Portfolio Construction based on Wavelet Neural Network and Deep Reinforcement Learning

Anran Wang, Jieding He

December 2019

Abstract

The application of machine learning methods in the financial field has become one of the hot topics, and the construction of investment portfolio is one of the key issues in the financial field. In this project, we use supervised learning and reinforcement learning to construct a portfolio respectively. In the supervised learning section, we combine wavelet decomposition with Long Short-Term Memory (LSTM) neural networks (WNN) to predict stock time series, and compare with the predicted value using FFT and original time series, where WNN has better prediction results. The mean-variance portfolio constructed based on the prediction results has a superior performance than the benchmark. In the reinforcement learning section, we use deep reinforcement learning (DRL) to build a single stock trading model that can get more profit than a passive position. Moreover, we also use a deep reinforcement learning framework that combines different structures of agent, state and reward to build a portfolio. The results show that the DRL using Multilayer Perceptrons, wavelet coefficients and Sharpe ratios performs better. The portfolios based on DRL also outperform benchmarks. The application of deep reinforcement learning in the financial field still has a lot of energy waiting to be tapped.

1 Introduction

Many professionals and researchers have focused on predicting stock prices for a long time. But thanks to non-stationary properties of stock market, the prediction of stock prices is always a challenging problem. In recent years, multiresolution theories have been successfully applied to financial time series study, such as wavelet transform. The wavelet transform is a signal processing technique that simultaneously analyzes the time domain and the frequency domain. Due to its powerful feature extraction capability, the wavelets can seize discontinuities, ruptures, and singularities in the original data.

Nowadays, machine learning techniques have been studied well and widely used in a lot of areas, such as classification, prediction, image recognition, etc. As a kind of recurrent neural network (RNN), long short-term memory (LSTM) has been proven to be very effective on financial time-series data because it is able to store past information, which is crucial in our project since the future price of a stock is based on its previous price.

Combining wavelet transform and LSTM, we believe we can accurately forecast stock prices. Using the future prices we predict as well as days of historical prices, we are able to construct a portfolio based on the mean-variance optimization (MVO) created by Markowitz in 1952. The MVO framework aims to accomplish security diversification by means of minimizing portfolio risk under determined expected return or maximizing expected return under decided portfolio variance, outputting optimal allocation of assets.

Not only do we want to build a portfolio based on supervised learning, we would also like to use reinforcement learning (RL) to construct a portfolio. As mentioned before, the wavelet neural network (WNN) gives the predicted stock prices. However, what if we do not do price predictions and do not learn the structure of the market implicitly but directly learning the policy of changing the weights dynamically in the continuously changing market, and that is how we come up with using RL to construct another portfolio. Moreover, we find it feasible to set neural networks as the agent ,which can enhance learning efficiency and performance, and combine it with RL algorithms into deep reinforcement learning (DRL). In addition, we will compare the performance of using different neural networks, different states and different reward functions, to demonstrate the feasibility and performance of using DRL to build a portfolio.

The structure of this report is as follows, in Section 3, we introduce the basics of supervised learning and reinforcement learning. In Section 4.1, we construct a portfolio based on WNN. In section 4.2 and 4.3, we establish a single stock trading model and a portfolio both based on a deep reinforcement learning method.

2 Literature review

Using WNN in finance, Luis Ortega and Khaldoun Khashanah proposed a wavelet neural network (neural wavelet) model for short-term stock returns in modern financial data in [11]. Luca Di Persio and Oleksandr Honchar proposed an artificial neural network (ANN) method to predict stock market indexes, especially in terms of predicting their up or down trends in [2]. Rania Jammazi and Chaker Alouicombined combined the dynamic characteristics of multi-layer back-propagation neural network and the recent Harr A wavelet decomposition, a hybrid model of HTW-MPNN was implemented to achieve outstanding prediction of crude oil prices in [3]. Wang, Huai-zhi et al proposed an advanced point prediction method based on wavelet transform and convolutional neural network is proposed for wind prediction in [12]. Salim Lahmiri presented a forecasting model that integrates the discrete wavelet transform (DWT) and backpropagation neural networks (BPNN) for predicting financial time series in [6]. Hui Liu, Xi-wei Mi and Yan-fei Lia proposed novel hybrid deep-learning wind speed prediction model, which combines the empirical wavelet transformation and two kinds of recurrent neural network in [8].

About deep reinforcement learning, DeepMind Technologies presented the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning in [9]. Pengqian Yu et al designed a deep reinforcement learning (RL) architecture with an autonomous trading agent such that, investment decisions and actions are made periodically, based on a global objective, with autonomy in [13]. Yue Deng et al tried to address this challenge by introducing a recurrent deep neural network (NN) for real-time financial signal representation and trading in [1]. Zhengyao Jiang and Jinjun Liang presented a model-less convolutional neural network with historic prices of a set of financial assets as its input, outputting portfolio weights of the set in [4]. Zhengyao Jiang et al also presented a framework consists of the Ensemble of Identical Independent Evaluators (EIIE) topology, a Portfolio-Vector Memory (PVM), an Online Stochastic Batch Learning (OSBL) scheme, and a fully exploiting and explicit reward function in [5]. John Moody proposed to train trading systems and portfolios by optimizing objective functions that directly measure trading and investment performance using recurrent reinforcement learning in [10]. Jae Won Lee proposed a method of applying reinforcement leaming, suitable for modeling and leaming various kinds of interactions in real situations, to the problem of stock price prediction in [7].

3 Theoretical framework

3.1 Supervised learning

3.1.1 Discrete wavelet transform

The wavelet transform is a mathematical method that allows transforming a given signal $x(t)$ into many frequency bands. More specifically, the signal $x(t)$ is decomposed into smooth coefficients s and detailed coefficients d, which are given by:

$$s_{j,k} = \int x(t)\phi_{j,k}(t)dt$$

$$d_{j,k} = \int x(t)\psi_{j,k}(t)dt$$

Where j and k are the scaling and translation parameters, respectively, and ϕ and ψ are the father and mother wavelets, respectively.

For the discrete wavelet transform, the decomposition is defined on a time domain $t = 0, 1, 2, \ldots, N-1$, where N is the number of samples in time series. In our project, we mainly utilize the Haar wavelet. The Haar father function is defined as:

$$\phi(2^{-j}t) = \begin{cases} 1, & [0, 2^j) \\ 0, & otherwise \end{cases}$$

where $j \subset N$. It is obvious that the Haar father function is a continuous pulse with a value of 1 on interval $[0, 2^j)$. Also, the Haar mother function is defined as:

$$\psi(t) = \begin{cases} 1, & [0, \frac{1}{2}) \\ -1, & [\frac{1}{2}, 1) \\ 0, & otherwise \end{cases}$$

And the Haar mother wavelet generates a set Ψ of functions $\psi_{j,k}$ that are given by

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k)$$

Moreover, the critical resolution level J, also known as the decomposition level, represents the degree of decomposition. The max level of J depends on the length of time series, the N_{ts} should be divisible by $2^{J_{max}}$.

