An Investigation on Quantitative Techniques for Computational Finance

Cheung Chun Lok
Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong, China
lok419@connect.hku.hk

***Abstract*—With the development of the AI technologies, the use of machine learning algorithms have achieved remarkable performance in many fields, including computational finance. Most of the research focuses on the fundamental or technical indicators prediction which often achieved 55% - 65% directional accuracy. Although some studies had attempted financial portfolio management by reinforcement learning, it was mainly based on discrete action spaces, which leads to inflexibility of portfolio construction. In this research, continuous control with deep reinforcement learning and deterministic policy gradient was adopted. This algorithm is able to provide sufficient flexibility of portfolio construction without growing too much complexity of neural networks, even with a large number of assets. The result was compared with different types of algorithm for online portfolio selection. Based on the back-testing, it was demonstrated that the proposed model was able to generate a portfolio value which is comparable with some of the existing algorithms.**

Keywords*—Machine Learning, Reinforcement Learning, Deterministic Policy Gradient, Continuous Control, Convolutional Neural Network, Portfolio management

1. INTRODUCTION

With the rise of computing power, much research was conducted to forecast the market performance by supervised learning, expecting the hidden patterns in financial data can be recognized. Nonetheless, its performance was not satisfied when comparing with the same application in other fields. It is mainly because market is driven by a group of emotion and events which are hardly predicted by only figures. Recently, with the application of deep reinforcement learning, another challenging task, portfolio management is attempted apart from price prediction, it optimizes the risk and return of the portfolio, which is the goal of this research.

1.1. Objective

This research attempted to implement a reinforcement learning framework and utilize Deterministic Policy Gradient algorithm to generate a portfolio selection vector, each number in the vector represents an exact percentage of total capital allocated to an individual stock in the portfolio during next trading interval. Different reward functions and models were implemented in this research, and the result was compared with different trading algorithms as benchmarks such that effectiveness of models could be evaluated. In addition, this research model attempted to integrate with a price prediction model, its daily price prediction result was designed to interrupt intermediate layer within this research model such that it further enhances profitability of model.

1.2. Outline

This paper firstly introduces the basic concepts of reinforcement learning, including all relevant techniques and algorithms which were adopted in this research. Then, it explains the methodology of modeling a portfolio management problem mathematically by reinforcement learning and the setting of trading simulation. In addition, the configuration of proposed models is clarified. The performance of different experimental settings is compared by figures and graphs, its results are further discussed and summarized.

2. BACKGROUND

This section will briefly describe the background knowledge of reinforcement learning, and the corresponding algorithms for updating a neural network.

2.1. Reinforcement Learning

Reinforcement learning enables the model to address a complex problem, learn and take actions to maximize the reward through interaction with the environment. This environment is modelled by a mathematical framework called Markov decision processes (MDP) which defines five major components:

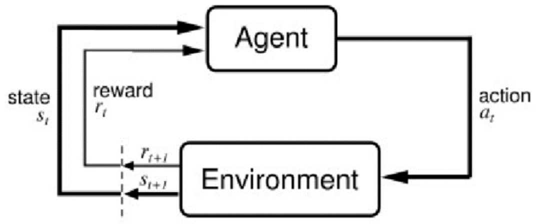

Figure 1. The typical framing of a Reinforcement Learning (RL) scenario

1. A set of (possibly infinite) states s_t
2. A set of actions a_t
3. Transition probability distribution that action A_t in state S_t leads to new state s_{t+1}
4. Reward function $r_a(\mathrm{s}, s_{t+1})$ after the transition by action A
5. Discount factor $\gamma \in [0,1]$ to penalize the future reward

Markov decision processes also provide one important property that state S_{t+1} entirely depends on S_t rather than any states before time t. The agent can establish a policy,

$$\pi_\theta(s) : \mathrm{S} \rightarrow \mathrm{A} \tag{1}$$

which determines the action to be chosen in current state. The methodology in this research adopted the concept of deterministic policy. It is a deterministic function which always produces the same action given the same state, hence,

$$\mathrm{a} = \pi_\theta(s) \tag{2}$$

In order to evaluate the performance, state value function V(s) and state-action value function Q(s, a) are introduced:

$$V_\pi(s) = E_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right) \tag{3}$$

$$Q_\pi(s,a) = E_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = \mathrm{a}\right) \tag{4}$$

V(s) is the expected reward in current state considering all possible actions while Q(s, a) is the expected reward of taking an individual action under current state.

2.2. Convolution layer

The Convolutional Neural Network (CNN) has shown excellent performance in many computer vision and machine learning problems which consists of high dimensional input feature [1]. For portfolio management task, CNN offers benefits of ease of implementation and spatial information.

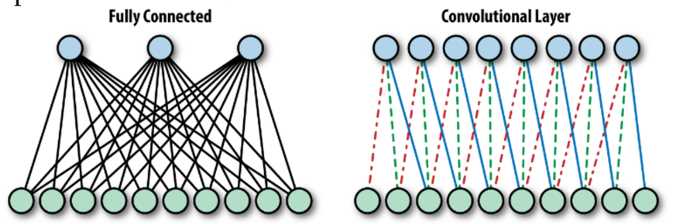

Figure 2. Comparison between Convolution and fully-connected layer

For fully connected layer, outputs from preceding layer are being connected to all input neurons in next layer. On the other hand, filters in convolution layer can extract a partial pattern from the input effectively. For example, if there is a set of stock price with 60 days, literally, stock price of first day and last day are not much correlated, but strongly correlated to their adjacent days. Therefore, a filter size of 5x1 can be applied to extract every 5-days movement within 60 days which might strengthen the learning ability of the network.

Much recent research had already concluded that CNN can actually produce a comparable performance to that of FNN and SVM in stock price prediction [2] [3]. Therefore, convolution layers were selected as the hidden layer in one of proposed network.

2.3. Long Short-Term Memory

Long short term memory (LSTM) is a building block of recurrent neural layer which is a type of layer aimed to deal with sequential data. LSTM is explicitly invented to solve the long-term dependency problem which is mainly due to vanishing gradient problem. It enables the cell to “remember” a value over arbitrary time intervals. Each individual gate is similar to conventional artificial neuron.

In this research, effectiveness of LSTM layer was experimented by two approaches. The first approach was to test the performance of LSTM solely, the second approach was to evaluate the performance when LSTM layer worked together with CNN. It is expected that LSTM will achieve a satisfactory performance on analyzing financial data which is a set of sequential data.

