The Design and Implementation of Quantum Finance-based Hybrid Deep Reinforcement Learning Portfolio Investment System

Yitao Qiu, Rongkai Liu and Raymond S T Lee*
Division of Science and Technology, Beijing Normal University - Hong Kong Baptist University United International College (UIC), China
*Email: raymondshtlee@uic.edu.cn

Abstract. With the rapid development of Deep Learning (DL) and Reinforcement Learning (RL), Deep Reinforcement Learning (DRL) has become one of the leading trends in automatic trading. However, research of RL in portfolio investment has difficulties in distributing investment funds, controlling profit and loss, and exploring the unseen environment. This paper introduces an intelligent portfolio investment system based on the integration of DRL, Quantum Finance Theory (QFT). Our proposed system consists of two agents: 1) A trading agent based on Deep Deterministic Policy Gradient (DDPG) algorithm to generate continuous actions for investment weighting; and 2) A risk-control agent based on Policy Gradient (PG) algorithm produces discrete actions according to each day’s Quantum Price Levels (QPLs). One significant merit of integrating two intelligent agents is that they can cooperate to make more reasonable and stable fund distribution adjustments in the portfolio investment. The experimental results reflect the flexibility and robustness of our system, as it achieves considerable profits in back-tests consisting of various combinations of FOREX products.

(Source code is available at: https://github.com/772435284/QF-portfolio-investment-system)

1. Introduction

1.1. Significance and Purpose of the Study
In traditional statistics, researchers usually study the time series by modeling them as a linear process, but financial time series have essential characters include complex, highly noisy, dynamic, non-linear, non-parametric, and even chaotic [1]. Deep neural network has become one of the most widely-used methods since it has the following advantages [2]: 1) all information that may be relevant to the prediction problem can be used as model input, 2) fit nonlinear complex relationships between input variables effectively and 3) avoid over-fitting problems of relatively simple structures.

Financial analysis has certain merits in market predictions, but how to construct an effective trading system specifying profitable trading strategies based on the prediction provided by forecast system is still a major challenge [3]. This highlights the importance of mechanism for formulating effective trading strategies. Both financial modeling and Deep Learning (DL) nowadays have limitations in generating trading strategies. For mathematical and financial modeling, they rely on assumptions that generally over-simplify problems. For DL, model performance highly depends on the prediction accuracy. This requires experienced traders to make the final decision, which has defeated the original objective of automatic trading.
Based on the above factors, current studies call for demands on model-free automatic trading system. Reinforcement Learning (RL) consists of agents that keep on making interaction with the environment to learn an optimal policy by trial and error for sequential decision-making problems, provides a feasible solution [4].

1.2. Brief Description of Our Model
Due to the particularity of financial engineering, there are some problems in financial transactions based on Deep Reinforcement Learning (DRL): 1) The Markov Decision Process model for trading tasks lacks a suitable definition. Without a universal MDP model, researchers need to decide the detailed representation of state space, action space, and reward function, which largely determines whether the algorithm can learn effective strategies. 2) Different from agents playing games and controlling robots, trading agents, not only need to optimize the reward gained from environment, but also to prevent potential loss in transactions by risk control.

In this paper, we proposed a DRL framework specially designed for the task of portfolio management. Our model directly generates the trading policy through adjusting the weight of each product chosen in the portfolio in order to maximize the expected accumulative return. Our model consists of 2 core parts: 1) A Deep Deterministic Policy Gradient (DDPG) agent used to work out appropriate weight vectors. 2) A Policy Gradient (PG) agent trained by the same observation spaces as that of DDPG, in order to select the key QPL which works as an effective indicator about at which price level the pending order should be closed. Additionally, the usage of QPL is inspired by theories of QF. Experiments on Forex products are delivered and comparisons with previous portfolio investment methods are conducted. Our system outperforms other methods and has strong adaption to the unseen environment.