3.1.2 Fast Fourier transform

To interpret why we use discrete wavelet transform, we consider comparing DWT with fast Fourier transform (FFT). Before elucidating FFT, we first introduce discrete Fourier transform (DFT). DFT is a kind of Fourier transform, which transforms a sequence of N equally-spaced samples, x_n, to a complex-valued sequence of frequency, X_k. The definition is:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi kn}{N}}$$

where $k = 0, 1, \ldots, N-1$, x_n is a sequence of N equally-spaced samples, i is the imaginary unit, X_k is the complex-valued sequence of frequency. After some mathematical deformation, we obtain:

$$X_k = \sum_{n=0}^{N-1} x_n cos(-b_n) + i \sum_{n=0}^{N-1} x_n sin(-b_n) = A_k + iB_k$$

where A_k and B_k are, respectively, the real coefficient and the imaginary coefficient.

The definition of FFT is the same as that of DFT, but the output of FFT depends on the parity of N instead of from X_0 to X_{N-1}. In particular, the output of FFT is:

$$X_0, X_1, \ldots, X_{\frac{N}{2}}, X_{1-\frac{N}{2}}, \ldots, X_{-1}, \quad N \quad is \quad even$$

$$X_0, X_1, \ldots, X_{\frac{N-1}{2}}, X_{1-\frac{N-1}{2}}, \ldots, X_{-1}, \quad N \quad is \quad odd$$

3.1.3 Long Short-term Memory

In order to construct Wavelet Neural Network, we need to implement Long Short-term Memory (LSTM) in addition to Wavelet Transform. LSTM is a kind of recurrent neural networks (RNN) and extends the memory of RNN. It can not only process single data points but also entire sequences of data.

Figure 1 displays a part of a LSTM chain that has three types of gates, which are Forget gate, Input gate and Output gate. We first use the forget gate to decide how much information we want to keep, next the input gate will update the old cell state into new cell state, then the output gate will decide what parts of the cell state we are going to output and put the cell state through *tanh* layer, and we get output.

The first step is to decide what information we are going to throw away. This decision is made by a *Sigmoid* layer called the "forget gate layer." It looks at h_{t-1} and x_t , and outputs a number between 0 and 1. 1 means "completely keep this" while 0 means "completely throw away this". The corresponding formula is:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

where

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

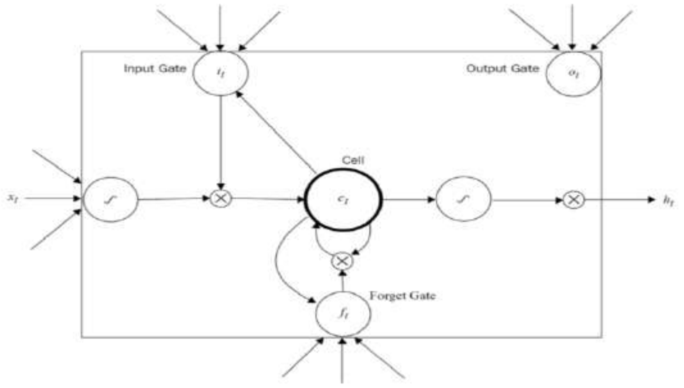


Figure 1: LSTM structure

W, b, $[h_{t-1}, x_t]$ are weights, biases and concatenation of h_{t-1} and x_t respectively.

The next step is to decide what new information we are going to store. First, a *Sigmoid* layer called the "input gate layer" decides which values we will update. Next, a tanh layer creates a vector of new candidate values, $\tilde{C_t}$. The corresponding formulas are:

$$i_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$\tilde{C_t} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

where

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

And the third step is to update the old cell state C_{t-1} into new cell state C_t. We multiply the old state by f_t to forget things, then we add $i_t \cdot \tilde{C_t}$, which is the new candidate value. scaled by how much we decided to update each state value. The corresponding formula is:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C_t}$$

Finally, we need to decide what we are going to output. First, we run a *Sigmoid* layer which decides what parts of the cell state we are going to output. Then, we put the cell state through *tanh* layer. And multiply it by the output of the *Sigmoid* layer. Mathematically, we have:

$$\sigma_t = \sigma(W_0 \cdot [h_{t-1}, x_t] + b_0)$$

$$h_t = o_t \cdot tanh(C_t)$$

3.1.4 Mean-variance analysis

Mean-variance analysis, or modern portfolio theory, first introduced by Harry Markowitz in 1952, is a mathematical framework for congregate a portfolio of securities so that the expected return of the portfolio is maximized for a given level of risk, or the volatility of the portfolio is minimized for a given expected return. In our project, we try to realize the latter. Specifically, we need to construct a portfolio such that:

$$Min \quad \frac{1}{2}\omega^T \Sigma \omega$$

subject to

$$\omega^T E = E(r_p)$$
$$\omega^T e = 1$$

where $\omega^T = (\omega_1, \omega_2, ..., \omega_n)$ is the weight vector of assets. Σ is the covariance matrix of assets. $E = [E(r_1), E(r_2), ..., E(r_n)]^T$ is the expected return vector. $e = (1, 1, ..., 1)^T$ is the n-vector of ones.

3.2 Deep reinforcement learning

3.2.1 Reinforcement learning

Reinforcement learning is an another area in machine learning that focuses on how to act on the environment to achieve the maximum expected benefits.

Reinforcement learning is different from supervised learning. Supervised learning is learning from a training set of labeled examples provided by a knowledgeable external supervisor. Each example is a combination set consists of the situations and the labels which the situations should belong to. The object of supervised learning is to extrapolate the responses so that it acts correctly in situations not present in the training set.

Reinforcement learning is also different from unsupervised learning, which is typically about finding structure undiscovered relationship between the unlabeled data.

Whereas, reinforcement learning is a vital way to achieve artificial intelligence. The learning process is very similar to human learning. In particular, the rise of deep neural networks in recent years, deep reinforcement learning combined with deep learning and reinforcement learning indicates a new direction for real artificial intelligence.

The key elements of reinforcement learning are: environment, reward, action, and state. With these elements we can build a reinforcement learning model. The problem that reinforcement learning solves is to get an optimal policy for a specific problem, so that the reward obtained under this policy is

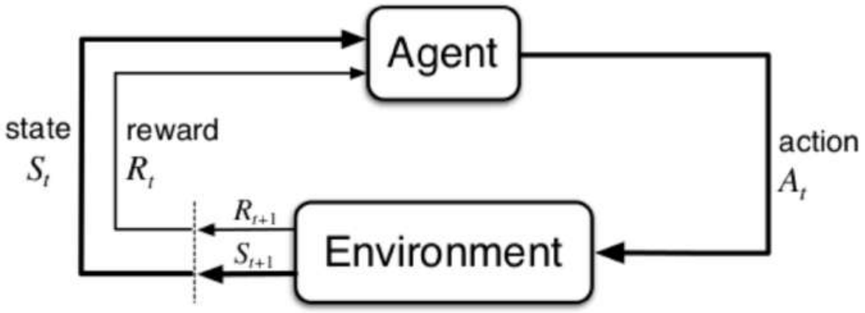


Figure 2: Reinforcement learning structure

the largest. The so-called policy is actually a series of actions.