2.4. Deterministic Gradient Policy

The concept of DPG is to directly maximize the Q-value by gradient ascending method. Firstly, by considering a deterministic policy μ_θ with trainable model parameters θ which are weights of neural network, a performance objective can be defined as,

$$\mathrm{J}(\mu_\theta) = \mathrm{Q}(\mathrm{s}, \mu_\theta) \tag{5}$$

This is simply an action-value function with respect to network parameters (or policy). Next, the gradient of performance objective can be estimated by a mini-batch which comprises N observed states,

$$\nabla_\theta J(\mu_\theta) \approx \frac{1}{N}\sum_i \nabla_\theta \mu_\theta(s_i) * \nabla_a Q(s_i, a_i)|_{a=\mu_\theta} \tag{6}$$

Therefore, the trainable model parameters are updated by

$$\theta' = \theta + \alpha \nabla_\theta J(\mu_\theta) \tag{7}$$

where α is the learning rate and the plus sign indicates that these parameters are updated toward maxima of performance objective $\mathrm{J}(\mu_\theta)$.

This algorithm enables continuous action control, and the implementation is as simple as Q-learning with neural network. Hence, it was adopted in this research model.

3. METHODOLOGY

For an automated trading machine, financial assets in portfolio are reallocated regularly within a defined period, and it should be able to react to market trends. This section defines the portfolio management problems in a mathematics manner.

3.1. Hypotheses

In this research, some hypotheses should be made before carrying out back-test.

1 The liquidity of all traded asset is high enough that the exact price can be bought and sold instantly
2 Volume of each trading made by this experiment is small enough that there is no significant impact on overall markets
3 Continuity of Stock Unit: The agent can trade at any number of stock (e.g. 5.5 stocks)

3.2. Trading period

Under proposed model, the trading interval was defined on a daily basis. The trading period can be further divided into training, validating and testing period.

3.2.1. Training period

Training period enables model to learn and update the model's parameters in order to maximize a specific reward. Iterating the training period once is defined as one episode. Additionally, the feature set to be trained in each episode was randomly constructed by different stocks within the same GICS sector, this approach was aimed to prevent overfitting the same feature set.

3.2.2. Validating period

Validating period evaluates the performance of trained model with unseen data. The best episode was chosen according to its performance in validating period when serval tests can be carried out in order to obtain a set of best model hyper-parameters.

3.2.3. Testing period

Testing period evaluates how the trained model with a set of validated model hyper-parameter reacts to unseen data, it reflects the true performance of the model in a more realistic manner. In the proposed model, the concept of online learning was applied. That is the model was continuously receiving a reward and being updated during testing period, it enables the model to optimize a new policy towards the unseen data.

3.3. Data Treatments

The portfolio contains m assets. The input price tensor is a three-dimensional tensor, each of the dimension represents assets, time and financial indicator respectively.

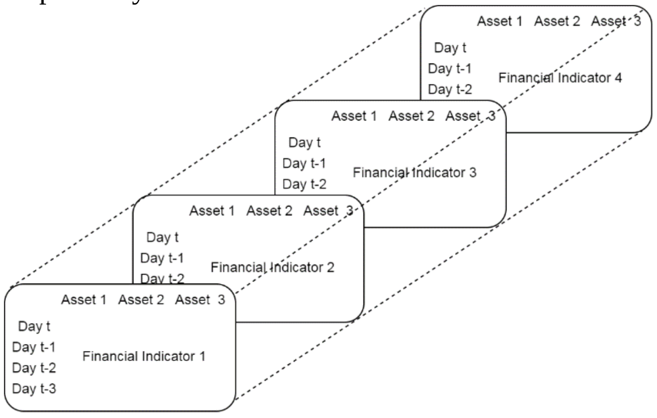

Figure 3. Illustration of input price matrices X_t

3.3.1. Asset Selection

Stocks in S&P500 were selected to be tested in the experiment and the number of assets in portfolio was set to 15 in this stage. For stock pre-selection, it was mainly based on beta β which indicates the volatility of individual asset compared to market.

Portfolio	GICS sector	Company
1 (High beta)	Information Technology	'AAPL', 'AMAT', 'AMD', 'CSCO', 'EBAY', 'GLW', 'HPQ', 'IBM', 'INTC', 'KLAC', 'MSFT', 'MU', 'NVDA', 'QCOM', 'TXN'
2 (Low beta)	Consumer Discretionary	'AZO', 'BBY', 'DHI', 'F', 'GPS', 'GRMN', 'HOG', 'JWN', 'MAT', 'MCD', 'NKE', 'SBUX', 'TJX', 'TWX', 'YUM'
3 (High-Low beta)	Industrials	'BA', 'CAT', 'CTAS', 'EMR', 'FDX', 'GD', 'GE', 'LLL', 'LUV', 'MAS', 'MMM', 'NOC', 'RSG', 'UNP', 'WM'

Table 1. Description of pre-selected assets

Three groups of stocks with different beta were pre-selected to construct portfolios for experiment. The correlation between portfolio return by this research model and beta was one of interests in this research.

During training phase, stocks within the corresponding sector were chosen randomly to construct a portfolio for each episode. This approach can prevent model from overfitting a same set of financial data and enable the model to learn overall market movement or specific correlation within a sector effectively.

3.3.2. Financial Indicator

It is expected that the model is able to recognize a specific pattern within financial indicators which leads to positive reward.

Name	Definition / Implication
Fundamental Indicators	
Close	Daily adjusted close price
Open	Daily adjusted open price
High	Daily adjusted high price
Low	Daily adjusted low price
Volume	Daily trading volume
Technical Indicators	
ROC	Price rate of change
MACD	Moving average convergence divergence
MA5	5-day simple moving average
MA10	10-day simple moving average
EMA20	20-day Exponential Moving Average
SP20	20-day Sharpe ratio

Table 2. Description of the input variables

3.3.3. Time series

A sliding window was defined as a set of delayed prices plus today price. In the experiment, the sliding window which comprised the daily price of last month

(20 business day) was proposed. Some pre-test was conducted which shown that a sliding window of 20-day achieved the best performance.

