Deep Reinforcement Learning for Portfolio Management

Yue Ma*
Northern Illinois University, DeKalb, IL

Ziping Liu † and Charles D. McAllister †
Southeast Missouri State University, Cape Girardeau, MO

Abstract

This paper discusses how to build deep reinforcement learning (DRL) agents to determine the allocation of money for assets in a portfolio so that the maximum return can be gained. The policy gradient method from reinforcement learning and convolutional neural network/recurrent neural network/convolutional neural network concatenated with the recurrent neural network from deep learning are combined to build the agents. With the proposed models, three types of portfolios are tested: stocks portfolio which has a positive influence due to the Covid-19; stocks portfolio which has a negative influence due to the Covid-19; and stocks, cryptocurrency combined portfolio which are randomly selected. The performance of our DRL agents is compared with that of equal-weighted agent and all the money fully invested in one-stock agents. All of our DRL agents showed the best performance on the randomly selected portfolio, which has an overall stable increasing trend. In addition, the performance of a linear regression model is also tested with the random selected portfolio, and it shows a poor result compared to other agents.

Key Words: Deep reinforcement learning; financial portfolio management; machine learning; neural networks; stock.

1 Introduction

Deep reinforcement learning shows its advantages in various fields, such as gaming [17, 7, 5], transportation [18, 11, 8] and the medical field [1, 24, 20]. These successful applications have inspired more and more researchers to apply this framework to other fields. For example, use in the financial market can expect to achieve the same success.

Investment in the financial market plays an important role in a global economy, especially the stock market investment [3]. However, the financial market is not only large and volatile [16], but also can be heavily impacted by all sorts of unpredictable natural [9], social [6], and political factors [13]. Hence, the impacts of actions taken on the financial market will not be measured easily and quickly, which adds difficulties on the building of effective financial models. In deep reinforcement learning, the goal is to maximize the reward. Because of this special feature, deep reinforcement learning has gained a lot of attention in the field of gaming. Bearing the goal to maximize the overall return in financial portfolio management, which is similar to gaming, deep reinforcement learning should be applicable to manage financial portfolios. Considering the impact of the volatility of Covid-19 on the financial market, three different types of portfolios were selected to test the hypothesis in this paper.

This paper investigates the applicability of deploying deep reinforcement learning agents in the design of portfolio management agents such that weights in the portfolio can be allocated automatically to maximize the overall return. Convolutional neural network and recurrent neural network are investigated in the construction of the policy network, and the performances between agents with different network structures are compared.

2 Literature Review

Financial portfolio management is about the allocation of financial assets to meet the financial goals of investors. Generally, most investors have the same goal: maximize the expected return. In many different fields, such as medical diagnosis, speech recognition, and market clustering analysis, machine learning methods have proved to be powerful analytical tools. The above successful cases have motivated researchers to design machine learning models for portfolio management.

In [4], Gah-Yi Ban et al. introduced performance-based regularization (PBR) and performance-based cross-validation for the portfolio optimization problem and investigated them in detail. Tianello [2] used regression analysis to predict how each type of asset performs on a risk adjusted basis, and then allocates funds to those assets proportionately. Compared to classic machine learning, deep reinforcement learning prefers to achieve a pursued long-term result and portfolio management has a similar goal to maximize the rewards. The aforementioned similar goal has driven many researchers to develop and implement different deep reinforcement learning methods to manage financial portfolios. Liang et al. in [23] used deep reinforcement learning algorithms with continuous action space for the asset allocation. The agents were tested with risk adjusted accumulative portfolio value as objective function and the inputs took the combinations of different features. In [22], Liang et al. implemented three state-of-art continuous reinforcement learning algorithms, Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO) and Policy Gradient (PG) in portfolio management and proposed

*e-mail: yuema@niu.edu
†e-mail: zliu,cdmcallister@semo.edu

an adversarial training method and showed that it can greatly improve the training efficiency and significantly raise average daily return and Sharpe ratio in backtests. Zhang et al. proposed a cost-sensitive portfolio selection method with deep reinforcement learning in [21], where a novel two-stream portfolio policy network was devised to extract both price series patterns and asset correlations. The proposed method also includes a new cost-sensitive reward function to maximize the accumulated return as well as to constrain both costs via reinforcement learning. Kanwar in [10], applied model free reinforcement learning algorithm which is a computationally efficient algorithm that directly estimates the optimal policy or value function through algorithms such as policy iteration or value iteration, with emphasis on policy gradient and Actor Critic Methods to build the trading agent in order to maximize its overall return. Opposite to [14], Yu et al. used a model-based approach which allows agents to plan ahead a range of possible choices to build the deep reinforcement learning agents. Jiang et al. also presented a model-free convolutional neural network with historic prices on a set of financial assets as its input, which outputs portfolio weights of the set in [26]. In [25], Jiang et al. proposed portfolio vector machine (PVM) which is an online stochastic batch learning scheme. In the proposed scheme, PVM works together with the policy network via convolutional neural network (CNN), recurrent neural network (RNN) and the Long Short-Term Memory (LSTM) to re-balance the cryptocurrency portfolio every 30 minutes. Selim Amrouni et al. in [15] extended the work of Jiang et al., applied the PVM to trade the daily stock and cryptocurrency data with a daily rebalance.

This research was motivated by both Jiang et al. [25] and Amrouni et al. [15], and used their work as reference for the basic policy network architecture of a convolutional neural network but implemented with a recurrent neural network and a mixed neural network (RNN+CNN) to build agents and used various financial asset combinations for testing. Since Covid-19, the stock market has experienced the most volatile ups and downs. Meanwhile, there is less related research taking Covid-19 into consideration regarding deep reinforcement learning on portfolio management. In order to gain insights, this research selected assets with different combinations to build the financial portfolio on testing the performance of deep reinforcement learning agents. To compare the performance of deep reinforcement learning with classic machine leaning approaches in portfolio management, linear regression is also tested for weights allocation.

3 Methodology

3.1 Deep Learning

Deep learning is one branch among the many fields of machine learning, and it is based on artificial neural networks. Convolutional neural networks are designed to work with grid-structured inputs, which have strong spatial dependencies in local regions of the grid. The most obvious example of grid-structured data is a 2-dimensional image, while other forms of sequential data like text, time-series, and sequences can also be considered special cases of grid-structured data with various types of relationships among adjacent items. The core concept of convolutional neural network is to simplify complicated questions. It reduced the dimensionality of a large number of parameters into a small number of parameters before processing.

Certain data types such as time-series, text, and biological data contain sequential dependencies among the attributes. Taking time-series data set as example, the values on successive timestamps are closely related to one another. If one uses the values of these timestamps as independent features, then key information about the relationships among the values of these timestamps is lost. For example, the value of a time-series at time t is closely related to its values in the previous window. However, this information is lost when the values at individual timestamps are treated independently of one another. A recurrent neural network, also known as RNN, can solve this problem. RNN is a class of neural networks which allow previous outputs to be used as inputs while having hidden states. The main advantage of recurrent neural network is that it can model sequence data, which is also the reason why RNN models are mostly being used by processing natural language.

3.2 Deep Reinforcement Learning

Learning can be divided into supervised learning, unsupervised learning and reinforcement learning. Supervised learning models receive features (X) and labels (y) as training data, using them to get predicted values (y); while unsupervised learning models receive only features (X) and to do further analysis; but reinforcement learning can be different with these learning types. Reinforcement learning (RL) [12] is the study of how an agent can interact with its environment to learn a policy which maximizes expected cumulative rewards for a task. Environment in DRL represents the outside world the agent interacts with. Agent is the individual which makes decisions to decide what actions to take in the environment. State is the current situation of agent, what the agent observed in the environment. Action is what the agent can do in the environment, the input provides to the environment. Reward is a feedback signal from the environment. Policy is used by an agent as a strategy to decide what actions should be taken given state s. Policy had deterministic policy and stochastic policy.

3.3 Financial Portfolio Management

Markowitz portfolio theory, which is also called Modern portfolio theory (MPT), is a theory on how risk-averse investors can construct portfolios to maximize expected return based on a given level of market risk. The investor can decide the risk or the expected rule, but there is a rule to follow, which is: More risk will be taken, when more money is expected in return. How we compute the expected rate of return for an individual investment

as shown in the formula below:

$$E(R_{port}) = \sum_{i=1}^{n} w_i R_i$$

where:

w_i = the weight of an individual asset in the portfolio, or the percent of the portfolio in Asset i

R_i = the expected rate of return for Asset i

The expected rate of return for a portfolio of investments is simply the weighted average of the expected rates of return for the individual investments in the portfolio.

