Project Report
Portfolio Optimization Using Deep Reinforcement Learning

Bishishta Mukherjee, Konanki Sai Charan, Manvita Markala, Romil Rathi

I. Summary

Financial portfolio optimization is the process of redistributing funds into multiple financial vehicles at a given timestamp to maximize returns while minimizing the risk at the same time. The goal of this project is to provide a solution framework which deals with this complex financial engineering problem. The framework will be implemented using a combination of Machine Learning and Reinforcement Learning (RL) techniques. Built RL framework will be trained and tested on stocks and cryptocurrency trading data.
The dataset consists of:

1) Historical trading data for Stocks: 15 assets from S&P 500 [1] Portfolio from 2005 to present.
2) Historical price data for six Cryptocurrencies from CoinMarketCap [2] from 2015 to present.

The price data follows the format of Open, High, Low and Close (OHLC) for a given trading day. Open is the price at which the stock begins trading, High is the highest value it attains, Low is the lowest value throughout the day and Close is the closing value. Usually, open price is equal to close price for the previous day, but cryptocurrency follows high frequency trading, thus we might not see open price for one day same as closing price for previous days.
To estimate/approximate the profit function, the solution framework will be built using a statistical machine learning model - Convolutional Neural Network (CNN) after which a goal-oriented algorithm - Recurrent Reinforcement Learning will be used to train the mentioned neural network model and make it learn how to maximize the return profit over time.

II. Methods

(a) Asset Pre-Selection

To explore the relationship between the different stocks and cryptocurrencies, heat maps have been plotted to identify the correlation as observed from Fig. 1 and Fig. 2 respectively.
Stocks for some of the companies like Apple, Amazon, Google and Boeing were highly correlated which indicated that the stocks always move in the same direction. This fact was used to select 15 assets with very fewer correlations for the inclusion of the portfolio to further minimize the risk factor. Majority of the times the returns of less correlated stocks move in opposite directions and hence a security factor comes into play if the portfolio is distributed amongst these assets, as the decrease in returns from one asset can be balanced by the increase in the other. Therefore, the portfolio size for stocks is 16 (cash + 15 assets).

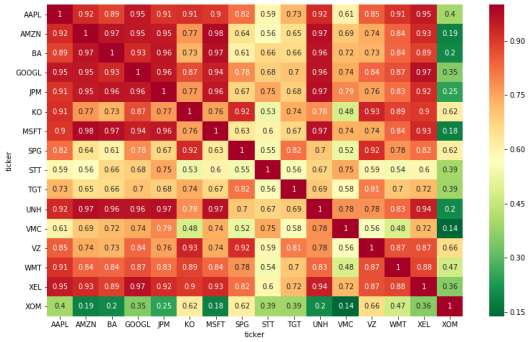

Fig. 1. Correlation plot for Stocks

Six different cryptocurrencies were selected based on the volume of trading on the index that is, the ones which are performing good in the market rather than based on the correlation values between them. The basic reason for doing so is that since cryptocurrency has been introduced in recent years, there is not enough data for the majority of them. In such a scenario, only the top trading ones provide us with sufficient data to build a well-trained framework. Therefore, the portfolio size for cryptocurrency is 7 (cash + 6 assets).

Fig. 2. Correlation plot for Cryptocurrency

(b) Data Pre-Processing

For stocks, the total data collected was for 3671 trading

days and contained no missing values. Therefore, stocks data needed no further tidying. But in case of cryptocurrency, the total data collected varied from 1500 to 2300 trading days approximately because certain cryptocurrencies like Bitcoin were launched as early as 2009 and cryptocurrencies like Ethereum launched much later in the market, around 2015. Due to this wide gap between the launch dates for different currencies, the time window for each of the crypto currency was set to the same dates as that of Ethereum, i.e. from August 2015 to present and hence captured the data for 1514 trading days.