3.3.4. Data normalization

For a neural network, data must be pre-processed before feeding into the network. In the experiment, most of the financial indicators were nominalized with its latest value, the mathematical expression is:

$$X_{price} = \left(1\,,\ \frac{\vec{X}_{price,\ t-1}}{\vec{X}_{price,\ t}}\,,\ \frac{\vec{X}_{price,\ t-2}}{\vec{X}_{price,\ t}}\,, \dots\dots\right)^{T} \qquad (8)$$

By this approach, each element within the input matrices now depends on the latest value. As a result, the whole matrix is an alternative representation of latest price to a certain extent. Yet, ROC and SP20 are the indicators that processed without normalization. This is because its range normally lies between 0 and 2 which is good enough to be processed in neural network without normalization.

3.3.5. Portfolio selection vector

The output vector is a one-dimensional vector with a size of m+1 (number of assets + 1). Each element represents the percentage of each asset or cash to be held in the portfolio at t day.

$$\vec{w}_t = (w_1, w_2, w_3 \dots\dots)^T \qquad (9)$$

$$\sum_{i=0}^{m} w_i = 1 \qquad (10)$$

The action of short selling was not considered in this experiment, so all weights are positive number. By adjusting parameter of cash bias manually, tendency of holding cash and aggressiveness of portfolio can be controlled.

3.4. Portfolio Return

In a real situation, it is impossible to access market close price and trade it on the same day, as the market has already closed once received OHLC price.

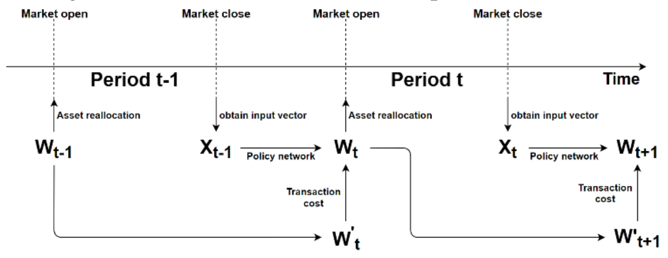

Figure 4. Illustration of trading simulation

A solution which was adopted is to define the actual trading (assets reallocation) which takes place on the next business day. The trading simulation procedures were as follows: An input price matrix X_{t-1} was retrieved from markets. Then a portfolio selection vector w_t was generated by the policy network. Lastly, the asset reallocation was executed at market open of time t. By hypotheses 1 and 3, it is assumed that order could be made on any exact open price.

Since all the orders were executed at open price, the day return should be determined by the difference of two successive adjusted open price. Let y_t denotes adjusted open price vector, each element represents adjusted open price of respective assets in the portfolio at time t. Portfolio day return can be derived as:

$$r_t = \vec{w}_{t-1} \cdot \frac{\vec{y}_t}{\vec{y}_{t-1}} \qquad (11)$$

$$\text{cumulative return} = \prod_{t=0}^{n} \vec{w}_{t-1} \cdot \frac{\vec{y}_t}{\vec{y}_{t-1}} \qquad (12)$$

The cash component (in U.S dollar) of y_t was defined as 1 at any time, such that

$$\frac{y_{cash,t}}{y_{cast,t-1}} = 1 \qquad (13)$$

It implies that there is no profit and loss for holding cash component.

3.4.1. Transaction cost

The transaction cost (in percentage) at time t can be approximated as:

$$\text{transaction cost} \approx c_s * \sum_{i=0}^{m-1} |w_{t,i} - w_{t-1,i}| \qquad (14)$$

where c_s is the transaction factor which is 0.25% of trade value. It ignores the weight change due to daily market movement, as the change is considered to be very small.

4. PROPOSED MODEL

This section introduces the design of the policy network, presents a reinforcement learning framework for problem defined at section 3.

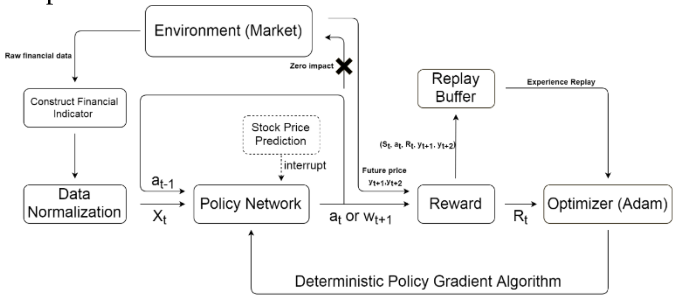

Figure 5. Flowchart of Proposed Model

Figure 5 demonstrates the framework of proposed model. The details of each procedures and sub-model will be discussed one by one below.

4.1. Environment and agent

The input price matrices X_t which contains a set of fundamental and technical indicators is regarded as external state. In contrast, the portfolio selection vector from previous time step was considered as internal state, as the last action was decided by RL agent itself. Since the difference between w_t and w_{t+1} is proportional to transaction cost and day return. It is expected that the RL agent can learn an optimized strategy without spending too much transaction cost.

Combining both internal and external, the state at time t can be interpreted as:

$$s_t = (X_t, w_t) = (X_t, a_{t-1}) \quad (15)$$

The action a_t generated from the policy network redistributes all capital in open market of t+1,

$$a_t = w_{t+1} = \pi_\theta(s_t) \quad (16)$$

where π_θ denotes the policy network given trainable parameters θ

4.2. Policy Network

In this research, four policy networks with different combination of convolution layers, fully connected layers and LSTM layers were proposed. All proposed networks follow the concept of Ensemble of Identical Independent Evaluators (EIIE) which was introduced by Jiang 2016 [3]. This concept is to evaluate the performance of each asset independently, but sharing the same parameters between columns in EIIE configuration. Therefore, the network does not bias to one or two specific investment combinations, since all data are passed independently and a voting score is submitted to softmax layer to determine an action.

The aims of testing four policy networks are to evaluate the effectiveness of CNN and LSTM in portfolio selection and explore the best combination.

4.2.1. CNN

This network consists of three convolution layers only to produce a set of voting score for each equity.