Reinforcement learning can be described by the following figure, which is to extract an environment from the task to be completed first, and abstract the state, action, and instant rewards for performing the action.

At each step t the agent:

- Receives state S_t

- Receives scalar reward R_t

- Executes action A_t

And the environment:

- Receives action A_t

- Emits state S_{t+1}

- Emits scalar reward R_{t+1}

Algorithms for reinforcement learning, such as Q-Learning, limited to both the action space and sample space, and generally in discrete situations. However, more complex tasks that are closer to the actual situation often have a large state space and continuous action space. When the input data are images and sounds, they often have high dimensions, and traditional reinforcement learning is difficult to handle that.

3.2.2 Deep reinforcement learning

After DeepMind published 'Playing Atari with Deep Reinforcement Learning' in 2013, deep reinforcement learning slowly entered people's field of vision. Deep reinforcement learning combines the perception of deep learning with the decision-making ability of reinforcement learning.

In deep reinforcement learning, deep neural networks as the agent replace the Q-table in traditional reinforcement learning, enabling deep reinforcement learning to handle complex and high-dimensional environments.

The basic structure of deep reinforcement learning is similar to reinforcement learning, except that during training, traditional reinforcement learning is by changing the value of the Q-table, and deep reinforcement learning is by training the weights of the deep neural network as an agent.

There are multiple algorithms for deep reinforcement learning, such as DQN, DDPG and Actor Critic.

3.2.3 Actor-Critic algorithm

The Actor-Critic algorithm consists of two parts: actor and critic. Where Actor is the Policy Gradient algorithm and Critic is Q-learning. So in fact, the actor-critic algorithm is a combination of Q-learning algorithm and policy gradient algorithm.

The value function of the policy of Critic is same as Q-Learning, which is:

$$V_\pi(s) = E_\pi[r + \gamma V_\pi(s\prime)]$$

The action value function of the policy is:

$$Q_\pi(s,a) = R_s^a + \gamma V_\pi(s\prime)$$

The advantage function A, which indicates how good the choice action a is in state s. If action a is better than average, then the advantage function is positive; otherwise, it is negative. So that the A should be:

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s) = r + \gamma V_\pi(s\prime) - V_\pi(s)$$

The Actor, which should use Policy Gradient theorem:

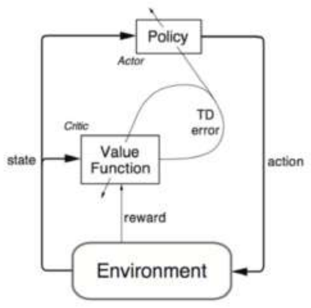
$$\nabla_\theta J(\theta) = \sum_{s \subset S} d(s) \sum_{a \subset A} \pi_\theta(s,a) \nabla_\theta \log \pi(a|s;\theta) Q_\pi(s,a)$$

Use $A_\pi(s,a)$ replaces $Q_\pi(s,a)$:

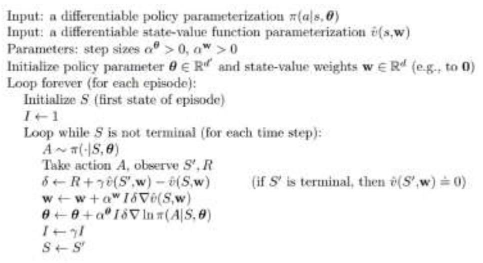
$$\nabla_\theta J(\theta) = \sum_{s \subset S} d(s) \sum_{a \subset A} \pi_\theta(s,a) \nabla_\theta \log \pi(a|s;\theta) A_\pi(s,a)$$

The update formula should be:

$$\theta_{t+1} \leftarrow \theta_t + \alpha A_{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(s,a)$$



(a) Structure



```
Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
    Initialize $S$ (first state of episode)
    $I \leftarrow 1$
    Loop while $S$ is not terminal (for each time step):
        $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
        Take action $A$, observe $S', R$
        $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$        (if $S'$ is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} I\delta\nabla\hat{v}(S,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I\delta\nabla\ln\pi(A|S,\boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$
```

(b) Algorithm

Figure 3: Actor-Critic

3.2.4 Combine with WNN

In deep reinforcement learning, observation and reward are used as inputs for deep neural network agents. Therefore, when dealing with time series problems, we can combine wavelet decomposition into the structure of deep reinforcement learning. The coefficient obtained by wavelet decomposition of the original time series is used instead of the original sequence as an observation of deep reinforcement learning. Similarly, the Fourier transform can also be combined with deep reinforcement learning. This way will give our deep reinforcement learning brain, which is the agent, the ability of observing the characteristics of time series on different time and space scales.

4 Experiment design and results

4.1 Wavelet neural network portfolio

4.1.1 Model construction

The wavelet neural network portfolio based on a supervised learning method consists of 50 stocks and each stock has a weight ω_i:

$$\sum_{i=1}^{50} = 1, \quad |\omega_i| \leq 0.1$$

Since the absolute value of each ω_i is no more than 0.1, whether we long or short stocks, we cannot exceed ten percent of the principal. And the structure of portfolio is that we first use Long short-term memory (LSTM) combined with Discrete Wavelet Transform (DWT) to predict the total 50 stock prices, then implement mean-variance optimization to construct the portfolio.

