1024 trading days of each agent. In the long-term perspective, the agents trading with USD100_M1 and Google have better performance. The accumulative return both grow steadily except there are sharp retracement in some trading days. The assets are both doubled in the last trading days. The accumulative return rate of Airbus has normal performance. It increased in the first 200 trading days but keep constant in later 200 trading days. Then it falls back by a small margin then increased steadily in the last 500 trading days. The accumulative return rate of XAU/USD fluctuated around the principal line in the first 500 trading days. Then it continuously rises from 500th trading day and keeps constant in the last 150 trading days. Similar to XAU/USD, the accumulative return rate of AUD/USD fluctuated around the principal line with a relatively small aptitude in the first 750 trading days. Then it keeps constant in 750th trading days to 900th trading days. And it rises since the 900th trading days.

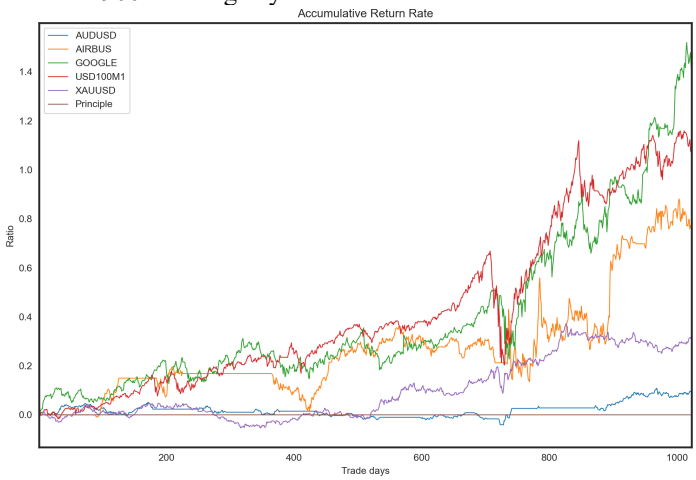

Fig. 3. Accumulative Return Rate of each Financial Product

Table V demonstrates the revenue data of each financial product under the trading agent. In general, the agent trading with Google has the best performance since it has the highest final accumulative return rate, the highest maximum accumulative return rate, and the lowest minimum accumulative return rate. Follows are the agent trading with USD100_M1, Airbus, and XAU/USD due to similar reason. Trading with AUD/USD has the worst performance since it has the lowest accumulative return rate, lower maximum accumulative return rate and the highest minimum accumulative return rate.

Table V
REVENUE DATA OF EACH PRODUCT

Product	Final ACCUM* Return Rate	Max. ACCUM* Return Rate	Min. ACCUM* Return Rate
AUD/USD	8.49%	10.88%	-4.14%
Airbus	81.35%	88.12%	-0.95%
Google	139.41%	151.98%	-0.10%
USD100_M1	103.46%	115.87%	-1.61%
XAU/USD	31.16%	37.28%	-5.42%

Figure.4. demonstrates each the transaction behavior of agent trading with AUD/USD. In the first 200 trading days, there is only one buy in transaction and one sell transaction. The frequency of trading behaviors become more and more frequent over time. The average trading frequency is about 20 days from 300th trading days to 450th trading days. And the average trading

frequency is about 3 days since 500th trading days. It is obvious that the frequency of trading become larger since the price of the product keep decreasing since 200th trading days. And the frequency became smaller along with the rise of price.