(c) Building Price Tensor

For each of the stocks and cryptocurrencies, 4 lists containing the scraped raw price values for Open, High, Low and Close (OHLC) over the entire time frame have been created and were then normalized by the opening price since only the changes in the prices will determine the performance of the portfolio management rather than the raw prices themselves. The lists have been normalized as follows:

$$\text{Close values list} = \left[\frac{\text{Close}_{(t-n-1)}}{\text{Open}_{(t-n-1)}}, ..., \frac{\text{Close}_{(t-1)}}{\text{Open}_{(t-1)}}\right]$$

$$\text{High values list} = \left[\frac{\text{High}_{(t-n-1)}}{\text{Open}_{(t-n-1)}}, ..., \frac{\text{High}_{(t-1)}}{\text{Open}_{(t-1)}}\right]$$

$$\text{Low values list} = \left[\frac{\text{Low}_{(t-n-1)}}{\text{Open}_{(t-n-1)}}, ..., \frac{\text{Low}_{(t-1)}}{\text{Open}_{(t-1)}}\right]$$

$$\text{Open values list} = \left[\frac{\text{Open}_{(t-n)}}{\text{Open}_{(t-n-1)}}, ..., \frac{\text{Open}_{(t)}}{\text{Open}_{(t-1)}}\right]$$

At the end of period t, the input price tensor to the neural network is of the shape (x, y, z) where x is the number of features(OHLC in case of portfolio management) and y is the number of non-cash assets and z is the trading days considered until time t. This price tensor for stocks data is of the shape (4,15,3670) and that for cryptocurrencies is of the shape (4,6,1513).

(d) Reinforcement Learning Environment Setup

In the solution framework presented, the financial market will be the environment[3] and the portfolio optimisation software will act as an agent allocating funds amongst different assets at a particular instance of time. The action taken by an agent at time t is to reallocate the funds between various assets and thus determines the current weight vector w_t.

$$\boxed{Action_t = w_t = [w_{cash}, w_{asset1}, w_{asset2}, ..., w_{assetn}]} \quad (1)$$

While reallocating the funds i.e. when buying or selling assets we need to account for a transaction cost which is normally a constant commission fee involved for each stock/cryptocurrency traded. Therefore the total transaction cost[4] at time t will be calculated while distributing the funds according to the new weight vector at time t i.e w_t from weight vector at time $t-1$ i.e. w_{t-1}.

$$\boxed{\begin{aligned}&\text{Total transaction cost}_{(t)} = \\ &\text{Portfolio value}_{(t-1)} * \text{Transaction Fee} * (w_{(t)} - w_{t-1})\end{aligned}} \quad (2)$$

Therefore every time the stock portfolio is updated, this transaction cost incurs on the portfolio value i.e. the total funds we have at timestamp t.

$$\boxed{\begin{aligned}&\text{Portfolio vector}_{(t)} = (\sum(\text{Portfolio value}_{(t-1)}) * w_t) \\ &- (\text{Total Transaction Cost} + [0] * \text{number of assets})\end{aligned}} \quad (3)$$

Portfolio vector is a vector of size: (number of assets +1) to store the value for each asset at time t.

$$\boxed{\begin{aligned}&\text{Cumulative Portfolio Vector}_{(t)} = \\ &(\sum(\text{Portfolio value}_{(t-1)}) * w_t) - (\text{Total Transaction} \\ &\text{Cost} + [0] * \text{Number of assets}) * \text{Return rate}_{(t)}\end{aligned}} \quad (4)$$

$$\boxed{\begin{aligned}&\text{Final Weight Vector}_{(t)} = \\ &\text{Portfolio vector}_{(t)}/\text{Cumulative Portfolio value}_{(t)}\end{aligned}} \quad (5)$$

Here return rate at time t contains the rate of interest for each of the asset which the agent will:

1) Get at each step if he has positive amount of money.
2) Pay if we have a negative amount of money.

The reward value at time t is the percentage change in the portfolio value and thus depends on Portfolio value$_{(t-1)}$ and Total transaction cost$_{(t)}$.

$$\boxed{\text{Reward}_{(t)} = (\text{Portfolio value}_{(t)}/\text{Portfolio value}_{(t-1)}) - 1} \quad (6)$$

This dependency is encapsulated by considering w_{t-1} and Portfolio value$_{(t-1)}$ as part of the input to the agent at time t. So the state at time t (state$_{(t)}$) consists of the price tensor at time t, weight vector at time $t-1$ (w_{t-1}) and portfolio value at time $t-1$ Portfolio value$_{(t-1)}$ where $W_0 = [1, 0, 0..., 0]^T$

$$\boxed{\text{State}_{(t)} = (\text{Price tensor}_{(t)}, w_{(t-1)}, \text{Portfolio value}_{(t-1)})} \quad (7)$$

Here, two environments are being setup. An environment for the portfolio optimization agent and a baseline environment for the agent assigning equal weights amongst the assets. Using this setup the adjusted reward for time stamp t is calculated as follows:

$$\boxed{\begin{aligned}&\text{Adjusted Reward}_{(t)} = \\ &\text{Reward}_{(t)} - \text{Baseline Reward}_{(t)} * \alpha * max(w_{(t)})\end{aligned}} \quad (8)$$

Where $\alpha * max(w_{(t)})$ is a term proportional to the maximum of the weight vector at time t. This term is taken into account so that the agent avoids to invest fully in one stock.