We use $J = 2$ Haar wavelet to decompose each stock daily close price time series into one smooth coefficient s and two detailed coefficients d_1 and d_2. Use

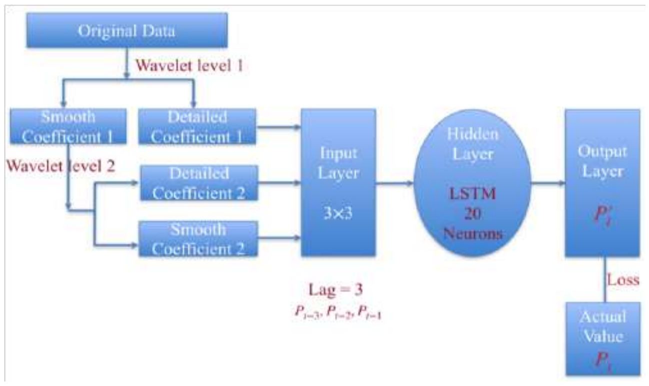
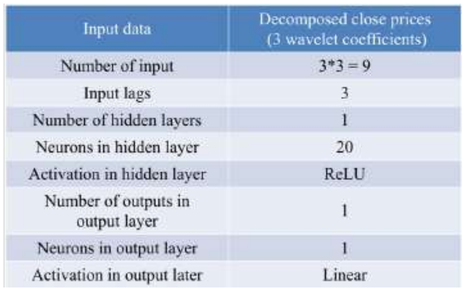


Figure 4: Wavelet Neural Network



Input data	Decomposed close prices (3 wavelet coefficients)
Number of input	3*3 = 9
Input lags	3
Number of hidden layers	1
Neurons in hidden layer	20
Activation in hidden layer	ReLU
Number of outputs in output layer	1
Neurons in output layer	1
Activation in output later	Linear

Figure 5: LSTM detailed structure

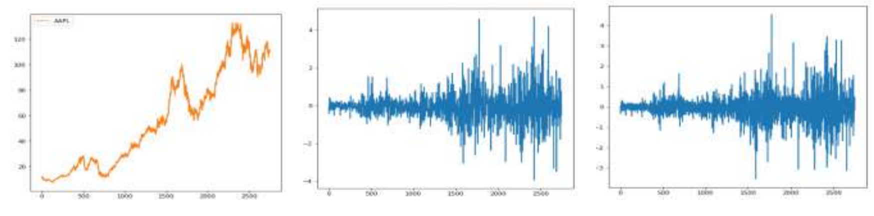
the total 3 coefficients instead of original time series as the input of a neural network to predict the next day's price. And the time lag equals 3. So that the input size of neural network should be 3×3 and the output is p_{t+1}. The Wavelet Neural Network (WNN) topology is shown in figure 4.

Moreover, we want to compare the different results between using wavelet coefficients, FFT coefficients and the original time series. The other two method compared with WNN only change the input of the neural network.

For neural network, here we use LSTM which is appropriate for a time series. The detailed structure of LSTM we used shown in figure 5.

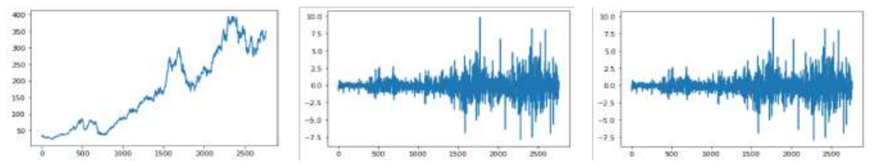
4.1.2 Results

Our data contain the daily close price time series of 50 stocks chosen from S&P500. The training set of this part is from 01/03/2006 to 12/30/2016, daily close price of 2768 trading days. The test set is 700 trading days from 01/03/2017.

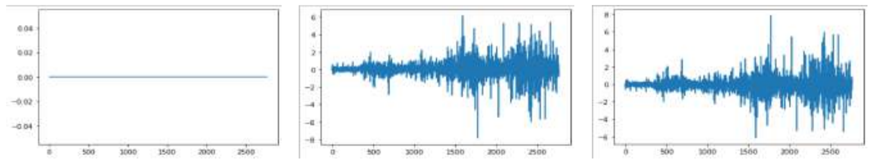


(a) Smooth (b) Detailed 1 (c) Detailed 2

Figure 6: AAPL wavelet coefficients



(a) Real 1 (b) Real 2 (c) Real 3



(d) Img 1 (e) Img 2 (f) Img 3

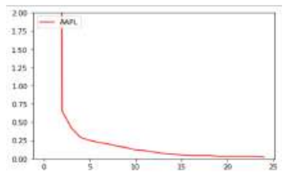
Figure 7: AAPL FFT coefficients

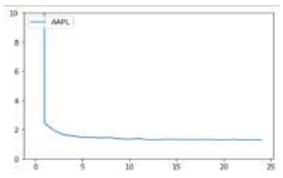
The key to creating a portfolio using supervised learning is accurate predictions of time series.We choose AAPL as an example to compare the prediction results using WNN, FFT-LSTM and Price-LSTM.

We use $J = 2$ Haar wavelet decomposition to AAPL time series. We can see 1 smooth coefficient and 2 datailed coefficients in figure 6. It can be seen that the smoothing coefficient is a sequence extracted after denoising the original sequence, which basically retains the trend of the original sequence. The detail coefficients are small fluctuation of the original sequence. The larger the order, the smaller the fluctuation.

Similarly, we also decompose the AAPL time series by FFT to get coefficients. In this experiment we use a lag order of 3, which means we focus on the data from the past three days. So using a 3rd-order FFT to decompose the original sequence to get three real coefficients and three imaginary coefficients, shown in figure 7.

We will train the neural network on 50 stocks separately, so there will be a total of 50 neural network models that predict the closing price of $t + 1$ for


(a) WNN


(b) FFT-LSTM

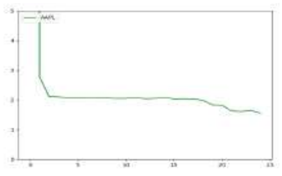
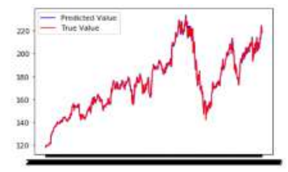
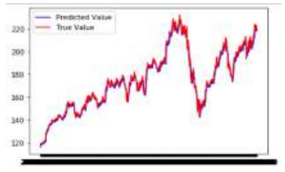

(c) Price-LSTM

Figure 8: AAPL training loss


(a) WNN


(b) FFT-LSTM

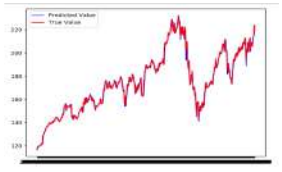

(c) Price-LSTM

Figure 9: AAPL prediction results

each stock. Recall that we use a $lag = 3$, so that the input of the WNN for each stock will be a 3×3 vector, FFT-LSTM will be a 2×3 vector, and using original price time series will be a 3×1 vector.

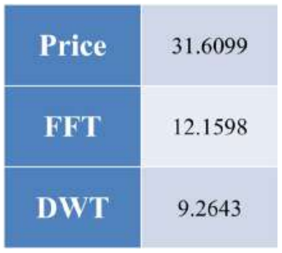
We first compare the losses during training of the three models, still using AAPL as an example. As can be seen from the figure 8, training using FFT-LSTM has the fastest convergence but maintain the training loss at around 1.5. WNN also converges fast, and can converge to a loss close to 0 after 20 epochs. Using original time series has the worst performance, which can not converge after 25 epochs and the training loss is still high, around 2.