4 Research Methodology

4.1 Overall Architecture

The concept of using efficient portfolio management agents to maximize the expected return and to avoid risk follows Markowitz portfolio theory. In this paper, a model with deep reinforcement learning (DRL) agents is studied to determine the allocation of money for assets in a portfolio so that the maximum return can be gained. The policy gradient method from reinforcement learning and convolutional neural network/ recurrent neural network/ recurrent neural network concatenated with the convolutional neural network from deep learning are combined together to build the agents. Figure 1 shows the overall architecture of the deep reinforcement learning agent. Data should be pre-processed based on different choices of neutral networks, and then the pre-processed data is passed to the neural networks. Softmax is applied in order to get the weight distribution of the assets. Input data, neural networks and the weight distribution formed the policy network, and reward is calculated based on the calculated weights. Reward is next fed into an objective function to update the policy network.

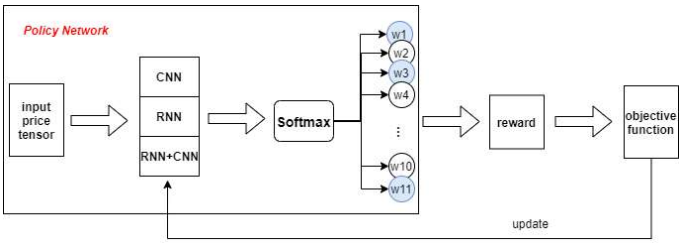

Figure 1: Overall deep reinforcement learning architecture

4.2 Data Collection

Considering the impact of the volatility of Covid-19 on the financial market, in this research, various combinations of stocks and cryptocurrency are selected to be tested, and all the data were collected from Yahoo! Finance [19].

4.2.1 Negatively Influenced Portfolio. The stocks which are heavily impacted by the Covid-19 are selected as the negatively influenced portfolio. These stocks are from the following companies: American Airlines Group (AAL), Delta Airlines (DAL),Southeast Airlines (LUV), United Airlines Holdings (UAL), Spirit Airlines (SAVE), Hilton Worldwide Holdings (HLT), Marriott International(MAR), Choice Hotels International (CHH), InterContinental Hotels Group PLC (IHG), and International Consolidated Airlines Group (ICAGY). Figure 2 shows the daily closing price evolution for the negatively influenced portfolio from January 25, 2016 to January 25, 2021. X-axis represents the days, and y-axis represents close price(US dollars).

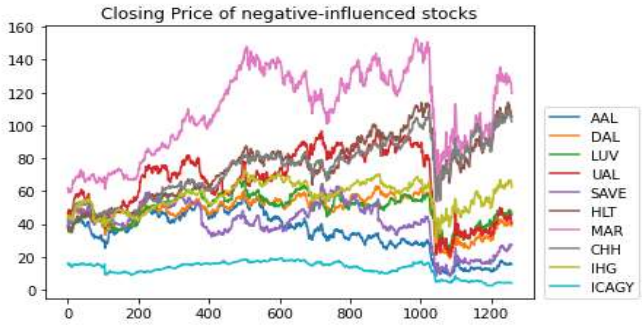

Figure 2: Negatively influenced stocks

4.2.2 Positively Influenced Portfolio. The stocks which have a positive influence because of the Covid-19 are selected as the positive influenced portfolio. These stocks are from the following companies: Alibaba Group Holding Limited (BABA), Walmart (WMT), eBay (EBAY), Best Buy (BBY), Allied Healthcare Products (AHPI), Amedisys (AMED), Viridian Therapeutics (MGEN), Meridian Bioscience (VIVO), NanoViricides (NNVC), Netflix (NFLX). Figure 3 shows the daily closing price evolution for the positive influenced portfolio from January 25, 2016 to January 25, 2021. X-axis represents the days, and y-axis represents close price (US dollars).

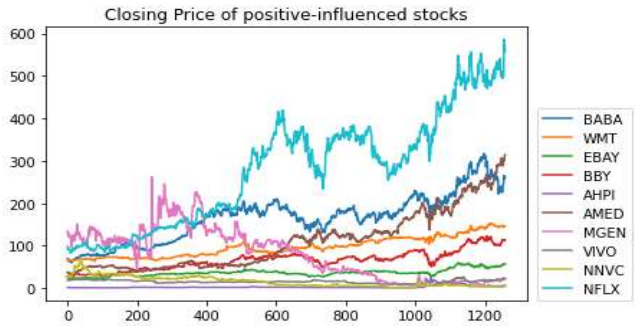

Figure 3: Positively influenced stocks

4.2.3 Randomly Selected Portfolio. To consider neither the positive impact nor the negative impact of Covid-19, seven random US stocks and three cryptocurrencies are selected from the following companies: American Airlines Group (AAL), Facebook (FB), Bank of America (BAC), Best Buy (BBY), Alibaba Group (BABA), Axon Enterprise (AAXN), BlackRock (BLK) , and cryptocurrency of Ethereum (ETH), Ripple (XRP),

Litecoin (LTC). Figure 4 shows the daily closing price evolution for the positive influenced portfolio from January 25, 2016 to January 25, 2021. X-axis represents the days, and y-axis represents close price (US dollars).

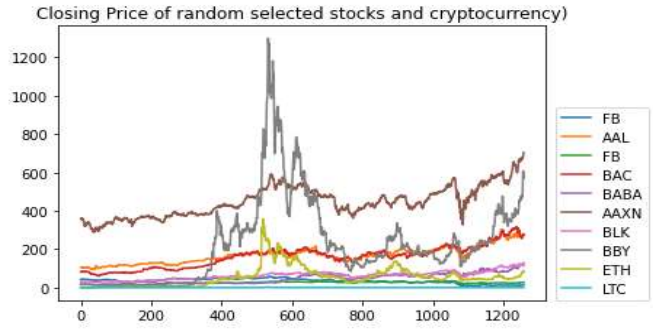

Figure 4: Randomly selected stocks and cryptocurrency

4.3 Data Preprocessing

For both stocks and cryptocurrencies, data between January 25, 2016 to January 25, 2021 were downloaded and they span totally five years, 1258 days for stocks and 1828 days for cryptocurrencies. The reason why only five years data were selected is that cryptocurrency does not have historical data before 2009 when the first cryptocurrency was adopted as bitcoin. Since then, more and more newly generated cryptocurrencies have come out. To make more use of stable cryptocurrency data in the model, our research restricted the data selection to the recent five years.

There was a problem when data for the random selected portfolio was processed, since the length of two types of investment data do not have the same length and has a gap of 569 trading days. In this research, for the days which cryptocurrency trading occurred while the stock trading did not occur, the price information for cryptocurrency was removed.

The data is divided as below:
Total days: 1258
Training days: 60% of total days
Validation days: 20% of total days
Test days: 20% of total days

Normal indicators for stock include ”Date,” ”Open,” ”High,” ”Low,” ”Close,” ”Adj Close” and ”Volume,” but for this research, only ”Open,” ”High,” ”Low” and ”Close” were used. Strictly speaking, the market for cryptocurrency never closes, but there is still data available as ”Open price” and ”Close price” provided for cryptocurrency. ”Open” generally refers to the price at 12:01 AM UTC of any given day and ”close” generally refers to the price at 11:59 PM UTC of any given day. Hence, stock data and cryptocurrency data can be mixed together.

Figure 5 shows the data set weight distribution for this research, testing data set contains 253 days.

4.3.1 CNN Price Tensor Pre-Processing. The input data are arranged into 3D in shape of (4,10,1258) as shown in Figure 6. The first dimension 4 represents the number of features, and they are:

Figure 5: Dataset weight distribution

- Close(t-1)/Open(t-1)
- High(t-1)/Open(t-1)
- Low(t-1)/Open(t-1)
- Open(t)/Open(t-1)

Here the prices are normalized to the same scale. The second value for the dimension is 10, where it represents ten asset items (stocks+cryptocurrency) in the portfolio. The third value for the dimension is 1258, which represents the time steps (days). 1258 was sliced to 50 days each before passing to the network.