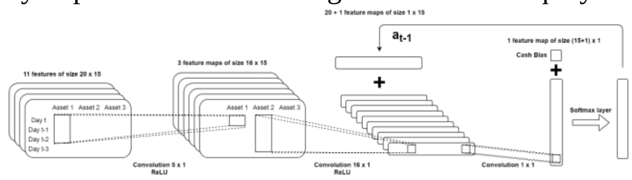
11 features of size 20 x 15
3 feature maps of size 16 x 15
20 + 1 feature maps of size 1 x 15
a_{t-1}
1 feature map of size (15+1) x 1
Cash Bias
Softmax layer
a_t
Convolution 5 x 1 ReLU
Convolution 16 x 1 ReLU
Convolution 1 x 1

Figure 6. Implementation of CNN network

All kernel sizes are nx1 which ensures all assets are being processed separately. Then the voting score plus a cash bias are passed by a softmax layer to generate a set of valid portfolio selection weights such that sum of output equals to one. As mentioned in section 4.1, each state included previous action in order to minimize transaction cost, the last action was inserted between second and third convolution layer.

4.2.2. CNN + Dense

This model consists of two convolution layers and two fully connected layers.

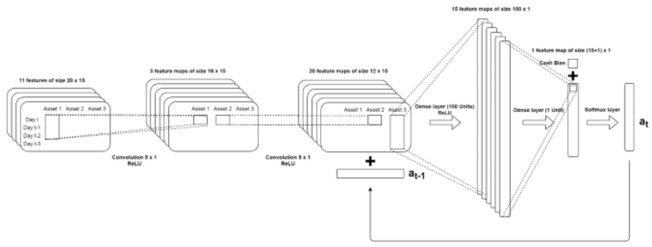
11 features of size 20 x 15
3 feature maps of size 16 x 15
20 feature maps of size 12 x 15
15 feature maps of size 100 x 1
1 feature map of size (15+1) x 1
Cash Bias
Dense layer (100 Units) ReLU
Dense layer (1 Unit)
Softmax layer
a_t
a_{t-1}
Convolution 5 x 1 ReLU
Convolution 5 x 1 ReLU

Figure 7. Implementation of CNN + Dense network

The first two layer are similar to CNN model except the kernel size of second convolution layer. The last layer was replaced by a fully connected layer which converts a 2-dimensional feature map to a voting score.

4.2.3. CNN + LSTM

This model consists of two convolution layers and two LSTM layer.

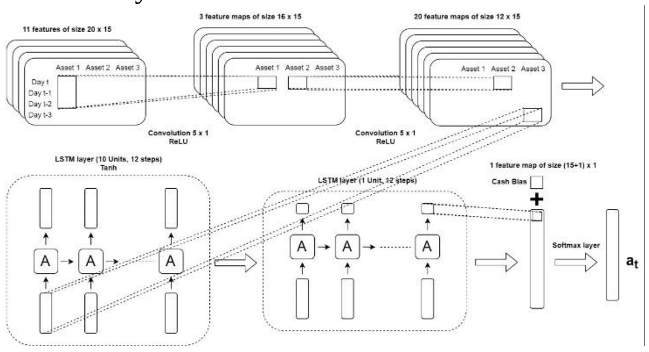
11 features of size 20 x 15
3 feature maps of size 16 x 15
20 feature maps of size 12 x 15
Convolution 5 x 1 ReLU
Convolution 5 x 1 ReLU
LSTM layer (10 Units, 12 steps) Tanh
LSTM layer (1 Unit, 12 steps)
1 feature map of size (15+1) x 1
Cash Bias
Softmax layer
a_t

Figure 8. Implementation of CNN network

The first two convolution layers are exactly same as CNN + Dense model. The following layers are replaced by two LSTM layers which process a set of sequential data to generate a voting score for each asset. The aim of this network is to explore the possibility of connecting CNN and LSTM in series to process and analyze a set of financial data.

4.2.4. LSTM

This model consists of two LSTM layers only.

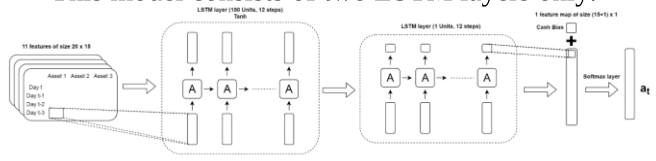
11 features of size 20 x 15
LSTM layer (100 Units, 12 steps) Tanh
LSTM layer (1 Units, 12 steps)
1 feature map of size (15+1) x 1
Cash Bias
Softmax layer
a_t

Figure 9. Implementation of LSTM network

The configuration of LSTM layers is same as CNN + LSTM model. Nonetheless, because of absence of convolution layer, the number of LSTM cell should be increased to strengthen the computational ability of network which might proportional to the number of trainable parameters in the network. This model was designed to test the effectiveness of solely using LSTM layers to analyze and process a set of financial data.

4.2.5. Cash Bias

Cash bias is a constant number associated with voting score layer. It determines the tendency of holding cash in portfolio. Therefore, the activeness of the trading strategy can be controlled by adjusting cash bias. It was tuned in validation phase according to its performance and range of voting score.

4.3. Reward Function

The main objective of policy network is to maximize the reward or the action-state value Q by computing gradient of reward with respect to network's trainable parameter. Under hypothesis 2, action from policy network would not affect any future state and price, thus, it was boldly proposed that the discount factor of Q-value was zero. It implies the model does not care about future reward other than reward made by current action.

In this experiment, four reward functions were created. By mini-batch updating, the reward function was fed by a set of time-ordered actions and future prices. Let t_b and n_b denote a specific point of time and size of each mini-batch respectively.

4.3.1. Average logarithmic cumulative return

$$R_1 = \frac{1}{n_b} \sum_{t=t_b}^{t_b+n_b-1} \ln(r_t) \quad (17)$$

where r_t is the day return defined in section 3.4.

The goal of the model is to generate a portfolio value as high as possible, therefore, it is reasonable to utilize an average logarithmic cumulative return within a specific period as reward function.

4.3.2. Relative average logarithmic cumulative return

$$R_2 = \frac{1}{n_b} \sum_{t=t_b}^{t_b+n_b-1} \ln(r_t) - ln\left(\frac{1}{m} sum\left\{\frac{\vec{y}_t}{\vec{y}_{t-1}}\right\}\right) \quad (18)$$

It is simply logarithmic difference between return generated by policy network and Uniform Constant Rebalanced Strategy (UCRP) which is uniformly holding each equity throughout the period. The main expectation of the model is not only obtaining a considerable profit, but also outperforming the average return.