2. Related Work
2.1. Quantum Finance Theory
With the integration of Quantum Field Theory (QFT) and Anharmonic Oscillatory Model (AOM), Dr. Lee [5] proposed Quantum Price Level (QPL) as an innovative indicator of market information that could be effectively applied in financial market predictions. Compared with traditional market features and technical indicators, one significant merit of QPL is that it can model the dynamics of financial instruments of worldwide financial markets as financial quantum particles with wave-particle duality characteristics. Since the motion and dynamics of these financial particles are subjected to their intrinsic Quantum Energy Fields, QPL could be used for the design of trading strategy as analogue to the resistance and support lines in technical analysis [6,7].

2.2. Deep Reinforcement Learning in Financial Trading
The performance of supervised learning models highly depends on prediction accuracy and convert price predictions into action needs implementation of additional logical layer [8]. If this layer is implemented in a hand-coded manner, then we cannot regard the whole approach as completely machine-learning. Therefore, the online learning process of DRL would be more suitable for formulating trading strategies. Jiang and his team [8] proposed a financial-model-free DRL framework, consisting of a Portfolio-Vector Memory (PVM) and do a series of experiments by implementing the DPG network in its framework with CNN, RNN and LSTM to find the best back-test results. A combination of PG with adversarial training method [9] also achieves a profitable portfolio investment system with more efficient training.

3. Methodology
3.1. Quantum Price Level based on Quantum Finance Theory
In Quantum Finance Theory (QFT) [5], dynamics of financial instruments of worldwide financial markets can be modeled as Financial Particles (FP) with wave-particle duality characteristics. Moreover, the motion and dynamics of these FPs are subjected to their intrinsic Quantum Energy Fields. Like physical particle, particle in financial area also has its equilibrium states. And if there is external stimulus able to excite particles, particles will be moved to another higher or lower energy levels. We call these
levels Quantum Price Levels (QPLs) in financial market, which can also be regarded as support and resistance. To calculate QPLs, we need to solve Quantum Finance Schrödinger Equation (QFSE), which is given by [5]:

\[
\begin{align*}
\frac{-\hbar}{2m} \frac{d^2}{dr^2} + \left( \frac{\gamma \eta \delta}{2} r^2 - \frac{\gamma \eta v}{4} r^4 \right) \psi(r) &= E \psi(r)
\end{align*}
\]  

(1)

where \( \frac{\hbar}{2m} \frac{d^2}{dr^2} \) is kinetic energy (KE), the second term represents potential energy (PE), \( \hbar \) is the Planck's constant, \( m \) stands for internal potential (e.g. market capital), \( E \) represents the corresponding Quantum Energy Levels of QFPs and \( \psi(r) \) is the wave-function, \( \eta \) is the market resistance coefficient, \( \delta \) denotes the harmonic dynamic term (trend following contribution), and \( v \) represents the anharmonic term (market volatility).

3.2. QPL-inspired Risk Control Mechanism

QPL is a strong indicator about resistance and support lines. The basic idea to make trading decisions based on QPL is to estimate the range of price movement, and then take corresponding actions. Figure 1 shows 3 conditions in the training process. There are three types of decisions that could be made: 1) Base on \( QPL_{+1} \); 2) Base on \( QPL_{+2} \); 3) Hold the order

1) In the first condition, the range \([QPL_{+1}, QPL_{+2}]\) is a subset of that day’s \([LOW, HIGH]\). If \( QPL_{+1} \) is evaluated as the most important factor for decision, then the agent will close orders of current product at \( QPL_{+1} \). Similarly, if \( QPL_{+2} \) is selected, the order would be closed at \( QPL_{+2} \). If the holding decision is made, then the order of current product is kept.

2) In the second condition, both QPLs are not within the range \([LOW, HIGH]\). In this case, making decision based on \( QPL_{+1} \) is the same as that in the first condition. But for \( QPL_{+2} \), once it is selected as the most important factor, the agent would be guided to hold current orders.

3) In the third condition, both \( QPL_{+1} \) and \( QPL_{+2} \) are not in the range of \([LOW, HIGH]\). In this case, the only choice is to hold the order.