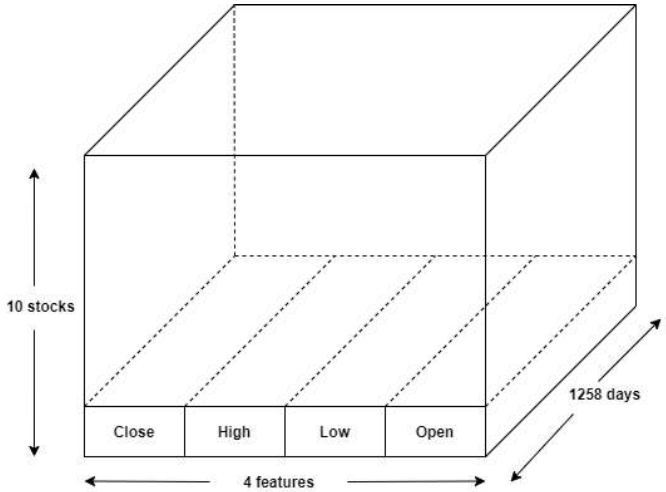

Figure 6: CNN data pre-processing

4.3.2 RNN and RNN+CNN Price Tensor Pre-Processing. For the architecture of recurrent neural network, and the architecture for the mixed (RNN+CNN) neural network, these two networks have the same input shape. There is only one feature (the mean value for the open price, close price, high price and the low price) used. The mean prices are passed to MinMaxScaler which helps to make all the value in the same scale.

$$MeanPrice = \frac{OpenPrice + ClosePrice + HighPrice + LowPrice}{4}$$

After data scaling, data was grouped to 50 days each. In the previous section’s CNN policy network, the length of tensor was set to be 50 days. In order to do a comparative analysis with the CNN policy network, the tensor should be processed as the same

length for batch processing. After transposing and dimension expanding, pre-processed input data has the shape (25,1,10,50), which is illustrated in Figure 7. The diagram shows that there are 25 batches of (1,10,50), for each step, one batch (1,10,50) from (25,1,10,50) was passed to the policy network to calculate the weight distribution. 1 means there is only one feature– mean value, 10 represents 10 stocks, and 50 is the length of the tensor(50 days).

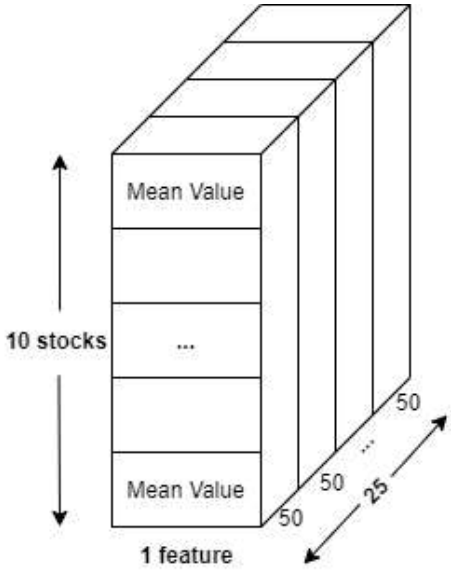

Figure 7: RNN and RNN+CNN data pre-processing

4.3.3 Linear Regression Price Tensor Pre-Processing. Data processing for linear regression is similar as for RNN. However, in linear regression, it needs X (features) and y (target values). In this research, the initial investment and mean value (open price, close price, high price and low price) of 10 stocks are chosen for X, which has 11 features in total. Initial investment is set to 10000 for every data time, and y is the portfolio value which is randomly generated between values of (10000,13000). After setting the portfolio value for each data time, the initial investment of each day will be adjusted to the previous portfolio value. Figure 8 shows the shape of pre-processing data for linear regression model.

Money	Stock1	Stock2	...	Stock10	Portfolio Value
10000	12.3	25.6		28.8	12500
12500	12.9	22.6		15.8	12907
12907	17.8	11.1		15.9	11008
...	...	...		...	...

Figure 8: Linear regression data pre-processing

4.4 Reinforcement Learning Framework

Markov Decision Process (MDP) provided the framework for describing the reinforcement learning problem. The state of the agent is the input matrix X_t and previous portfolio weights at time *t-1*.

$$state = S_t = (X_t, w_{t-1})$$

The action of the agent is the weight vector of portfolio at time t.

$$action = w_t = (w_1, ..., w_n)$$

The reward function is defined such as it is the agent's return minus a baseline return minus a term proportional to the maximum of the weight (this term is set up to make the agent avoid investing fully in one stock), and baseline is an equal weighted agent which divides the investment into the same proportion.

$$r_t = \sum w_i' y_i - \frac{1}{m} \sum y_i - \alpha max(w_1, ..., w_m)$$

where:

y_i is the return of single stock or cryptocurrency.

$\sum w_i' y_i$ is the sum of each stock or cryptocurrency's return. The formula for calculating the expected rates of return was used as reference.

m is number of items(stocks and cryptocurrency) in the portfolio.

α is the parameter of regularization, which can avoid investing fully in one stock.

The input price tensor is the state for the deep reinforcement learning model, and action is the weight distribution which was calculated by the policy network. Reward is the reward function in the overall architecture which helped to build the objective function. The three components work together to train the model to pursue maximum return.

4.5 Network Structure

The convolutional neural network structure was designed by Amrouni et al [15]. Motivated by the work in [15], this research also studied recurrent network, and a mixed network with RNN+CNN. The network structure design for the recurrent neural network and the mixed network is a new attempt for the training process which will be introduced in the next section.

4.5.1 Convolutional Neural Network. The detailed parameters for the convolutional neural network model are listed in Table 1. The price tensor starts in a shape of $[None, nb\_feature, m, n]$, where $nb\_features$ represents number of features, m represents the number of stocks and cryptocurrency, n is the length of tensor. It is then transposed to $[None, n, m, nb\_features]$ where it is used as the input for the convolutional neural network.

The policy network which uses convolutional neural network comprises input layer and three convolution layers where conv1 extracted and integrated effective features to 2 kernels, conv2 kept on the process and output 20 kernels, and in the end the

conv3 will output the last feature map, which concatenated with the cash bias. After the concatenation with the cash bias, the tensor got squeezed, and in the end, softmax function is used to get the probability distribution which is also the weight distribution of the portfolio. The convolutional neural network model is shown in Figure 9. Batch size was set to 50 in this research, which is a common number picked by most researchers and it can also be changed to another value.

The output of the network is then used to calculate the adjusted reward. At first it needs to calculate the instantaneous reward as shown in the formula below:

$$InstantaneousReward = \frac{PortfolioValue - PrevPortfolioValue}{PrevPortfolioValue}$$

In the formula, the trading cost is also taken into consideration when the portfolio value is calculated.

Considering the formulation of the objective function, since there is no maximization function for the reward, minimizing the negative of the original value is equivalent to maximizing the original value; therefore, minimize (-adjusted reward) was chosen to maximize the reward over the batch.

Table 1: Parameters for convolutional neural network model

Input	(?,4,10,50)	
Conv1	(?,50,10,2)	activation=tf.nn.relu, filters=2, strides= (1, 1), kernel_size=(1,3), padding='same'
Conv2	(?,1,10,20)	activation=tf.nn.relu, filters=20, strides= (50, 1), kernel_size= (1, 50), padding='same'
Conv3	(?,1,10,1)	activation=tf.nn.relu, filters=1, strides= (21, 1), kernel_size= (1, 1), padding='same'
Tensor4	(?,1,11,1)	Concatenate conv3 with cash_bias
Squeezed_tensor	(?,11)	
Action	(?,11)	tf.nn.softmax

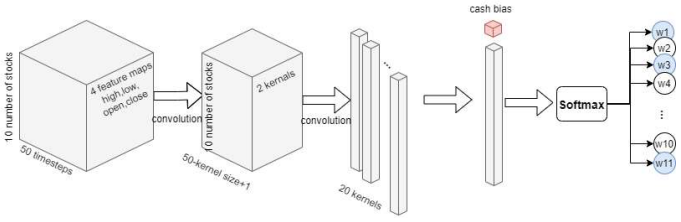

Figure 9: Convolutional neural network model

4.5.2 Recurrent Neural Network. The detailed parameters for the recurrent neural network model are listed in Table 2. The tensor which is the input to the network has the shape of (None, m, n), where *m* is the number of stocks and *n* is the length (time step) of the tensor. It has two recurrent layers. For the first recurrent layer, the number of recurrent units was set to 20, and it came from the inspiration from figure 4.9's CNN of filters=20. For the second recurrent layer, the number of the recurrent units was set to 1. The output dimension after the two recurrent layers was expanded to 4D. And after that it can be concatenated with a 4D-shape cash-bias. The following steps such as to squeeze the tensor, to use softmax to get the probability distribution, and to calculate the reward are the same as in the convolutional neural network. The recurrent neural network model is shown in Figure 10.