We use the trained models to generate predictions on the test set and compare it with the true value. Use AAPL as example and shown in figure 9. We see that the predicted values of the three methods are very close to the real values. To be more explicit, we calculated mean squared error (MSE) as shown in figure 10:

$$MSE = E((P_{prediction} - P_{true})^2)$$

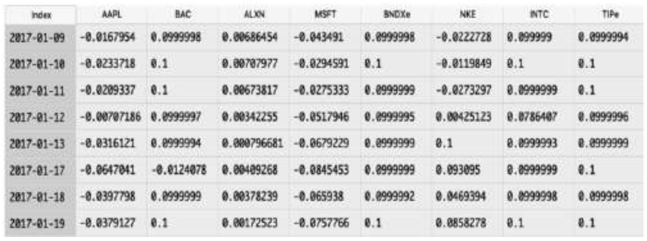
Same as intuition, there is a minimal MSE using WNN. FFT-LSTM is next, Price-LSTM is the worst. Therefore, we conclude that WNN is the best method to predict the stock time series.

Based on the predictions obtained on the test set using WNN, we can construct a mean-variance portfolio based on WNN. We set the annual target return equals 20%, which convert to daily let the $E(r_p) = 4.99 \times 10^{-4}$. And the absolute value of weight for each stock cannot over 0.1. This nonlinear programming should be:



Price	31.6099
FFT	12.1598
DWT	9.2643

Figure 10: MSE



Index	AAPL	BAC	ALXN	MSFT	BNDXe	NKE	INTC	TIPe
2017-01-09	-0.0167954	0.0999998	0.00686454	-0.043491	0.0999998	-0.0222728	0.099999	0.0999994
2017-01-10	-0.0233718	0.1	0.00707977	-0.0294591	0.1	-0.0119849	0.1	0.1
2017-01-11	-0.0209337	0.1	0.00673817	-0.0275333	0.0999999	-0.0273297	0.0999999	0.1
2017-01-12	-0.00707186	0.0999997	0.00342255	-0.0517946	0.0999995	0.00425123	0.0786407	0.0999996
2017-01-13	-0.0316121	0.0999994	0.000796681	-0.0679229	0.0999999	0.1	0.0999993	0.0999999
2017-01-17	-0.0647041	-0.0124078	0.00409268	-0.0845453	0.0999999	0.093095	0.0999999	0.1
2017-01-18	-0.0397798	0.0999999	0.00378239	-0.065938	0.0999992	0.0469394	0.0999998	0.0999998
2017-01-19	-0.0379127	0.1	0.00172523	-0.0757766	0.1	0.0858278	0.1	0.1

Figure 11: Daily weights fragment

$$Min \quad \frac{1}{2}\omega^T \Sigma \omega$$

subject to

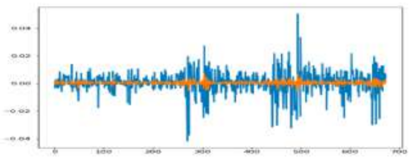
$$\omega^T E = 4.99 \times 10^{-4}$$

$$\omega^T e = 1$$

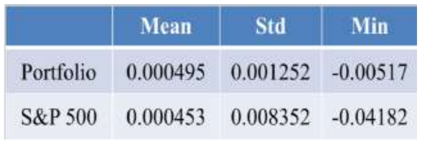
$$|\omega_i| \leq 0.1$$

Use a optimizer for this quadratic programming problem, we get daily weights for each stock. Based on these weights, we calculated the daily returns of the portfolio and compare it to the S&P500 over the same period.

It is learned from the figure 12 that because WNN method has good predictions for all 50 stocks, our mean-variance portfolio can very well reach our



(a) Returns time series



	Mean	Std	Min
Portfolio	0.000495	0.001252	-0.00517
S&P 500	0.000453	0.008352	-0.04182

(b) Returns statistics

Figure 12: Returns

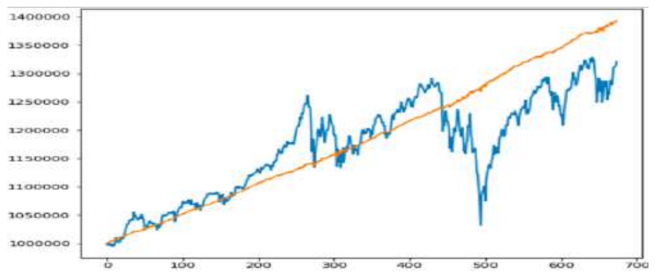


Figure 13: Balance values

preset goals. Portfolios can have higher returns and less risk than S&P500. If we invest one million in our portfolio and spy at the same time, as shown in figure 13, our portfolio can have a steadily rising balance value and can avoid fluctuations and losses when the entire market falls.

4.2 Single stock trading

4.2.1 Model construction

We want to test the performance of deep reinforcement learning, so we designed experiments to trade on a single stock sequence.

AAPL is selected for a single stock sequence, and the training set is 2768 trading days daily price data from January 3, 2016 to December 30, 2016. The test set are 600 days from January 3, 2017.

In the construction of deep reinforcement learning structure, we chose a two-layer, multi-layer perceptron(MLP) with 64 neurons as the agent. A MLP is a simple, forward-propagating, fully-linked artificial neural network. Because we want to simply test the performance of deep neural networks on time series problems, a simpler MLP is more suitable.

We set the action generated by the agent to be a two-dimensional vector. The first dimension is a continuous value from 0 to 3, and the second dimension is a continuous value from 0 to 1. The first dimension represents the direction of the transaction issued by the agent at the current moment. 0 to 1 represents buying, 1 to 2 represents selling, and 2 to 3 represents holding. The second dimension represents the ratio of the number of transaction shares to the maximum number of shares that can be traded by all the current funds. That is, the action:

$$a = (x_1, x_2)$$

where

$$x_1 = \begin{cases} [0,1), & \text{Buy} \\ [1,2), & \text{Sell} \\ [2,3], & \text{Hold} \end{cases}$$

$$x_2 = \frac{Vol_{transaction}}{Vol_{max}}$$

At the same time, the training environment is the price data of 2768 trading days. The environment will gives us the current state, which is the observation. Here we divide the observations into two types. The first is the open, max, min, close and adjusted close price for each of the past five days at the current moment, a 5×5 vector:

$$Observation_1 = \begin{pmatrix} p_{open,t-5} & p_{max,t-5} & p_{min,t-5} & p_{close,t-5} & p_{adj.close,t-5} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{open,t-1} & p_{open,t-1} & p_{open,t-1} & p_{open,t-1} & p_{open,t-1} \end{pmatrix}$$