4.3.3. Sharpe Ratio

$$R_3 = \frac{\sqrt{n_b}\left(\frac{1}{n_b}\sum_{t=t_b}^{t_b+n_b-1} \ln(r_t) - \ln(r_f)\right)}{std\{\ln(r_t), \ln(r_{t_b+1}), \dots\dots, \ln(r_{t_b+n_b+1})\}} \quad (19)$$

It measures the risk-adjusted which is Sharpe ratio return across a specific period of time. r_f represents daily risk-free rate of return which is a return of an absolutely risk-free investment over a period of time.

4.3.4. Calmar Ratio

$$\mathrm{R_4} = \frac{\left(\prod_{\mathrm{t=t_b}}^{\mathrm{t_b+n_b-1}} \mathrm{r_t}\right) - 1}{\mathrm{MDD}} \quad (20)$$

It measures the risk-adjusted return across a specific period of time. The numerator represents the overall compounded return while denominator is the maximum drawdown which measures maximum cumulative loss from a peak to a following bottom [4].

4.4. Experience Replay

Experience replay is one of the important technique to fasten the learning process of neural network. The concept of experience replay is to reuse previous states to update model by mini-batch. With experience stored in a replay buffer, it enables the model to eliminate the temporal correlation by combining more and less recent experiences for update, and rare experience can be reused [5]. It provides advantages of enhancing the sample efficiency by allowing mini-batch updates, and improving the computational efficiency [6].

4.4.1. Stochastic Time-Order Batching

Market movement and action made by trading machine is continuous with time, it is reasonable that batching in this model should be a set of time-ordered prices, states and actions. In consequence, the overall experience replay procedure can be illustrated as:

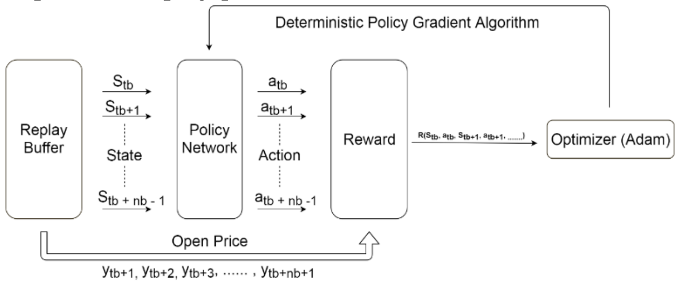
Deterministic Policy Gradient Algorithm
Replay Buffer
S_{tb}
S_{tb+1}
State
$S_{tb+nb-1}$
Policy Network
a_{tb}
a_{tb+1}
Action
$a_{tb+nb-1}$
Reward
$R(S_{tb}, a_{tb}, S_{tb+1}, a_{tb+1}, \dots)$
Optimizer (Adam)
Open Price
$y_{tb+1}, y_{tb+2}, y_{tb+3}, \dots\dots, y_{tb+nb+1}$

Figure 10. Detailed diagram of experience replay

In training phase, experience replay was processed once each time step only, but the same period with random portfolio was iterated for many times. Conversely, since validating or testing period was only iterated once, it is necessary to increase the number of experience replay for each time step in order to enable the network to learn the recent market movement.

4.4.2. Weighted experience

Whenever a mini-batch of experiences is drawn from replay buffer, the chance of each mini-batch to be selected follows a probability distribution. More recent experiences are more important than less recent experiences. Therefore, each mini-batch of experience was allocated a weight which enlarges or dismisses its probability of being picked. A sample bias determines the tendency of selecting recent experience to train, it is defined in a range from 1 to 1.15, which a value above causes the model to overly bias the latest data and degrade the overall performance eventually.

4.5. Benchmarking

To measure the performance, the profit produced by the proposed models should be compared with certain benchmarks. In this experiment, the following benchmarks were proposed:

1. Uniform Constant Rebalanced Portfolio (UCRP)
2. Uniform Buy and Hold Portfolio (UBHP)
3. Robust Median Reversion (RMR)
4. On-line portfolio selection with moving average reversion (OLMAR)
5. Passive aggressive mean reversion strategy (PAMR)

6. Anti-correlation (Anticor)

4.6. Price prediction model

The proposed model in this research was integrated with a price prediction model in order to enhance the overall profitability.

4.6.1. Model Configuration

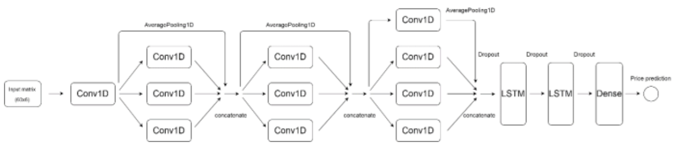

Figure 11. Implementation of price prediction model

The price prediction model is composed of 1D convolution layers, LSTM layers and fully connected layers. The input sliding window was defined as 60-day and the output (label) of this model was logarithm change of next day close price. This price prediction model was trained with almost every stock in S&P500.

4.6.2. Price Prediction Vector

The price prediction vector f_t is defined as predicted daily change of price of all equites held in a portfolio at time t. Since outputs from price prediction model is logarithm change r_i', the price prediction vector can be expressed as:

$$f_t = \left(e^{r_0'}, e^{r_1'}, \dots\dots, e^{r_{14}'}\right) \qquad (21)$$

Therefore, each element in f_t representing a daily change prediction of a stock in portfolio. Then f_t was multiplied to the voting score layer to adjust the action generated by portfolio selection model.

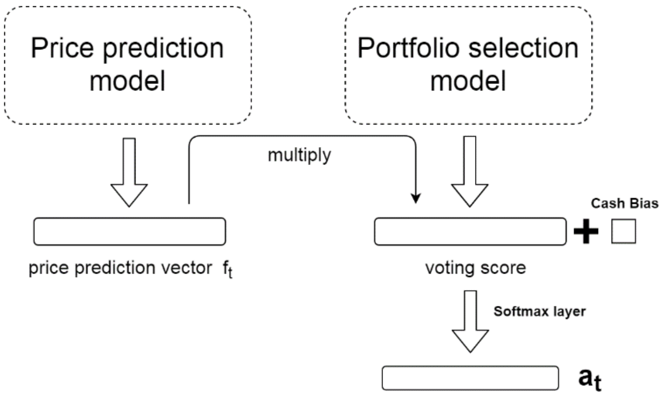

Figure 12. Connection of portfolio selection model and price prediction model

Each voting score was multiplied by corresponding price prediction one by one. Mathematically, a new voting score $v_{t,i}'$ of an equity at time t can be expressed as,

$$v_{t,i}' = v_{t,i} * {f_{t,i}}^{d} \qquad (22)$$

where $v_{t,i}$ is the voting score of an equity directly generated from portfolio selection model, d is dependent factor which represents the tendency of final action relying on price prediction. If d is large, it implies output of portfolio selection model is mainly determined by price prediction model rather than itself. If d is zero, it indicates that model works without the aid of price prediction. Therefore, the price prediction vector can be treated as another input tensor to the network other than input price matrix X_t, last action a_{t-1}.