Table 2: Parameters for recurrent neural network model

Input	(?,10,50)	
Rnn1	(?,10,20)	num_units=20, activation=tf.nn.relu
Rnn2	(?,10,1)	num_units=1, activation=tf.nn.relu
Outputs(expand dimension)	(?,1,10,1)	
Tensor4	(?,1,11,1)	Concatenate Outputs with cash_bias
Squeezed_tensor	(?,11)	
Action	(?,11)	tf.nn.softmax

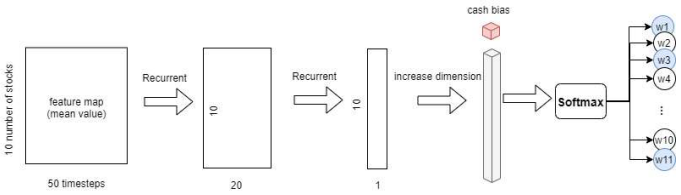

Figure 10: Recurrent neural network model

4.5.3 Mixed Neural Network. The mixed neural network model combines both the convolution layers and recurrent layers when processing the price tensor. Table 3 shows the details of the parameters. The input tensor has the same shape as the one of the recurrent neural networks and it is (None,10,50). First, it is passed to the recurrent layer, which has 20 recurrent units. The output of the recurrent layer's dimension is then expanded in order to be processed by the convolutional neural network. For the convolution layer, there is one filter. After the dimension expansion, all the following steps are the same as the convolution neural network's. The mixed neural network structure is as the Figure 11 below.

Table 3: Parameters for CNN+RNN model

Input	(?,10,50)	
Rnn1	(?,10,20)	num_units=20, activation=tf.nn.relu
Outputs(expand dimension)	(?,1,10,20)	
Conv3	(?,1,10,1)	activation=tf.nn.relu, filters=1, strides=(21, 1), kernel_size=(1, 1), padding='same'
Tensor4	(?,1,11,1)	Concatenate conv3 with cash_bias
Squeezed_tensor	(?,11)	
Action	(?,11)	tf.nn.softmax

4.5.4 Linear Regression Model. The training data is passed to the standard scaler first to be processed into the same scale.

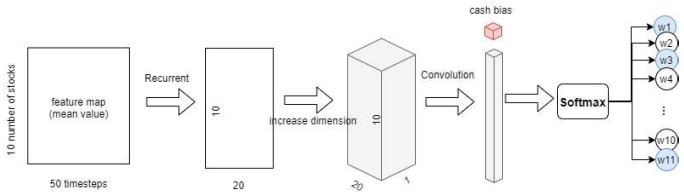

Figure 11: Mixed neural network model

Next, the processed data passes into the linear regression model in order to get the coefficient list which is also the weight list. The accumulated value for each value in the portfolio weight list should be 1, so the proportional value is calculated. The linear regression model is shown in Figure 12.

$$sum\_coef = |coef_1| + |coef_2| + ...|coef_{11}|$$

$$weight\_list = [\frac{|coef_1|}{sum\_coef}, \frac{|coef_2|}{sum\_coef} ... \frac{|coef_{11}|}{sum\_coef}]$$

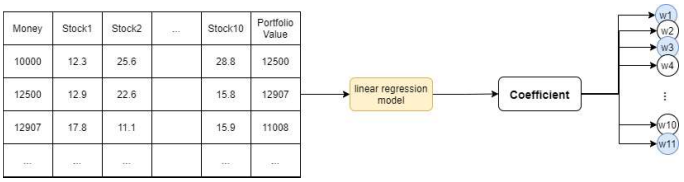

Figure 12: Linear regression model

4.6 Network Training Process

Portfolio Vector Memory was introduced by Jiang et al. in [9], and was implemented by Selim Amrouni in [10] for convolutional neural network model training. Figure 13 shows the CNN model training process. For each episode (total of 2 episodes in this paper), the PVM was initialized with training parameters. For each batch (total of 10 batches in this paper, 1 batch has 50 days), the starting point of the batch, initial portfolio value and initial weights were drawn from the PVM at time t-1. Based on the starting point, a price tensor which has the length of 50 was passed to the policy network to get the weight distribution. The weight distribution which is also the action wrote to the PVM at time t and made the agents step to the next day. The whole process recursively performed until all the batches and episodes ended.

This paper also implemented the PVM with the recurrent neural network model and the mixed RNN+CNN model. Figure 14 shows the process. The main difference between them is that for RNN and the mixed model, the price tensor retrieval and days traversal are separated. The starting point which was fed from the PVM was generated from the RNN pre-processed data, instead of generating the price tensor from CNN pre-processed data. CNN pre-processed data was here to help traverse the days. One price tensor which has length of 50 was retrieved and sent to the policy network to get the weight distribution. The remaining steps were similar as the CNN's.

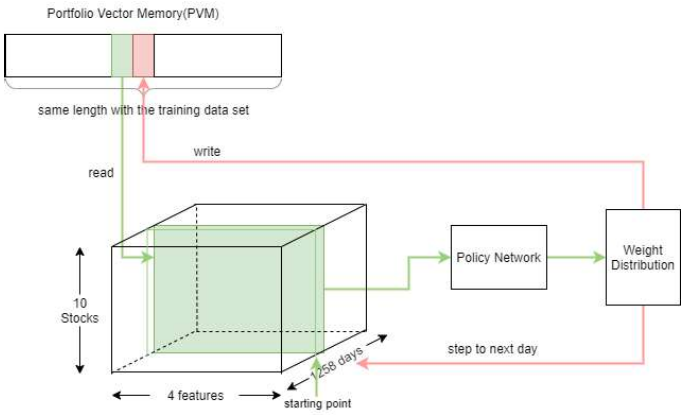

Figure 13: CNN training process

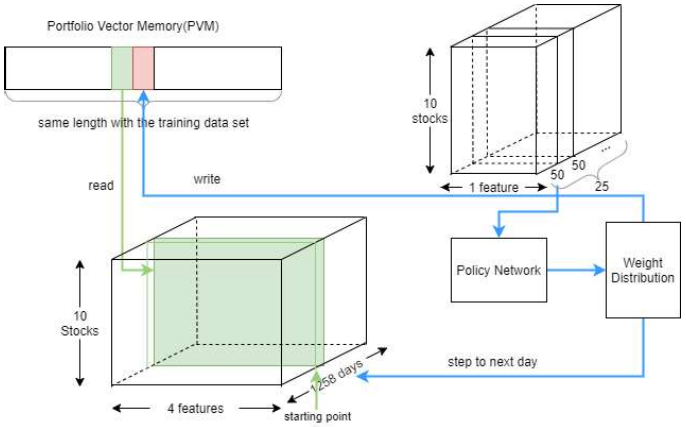

Figure 14: RNN and RNN+CNN training process

5 Results

5.1 Overall Results

Three different types of portfolio and three different networks were tested. For linear regression, the random selected portfolio was tested. The portfolio value evolution, min and max value, which represents minimum and maximum portfolio value, and the weight evolution during testing are presented in the following subsections. For testing, result figures are presented in the next subsections, except three deep reinforcement learning agents, where are other financial portfolio management agents being tested. Equal-weighted agent is an agent used for all the financial assets. Initial investment is 10000 dollars which acts as a baseline to show a clear performance comparison. For the other "full stock AAL, DAL, LUV..." means the money fully invest on the single stock.

5.1.1 Negatively Influenced Portfolio. Figure 15 shows the negatively influenced portfolio value evolution for 253 days. For both three types of deep reinforcement learning agents, there exist certain periods in which portfolio value is greater than the initial investment. But overall, they do not show an increasing trend. Among the three models, RNN model has

the best performance(most return) at the end of the testing day, however it still loses 45.58% off the initial investment. Table 4 shows the maximum portfolio value and the minimum portfolio value for testing data set, CNN reached 10749 dollars which is the maximum portfolio value compared to the other two deep reinforcement learning agents. Tables 5, 6 and 7 shows the weight evolution every 50 days during testing for these three deep reinforcement learning agents.