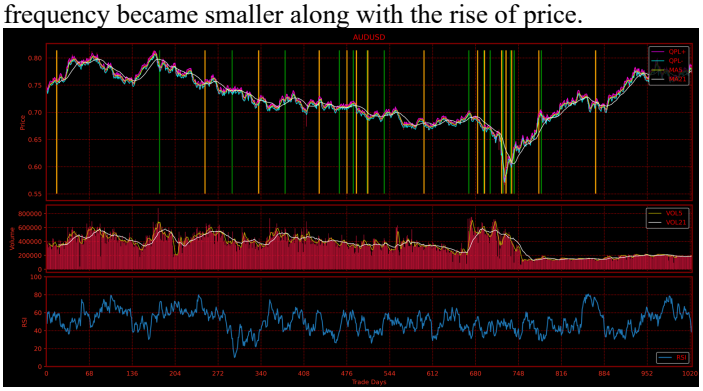

Fig. 4. Transaction Log Graph of AUD/USD

Fig. 5. Demonstrates the transaction log graph of Airbus. There are only 9 buy in transactions and 8 sell transactions. Thus, the frequency of trading of Airbus is relatively lower than other products. In the first 600 trading days, the agent always bought in a relatively low price and sell out in a relatively high price, which make the accumulative return rate grows. There is zero transaction between 360th trading days and 615th trading days when the agent bought in the product in 359th trading days. The reason may be the fluctuation of the price is too extreme and the agent needs enough observations. When the price rises to a stable level on about 900th trading days the agent started to trade.

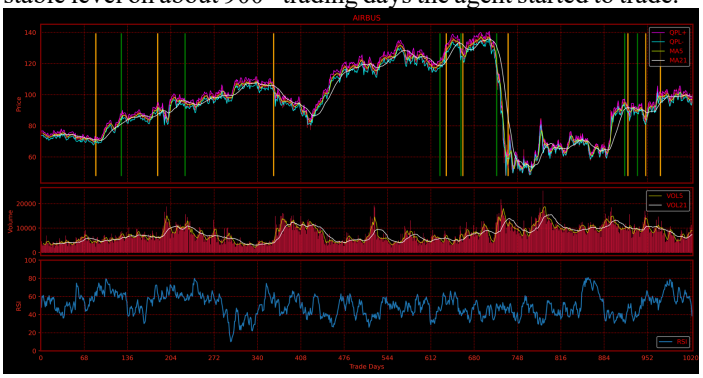

Fig. 5. Transaction Log Graph of Airbus

Fig.6. demonstrates the transaction log graph of Google. It is clear that the transaction agent made shows a certain periodicity and the gap between buy in action and sell action is extremely short. For example, if the agent sells the product in a trading day, there must be buy in behaviors in the next few trading days. In addition, the times of buy in is much more than the times of sell. The probable reason is that the agent learned the long-term tendency of the price and it understands that buy in actions may lead to the increasement of account balance.

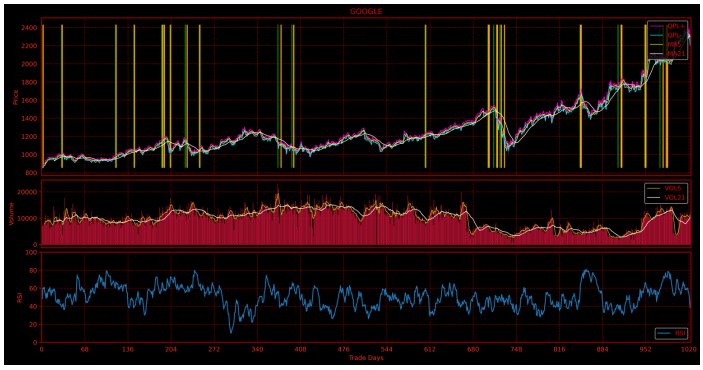

Fig. 6. Transaction Log Graph of Google

Fig.7. demonstrates the transaction log graph of product US100_M1. In the first 350 trading days there is zero transaction behaviors happened. In the next 100 trading days, there are frequency transactions happened. The gap between buy in behaviors and sell behaviors are short. Since the 500th trading days, the agent has bought in 6 times and sell out 2 times, along with the price continuous increasing.

Fig. 7. Transaction Log Graph of US100_M1

Fig.8. is the transaction log graph of XAU/USD. The transaction distribution of XAU/USD is different from other products. Once the agent sold the position in a trading day, the agent must buy in the product in the next trading days. And the agent buy in many times before the price rises to 1,400 USD. When the price rises to the peak, the agent still buys in instead of selling the positions. The reason may be the price is too stable before it rises significantly, and the agent still cannot learn the state of XAU/USD.