This implementation gives three advantages. The first one is that two models, portfolio selection model and price prediction model, can be developed and trained separately. The second advantage is that improvement is guaranteed when directional accuracy of price prediction is high enough. Thirdly, it provides the flexibility of depending or not depending the price prediction model.

5. EXPERIMENT

Each experiment was conducted by training, validating and testing phase. The following features were tested experimentally,

- Effect of learning rate
- Effect of cash bias
- Performance of reward functions
- Performance of models
- Performance of proposed model integrating with stock prediction

For each of experiment, cumulative return (CR), maximum drawdown (MDD), Sharpe ratio (SR) and volatility were measured one by one against different model hyper-parameters. In this experiment, volatility is defined as standard derivation of daily return throughout a specific period.

A. Experimental settings

Training period	From 2008-01-04 to 2014-12-31	
Validating period	From 2015-01-02 to 2015-12-31	
Testing period	From 2016-01-04 to 2017-12-31	
Episode	100 – 600	
Hyper-parameters	Value	Description
Training step	1759	Total number of iteration for each episode during training
Window size	20	Number of rows (lookback days) in input price matrix
Transaction factor	0.0025	Cost of equity transactions defined in 3.4.1
Replay Buffer size	200	Number of latest historical data available for online training
Batch size	10 / 60	Number of historical data (time-ordered) for each online training step
Learning rate	$2\text{x}10^{-6}$ – $9\text{x}10^{-4}$	Parameter α (the step size) of the Adam optimization
Rolling steps	0 – 30	Number of online training steps in validating period and testing period
Cash bias	0 – 50	Tendency of holding cash

Sample bias	1 – 1.1	Tendency of picking recent historical data for online training

Table 3. General Preliminary setting of experiments

A batch size of 60-days was selected when reward function was Sharpe ratio (R_3) or Calmar ratio (R_4). As these financial indicators usually require a longer period of time to be estimated especially for its risk component. Additionally, some hyper-parameters are specified in a range, as it varied according different models or portfolios. Learning rate, rolling steps, cash bias and sample bias were tuned in validating period. Unless specified otherwise, reward function of all tests is average logarithmic cumulative return $\boldsymbol{R_1}$.

5.1. Effect of Learning Rate

This section demonstrates how the learning rate (LR) of the model affects the trading strategy and performance.

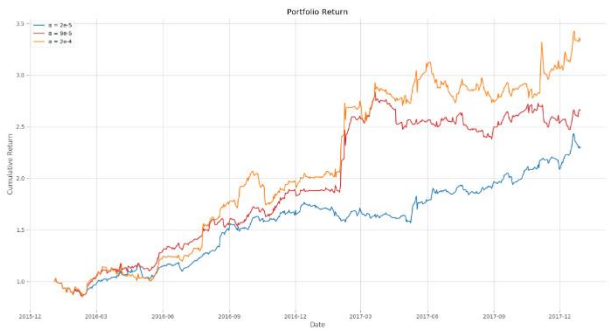

Figure 13. Portfolio 1: Cumulative return with different learning rates (CNN model)

LR	CR	MDD	SR	Volatility
$2\text{x}10^{-5}$	2.29	**0.171**	2.22	**0.0141**
$9\text{x}10^{-5}$	2.66	0.175	**2.32**	0.0164
$2\text{x}10^{-4}$	**3.33**	0.173	2.30	0.0209

Table 4. Portfolio 1: Performance of CNN model with different learning rates

The result demonstrated that the model generally achieved better performance for larger learning rate. During beginning of February 2017, the model with larger learning had established a strategy which rose the cumulative return dramatically in only few days. Nonetheless, small learning rate obtained the lowest maximum drawdown and volatility which are one of its benefits. Furthermore, from the daily assets allocation, model with smaller learning rate tended to hold two or three stocks in each interval. In contrast, model with larger learning rate only held one stock for most of the time. It can be concluded that learning rate directly relates to the aggressiveness of trading strategy, a large learning rate implies a strategy of high-risk high-return and vice versa.

5.2. Effect of Cash Bias

This section demonstrates how the cash bias (CB) defined in section 4.2.5 affects the trading strategy established by model.

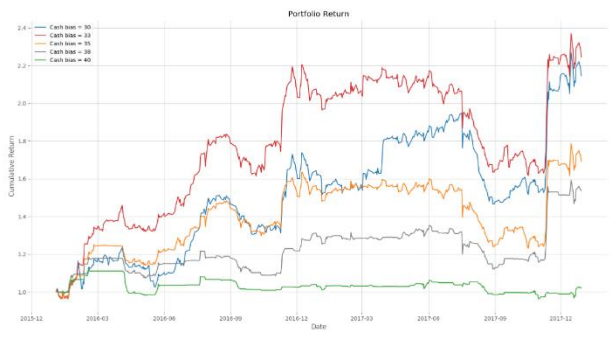

Figure 14. Portfolio 2: Cumulative return with different cash bias (CNN + Dense model)

CB	CR	MDD	SR	Volatility
30	2.14	0.248	1.43	0.0198
33	**2.24**	0.267	**1.60**	0.0191
35	1.69	0.247	1.01	0.0177
38	1.53	0.171	0.931	0.0146
40	1.02	**0.141**	-0.707	**0.0063**

Table 5. Portfolio 2: Performance of CNN + Dense model with different learning rates

The result illustrated that the cash bias value of 33 achieved the best performance which outperformed others, though, it is not an absolute value which can apply to every situation. But it implies a reasonable tendency of holding cash can actually enhance the profitability. From Table 5, it can be concluded that model with large cash bias value often achieved best maximum drawdown and volatility. Therefore, value of cash bias directly relates to the aggressiveness of trading strategy.

