Long Short-Term Memory - Monte Carlo approach in High Frequency Data: a Portfolio Optimization approach

Fernando Victoria Valpuesta
Department of Computer Science
University College London
London, UK

Abstract

A Deep Learning-Monte Carlo Sampling method was introduced in the context of Portfolio Allocation for cryptocurrencies. Different LSTM-based Neural Networks were trained on high granularity Limit Order Book data in order to predict future returns. The predictions were then used to perform Monte Carlo sampling to approximate the optimal weights of the portfolio, that maximise the Sharpe Ratio. The training was performed on several contiguous days of high frequency data, the last ten hours of which constitute the testing set. We compare the Deep Learning methods to other portfolio construction methods such as $1/N$, random, and Monte Carlo simulations (directly on training data). The Sharpe Ratio, Expected Return and Volatility were used as metrics to measure how efficient a portfolio is. A comparative analysis using these metrics was then performed, and the Deep Learning methods were found to be outperformed by the $1/N$ portfolio.

1 Introduction

Modern Portfolio Theory (MPT) was pioneered by Harry Markowitz in 1952, ((Markowitz, 1952)) when he proposed an optimisation problem constituting of the selection of the best asset distribution within a portfolio, in order to maximize the expected return, conditionally on any given amount of risk. The main advantage of this method is the diversification in asset selection, as a higher expected return per risk is achieved, rather than when trading a single asset. This complex problem which remains an open challenge over the last decades (Beasley, 2013), falls into the category of non-deterministic polynomial-time hardness (NP-Hard). Following the mean-variance approach introduced by Markowitz, there have been other similar approaches (Milhomem and Dantas, 2020). The mean-variance method and its modifications are an appealing framework because of the computational efficiency they present. However, in (Michaud, 1989), a study on this method, it was concluded that the mean-variance approach can lead to non-optimal asset allocation from a financial perspective. In practice, few models proved to be efficient alternatives to the Markowitz's model. This is because despite having a solid theoretical background, in practice they are not feasible to implement, as the universe of assets is huge and they rapidly become computationally inefficient.

Another field of study that has struggled with optimisation when large-scale problems were presented, is Machine Learning (ML). Indeed, ML algorithms struggle to generalize well on high dimensional challenges. However, the recent advances in hardware that led to a significant increase in computational power, have made ML an efficient alternative to many problems, and in some cases, leading to state-of-the-art performance (Mnih et al., 2013). Machine Learning has been vastly applied to Finance, and has had major impact in Algorithmic Trading, Fraud Detection and Portfolio Management (Rundo et al., 2019). In general, the adaptation of Machine Learning in this domain is a challenging task, as the stock market is known for being volatile, dynamic and non-linear, and many factors affect the price of financial products(securities, commodities and currencies, and, more recently, cryptocurrencies (Kyriazis, 2019)). However, as recent research suggests (Perrin and Roncalli, 2019), Portfolio Allocation is a suitable problem for Machine Learning techniques. In fact, the capability of ML models to process large amounts of data, extract patterns from it and build non-linear relationships, make this technique suitable for this problem.

Portfolio Construction based on the predictive results of Machine Learning models is an active research area. In (Krauss et al., 2017), a Multi Layer Perceptron (MLP), random forests (RAF), gradient-boosted trees(GBT) and other methods were implemented in the context of statistical arbitrage. The price of securities is predicted with these methods, and based on the output, the highest K probabilities are translated to a long-position, and the lowest K to a short position. In (Yang et al., 2019), portfolio construction is divided into two steps: stock prediction and stock scoring. The first step is done by applying extreme learning machine on returns, and with the outputs of this, an optimization model is built with the help of other technical indices. The novel Portfolio Optimization technique introduced in (Freitas et al., 2009) consisted of using autoregressive Neural Networks for the prediction task on returns, and then applied predictive errors for the optimization problem. This approach was empirically proven to outperform the Mean-Variance model, as, for the same amount of risk, it yielded greater returns.