(e) Policy network architecture

Convolutional Neural Network is used to design the policy function. The input to this network is the price tensor (Price tensor) and the output is the weight vector $(w_{(t)})$, the portfolio value at t (Portfolio value$_{(t)}$) and adjusted reward at t

(Adjusted Reward$_{(t)}$). The first convolution layer is realised in a small tensor and has 2 filters. The activation function used for the first convolution layer is $Tanh$ for stocks data and $ReLU$ for cryptocurrency data. The second convolution layer is made resulting in a second vector (number of assets * 1 * 1) and has 20 filters. The previous output vector is stacked. The activation function used here is $ReLU$ for both stocks and cryptocurrency data. For the neural network to minimize the transaction cost, portfolio vector at timestamp t-1 is inserted before the last layer. The portfolio vector produced by the network at each time step is stored in a Portfolio Vector Memory to be used at the next timestamp. The last layer is a terminate convolution layer resulting in a unique vector of size same as a number of assets. Then a cash bias is added and a softmax applied. An important feature of this network is that the network parameters are common for all the assets but they are processed independently. When the softmax function is applied, it is ensured that the weights allotted are non-negative and sum up to unity.

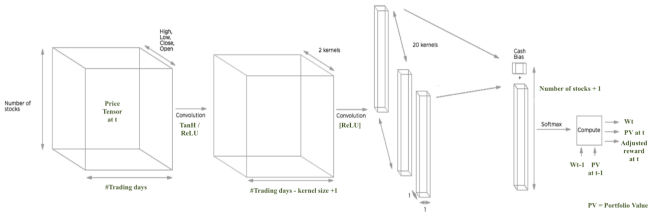

Fig. 3. CNN Architecture for Portfolio Optimization

(f) Evaluation Metrics

The first evaluation metric is based on benchmark testing[5] where the outcomes produced by built optimizer agent will be compared with outcomes produced by other agents. The outcome here is the cumulative portfolio value returned over a period of time and the other agent is the one which distributed the weights equally amongst all the available assets. According to the hypothesis, a good optimizer agent should outperform the equiweight agent. But this strategy does not accommodate the risk factor involved while calculating the returns. Therefore we look into another evaluation metric which involves backtesting and is called the Sharpe Ratio.

Backtesting is a term used in modelling to refer to testing a predictive model on historical data and assess the viability of a training strategy by discovering how it would play out using historical data. It is a key component of effective trading system development.

The two key assumptions[6] made for back-testing are :

1) Zero Slippage : The liquidity of all market assets is high enough that each trait can be carried out immediately at the last price.
2) Zero Market Impact : The capital invested by the software trading agent is so insignificant that it has no influence on the market.

These two assumptions are near to reality in the real world trading environment if the trading volume in the market is high enough.

The Sharpe ratio is a measure of the risk-adjusted return. It is calculated using the following equation:

$$\boxed{S = \frac{R_a - R_f}{\sigma}} \tag{9}$$

where,

S = Sharpe ratio

R_a is the asset return which is nothing but the mean of the weighted daily asset returns. Here the returns are calculated as the percentage change in closing price each day.

R_f is risk free rate of return. For general scenarios in the real world, this component is taken as 0.

σ is the asset volatility which is calculated as the standard deviation of the daily asset returns.

If the Sharpe ratio generated by the current strategy is greater than 1, then it is considered to be fairly good, if it is higher than 2, then it is considered to be very good, if it is higher than 3 then it is considered to be excellent and if it is under 1, the strategy developed is considered to be sub-optimal and has a great scope of further optimization.

III. Results

On testing the performance of optimizing agent over stocks data, the final weights allocated to different assets can be seen in Fig. 4. The highest weight was assigned to Coca-Cola Co. followed by Verizon Communications Inc and Walmart Inc. The least weight was assigned Simon Property Group Inc followed by Xcel Energy Inc.