3.3. Assumptions and Mathematical Formalism

There are two assumptions in our model: 1) the trading volume is considerably sufficient in the market. This means that all the operations done by our agent will not have influence on the market due to their insignificance; 2) There is no slippage occurred in the market, and the market environment simulated is in high liquidation. So, each trading can be executed immediately at the last price when orders are open. If we select \( m + 1 \) assets to be traded in the portfolio, the closing price of them could comprise a price vector at period \( t \), denoted as \( v_t = (v_{0,close}^t, v_{1,close}^t, \ldots, v_{m,close}^t)^T \). \( v_{i,t} \) indicates the \( i \)th asset’s closing price at period \( t \). A special term \( v_{0,close}^t \) in portfolio called quoted currency, indicating the remaining balance used for purchasing other forex products. As a risk-free asset, we set its value to be always at 1. Moreover, define price relative vector of \( t_{th} \) trading period as \( y_t \):

Figure 1. Three typical locations of QPLs compared with price movements of a specific day.
\[ y_t = \begin{pmatrix} v_{1,t+1}^{\text{close}} & v_{2,t+1}^{\text{close}} & \cdots & v_{m,t+1}^{\text{close}} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1,t}^{\text{close}} & v_{2,t}^{\text{close}} & \cdots & v_{m,t}^{\text{close}} \end{pmatrix}^T, \quad \text{when} QPL \notin [v_{m,t+1}^{\text{low}}, v_{m,t+1}^{\text{high}}] \]

(2)

where \( i \) is an integer, it indicates the specific selected QPL.

The elements of \( y_t \) indicate the quotients of closing prices for each asset in the period \( t \). If we define the portfolio value at the end of period \( t \) as \( p_t \), then \( p_{t-1} \) indicates the portfolio value at the beginning of period \( t \). The cumulative return \( R \) at time \( t \) is \( R_t = \frac{p_t}{p_0} \) and the return \( u \) at time \( t \) is \( u_t = \frac{p_t}{p_{t-1}} \).

In real-world market, all operations need commission fee. Transaction costs may outweigh returns, resulting further loss. Hence, it is necessary to introduce transaction cost to simulate a real-world market. At the beginning of period \( t \), the portfolio weight is indicated by \( w_{t-1} \). Since price movements always happening in the market, the weight vector would change correspondingly into [10]:

\[ w_t = \frac{y_{t-1} \odot w_{t-1}}{y_{t-1} \cdot w_{t-1}} \]

(3)

where \( \odot \) indicates element-wise multiplication.

Assume the commission rate is a constant \( C \), then the cost at time \( t \) is given as:

\[ \mu_t = C \sum_{i=1}^{m} |w_{i,t} - w_{i,t}'| \]

(4)

The return at time \( t \) and cumulative return from 0 to \( T \) can be given as:

\[ u_t = (1 - \mu_t)w_t \cdot y_t \]

\[ R_T = \prod_{t=1}^{T} u_t = \prod_{t=1}^{T} (1 - \mu_t)w_t \cdot y_t \]

(5)

(6)

3.4. Deep Reinforcement Learning

3.4.1 Definition of State Space, Action Space and Reward Function

**State:** A 2-dimensional observation space is built in our model with a shape \((m, n)\), where \( m \) represents the number of products contained in the portfolio, and \( n \) stands for the size of sliding window. For each row, the state space consists of

\[ p_t = \left( v_{1,t}^{\text{open}}, v_{2,t}^{\text{open}}, \ldots, v_{m,t}^{\text{open}} \right) \]

(7)

where \( p_t \) corresponds to market information of the \( i \)th product in the portfolio and \( i \in [1, m] \).