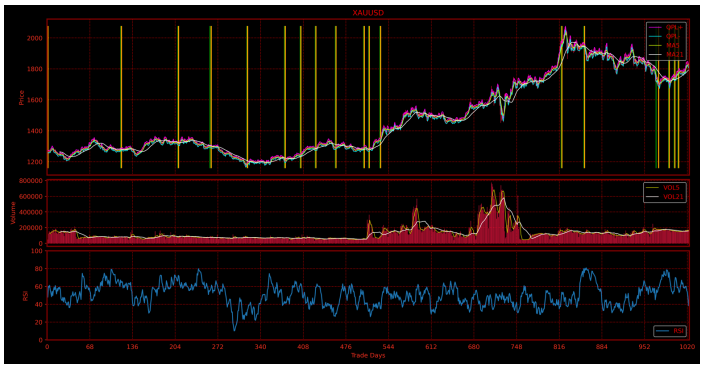

Fig. 8. Transaction Log Graph of XAU/USD

In general, according to the rate of return, the deep Q-learning model is quite effective in trading with indexes futures, CFDs, and shares. And performance on trading with forexes and spot gold shows that there is room to develop. A possible reason is that the fluctuation of forexes and spot gold are relatively small

and the direction changes relatively sudden. It is difficult for the model to capture useful information and make correct decisions.

V. Discussion

Qualitative trading takes advantage of computer technology and mathematics for making rational trading decisions according to the market data. We proposed a deep reinforcement learning based model to execute instructions which is computed by the algorithm to make profits in the financial market.

Reinforcement learning algorithm can let the trading agent learn interaction between actions and environment and make optimal decisions based on any situation. The mechanism naturally suitable for trading in the financial market.

To build the environment, in addition to use the transaction data of financial products and financial indicators, we also introduced the concept of Quantum Price Levels. As a direct analog to the energy levels in an atom, Quantum Price Levels can be considered as invisible energy levels that exist in every financial market, thus are intrinsic in nature. It provides a clear definition of multiple support and resistance of the product.

Since Q-learning requires the memory to store a large Q-table while most of the variables in the financial market are continuous. It is hard for common computer to run the algorithm. Therefore, a deep neural network is used to approximate the original value function. It is clear that this kind of approximation is useful and effective.

From the whole project's perspective, there is some room for improvement. The most important thing in reinforcement learning is the design of reward function. In the project, the reward function is related to the price change of next trading days and current behaviors. The idea is correct, but it makes the agent only care about the short-term profit instead of the long-term profit. It is well-known that there are certain periods for financial markets especially in stock market. For example, many stocks will rise astonishingly in a bull market. If the agent can buy in and hold the position in the early in a bull market, the profit in the end of the bull market will be much higher than expected. Therefore, the design of the reward function should consider both the short-term period and long-term period.

Second is the design of the deep neural network. In our project, the backbone of the neural is fully connected layers with non-linear activation function plus some regularization methods such as batch normalization and dropout. However, the financial data can be considered as the time series data. Neural networks with recurrent units, such as gated recurrent unit and long short term memory cells, will probably fit in the target value function effective and efficiently.

In addition, in the real situation, a portfolio investment is common. If the agent is able to select optimal combination of financial products each day and make optimal transaction behaviors, the rate of return should be higher than only consider on single financial products.

VI. Conclusion

We propose Deep Q-Learning model to build an automated trading agent in the financial market. The environment is a one-dimension vector which contains: [Open Price, Close Price, Highest Price, Lower Price, Volume, QPL,QPL+, QPL-, MA5, MA21, RSI, Account Balance, Account fund, Account Position]. There are three transaction behaviors that an agent can execute: [Buy, Sell, Keep]. To simplify the problem, we set some transaction rules that the agent must obey. Then deep neural network will output the Q-value of each action. The agent will choose the action with largest Q-value in each trading days. In addition, to find out the best combination of parameters, a grid searching is used to search the optimal combination of parameters to maximize the profit.