Whereas the second type observation is the $J = 2$ Haar wavelet decomposition coefficients of each element in the first type, which will give us one smooth coefficients and two detailed coefficients for each element in $Observation_1$. So that the $Observation_2$ is a $5 \times 5 \times 3$ vector:

$$Observation_2 = \begin{pmatrix} \begin{bmatrix} Wf_{s,open,t-5} \\ Wf_{d1,open,t-5} \\ Wf_{d2,open,t-5} \end{bmatrix} & \cdots & \begin{bmatrix} Wf_{s,adj.close,t-5} \\ Wf_{d1,adj.close,t-5} \\ Wf_{d2,adj.close,t-5} \end{bmatrix} \\ \vdots & & \vdots \\ \begin{bmatrix} Wf_{s,open,t-1} \\ Wf_{d1,open,t-1} \\ Wf_{d2,open,t-1} \end{bmatrix} & \cdots & \begin{bmatrix} Wf_{s,adj.close,t-1} \\ Wf_{d1,adj.close,t-1} \\ Wf_{d2,adj.close,t-1} \end{bmatrix} \end{pmatrix}$$

The environment will give a reward based on the current action a. We define reward function:

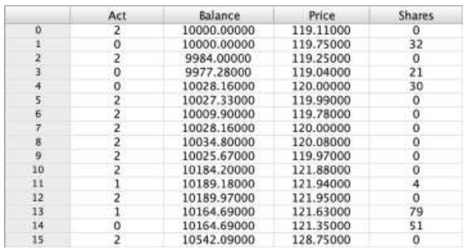
$$r = V_{balance} \times Mod$$

where

$$Mod = \frac{t_{current}}{t_{total}}$$

The $V_{balance}$ is the balance value at current time.

We use Actor-Critic algorithm for training the weights of the agent – MLP neural network.



	Act	Balance	Price	Shares
0	2	10000.00000	119.11000	0
1	0	10000.00000	119.75000	32
2	2	9984.00000	119.25000	0
3	0	9977.28000	119.04000	21
4	0	10028.16000	120.00000	30
5	2	10027.33000	119.99000	0
6	2	10009.90000	119.78000	0
7	2	10028.16000	120.00000	0
8	2	10034.80000	120.08000	0
9	2	10025.67000	119.97000	0
10	2	10184.20000	121.88000	0
11	1	10189.18000	121.94000	4
12	2	10189.97000	121.95000	0
13	1	10164.69000	121.63000	79
14	0	10164.69000	121.35000	51
15	2	10542.09000	128.75000	0

Figure 14: Trading action segment

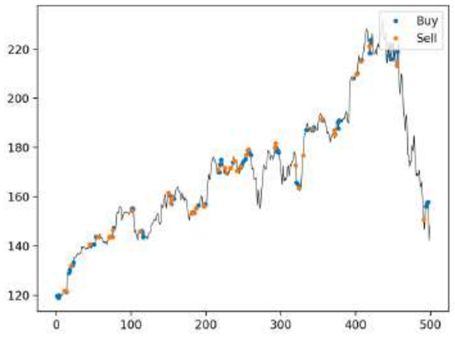


Figure 15: Transaction points

4.2.2 Results

We use the trained agent to run on the test set which is are 600 daily price data from January 3, 2017.

The trained agent will generate actions at each moment according to the environment of the test set. We can show actions as time series. It can be seen that the action will generate a two-dimensional vector according to the environment. The first dimension is shown in Figure 14 as the Act, where 0 is a buy, 1 is a sell, and 2 is a hold. The result of the ratio represented by the second dimension is shown in the figure as shares.

In Figure 15, we can directly observe the point at which the agent transacted throughout the test cycle. Unlike human traders, the trained agent will maximize the value of the account in this experiment according to the training it receives. In general, the agent of deep reinforcement learning does have some ability to judge whether it is the time to buy or sell.

Because the ultimate goal of this problem is to maximize the balance value of the investment account. We compare the balance value of using the agent of deep reinforcement learning with the balance value of holding shares without

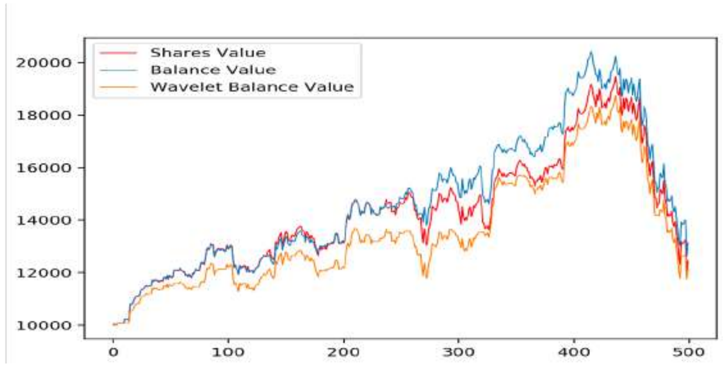


Figure 16: Balance value

trading, which passive management, shown in figure 16. We can see that the account represented by the agent using the $Observation_1$ can have a higher balance value when the stock price rises than the account that only holds. However, using the agent can not feel the decline in the stock price in the later period and immediately clear the position to prevent losses, resulting in holding a certain position when the decline, and ultimately the same as passive holding.

We also compared the accounts of the agents using the $Observation_2$. It can be found that using wavelet coefficients as an observation does not bring good results in this experiment, even the balance value is not as good as passive holding.

4.3 Deep reinforcement learning portfolio

4.3.1 Model structure of deep reinforcement learning portfolio

Now we start to focus on how to use deep reinforcement learning to construct a portfolio. When a human investment manager faces a portfolio management problem, he will uses the information obtained to determine the weight of each stock in the portfolio. Therefore, in a deep reinforcement learning portfolio, our agent should be able to generate actions by sensing the current state given by the environment.

Therefore, we set the action a generated by the agent in the deep reinforcement learning portfolio as a weight vector of the portfolio at this moment t:

$$a_t = (w_1, w_2, \cdots, w_{50}), \qquad 0 \le w_i \le 1$$

Please keep in mind that our stock pool is the same as the previous mean-variance portfolio, which is still 50 stocks chosen from the S&P500 constituents.

In this problem, we want to compare the impact on the final result by giving different components of deep reinforcement learning different characteristics. So first, we use two different neural networks as agents, one is a Multilayer Perceptron, and the other is Long Short-Term Memory. Both neural networks use three hidden layers with 512, 256, and 128 neurons, respectively.