In this work, different approaches to portfolio allocation were presented, ranging from Monte Carlo simulations to Deep Learning methods. These aproaches were compared with the $1/N$ portfolio and a randomly weighted portfolio, which acted as benchmarks. These techniques were applied to high granular Limit Order Book data from the Coinbase Exchange, and the major cryptocurrencies, by market capitalization, were considered. The goal of this research is the exploration of some of the existing methods for Portfolio Allocation, and of some novel Machine Learning, and in particular Deep Learning, approaches.

Firstly, the simple Monte Carlo simulations were introduced, which are methods that aim to test the long term expected portfolio growth. These techniques allow us to vary risk assumptions under the set of parameters and provide the optimal asset weights, in order to maximize or minimize an objective (eg: Sharpe Ratio or Volatility). Next, two Deep Learning models based on different Recurrent Neural Networks architectures were introduced. The Neural Networks utilized Long Short-Term Memory layers as their main component. These are used to predict the Expected Return and to have a better sense of future performance of the cryptocurrencies, which is a crucial step in building a portfolio of risky assets. After these predictions were made, Monte Carlo methods were used to allocate weights based on these predictions, thus, seeking to increase the Sharpe Ratio by setting more portfolio weights to the cryptocurrencies that were predicted to perform better. Lastly, two other techniques for Portfolio Allocation were introduced. The first one, the $1/N$ portfolio, seeks to allocate an equal share of wealth to each of the assets. The second strategy, is a randomly weighted portfolio, which will give us a sense of how well one can do with a random allocation strategy.

To the best of my knowledge, this is the first attempt to combine Deep Learning and Monte Carlo sampling for Portfolio Construction. The novelty of this methodology does not only originate from the above combination of methods, but also because of the universe of assets considered, namely cryptocurrencies. The high granularity of the data adds some points on the novelty of the research, as Portfolio Construction is normally treated with low frequency data.

2 Methodology

2.1 Models

In this subsection, the Markowitz portfolio optimization is briefly discussed and this paper's notations are defined. Assuming that we have N risky assets, the problem is formulated as follows:

$$\max_{w}(\mu^T w - \lambda w^T \sum w), \tag{1}$$

so that:

$$w_i \geq 0, \text{ for i = 1, ..., N}$$

and

$$\sum_{i=1}^{N} w_i = 1,$$

where $\sum$ is a n x n variance covariance matrix, μ is an n-dimensional vector of returns of the assets, w are the weights assigned to each asset $(w_i(i = 1, ..., N).)$, and λ is a positive risk aversion constant.

2.1.1 Monte Carlo Sampling

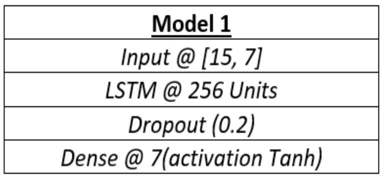
The Monte Carlo methods are computational algorithms that rely on sampling randomly to solve statistical problems. The simulation of this process gives a distribution of the sampled results with different inputs that are sampled many times.
In the purposes of this paper, we sample the weights of the assets of the portfolio randomly. This way,

we obtain the Expected Return and Volatility of randomly sampled portfolios. With sufficient points, we can approximate the desired portfolio (maximum Sharpe Ratio, minimum Volatility, etc) and have an understanding on how the Expected Return relates to the Volatility when sampling weights. This first method was used alone to sample the input data, as opposed to the Deep Learning approach.

2.1.2 Deep Learning approach

Recurrent Neural Networks (RNN) are a type of Neural architecture designed to process sequential data, such as timeseries. Long Short-Term Memory (LSTM) model are used in this case to handle the temporal dynamics of the system. This architecture explicitly models temporal and sequential dynamics in the input information, which helps to capture the temporal dimension that offers the data. LSTM models were proposed specifically to deal with the exploding/vanishing gradient problem. This consists of having a deep network when unrolling the RNN, and when using gradient descent, the error gradient vanishing (or exploding) quickly. For the purpose of this research, the chosen architectures consists of:

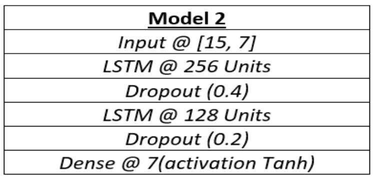
A simple LSTM architecture consisting of an initial LSTM layer with inputs $(15, 7)$ and a total of 256 units. Then, a dropout layer (20%) is incorporated in order to decrease overfitting. Finally, a Dense layer with 7 outputs, one for each currency, is incorporated with an activation function. This is shown in Figure 1



Model 1
Input @ [15, 7]
LSTM @ 256 Units
Dropout (0.2)
Dense @ 7(activation Tanh)

Figure 1: Graphical representation of the first LSTM model.