As seen in Fig. 5 that represents the cumulative portfolio value

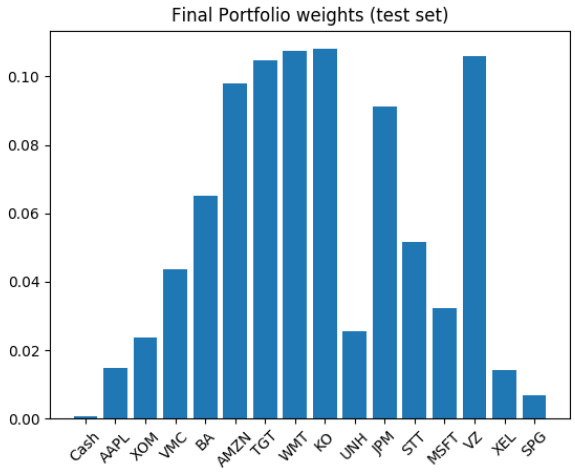

Fig. 4. Weights obtained for stocks portfolio

across test steps, the value returned by the built optimizing agent increased with each test step and performed better when compared to the values returned by the agent assigning equal weights amongst all the assets.

The optimizing agent exhibited an increase of 25-30% in cumulative portfolio value over the initial portfolio value.

The mean Sharpe ratio obtained for the optimizing agent in case of stocks data is 0.735. As mentioned earlier since the

Sharpe ratio is less than 1, the results obtained in Fig. 5 is misleading and the trained agent gives us sub-optimal results for stocks data.

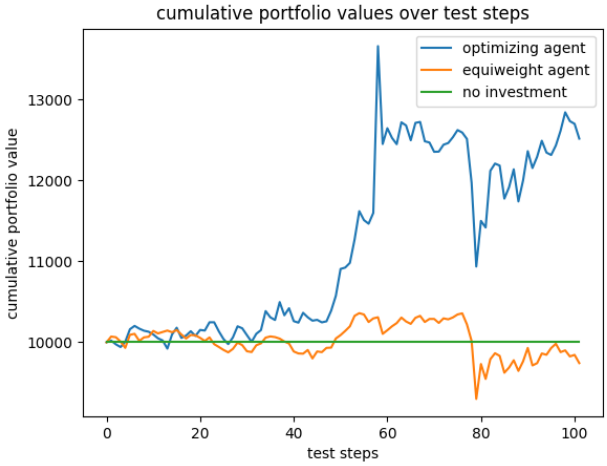

Fig. 5. Cumulative Portfolio Value for Stocks portfolio

On testing the performance of optimizing agent over crypto data, the final weights allocated to different assets can be seen in Fig. 6. The highest weight was assigned to Litecoin followed by XRP(Ripple) and Ethereum. The least weight was assigned to Monero followed by NEM(XEM).

As seen in Fig. 7 that represents the cumulative portfolio value

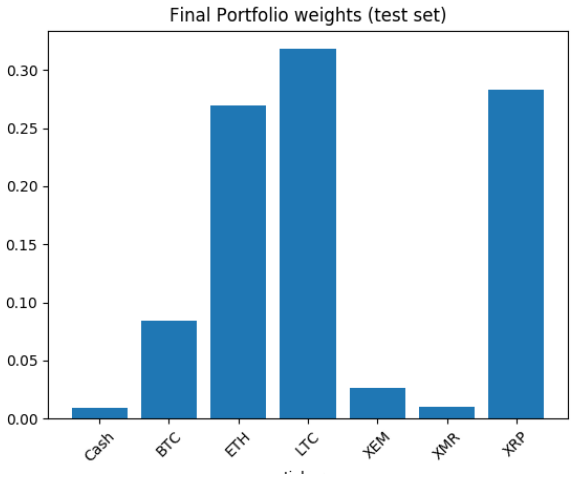

Fig. 6. Weights obtained for Crypto portfolio

across test steps, the value returned by the built optimizing agent increased with each test step and performed better when compared to the values returned by the agent assigning equal weights amongst all the assets. The optimizing agent exhibited an increase of 300-400% in cumulative portfolio value over the initial portfolio value. The equiweight agent also performed well here exhibiting an increase of around 200% in cumulative portfolio value over the initial portfolio value.