5.3. Performance of Reward Function

This section measures the performance of reward functions defined section 4.3 which two of them are based on cumulative return and others are based on risk-adjusted return.

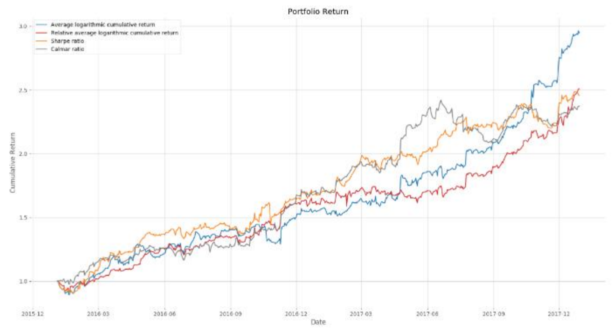

Figure 15. Portfolio 3: Cumulative return with four reward functions (CNN + Dense model)

Reward function	CR	MDD	SR	Volatility
R_1	**2.94**	0.109	**3.24**	0.0131
R_2	2.50	**0.0830**	3.10	**0.0114**
R_3	2.45	0.107	2.77	0.0124
R_4	2.37	0.137	2.75	0.0119

Table 6. Portfolio 3: Performance of CNN + Dense model with different reward functions

The results indicated that reward function of Average logarithmic cumulative return (R_1) achieved the best performance in terms of both cumulative return and

Sharpe ratio, though, it did not intent to optimize the Sharpe ratio in the first place. This is because once the average return is large enough, the Sharpe ratio would be large regardless of its risk. In term of risk management, reward function of Relative average logarithmic cumulative return (R_2) maintained the lowest maximum drawdown and derivation of return throughout the period. It is out of original expectation since both reward function of Sharpe ratio (R_3) and Calmar ratio (R_4) did not show any outstanding performance among all reward functions.

5.4. Performance of Models

Three tests with three portfolios were carried out to evaluate the performance of four models defined in section 4.2. All models used average logarithmic cumulative return (R_1) as reward function.

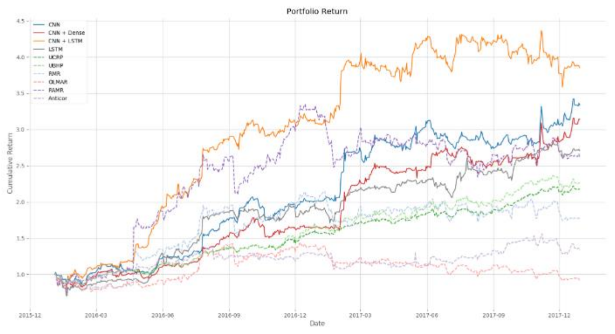

Figure 16. Portfolio 1: Cumulative return of four models and benchmarks

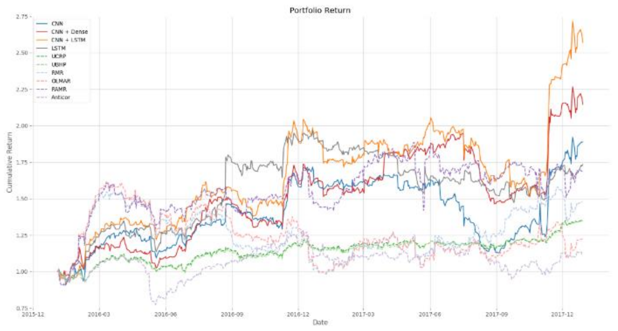

Figure 17. Portfolio 2: Cumulative return of four models and benchmarks

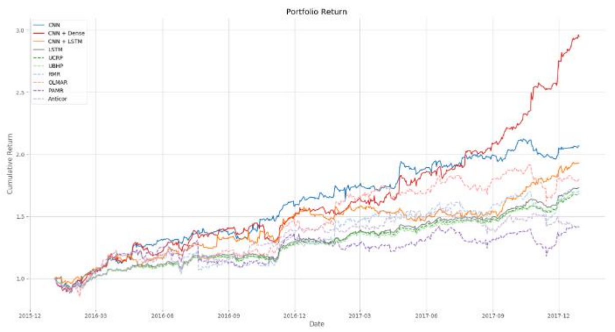

Figure 18. Portfolio 3: Cumulative return of four models and benchmarks

From the results, it was highlighted that the proposed models normally achieved a satisfactory result compared to benchmarks defined in section 4.5. From back-test on Portfolio 1, CNN + LSTM model almost outperformed all other algorithms throughout entire testing period while CNN model had a slightly better performance than CNN + Dense model. From back-test on Portfolio 2, CNN + LSTM model earned a highest portfolio profit again. Noted that all CNN models reacted to a sharp rising trend of a single stock (MAT) at the beginning of November 2017 and decided to distribute all capital into it, then earned a considerable profit eventually. From back-test on Portfolio 3, CNN + Dense model had the highest cumulative return followed by CNN model and CNN + LSTM model. It is observed that all algorithms suffered a small loss during November 2017 except CNN + LSTM model.

From all experiments, CNN, CNN + Dense and CNN + LSTM models obtained the highest cumulative return. It reveals that the effectiveness of CNN is significantly better than solely use of LSTM layer. Furthermore, the result indicated that additional layers of Dense or LSTM can actually enhance the profitability of CNN model. Generally, all proposed models cannot maintain low maximum drawdown and volatility, though, it is still slightly lower than algorithms of RMR, OLMAR and PAMR which are regarded as active trading strategies in some cases. It is mainly attributed to the use of reward function of average logarithmic cumulative return, the model did not optimize any risk but cumulative return only. Overall, in terms of both absolute return and relative return compared to UCRP, almost all proposed models generated their best results on Portfolio 1 which is a group of high beta stocks. It implies that CNN models are able to earn a higher profit when trading a high-risk portfolio. Since an aggressive strategy established by proposed model is easier to maximize its reward when all traded stocks swing more than the market.

5.5. Performance of proposed model integrating with price prediction

The methodology of integrating with price prediction model was tested. Different dependent factors defined in section 4.6 were conducted as well, it is a variable which determines the dependency on price prediction model.