The second model that relies on LSTMs consists of a stacked LSTM model. This architecture is similar to the previous one, but an extra LSTM layer is incorporated, in order to make the model deeper and more complex. Figure 2 shows the exact architecture.



Model 2
Input @ [15, 7]
LSTM @ 256 Units
Dropout (0.4)
LSTM @ 128 Units
Dropout (0.2)
Dense @ 7(activation Tanh)

Figure 2: Graphical representation of the stacked LSTM model.

These Deep Learning models output the predicted returns of the inputs. The first term in the input (15), corresponds to the window size used for the timeseries input, thus, 15 data points were fed in the Network. The other term (7), corresponds to the cryptocurrencies. For all the sets of data (each corresponding to one cryptocurrency), a series of points were outputted, corresponding to the forecasted returns. As explained in the previous section, Monte Carlo sampling was implemented on the forecasted returns, thus, allocating the weights that maximize the Sharpe Ratio (Expected asset Return divided by Volatility of that asset). The rational behind this, is that we seek to build a portfolio that will maximize the Sharpe Ratio in the future (forecasted). As opposed to the vanilla Monte Carlo sampling explained above, here the sampling is done on the outputs of the Neural Networks.

2.1.3 1/N portfolio

The $1/N$ portfolio is one of the preferred benchmarks to which researchers tend to compare their portfolio allocation techniques, and in many cases, fail to outperform it (DeMiguel et al., 2007). It simply consists in allocating an equal amount of wealth to each of the assets. In this research, 14.3 % of each of the cryptocurrencies is present under this strategy.

2.1.4 Randomly weighted portfolio

As the name indicates, this portfolio randomly allocates weights to the assets. It is always useful to include a random process for comparative purposes in order to have a better sense of how good or bad the proposed methods are.

2.2 Data

High quality Limit Order Book data from the Coinbase Crypto Exchange represents the basis of the research presented in the current paper. This exchange was selected due to its large daily trading volume, which positions it in one of the top Crypto Exchanges in the world. The data retrieval was done with the help of a data provider: Tardis.dev (https://tardis.dev/) – a digital asset data platform that provides Tick-by-Tick Order Book trades and updates. As their limited service provided a noisy but accurate event-by-event snapshot, some cleaning, formatting and processing of data had to be done in order to facilitate the data manipulation.

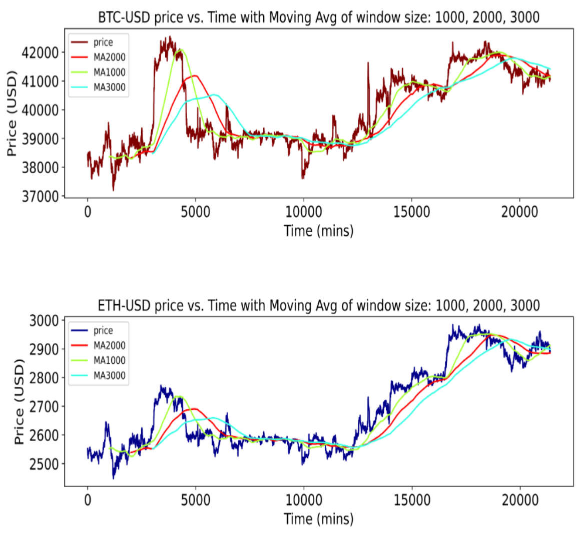


Figure 3: Graphical representation of two min-to-min Timeseries (Top: BTC-USD, Bottom: ETH-USD) fetched from Coinbase Exchange.