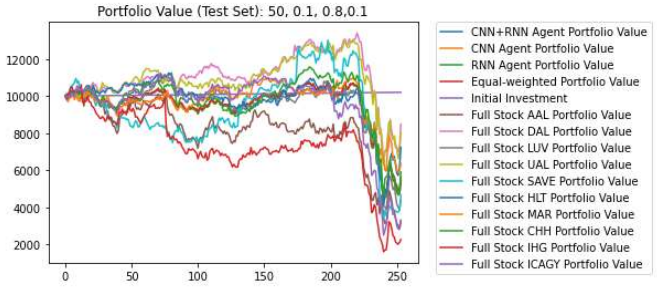

Figure 15: Negatively influenced portfolio

Table 4: Negatively influenced portfolio

	RNN	CNN	RNN+CNN
MAX	10744.384170389269	10749.582654440033	10746.332953411205
MIN	4296.119453575351	4257.566520880737	4282.784544420906

Table 5: Weight evolution during testing of CNN network

	Money	AAL	DAL	LUV	UAL	SAVE	HLT	MAR	CHH	IHG	ICAGY	PortfolioValue
Start	1	0	0	0	0	0	0	0	0	0	0	10000
50 days	0.04202312	0.09757042	0.09566913	0.09583758	0.09606861	0.09529205	0.09550374	0.09521315	0.09623 07	0.09488089	0.09571061	9869
100 days	0.04302808	0.09447327	0.09794327	0.09643734	0.09649739	0.09397549	0.09667 79	0.09640097	0.09637725	0.09359015	0.09459889	9043
150 days	0.04197768	0.09555684	0.09728851	0.09613146	0.09501 63	0.09509232	0.09558895	0.09559 92	0.09481207	0.09690978	0.09602688	10111
200 days	0.04253444	0.09510022	0.09617578	0.09571166	0.09539249	0.09750834	0.09640942	0.09539837	0.09467928	0.09564289	0.09544712	10471
250 days	0.04445002	0.09371601	0.10153517	0.09098017	0.09698123	0.09955532	0.09356 5	0.09537336	0.10010976	0.09723239	0.08650158	4815

5.1.2 Positively Influenced Portfolio. Figure 16 shows the positively influenced portfolio value evolution for 253 days. Due to the explosive increasing trend for the full stock on BABA portfolio value, the graph does not show a clear trend for the

Table 6: Weight evolution during testing of RNN network

	Money	AAL	DAL	LUV	UAL	SAVE	HLT	MAR	CHH	IHG	ICAGY	PortfolioValue
Start	1	0	0	0	0	0	0	0	0	0	0	10000
50 days	0.05115532	0.09664 03	0.09475713	0.09492398	0.09515281	0.09438365	0.09459332	0.09430 55	0.09531335	0.09397641	0.09479822	9870
100 days	0.05236406	0.09355161	0.09698776	0.09549653	0.09555599	0.09305869	0.09573473	0.09546051	0.09543702	0.09267 71	0.09367601	9052
150 days	0.05109973	0.09464697	0.09636215	0.09521612	0.09411158	0.09418687	0.09467877	0.09468893	0.09390 93	0.09598703	0.09511254	10111
200 days	0.05176996	0.09418 29	0.09524809	0.09478844	0.09447235	0.09656779	0.09547947	0.09447817	0.09376602	0.09472033	0.09452645	10469
250 days	0.05408524	0.09277103	0.10051135	0.09006277	0.09600332	0.09855146	0.09262155	0.09441167	0.09910031	0.09625195	0.08562935	4854

Table 7: Weight evolution during testing of RNN+CNN network

	Money	AAL	DAL	LUV	UAL	SAVE	HLT	MAR	CHH	IHG	ICAGY	PortfolioValue
Start	1	0	0	0	0	0	0	0	0	0	0	10000
50 days	0.04798819	0.09696288	0.09507342	0.09524083	0.09547042	0.09469869	0.09490906	0.09462028	0.09563149	0.09429009	0.09511465	9870
100 days	0.04912673	0.09387 12	0.09731909	0.09582276	0.09588243	0.09337659	0.09606178	0.09578662	0.09576306	0.09299371	0.09399602	9049
150 days	0.04793 61	0.09496253	0.09668343	0.09553357	0.09442535	0.09450089	0.09499443	0.09500462	0.09422239	0.09630705	0.09542965	10111
200 days	0.04856724	0.09450101	0.09556 98	0.09510 86	0.09479144	0.09689396	0.09580196	0.09479728	0.09408273	0.09504026	0.09484572	10470
250 days	0.05074487	0.09309864	0.10086629	0.09038082	0.09634235	0.09889948	0.09294863	0.09474508	0.09945027	0.09659185	0.08593173	4841

other agents. Actually, for the three types of deep reinforcement learning agents, during more than half of the testing days, the portfolio value seldom exceeds the initial investment, but at the end of the testing days, all agents show an increasing trend. For CNN model which has the best performance (most return) at end of the testing days, it gains 110.87% on the initial investment. Table 8 shows the maximum portfolio value and the minimum portfolio value for the testing data set, CNN reached 22784.7 dollars which is the maximum portfolio value compared to the other two deep reinforcement learning agents. Tables 9, 10 and

11 show the weight evolution during testing for these three deep reinforcement learning agents.

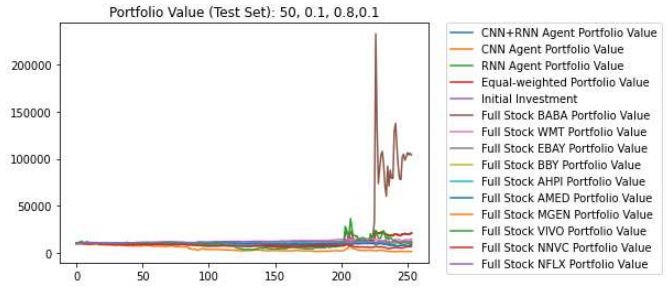

Figure 16: Positively influenced portfolio

Table 8: Positively influenced portfolio

	RNN	CNN	RNN+CNN
MAX	22696.510164433843	22784.707536769543	22604.74045890975
MIN	7621.239917016383	7607.848561432198	7635.092109920798

Table 9: Weight evolution during testing of CNN network

	Money	BABA	WMT	EBAY	BBY	AHPI	AMED	MGEN	VIVO	NNVC	NFLX	PortfolioValue
Start	1	0	0	0	0	0	0	0	0	0	0	10000
50 days	0.04045456	0.09421504	0.09656 75	0.09705374	0.09904441	0.09744479	0.09691681	0.09784962	0.08812246	0.09594888	0.09638219	9509
100 days	0.04151763	0.09493009	0.09508458	0.09717265	0.09681507	0.09684529	0.09672782	0.09159769	0.09385134	0.09725982	0.09819801	8612
150 days	0.04063 14	0.09541863	0.09672963	0.09640315	0.09806951	0.09753956	0.09593144	0.08993582	0.09915782	0.09297967	0.09720337	8168
200 days	0.04009868	0.09856441	0.09598776	0.09656359	0.09593668	0.09655176	0.09554054	0.08520287	0.10651 74	0.09353385	0.09550245	9840
250 days	0.04078758	0.10094165	0.09346066	0.09511324	0.09493542	0.09390534	0.09331918	0.09932903	0.08692816	0.10371155	0.09756818	19948

Table 10: Weight evolution during testing of RNN network

	Money	BABA	WMT	EBAY	BBY	AHPI	AMED	MGEN	VIVO	NNVC	NFLX	PortfolioValue
Start	1	0	0	0	0	0	0	0	0	0	0	10000
50 days	0.04617789	0.09365308	0.09599151	0.09647485	0.09845364	0.09686356	0.09633874	0.09726599	0.08759685	0.09537658	0.09580 73	9512
100 days	0.04738496	0.09434898	0.09450253	0.09657781	0.09622242	0.09625245	0.09613 57	0.09103698	0.09327683	0.09666445	0.09759 69	8621
150 days	0.04638161	0.09484672	0.09614986	0.09582533	0.09748171	0.09695493	0.09535646	0.08939677	0.09856349	0.09242237	0.09662076	8179
200 days	0.04577628	0.09798143	0.09542002	0.09599244	0.09536923	0.09598068	0.09497544	0.08469892	0.10588738	0.09298062	0.09493757	9843
250 days	0.04656188	0.10033 4	0.09289805	0.09454068	0.09436393	0.09334004	0.09275742	0.09873109	0.08640487	0.10308722	0.09698083	19888

Table 11: Weight evolution during testing of CNN+RNN network

	Money	BABA	WMT	EBAY	BBY	AHPI	AMED	MGEN	VIVO	NNVC	NFLX	PortfolioValue
Start	1	0	0	0	0	0	0	0	0	0	0	10000
50 days	0.05211806	0.09306983	0.09539 37	0.09587403	0.09784 05	0.09626032	0.09573876	0.09666024	0.08705131	0.09478 26	0.09521064	9515
100 days	0.05347262	0.09374604	0.09389861	0.09596063	0.09560751	0.09563735	0.09552135	0.09045521	0.09268075	0.09604672	0.09697 32	8630
150 days	0.05234932	0.09425317	0.09554816	0.09522566	0.09687167	0.09634819	0.09475972	0.08883733	0.09794668	0.09184 4	0.09601611	8190
200 days	0.05166957	0.09737629	0.09483 07	0.09539959	0.09478023	0.09538 79	0.09438887	0.09417582	0.10523341	0.09240637	0.09435123	9846
250 days	0.05255428	0.09970339	0.09231418	0.09394649	0.09377084	0.09275 34	0.09217443	0.09811056	0.08586181	0.10243931	0.09637131	19825

5.1.3 Randomly Selected Portfolio. Figure 17 shows the randomly selected portfolio value evolution for 100 days. For the three types of deep reinforcement learning agents, they all show a stable increasing trend. The deep reinforcement learning agents have the best performance on this type of portfolio compared to the other two portfolios. The RNN model, which has the best performance (most return) gained 47.72% initial investment for 253 days. Table 12 shows the maximum portfolio value and the minimum portfolio value during testing, RNN reached 14772 dollars which is the maximum portfolio value compared to the other two deep reinforcement learning agents. Tables 13, 14 and 15 show the weight evolution during testing for these three deep reinforcement learning agents.