**Action:** The expected weight vector at period \( t \), is given as:

\[ w_t = (w_{0,t}, w_{1,t}, w_{2,t}, \ldots, w_{m,t})^T \]

(8)

**Reward:** We use logarithmic rate of return considering transaction cost as the reward function:

\[ r_t = \ln u_t = \ln((1 - \mu_t)w_t \cdot y_t) \]

(9)

3.4.2 Deep Deterministic Policy Gradient and Policy Gradient

The Deep Deterministic Policy Gradient (DDPG) in our model is trained for distributing investment fund through adjusting the weight vector. According to DeepMind’s work [11], the following Algorithm 1 illustrates each step in DDPG:

**Algorithm 1 Deep Deterministic Policy Gradient Algorithm**

1. Randomly initialize critic \( Q(s,a|\theta^Q) \) actor \( \mu(s|\theta^\mu) \)
2. Initialize \( Q' \) with parameter \( \theta^Q' \leftarrow \theta^Q \) and \( \mu' \) with parameter \( \theta^\mu' \leftarrow \theta^\mu \)
3. Initialize Experience Replay Buffer \( R \)
4. For episode = 1 to M do
   1. \( \mathbb{E} \left[ \sum_{t=1}^{T} r_t \right] \) for each \( s_t \)
   2. \( \mathbb{E} \left[ \sum_{t'=1}^{T'} Q'(s', a'|\theta^Q') - Q(s_t, a_t|\theta^Q) \right] \)
   3. \( \mathbb{E} \left[ \sum_{t'=1}^{T'} \nabla_{\theta^\mu} \mu(s'|\theta^\mu) \sum_{t'=1}^{T'} Q'(s', a'|\theta^Q') \right] \)
5. \( \theta^Q, \theta^\mu \) update by gradient descent
6. For episode = 1 to M do
Initialize a random process $N$ for action selection
Receive initial observation state $s_1$

For $t = 1$ to $T$

For $t = 1$ to $T$

End For

End For

3.4.3 Policy Gradient
In 3.2, we mentioned that the QPL-inspired risk control mechanism and the 3 types of actions that can be taken. This mechanism can be implemented by Policy Gradient algorithm [12] [13]. In PG, policy is directly updated. Define a trajectory as $\tau = s_1, a_1, s_2, ...$, the probability of each trajectory could be calculated through:

$$p_\theta(\tau) = p(s_1)p_\theta(a_1|s_1)p(s_2|s_1, a_1)p_\theta(a_2|s_2) ... = p(s_1)\prod_{t=1}^{T} p_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)$$

(10)

Though taking a specific action, the state return by environment is still random. Therefore, in PG, the goal is to maximize the cumulative expected reward, given as:

$$\bar{R}_\theta = \sum_{\tau} R(\tau)p_\theta(\tau) = E_{\tau \sim p_\theta(\tau)}[R(\tau)]$$

(11)

The action of PG at time $t$ in our model is defined as:

$$a_t = \left(a_{\text{hold}}, a_{\text{QPL}+1}, a_{\text{QPL}+2}\right)$$

(12)

where each element in $a_t$ corresponds to the probability of each type of decision. Then PG agent achieves risk control by selecting the decision with the largest probability, and the decision is given as:

$$d_t = \text{argmax}(a_t)$$

(13)

4. System Implementation and Experiments
4.1. System Architecture
Our portfolio investment system consists of two main parts: 1) DDPG Agent 2) PG Agent. The cooperation of these two agents is shown in the following Figure 2:

Sensing current state $s_t$, DDPG agent would respond with current action $a_t = w_t$, which is the desired weight vector at time $t$. After PG agent makes decision about holding the activating orders or closing activating orders based on a specific QPL, it will receive the weight vector $w_t$ given by DDPG agent. For both agents, $w_t$ is used in calculating the return $u_{\text{DDPG},t}$ and $u_{\text{PG},t}$. With pre-defined constant commission rate $C$, transaction cost could be estimated, and then cumulative reward are computed accordingly. After updating the parameters of both agents, current iteration is completed.
4.2. Experiment Settings
Nine forex products are selected to compose a portfolio including: 1) AUDCAD, 2) AUDUSD, 3) EURAUD, 4) EURCAD, 5) EURUSD, 6) GBPUSD, 7) NZDCHF, 8) NZDUSD and 9) USDCHF. The datasets of nine products are obtained from Metatrader 4. There are 2048 trading days in each dataset from March 2014 to October 2020, with the first 80% used to construct the observation space in training, and the rest 20% used for testing. The size of sliding window for observation space is set to $n = 3$. In the back-testing, the initial portfolio value $p_0 = 10000$ (USD) with commission rate $C = 0.25\%$.