Firstly, the data was limited to several days of trading activity, which ranged from the 7th of March 2022 to the 22nd of March 2022, thus spanning 15 days (note that this market does not close overnight). These days were chosen because they provided sufficient datapoints and because it was as close to present as possible, like this, having an analysis that can be of use in the current market trends. Secondly, the number of cryptocurrencies taken into account in the research were limited to seven, who were traded against the U.S. Dollar, which were: Bitcoin (BTC), Ethereum (ETH), Cardano (ADA), Dogecoin (DOGE), Polkadot (DOT), Shiba Inu (SHIB) and Solana (SOL). This selection was done based on their total cryptocurrency market capitalization, which is of around 70%. The data was processed so only the minute-to-minute price, size, side and timestamp of the executed trades of these 7 currencies was selected. Note that a minimum tick size of $\theta = \$0.01$ is present in all cryptocurrencies in Coinbase Exchange. In order to have a better understanding of these timeseries, the returns, Log-returns and moving-window standard deviation were computed.

In Figure 4 correlation between the different

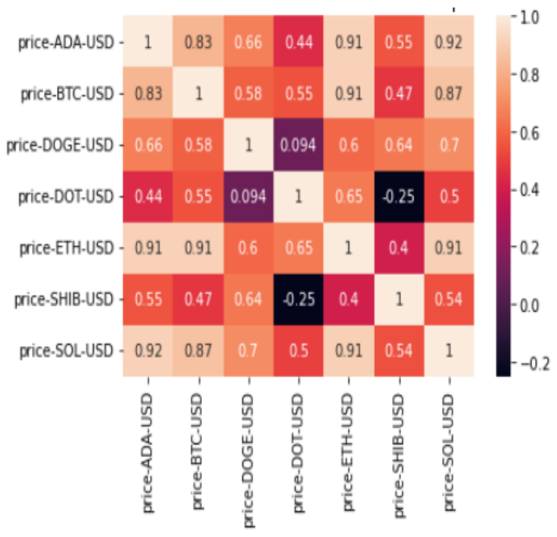


Figure 4: Correlation matrix of the 7 cryptocurrencies prices.

assets in this paper is illustrated. This measurement is crucial in Portfolio Allocation. When it comes to diversified portfolios, the degree of relationship between price movements of different assets included in the portfolio is represented by this. It is worth noting that if the correlation between any of the pairs is $+1$, the prices move jointly, and a -1 means they move in opposite directions. By looking at the correlation matrix, we see that one of the highest correlations is seen between BTC and ETH, which is further seen in Figure 3.

2.2.1 Training and testing pipeline

For each model that required a training process, the following procedure was applied:

The returns data was split in 3 sets: training, validation and testing set. The training set was of dimensions (14,000, 15, 7), which indicated that it trained on 14,000 rows that incorporated the last 15 returns for the 7 cryptocurrencies. This was labeled with the next price in time, thus indicating that is the value it should predict.

The DL algorithms trained on batches of size 64, which were empirically selected as being the optimal. The epochs were set to 50. This was done manually exploring the set of hyperparameters, and it was observed that these epochs performed well, accounting for the computational resources constraint.

The selected optimizer was Adam, and the Learning Rate was set to 0.001. The loss function was set to be the Mean Squared Error. The activation function was set to be $tanh$. This function takes in any real value and outputs values in the range $[-1, 1]$, which is ideal when dealing with returns.

Finally, the total numbers of trainable parameters for the two Deep Learning models were $270,000$ and $470,000$ for the single LSTM and stacked LSTM.

For the testing phase, an offline learning approach was taken. The data was fed to the Neural Networks, and it had to predict the n next steps (indicating the length of the testing set).

3 Results

This section includes the results of the methods and techniques implemented. It ranges from the training and testing processes of the NNets, to the comparison of the different portfolios performances. The weights allocated to each asset in each individual portfolio is also presented. The metrics that are used to analyse the portfolio are the Sharpe Ratio, Volatility and Expected Return.