5.1.4 Linear Regression on Randomly Selected Portfolio. Since randomly selected portfolio shows a stable increasing performance, we tested linear regression using the randomly selected portfolio. The linear regression agent's result is compared with the equal-weighted portfolio value and the initial investment in Figure 18. From the comparison, it can be seen

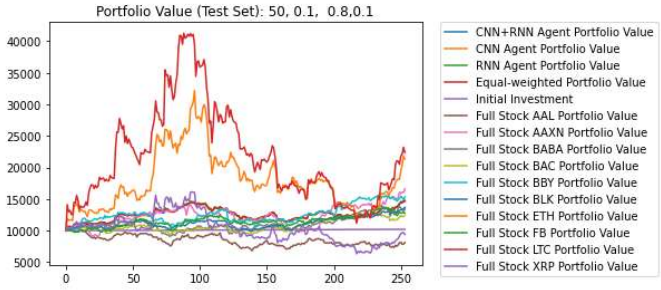

Figure 17: Randomly selected portfolio

Table 12: Randomly selected portfolio

	RNN	CNN	RNN+CNN
MAX	14772.48989557328	14679.754959892462	14764.415960353592
MIN	10000	10000	10000

that linear regression model shows a bad performance which has a clear downward trend. The weight evolution during testing is shown in Figure 19, the weight adjustment during testing is volatile for the linear regression agent.

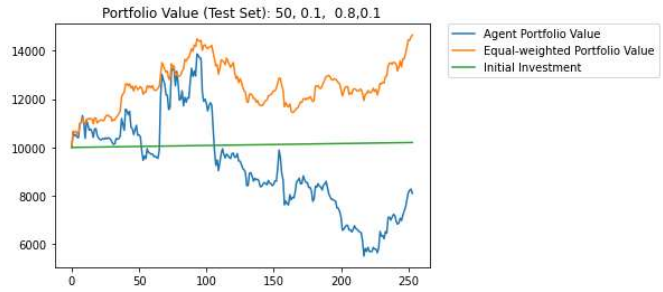

Figure 18: Linear regression on random selected portfolio

5.2 Portfolio Comparison Results

In this section, the performance results from the three different types of portfolios are discussed. Figure 20 shows the CNN model results for the three types of portfolio. Figure 21 shows the RNN model's results, and Figure 22 shows the RNN+CNN model's results. Among the three models, the negatively selected portfolio value presents a decreasing trend, while the positively selected portfolio value presents an explosive increasing trend around day 200, and the randomly selected portfolio presents an overall stable state with positive returns.

5.3 Activation Functions

The performances of activation functions were compared using random selected portfolio. Three different activation functions were compared: sigmoid, ReLU and tanh. *X* axis shows the 253 days, and the *y* axis is the portfolio value. Figure

Table 13: Weight evolution during testing of CNN network

	Money	AAL	AAXN	BABA	BAC	BBY	BLK	ETH	FB	LTC	XRP	PortfolioValue
Start	1	0	0	0	0	0	0	0	0	0	0	10000
50 days	0.07450122	0.09033511	0.09407107	0.09289475	0.09233032	0.09176801	0.09219527	0.09322398	0.09446668	0.09161997	0.09259363	12429
100 days	0.07540939	0.09131607	0.09225253	0.09355897	0.09278396	0.09257833	0.09294329	0.09269756	0.09382129	0.09080885	0.09182977	14157
150 days	0.07423953	0.09346664	0.09401267	0.09278013	0.09427755	0.09246214	0.09293781	0.09294169	0.09060269	0.09087423	0.09140374	12331
200 days	0.07616618	0.09397048	0.09440434	0.09629674	0.09472538	0.09163484	0.09343745	0.08661038	0.09466618	0.09067046	0.09138125	12397
250 days	0.07331774	0.09164177	0.08960015	0.08934477	0.09293649	0.09313318	0.09140766	0.09780822	0.08983148	0.09065893	0.09043927	14475

Table 14: Weight evolution during testing of RNN network

	Money	AAL	AAXN	BABA	BAC	BBY	BLK	ETH	FB	LTC	XRP	PortfolioValue
Start	1	0	0	0	0	0	0	0	0	0	0	10000
50 days	0.05744246	0.09200017	0.09580498	0.09460698	0.09403215	0.09345947	0.09389461	0.09494228	0.09620789	0.09330087	0.09430031	12477
100 days	0.05815997	0.09301998	0.09397392	0.09530473	0.09451527	0.09430579	0.09467756	0.09442725	0.09557194	0.09250329	0.09354326	14243
150 days	0.05723598	0.09518336	0.09573942	0.09448543	0.09600916	0.09416404	0.09464481	0.09464876	0.09226068	0.09254333	0.09308257	12372
200 days	0.05874964	0.09574646	0.09618409	0.09811217	0.09651118	0.09336238	0.09519897	0.09824032	0.09645087	0.09833702	0.09310402	12439
250 days	0.05651004	0.09330393	0.09122665	0.09096526	0.09462212	0.09482238	0.09306556	0.09958222	0.09146208	0.09833674	0.09610431	14563

23 shows the RNN activation results. Figure 24 shows the combined model activation result, and Figure 25 shows the CNN activation results.

5.4 Optimizer

The performances of various optimizers are compared using random selected portfolio. Four types of optimizers are compared: Adam, RMSProp, Gradient Descent and Adagrad. *X* axis shows the 253 days, and *y* axis is the portfolio value. For all the models, Adam optimizer and Gradient Descent optimizer

Table 15: Weight evolution during testing of RNN+CNN network

	Money	AAL	AAXN	BABA	BAC	BBY	BLK	ETH	FB	LTC	XRP	PortfolioValue
Start	1	0	0	0	0	0	0	0	0	0	0	10000
50 days	0.05896482	0.09185157	0.09565025	0.09445417	0.09388027	0.09330852	0.09374295	0.09478894	0.09605 25	0.09315 8	0.09414801	12472
100 days	0.05969696	0.09286789	0.09382027	0.09514 89	0.09436073	0.09415 16	0.09452276	0.09427286	0.09541568	0.09235205	0.09339031	14236
150 days	0.05875333	0.09503016	0.09558533	0.09433335	0.09585463	0.09400885	0.09449248	0.09449642	0.09211 83	0.09239438	0.09293275	12369
200 days	0.06030452	0.09558829	0.09602 52	0.09795 01	0.09635175	0.09320815	0.09504171	0.08809743	0.09629154	0.09819109	0.09295021	12435
250 days	0.05800961	0.09315563	0.09108165	0.09082068	0.09447173	0.09467167	0.09291765	0.09942394	0.09131543	0.09818044	0.09595156	14555

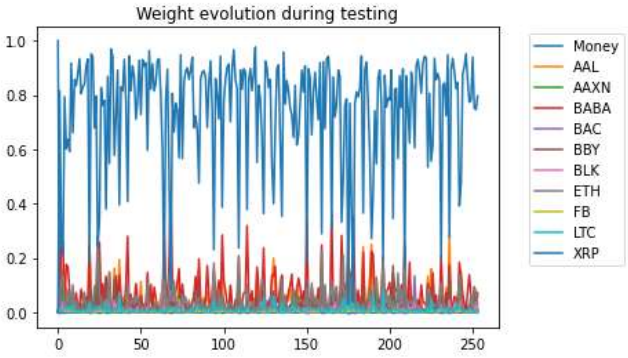

Figure 19: Weight evolution of linear regression

showed the best result. Figure 26 shows the RNN activation results, Figure 27 shows the combined model activation result, and Figure 28 shows the CNN activation results.