4.3. Network Structure and Training Settings
According to previous works [8] in the construction of the DDPG’s actor and critic network, CNN shows a better performance than LSTM and RNN in many situations. Therefore, as shown in Figure 2, two layers of CNN are used in actor and critic network. In the actor network, the CNN network is followed by two fully connected linear layers and a softmax layer. In the critic network, the structure of CNN part is the same as that of actor network, but there is a dedicated channel for action space. The outputs of both state and action channels are combined in the fully connected layer, then Q-value is generated. For the policy gradient agent, LSTM is efficient and effective at processing sequential data [14], so we first input observations into LSTM network to do a feature extraction. The output of LSTM is then imported to two fully connected linear layers followed by a softmax layer. Finally, the action is sampled from the policy generated previously.

About the training settings, both DDPG agent and PG agent are trained for 100 epochs using the Adaptive Moment Estimation (ADAM) optimizer. For DDPG agent, the learning rates of actor and critic network are 0.0001 and 0.001, respectively. For networks in PG agent, the learning rate is set to be 0.0001.

4.4 Model Settings
In order to compare the performance of QF-Portfolio Investment System with previous portfolio investment methods, we selected two model-based strategies and two reinforcement learning models as baselines: 1) The Best Stock [8], 2) Online Newton Step (ONS) [15], 3) DDPG and 4) Adversarial PG [9]. We implemented two versions of QF Portfolio Investment System (QFPIS): 1) QFPIS-1: the PG agent can generate two actions: a) holding, b) close order at QPL+1, and 2) QFPIS-2: the PG agent can generate three actions: a) holding, b) close order at QPL+1, and c) close order at QPL+2.

4.5 System Performance
To evaluate the performance of different models, we design the performance metrics consisting of Maximum Drawdown (MDD), the final Accumulated Portfolio value (fAPV) and the Sharpe ratio (SR).
The results in Table 1 show that our QF portfolio investment system outperforms other baselines, from which we can summarize these outcomes: 1) In terms of fPAV, QFPIS-2 has the best performance, whose final portfolio value is three times more than the original portfolio value. This shows the powerful profitability of QF Portfolio Investment System; 2) In terms of Sharp Ratio, our two systems’ SR both reach significantly high levels, while our baseline models’ SR are still less than 1. This shows the strong ability in risk control of our models; 3) QFPIS-2 performs better than QFPIS-1 when compared to fPAV and SR, which indicates that the larger action space indeed contributes to the exploration of PG agent, further helping to achieve a higher portfolio value.

### Table 1. Performance Metrics.

| Models         | MDD     | fPAV    | SR      |
|----------------|---------|---------|---------|
| QFPIS-1        | 7.2150% | 2.7659  | 1.6132  |
| QFPIS-2        | 7.6154% | 3.0746  | 1.9874  |
| DDPG           | 7.7579% | 1.3833  | 0.6790  |
| Adversial PG   | 15.4834%| 1.0179  | 0.7582  |
| ONS            | 7.7746% | 1.0128  | 0.0712  |
| Best stock     | 21.9226%| 1.1597  | 0.2727  |

The result of back-testing is shown in Figure 3. In the first half period, i.e. July 2019 to March 2020, almost all the baseline models plateau, but our system still achieved relatively considerable profits. In the second half period, i.e. March to October 2020, both two QFPIS models witnessed a continuous increase. Although almost all the baseline models (except ONS) are able to make profits with a risk control mechanism, our system achieved significantly high portfolio values showing its profitability and robustness.