In state, or observation, we also define two different features. One is the daily close price of 50 stocks for the past five days, which is a 5×50 vector. The other is the $J = 2$ Haar wavelet decomposition coefficients of $Observation_1$, which is a $5 \times 50 \times 3$ vector:

$$Observation_1 = \begin{pmatrix} p_{t-5,1} & p_{t-5,2} & \cdots & p_{t-5.50} \\ \vdots & \vdots & & \vdots \\ p_{t-1,1} & p_{t-1,2} & \cdots & p_{t-1,50} \end{pmatrix}$$

$$Observation_2 = \begin{pmatrix} \begin{bmatrix} Wf_{s,t-5,1} \\ Wf_{d1,t-5,1} \\ Wf_{d2,t-5,1} \end{bmatrix} & \cdots & \begin{bmatrix} Wf_{s,t-5,50} \\ Wf_{d1,t-5,50} \\ Wf_{d2,t-5,50} \end{bmatrix} \\ \vdots & & \vdots \\ \begin{bmatrix} Wf_{s,t-1,1} \\ Wf_{d1,t-1,1} \\ Wf_{d2,t-1,1} \end{bmatrix} & \cdots & \begin{bmatrix} Wf_{s,t-1,50} \\ Wf_{d1,t-1,50} \\ Wf_{d2,t-1,50} \end{bmatrix} \end{pmatrix}$$

Finally, our reward function is also divided into two types. The first is the return of portfolio at time t:

$$R_{1,t} = a_t \cdot r_t$$

where

$$r_t = (\frac{p_{1,t}}{p_{1,t-1}}, \cdots, \frac{p_{50,t}}{p_{50,t-1}})$$

The second is the Sharpe ratio of the for the past 5 days at time t:

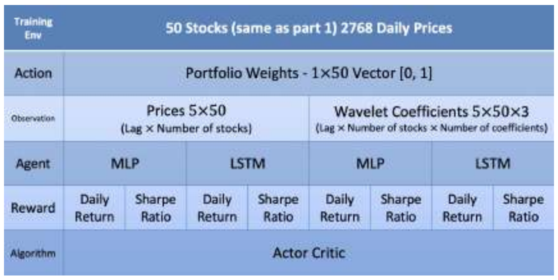
$$R_{2,t} = \frac{\mu_t}{\sigma_t}$$

where

$$\mu_t = \overline{(R_{1,t}, \cdots, R_{1,t-4})}$$

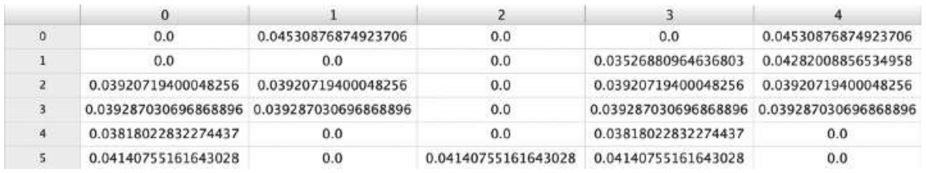
$$\sigma_t = SD(R_{1,t}, \cdots, R_{1,t-4})$$

Therefore, our deep reinforcement learning portfolio will have 8 different structures. The training algorithm is still Actor Critic.



Training Env	50 Stocks (same as part 1) 2768 Daily Prices							
Action	Portfolio Weights - 1×50 Vector [0, 1]							
Observation	Prices 5×50 (Lag × Number of stocks)				Wavelet Coefficients 5×50×3 (Lag × Number of stocks × Number of coefficients)			
Agent	MLP		LSTM		MLP		LSTM	
Reward	Daily Return	Sharpe Ratio	Daily Return	Sharpe Ratio	Daily Return	Sharpe Ratio	Daily Return	Sharpe Ratio
Algorithm	Actor Critic							

Figure 17: Model structure



	0	1	2	3	4
0	0.0	0.04530876874923706	0.0	0.0	0.04530876874923706
1	0.0	0.0	0.0	0.03526880964636803	0.04282008856534958
2	0.03920719400048256	0.03920719400048256	0.0	0.03920719400048256	0.03920719400048256
3	0.039287030696868896	0.039287030696868896	0.0	0.039287030696868896	0.039287030696868896
4	0.03818022832274437	0.0	0.0	0.03818022832274437	0.0
5	0.04140755161643028	0.0	0.04140755161643028	0.04140755161643028	0.0

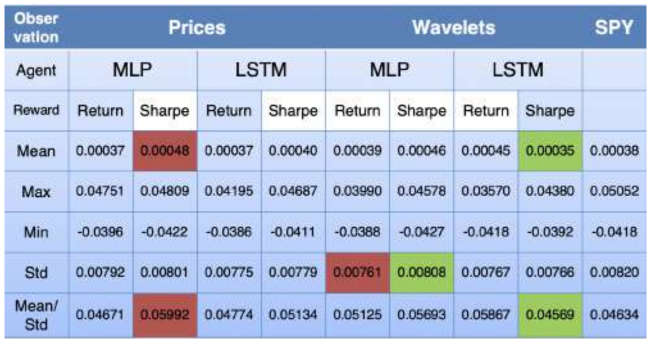
Figure 18: Portfolio weights segment

4.3.2 Results

We test the trained agent on the test set. The test set is 600 trading days from January 3, 2017. The trained agent will generate actions according to the environment, that is, the weights of the portfolio at each time point t.

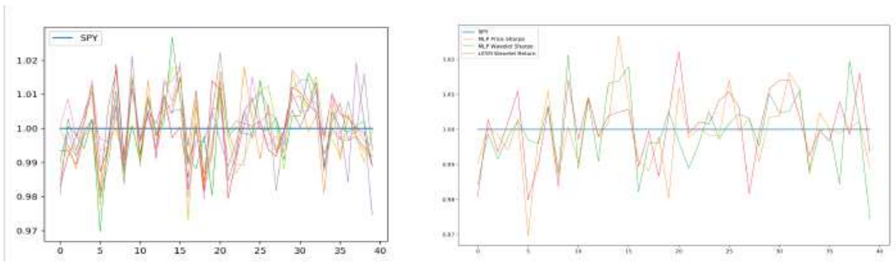
Through the daily weights generated by the agents in these 600 days, we can get the daily return time series of different portfolios. Calculate their statistics and compare them with SPY as a benchmark over the same period in figure 19.

We can see that five of the deep reinforcement learning portfolios have higher



Observation	Prices				Wavelets				SPY
Agent	MLP		LSTM		MLP		LSTM		
Reward	Return	Sharpe	Return	Sharpe	Return	Sharpe	Return	Sharpe	
Mean	0.00037	0.00048	0.00037	0.00040	0.00039	0.00046	0.00045	0.00035	0.00038
Max	0.04751	0.04809	0.04195	0.04687	0.03990	0.04578	0.03570	0.04380	0.05052
Min	-0.0396	-0.0422	-0.0386	-0.0411	-0.0388	-0.0427	-0.0418	-0.0392	-0.0418
Std	0.00792	0.00801	0.00775	0.00779	0.00761	0.00808	0.00767	0.00766	0.00820
Mean/Std	0.04671	0.05992	0.04774	0.05134	0.05125	0.05693	0.05867	0.04569	0.04634

Figure 19: Portfolio returns



(a) All portfolios

(b) Top 3 returns

Figure 20: 15-day returns compared with SPY

average daily returns than SPY. The combination of $observation_1$, MLP agent and R_2 has the highest average return. All portfolios have less volatility than SPY. If we used $\frac{\mu}{\sigma}$ as an indicator, 7 portfolios are better than SPY.