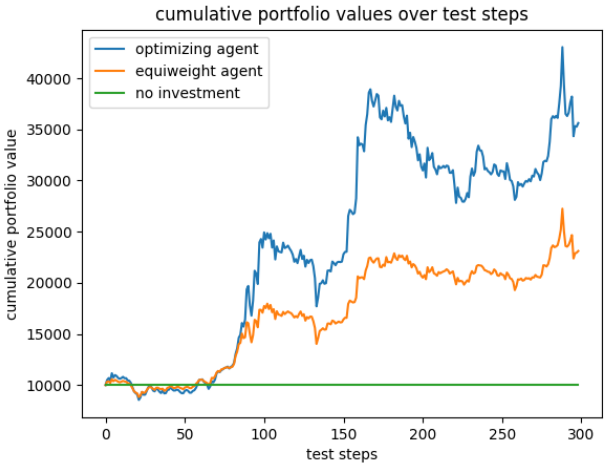

Fig. 7. Cumulative Portfolio Value for Crypto portfolio

The mean Sharpe ratio obtained for the optimizing agent in case of cryptocurrency data is 1.652. As mentioned earlier, since the Sharpe ratio is greater than 1, the results obtained in Fig. 7 is acceptable and the trained agent gives us nearly optimal results for cryptocurrency data.

IV. Discussion

As seen from the results obtained for stocks data, the plot for cumulative portfolio value over test steps gave misleading conclusions that the agent built was optimal but on calculating the mean Sharpe ratio, it was evident that the plot was unable to accommodate the risk factor involved while calculating the returns. While in the case of cryptocurrency data, the result shown in the plot was verified to be true by mean Sharpe ratio and the agent built was near-optimal one. For the second phase of the project, one of the goals would be to achieve a more optimal agent for both stocks data and cryptocurrency data than the one obtained in the first phase by experimenting with a different network architecture like Long Short-Term Memory (LSTM) to build the policy function.

Furthermore, the first phase of the project discretizes the trading period for cryptocurrency data by considering a time window of 24 hours which does not efficiently capture the fact that the exchanges are open 24/7 without restricting frequency of trading and should not ideally be directly applied to portfolio selection problems. But capturing the continuous nature is found to be difficult and even unstable sometimes. Therefore, non-stop markets are ideal for machines to learn in the real world in shorter timeframes and thus the results generated by considering a trading period of every 30 minutes will be explored and tested in the second phase.

Although Backtesting using Sharpe Ratio is a better evaluation metric than benchmark testing as it considers the volatility of the portfolio values but it equally treats upwards and downwards movements. In reality, upwards volatility contributes to positive returns, but downwards to loss. To highlight the downward deviation, Maximum Drawdown (MDD) evaluation metric will also be used. A maximum drawdown (MDD) is

the maximum observed loss from a peak to a trough of a portfolio before a new peak is attained. Maximum drawdown is an indicator of downside risk over a specified time period.

V. Statement of Contribution

Stage of Project	Member Contribution
Data Scraping	Romil Rathi, Sai Charan Konanki
Data tidying and pre-processing	Bishista Mukherjee, Manvita Markala
Building price tensor	Bishista Mukherjee, Manvita Markala
Building the RL environment	Romil Rathi, Sai Charan Konanki
Building the Policy Network Architecture	Bishista Mukherjee, Manvita Markala, Romil Rathi, Sai Charan Konanki
Tuning hyperparameters for the network for stocks	Bishista Mukherjee, Manvita Markala
Tuning hyperparameters for the network for cryptocurrency	Romil Rathi, Sai Charan Konanki
Generating plots for evaluating the cumulative portfolio values	Bishishta Mukherjee, Manvita Markala
Calculating Sharpe ratio	Romil Rathi, Sai Charan Konanki
Documentation (Creating report, presentation and proposals)	Bishishta Mukherjee, Manvita Markala, Romil Rathi, Sai Charan Konanki

References

[1] Historical stocks data of 15 assets from SP 500 portfolio https://www.barchart.com

[2] Historical data for 6 Cryptocurrencies from CoinMarketCap https://coinmarketcap.com

[3] Environment Reference RL Environment for Portfolio Management

[4] Chi Zhang, Corey Chen, Limian Zhang. Deep Reinforcement Learning for Portfolio Management. https://www-scf.usc.edu/~zhan527/post/cs599/

[5] Olivier Jin, Hamza El-Saawy Portfolio Management using Reinforcement Learning Dept. of Computer Science, Stanford, USA. https://pdfs.semanticscholar.org/90ad/8fed17daebaee0a743f9f9847e022739a534.pdf?_ga=2.116929670.1018518321.1572860456-1698436213.1572860456

[6] Zhengyao Jiang, Dixing Xu, and Jinjun Liang A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem. Xian Jiaotong-Liverpool University, Suzhou, SU 215123, P. R. China. https://arxiv.org/pdf/1706.10059.pdf

VI. Appendix

Git link for code: Portfolio Optimization using Deep Reinforcement Learning