The result suggests that the deep reinforcement learning model can have positive rate of return in each financial market. Among those five different financial products, the assets of agent with Google and US100M1 are doubled in the last trading days. And the accumulative return rate of trading with Airbus is 81.35%. Compared to these three products, the accumulative return rate of trading with AUD/USD and XAU/USD is relatively low. The rate of return of XAU/USD is 31.16%, which is similar to the increase of US indexes in 1024 trading days. And the rate of return of AUD/USD is even lower, 8.49%. It is lower than the total interest of 3-year bank deposit. The revenue data suggests that the deep reinforcement learning model is effective in trading with stock, futures, and CFD. And performance of trading with forex and spot gold is not ideal.

There are possible room for improvement. The redesign of reward function should let the agent aware to balance the short-term profit and long-term profit. The redesign of deep neural network such as adding the memory cells can make the network fits on the value function better.

References

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

[2] Yitao Qiu, Rongkai Liu and Raymond Lee (2020) The Design and Implementation of Quantum Finance-based Hybrid Deep Reinforcement Learning Portfolio Investment System, 2020 International Symposium on Automation, Information and Computing (ISAIC 2020), 2 - 4 Dec 2020, Beijing, China.

[3] Bao W and Liu X 2019 Multi-agent deep reinforcement learning for liquidation strategy analysis arXiv: 1906.11046

[4] Quantum Finance Forecast Center URL http://qffc.org

[5] Jiang Z, Xu D and Liang J 2017 A deep reinforcement learning framework for the financial portfolio management problem arXiv: 1906.11046

[6] Whittle, P., & Puterman, M. L. (1994). Markov Decision Processes. *Journal of The Royal Statistical Society Series A-statistics in Society*, 158(3).

[7] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6), 1087-1092.

[8] Ziogos, N. P., Tellidou, A. C., Gountis, V. P., & Bakirtzis, A. G. (2007). A Reinforcement Learning Algorithm for Market Participants in FTR Auctions. *ieee powertech conference*.

[9] Guo, M., Liu, Y., & Malec, J. (2004). A new Q-learning algorithm based on the metropolis criterion. *systems man and cybernetics*.

[10] Li, JA., Dong, D., Wei, Z. *et al.* Quantum reinforcement learning during human decision-making. *Nat Hum Behav* **4,** 294–307 (2020). https://doi.org/10.1038/s41562-019-0804-2

[11] Raymond Lee (2021) Quantum Finance Forecast System with

Quantum Anharmonic Oscillator Model for Quantum Price Level Modeling, International Advance Journal of Engineering Research, 4(2), 1 - 12.

[12] Henrik Tünnermann and Akira Shirakawa, "Deep reinforcement learning for coherent beam combining applications," Opt. Express 27, 24223-24230 (2019)

[13] Carapuço, J. et al. (2018) “Reinforcement learning applied to Forex trading,” Applied Soft Computing Journal 73: 783–794.

[14] Jiang, Z., Xu, D., and Liang, J.(2017), “A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem”.

[15] Niaki, S. T. A. and Hoseinzade, S., “Forecasting S&P 500 index using artificial neural networks and design of experiments”, Journal of Industrial Engineering International, vol. 9, 2013. doi:10.1186/2251-712X-9-1.

[16] Sutton R, Mcallester D, Singh S and Mansour Y 2000 Policy gradient methods for reinforcement learning with function approximation Advances in Neural Information Processing Systems Conference 1057-63

[17] Lee RST 2020 Quantum Finance–Intelligent Financial Forecast and Program Trading Systems, Springer NATURE, Singapore

[18] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. IEEE transactions on neural networks and learning systems,28(3):653–664, 2016.

[19] Chien Yi Huang. Financial trading as a game: A deep reinforcement learning approach. arXiv preprint arXiv:1807.02787, 2018.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529, 2015.

[21] John Moody and Matthew Saffell. Learning to trade via direct reinforcement. IEEE transactions on neural Networks, 12(4):875–889, 2001.

[22] Shin, W., Bu, S.-J., and Cho, S.-B., “Automatic Financial Trading Agent for Low-risk Portfolio Management using Deep Reinforcement Learning”, 2019

[23] Silver, David. Deep Reinforcement Learning. Accessed: 2016-12-14.

[24] S. Toulson. Use of neural network ensembles for portfolio selection and risk management, 1996.