By fixing other variables, we can calculate the average daily return of the portfolios using MLP agent is 4.25×10^{-4}, using LSTM agent is 3.93×10^{-4}. Using $Observation_1$ is 4.05×10^{-4}, using $Observation_2$ is 4.13×10^{-4}. And using R_1 is 3.95×10^{-4}, using R_2 is 4.23×10^{-4}. On the whole, we can say that using MLP as the agent, wavelet coefficients as the state and Sharpe ratio as the reward function may get better results.

We want to focus on the relative relationship between portfolio returns time series and the SPY. So we calculated $\frac{r_{p,t}}{r_{spy,t}}$. In the figure 20(a), we can see that the fifteen-day returns of the portfolios are oscillating relative to SPY. The returns of the portfolios are not stable compared to the SPY. Moreover, we can see that the returns of the eight different structured portfolios are highly correlated in the early stage, but divergences start to appear in the later stage. But the overall fluctuations relative to spy are still roughly the same.

We also only select the three portfolios with the best performance, that is, the highest average returns, which are MLP agent, $Observation_1$ and R_2, MLP agent, $Observation_2$ and R_2 and LSTM agent, $Observation_2$ and R_1. By comparing the returns of these three portfolios with SPY, shown in figure 20(b), we find that the returns of the deep reinforcement learning portfolio are not as good as spy in the first 200 days, but the performance of the returns has improved afterwards.

In order to be more intuitive, let's assume that we invest 10,000 in the three best portfolios and the SPY in the beginning separately, and compare their balance value. As can be seen from the figure 21, the balance values of the three portfolios in the first 200 days are not comparable to SPY. But it started to improve after 200 days and reached the largest differences at 570 day. The

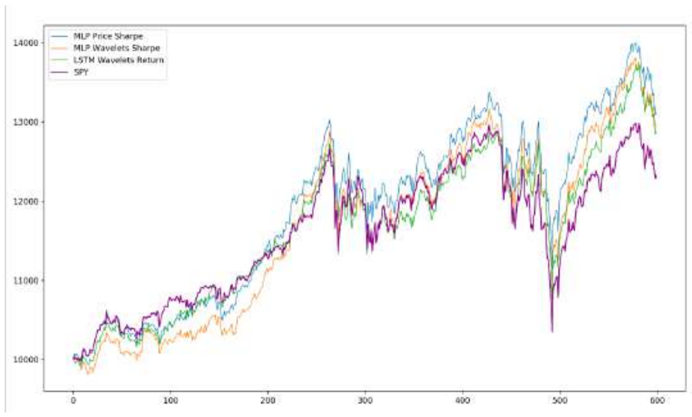


Figure 21: Portfolio balance values compared with SPY

best portfolios have nearly 9,000 additional returns compared to SPY. But after 570 days as the SPY fall, the values of the portfolios also begin to fall. And on the whole, the balance values of the deep reinforcement learning portfolios are highly correlated with SPY, and cannot avoid the decline of the overall market. But because our stock pool is all constituent stocks of S&P500 and will not be updated, our portfolio is not seeking for alpha.

5 Conclusion

In the section on building portfolios using supervised learning, we found that using wavelet decomposition combined with neural networks (we use LSTM) has a good effect in predicting time series. And compared with the use of FFT and original time series, there are higher accuracy and smaller errors. Based on the accurate time series prediction results provided by WNN, our mean-variance portfolio constructed using 50 S&P500 constituent stocks has a good performance. Compared with S&P500, the investment portfolio has higher yield and lower risk. WNN's ability in stock time series prediction is yet another way to build a portfolio based on supervised learning.

In the construction of a single stock trading model based on deep reinforcement learning, we found that the deep neural network agent of deep reinforcement learning can effectively use the preset reward function and states to train a transaction commander that can automatically make decisions based on the current state. And in terms of trading performance, it has the ability to surpass passive positions.

In building a portfolio using deep reinforcement learning, We use different structures in different components of deep reinforcement learning. By comparing the yield results, it is shown that MLP as an agent, wavelet coefficient as state, and sharpe ratio as reward have advantages in such problems. The ability of deep reinforcement learning portfolios to surpass benchmarks, but returns are not stable. The best investment portfolio in different portfolios can surpass SPY as the benchmark over the same period in balance value, but has limited advantages. But keep in mind that the construction of deep reinforcement learning structures is much more complicated than a supervised learning with labels. Many factors need to be improved and our model is still rough and simple. There is no doubt that the application of deep reinforcement learning in the portfolio still has a lot of energy to be developed, and as of now, this is the most promising way to create a true artificial intelligence fund manager.

References

[1] 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.

[2] Luca Di Persio and Oleksandr Honchar. Artificial neural networks architectures for stock price prediction: Comparisons and applications. *International journal of circuits, systems and signal processing*, 10:403–413, 2016.

[3] Rania Jammazi and Chaker Aloui. Crude oil price forecasting: Experimental evidence from wavelet decomposition and neural network modeling. *Energy Economics*, 34(3):828–841, 2012.

[4] Zhengyao Jiang and Jinjun Liang. Cryptocurrency portfolio management with deep reinforcement learning. In *2017 Intelligent Systems Conference (IntelliSys)*, pages 905–913. IEEE, 2017.

[5] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017.

[6] Salim Lahmiri. Wavelet low-and high-frequency components as features for predicting stock prices with backpropagation neural networks. *Journal of King Saud University-Computer and Information Sciences*, 26(2):218–227, 2014.

[7] Jae Won Lee. Stock price prediction using reinforcement learning. In *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*, volume 1, pages 690–695. IEEE, 2001.

[8] Hui Liu, Xi-wei Mi, and Yan-fei Li. Wind speed forecasting method based on deep learning strategy using empirical wavelet transform, long short term memory neural network and elman neural network. *Energy conversion and management*, 156:498–514, 2018.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[10] John Moody, Lizhong Wu, Yuansong Liao, and Matthew Saffell. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, 17(5-6):441–470, 1998.

[11] Luis Ortega and Khaldoun Khashanah. A neuro-wavelet model for the short-term forecasting of high-frequency time series of stock returns. *Journal of Forecasting*, 33(2):134–146, 2014.

[12] Huai-zhi Wang, Gang-qiang Li, Gui-bin Wang, Jian-chun Peng, Hui Jiang, and Yi-tao Liu. Deep learning based ensemble approach for probabilistic wind power forecasting. *Applied energy*, 188:56–70, 2017.

[13] Pengqian Yu, Joon Sern Lee, Ilya Kulyatin, Zekun Shi, and Sakyasingha Dasgupta. Model-based deep reinforcement learning for dynamic portfolio optimization. *arXiv preprint arXiv:1901.08740*, 2019.