5.5 Learning Rate

The performances of different learning rates are also compared. Four different learning rates with 0.1, 0.01, 0.001 and 0.0001 were calculated. Figure 29 shows the RNN learning rate results. Figure 30 shows the combined model learning rate result, and Figure 31 shows the CNN learning rate results.

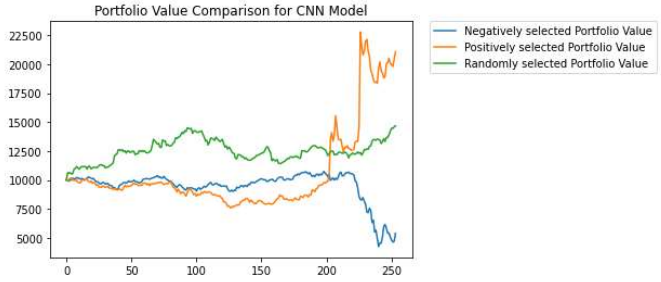

Figure 20: CNN model results

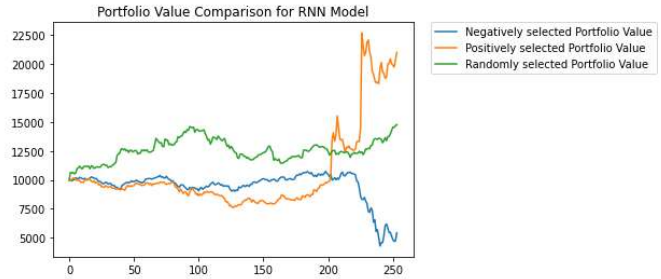

Figure 21: RNN model results

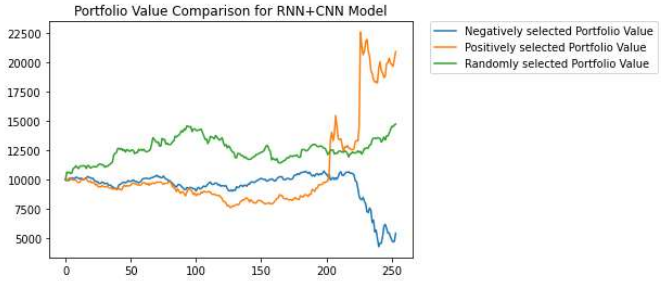

Figure 22: RNN+CNN model results

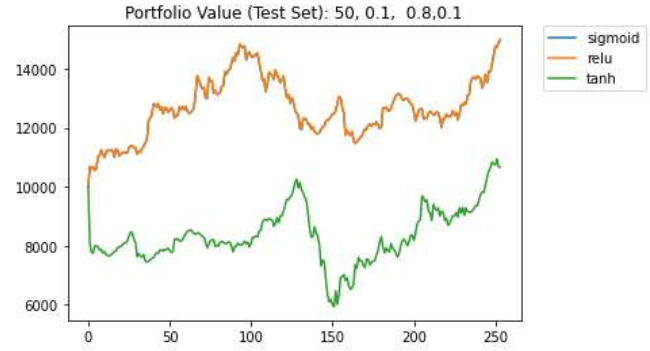

Figure 23: RNN results with activation functions

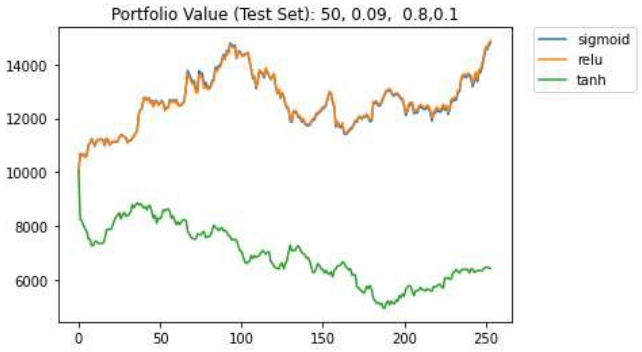

Figure 24: RNN+CNN results with activation functions

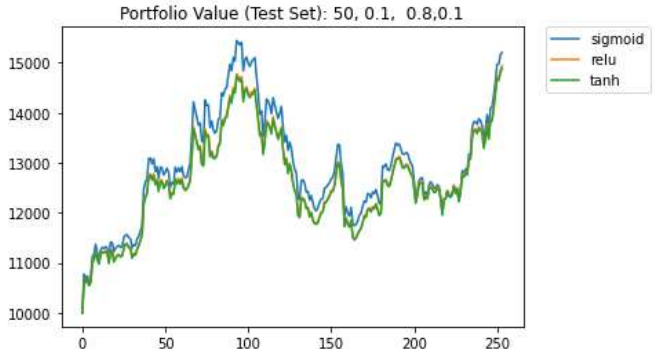

Figure 25: CNN results with activation functions

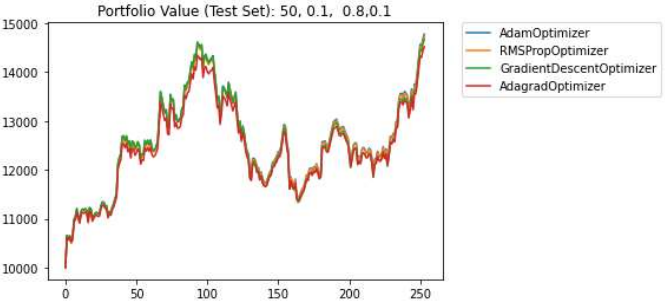

Figure 26: RNN results with different optimizers

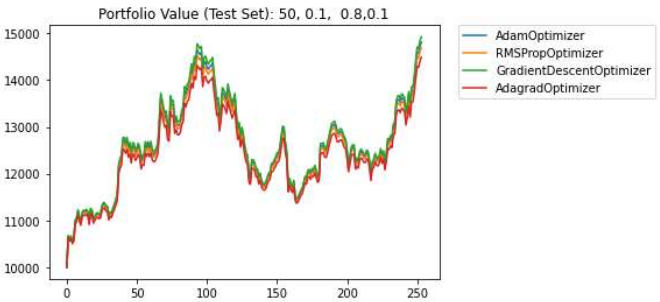

Figure 27: RNN+CNN results with different optimizers

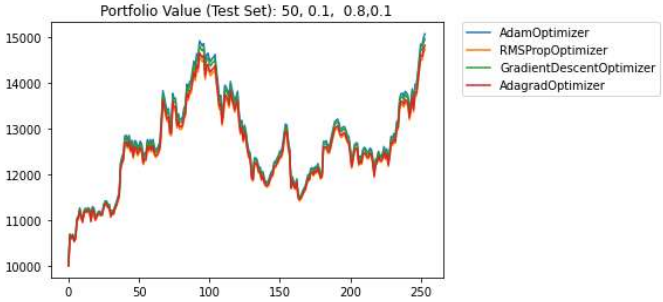

Figure 28: CNN results with different optimizers

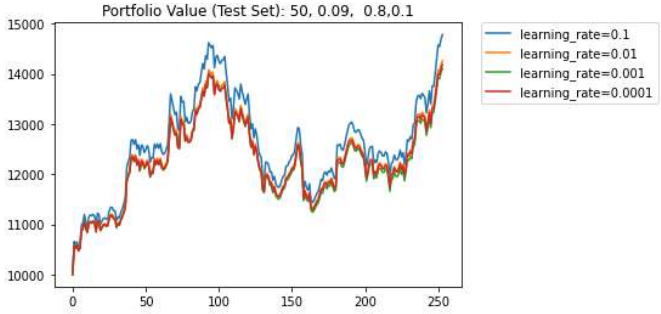

Figure 29: RNN results with different learning rates

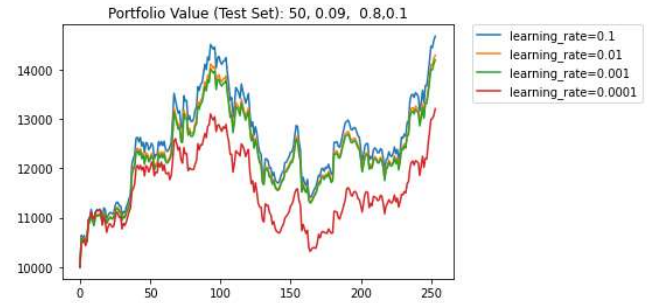

Figure 30: RNN+CNN results with different learning rates

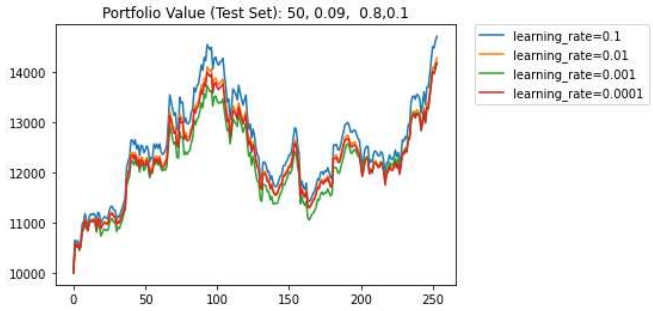

Figure 31: CNN results with different learning rates

6 Conclusion

This paper implemented deep reinforcement learning method to allocate weights for stocks and cryptocurrency in portfolio management and compared different policy network models. The training results from various policy networks constructed with convolutional neural network, recurrent neural network, or the mixed one show similar results. In this research, three different portfolios were considered, and they include stocks that have been negatively influenced by the Covid-19, and the stocks that have been positively influenced by Covid-19, and the stocks and cryptocurrency that were randomly selected. Among the three portfolios, it was found that all the agents have the best performance on the randomly selected portfolio, which presents overall stable moving-up portfolio values as time goes on. The average return rate for all three deep reinforcement learning agents reached 47.38%. It is known that the financial market is filled with uncertainties and unpredictability due to many different factors, so the training results from this research also show the instability when dealing with positively influenced portfolio and negatively influenced portfolio. The linear regression model for portfolio optimization shows poor results, but the weight allocation changed more when compared with deep reinforcement learning agents. Overall, the linear regression model does not meet the goal of financial portfolio management. On the other hand, the deep reinforcement learning agents show a better performance when cryptocurrency was added to the portfolio.

This paper considers that cryptocurrency does not have a long history, so only 5 years data were used. For future studies, data from 10 or more years can be collected to research the performance of deep reinforcement learning agents. The paper only applied to policy gradient in reinforcement learning; in the future, deep Q network and Deep Deterministic Policy Gradient can also be added to do a comparison study with the agents in the paper. For financial fields, there are many financial portfolio optimization methods, but in this research only equal weighted allocation strategy was used to compare with deep reinforcement learning agents. To do a further study, other portfolio optimization methods can also be applied and tested.

References

[1] A. Jonsson, "Deep Reinforcement Learning in Medicine," *Kidney Diseases*. 5:18-22, 2019.

[2] A. Tianello, *Using a Multi-Factor Regression Model to Determine Asset Allocation*, https://ufdcimages.uflib.ufl.edu/AA/00/05/75/50/00001 /altianello-Thesis.pdf.

[3] C. Adjasi and N. Biekpe, "Stock Market Development and Economic Growth: The Case of Selected African Countries," *African Development Review*, 18:144-161, 2006.

[4] G. Ban, N. El Karoui, and A. Lim, "Machine Learning and Portfolio Optimization," *Management Science*, 64:1136-1154, 2018.

[5] G. Lample, and D. Chaplot, "Playing FPS Games with Deep Reinforcement Learning," *Thirty-First AAAI Conference On Artificial Intelligence, San Francisco, California, USA*, 2017.

[6] J. Pineiro-Chousa, M. Vizcaíno-Gonzalez, and A. Perez-Pico, "Influence of Social Media over the Stock Market," *Psychology and Marketing*, 34: 101-108, 2017.

[7] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, and S. Levine, "Model-Based Reinforcement Learning for Atari," *ArXiv Preprint ArXiv:1903.00374*, 2019.

[8] L. Schultz, and V. Sokolov, "Deep Reinforcement Learning for Dynamic Urban Transportation Problems," *ArXiv Preprint ArXiv:1806.05310*, 2018.