![Figure 3. Left panel: Back-testing results of 2 models proposed and 4 baseline models based on test dataset dating from 2019.07 to 2020.10; Right panel: Back-testing results of 2 models proposed and 4 baseline models based on unseen environment dating from 2019.08 to 2020.11.](image)

### 4.6. Performance of Unseen Products

In order to further explore the robustness of our QF portfolio investment system, we selected other nine unseen products to do testing which are: 1) AUDCHF, 2) AUDNZD, 3) EURNZD, 4) EURGBP, 5) GBPAUD, 6) GBPNZD, 7) USDCAD, 8) USDSGD, and 9) NZDCAD.

The back-testing result is shown in Figure 3. For unseen products, our two models (QFPIS-1 and QFPIS-2) maintain their profitability. However, for the sole-DDPG agent, its overall performance is the worst among all the models tested. Although the DDPG agent shows a relatively outstanding performance in the back-testing on training products, its single-DRL framework is still limited to have both profitability and strong adaptability at the same time. Hence, this comparison obviously shows that the PG agent added did indeed cooperate with the DDPG agent and prevent potential loss through closing orders at appropriate times.
5. Conclusion and Future work
This paper designs and implements a Quantum Finance based Hybrid Deep Reinforcement Learning Portfolio Investment System. In experiments, our system has better performance over a range of previous methods in the test dataset and unseen dataset. Currently, the QPL-inspired risk control mechanism would do the same operation on each product in the portfolio when a specific close decision is made. Also, although PG agent would make decision by maximizing the desired portfolio value, the portfolio value is calculated by allowing all products do the same operations. In real-life situations, it is more appropriate to allow operations of each product in the portfolio to be separate, and we hope that this issue could be solved through a multi-agent mechanism.

Acknowledgment
The authors would like to thank for Division of Science and Technology of UIC for the provision of computer equipment and facilities. This project is supported by UIC research grant R202008.

Reference
[1] Si Y and Yin J 2013 OBST-based segmentation approach to financial time series Eng. Appl. Artif. Intell 26 2581-96
[2] Heaton J B, Polson N G and Witte J H 2016 Deep learning in finance Appl. Stoch. Models. Bus. Ind 33 3-12
[3] Wang J, Zhang Y, Tang K, Wu J and Xiong Z 2019 AlphaStock: a buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining DOI: 10.1145/3292500.3330647
[4] Bao W and Liu X 2019 Multi-agent deep reinforcement learning for liquidation strategy analysis arXiv: 1906.11046
[5] Lee RST 2020 Quantum Finance–Intelligent Financial Forecast and Program Trading Systems, (Springer Nature, Singapore)
[6] Meng X, Zhang J, Xu J and Guo H 2015 Quantum spatial-periodic harmonic model for daily price-limited stock markets Physica. A 438 154-60
[7] Qiu Y, Yuan Y, Chen Z, Lee RST 2019 Hybrid chaotic radial basis function neural oscillatory network (HCRBFNON) for financial forecast and trading system IEEE Symposium Series on Computational Intelligence (SSCI) pp 2799-806
[8] Jiang Z, Xu D and Liang J 2017 A deep reinforcement learning framework for the financial portfolio management problem arXiv: 1906.11046
[9] Liang Z, Chen H, Zhu J, Jiang K and Li Y 2018 Adversarial deep reinforcement learning in portfolio management arXiv:1808.09940
[10] Ormos M and Urbán A 2011 Performance analysis of log-optimal portfolio strategies with transaction costs Quant. Finance 13 pp 1587-97
[11] Lillicrap T, Hunt J, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D 2016 Continuous control with deep reinforcement learning International Conference on Learning Representations (ICLR) arXiv:1509.02971
[12] Silver D, Lever G, Heess N, Degris T, Wierstra D and Riedmiller M 2014 Deterministic policy gradient algorithms 31st International Conference on Machine Learning (ICML) 32(1) 387-95
[13] 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 pp 1057-63
[14] Ger F A, Schmidhuber J and Cummins F 1999 Learning to forget: continual prediction with lstm 9th International Conference on Artificial Neural Networks 2 850-5
[15] Agarwal A, Hazan E, Kale S and Schapire R E 2006 Algorithm for portfolio management based on newton method Proceedings of the 23rd international conference on Machine learning pp 9-16