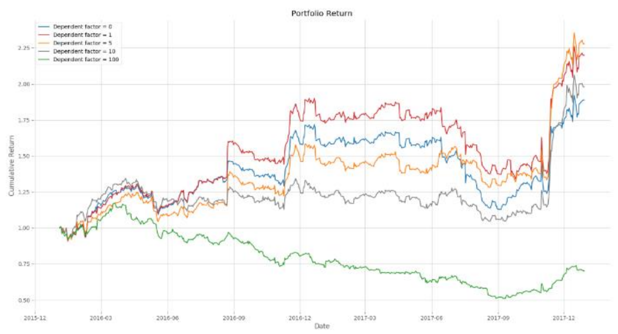

Figure 19. Portfolio 2: Cumulative return of proposed model integrating with price prediction

From the overall performance, it can be concluded that the final portfolio value could be improved with an appropriate dependency on price prediction. The dependent factor of 5 achieved the best cumulative return, whereas the performance degraded when dependent factor kept increasing. This is an example illustrating how integration of price prediction can enhance the overall profitability, nonetheless, it is also possible that the price prediction might worsen the overall profitability because of its poor accuracy especially in downturn period.

Therefore, the dependent factor should be varied smartly according to its past performance. For instance, the dependent factor should increase when its directional accuracy appears satisfied in last three months, and vice versa. This smart integration between proposed model and price prediction model is one of the future work to explore.

6. CONCLUSION

In present stage, the main contribution of this work is to evaluate the feasibility of adopting convolution layers, LSTM layers and fully connected layers for analyzing financial data.

This research built a portfolio management framework to produce a portfolio selection vector on a daily basis in order to optimize overall return. The procedures of this framework were as follows: First, a set of fundamental and technical indicators within a sliding window were normalized such that an input price matrix was constructed. Second, the input price matrix was computed by a policy network. Third, action from policy network was fed into reward function, and corresponding experience was added to replay buffer memory. Fourth, trainable parameters of policy network were updated by Deterministic Policy Gradient Algorithms (DPG) through experience replay.

The performance of different learning rates, cash bias, reward functions, models and portfolios was tested by four experiments. Firstly, the result indicated that learning rate and cash bias directly relates to the aggressiveness of trading strategy established by model. Secondly, reward function of Average logarithmic cumulative return achieved the best performance among all proposed reward functions. The reward functions of Sharpe ratio and Calmar ratio did not obtain any outstanding performance, thus, this model might not be suitable to be optimized by risk-adjusted return. Thirdly, convolution layers with the aid of LSTM layer or fully connected layer had a better performance than other models including all benchmarks. Furthermore, a price prediction model from other research was integrated with this model, and it demonstrated that the profitability could be improved by making use of price prediction result which was fed to voting score layer.

However, additional work is required for improving policy network and reward function. Most of the experimental settings which obtained a good performance in this research built a high-risk strategy which tends to distribute all capital to a single asset. It is a major limitation of this research as a slightly lower-risk strategy which holds two or more stocks are preferred. Although some reward function which was supposed to optimize risk-adjusted return was attempted, it could not create a low-risk strategy effectively.

ACKNOWLEDGMENT

I am honored to express my thanks of gratitude to my supervisor Dr. Vincent W. L. Tam, a responsible, patient and respectable scholar for his patient guidance, enthusiastic encouragement and expert advice of this project. Secondly, I heartily thank my groupmate Chow Hui Zhuo Garrett, a hard-working and motivated student from computer engineering. He helped me in finalizing my project within the limited time frame.

REFERENCE

[1] J. Wu, "Introduction to Convolutional Neural Networks," in *Nanjing University*, China, 2017.

[2] T. Aamodt, "Predicting Stock Markets with Neural Networks," in *University of Oslo*, 2015.

[3] Z. Jiang and J. Liang, "Cryptocurrency Portfolio Management with Deep Reinforcement Learning," in *Xi'an Jiaotong-Liverpool University*, 2016.

[4] Saud Almahdi and Steve Y. Yang, "An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown," *Expert Systems With Applications*, pp. 267-279, 2017.

[5] T. Schaul, J. Quan, I. Antonoglou and D. Silver, "PRIORITIZED EXPERIENCE REPLAY," *ICLR*, 2016.

[6] T. d. Bruin, J. Kober, K. Tuyls and R. Babuska, "The importance of experience replay database," Delft Center for Systems and Control Delft University of Technology, 2015.

[7] M. Volodymyr, K. Kavukcuoglu, S. David, V. Joel and A. Graves, "Human-level control through deep reinforcement," *NATURE*, p. 531, 2015.

[8] O. Jin and H. El-Saawy, "Portfolio Management using Reinforcement Learning," in *Stanford University*.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel and N. Heess, "CONTINUOUS CONTROL WITH DEEP REINFORCEMENT," *ICLR*, 2016.

[10] D. Silver and G. Lever, "Deterministic Policy Gradient Algorithms," *PMLR*, 2014.

[11] S. Qiu and B. Cai, "Flexible Rectified Linear Units for Improving Convolutional Neural Networks," in *South China University of Technology*, China, 2017.

[12] INVESTOPEDIA, "What is the ideal number of stocks to have in a portfolio?," INVESTOPEDIA, New York, NY, 7 2005. [Online]. Available: https://www.investopedia.com/ask/answers/05/optimalportfoliosize.asp. [Accessed 12 2017].

[13] J. Krohnfeldt, "3 Reasons Cash Is a Smart Position in Your Portfolio," INVESTOPEDIA, New York, NY, 7 2016. [Online]. Available: https://www.investopedia.com/articles/investing/072316/3-reasons-cash-smart-position-your-portfolio.asp. [Accessed 3 2018].

[14] A. Kalai and S. Vempalay, "Efficient Algorithms for Universal Portfolios," *Journal of Machine Learning Research*, pp. 423-440, 2002.

[15] D. Huang, J. Zhou and B. Li, "Robust Median Reversion Strategy for On-Line Portfolio Selection," *Twenty-Third International Joint Conference*, 2006.

[16] B. Li and S. C. H. Hoi, "On-Line Portfolio Selection with Moving Average Reversion," *International Conference on Machine Learning*, 2012.

[17] Bin Li, Peilin Zhao, Steven C. H. Hoi and Vivekanand Gopalkrishnan, "PAMR: Passive aggressive mean reversion strategy," *Mach Learn*, p. 221, 21 2 2012.

[18] Allan Borodin, Ran El-Yaniv and Vincent Gogan, "Can We Learn to Beat the Best Stock," University of Toronto, 2005.