[9] L. Wang, and A. Kutan, "The Impact of Natural Disasters on Stock Markets: Evidence from Japan and the US," *Comparative Economic Studies*, 55:672-686, 2013.

[10] N. Kanwar, *Deep Reinforcement Learning-based Portfolio Management,* https://rc.library.uta.edu/uta-ir/bitstream/handle/10106/28108/KANWAR-THESIS-2019.pdf?sequence=1&isAllowed=y, 2019.

[11] N. Kumar, S. Rahman, and N. Dhakad, "Fuzzy Inference Enabled Deep Reinforcement Learning-Based Traffic Light Control for Intelligent Transportation System," *IEEE Transactions On Intelligent Transportation Systems*. 22:4919-4928, 2020.

[12] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep Reinforcement Learning that Matters," *Proceedings Of The AAAI Conference On Artificial Intelligence, Louisiana, USA*, 392:3207-3214, 2018.

[13] P. Santa-Clara, and R. Valkanov, "The Presidential Puzzle: Political Cycles and the Stock Market," *The Journal Of Finance*, 58:1841-1872, 2003.

[14] P. Yu, J. Lee, I. Kulyatin, Z. Shi, and S. Dasgupta, "Model-Based Deep Reinforcement Learning for Dynamic Portfolio Optimization," *ArXiv Preprint ArXiv:1901.08740*, 2019.

[15] S. Amrouni, A. Moulin, and P. Mizrahi, *A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem*, https://selimamrouni.github.io/portfolio/air-elle.html, 2018.

[16] S. Bartram, G. Brown, and R. Stulz, "Why are US Stocks more Volatile?" *The Journal Of Finance*, 67:1329-1370, 2012.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *ArXiv Preprint ArXiv:1312.5602*, 2013.

[18] X. Liu, Z. Ding, S. Borst, and A. Walid, "Deep Reinforcement Learning for Intelligent Transportation Systems," *ArXiv Preprint ArXiv:1812.00979*, 2018.

[19] Yahoo Finance. https://finance.yahoo.com/.

[20] Y. Liu, B. Logan, N. Liu, Z. Xu, J. Tang, and Y. Wang, "Deep Reinforcement Learning for Dynamic Treatment Regimes on Medical Registry Data," *2017 IEEE International Conference On Healthcare Informatics (ICHI), Park City, Utah, USA*, pp. 380-385, 2017.

[21] Y. Zhang, P. Zhao, Q. Wu, B. Li, J. Huang, and M. Tan, "Cost-Sensitive Portfolio Selection via Deep Reinforcement Learning," *IEEE Transactions On Knowledge And Data Engineering*, 34: 236-248, 2020.

[22] Z. Liang, H. Chen, J. Zhu, K. Jiang, and Y. Li, "Adversarial Deep Reinforcement Learning in Portfolio Management," *ArXiv Preprint ArXiv:1808.09940*, 2018.

[23] Z. Liang, K. Jiang, H. Chen, J. Zhu, and Y. Li, "Deep Reinforcement Learning in Portfolio Management," *ArXiv Preprint ArXiv:1808.09940*, 2018.

[24] Z. Liu, C. Yao, H. Yu, and T. Wu, "Deep Reinforcement Learning with its Application for Lung Cancer Detection in Medical Internet of Things," *Future Generation Computer Systems*. 97: 1-9, 2019.

[25] Z. Jiang, D. Xu, and J. Liang, "A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem," *ArXiv Preprint ArXiv:1706.10059*, 2017.

[26] Z. Jiang and J. Liang, "Cryptocurrency Portfolio Management with Deep Reinforcement Learning," *2017 Intelligent Systems Conference (IntelliSys), London, United Kingdom*, pp. 905-913, 2017.

Yue Ma is a PhD student at Northern Illinois University, where she is pursuing her passion for computer science. She received her Master's degree from Southeast Missouri State University, bringing a wealth of knowledge and experience to her studies. Her research interests include data visualization, augmented reality (AR), and machine learning. She is dedicated to furthering her knowledge, and actively working towards making more publications and contributing to her research fields in meaningful ways.

Ziping Liu received her PhD in Engineering Science from Southern Illinois University at Carbondale in 1999 and began her computing career at Motorola, where she developed software for mobile phones. Currently, she is a professor of Computer Science at Southeast Missouri State University, where she has been teaching since 2001. Her research interests encompass a wide range of topics, including machine learning, cloud computing, secured software design, wireless ad hoc networks and sensor networks, distributed computing and game development.

Charles D. McAllister is currently a professor of business analytics at Southeast Missouri State University. He has previously served the university in such roles as vice provost and dean of graduate studies, interim dean of business, interim chairperson of computer science, and interim director of institutional research. He received a Ph.D. in industrial engineering and operations research from The Pennsylvania State University in 2002, M.S. in industrial engineering from Purdue University in 1999, and B.S. in industrial and management systems engineering from the University of Nebraska in 1998. His research interests include optimization, analytics, simulation, pedagogy, and higher education administration.