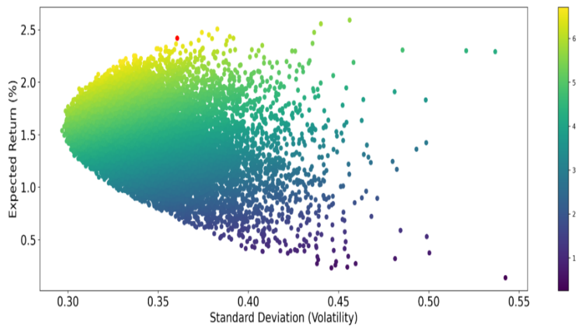


Figure 5: Monte Carlo sampling on the predicted (Single LSTM) Expected Returns of the assets. This shows the Expected Return and Volatility of the predictions, and locates the highest Sharpe Ratio set of weights (red dot). The colorbar corresponds to the Sharpe Ratio, the more yellow it is, the highest the Sharpe Ratio.

For the Machine Learning based models, the prediction task for future returns had to be done before building the portfolio. As seen in Table 1 the Mean Squared Errors fucnction was the loss function that was being monitored by the models. During the training, validating and testing process, the Stacked LSTM model performed slightly better than the single LSTM. As previously mentioned, the datasets to perform these tasks corresponded to 85%, 10% and 5% of the original dataset.

After having tested the two Recurrent Neural Networks, we end up with a prediction of the future returns. With these predictions, further Monte Carlo simulations were performed, as seen in Figure 5 and 6 for single LSTM and stacked LSTM, respectively. This enabled sampling the predictions, and arranging the portfolio weights at time t that will give us highest Returns (or in this case highest Sharpe Ratio) in the future (future length depends on test set, which in this case was t + 10hours).

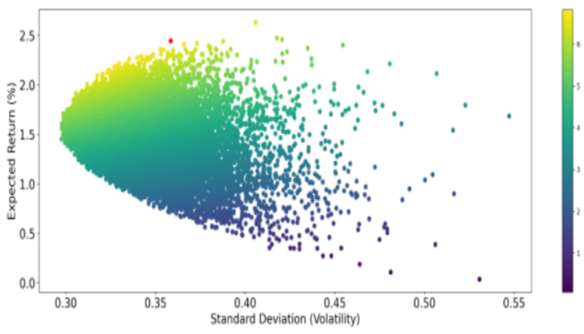


Figure 6: Monte Carlo sampling on the predicted (Stacked LSTM) Expected Returns of the assets. This shows the Expected Return and Volatility of the predictions, and locates the highest Sharpe Ratio set of weights (red dot). The colorbar corresponds to the Sharpe Ratio, the more yellow it is, the highest the Sharpe Ratio.

For the regular Monte Carlo sampling, the training set was taken and sampled in order to locate the Efficient Frontier. With this, the weights that gave the maximum Sharpe Ratio were located and used to build the portfolio. In this case, no prediction was done. Simply, the portfolio percentages were determined based on the training set, and expected to perform well in the future. Figure 7 reflects the simulations in this case.

For the $1/N$ portfolio, the weights were located equally in all assets. Around 14.3 % of each asset was present in the portfolio. For the randomly allocated portfolio, the weights were randomly selected, so each time the weights were re-computed, a different performance was seen.

Finally, the weights for the simple Monte Carlo method, single LSTM and stacked LSTM

Model	Training MSE	Validation MSE	Testing MSE
Single LSTM Neural Netwotk	3.02 %	3.23 %	3.45%
Stacked LSTMs Neural Network	**2.97 %**	**3.13 %**	**3.33%**

Table 1: Results for the Mean Squared Error during Training, Validating and Testing sets for the two NNet models.

Portfolio allocation technique	Anual Expected Return	Anual Expected Volatility	Anual Sharpe Ratio
Monte Carlo	652%	46.5%	14.0
Single LSTM Neural Network	1120%	45.8%	24.4
Stacked LSTM Neural Network	1804%	45.7%	39.5
1/N	**3136%**	**41.9%**	**74.9**
Random	186%	50.1%	3.7

Table 2: The Anual Expected Return, Volatility and Sharpe Ratio for the Monte Carlo sampled portfolio, single LSTM, stacked LSTM, 1/N and random.

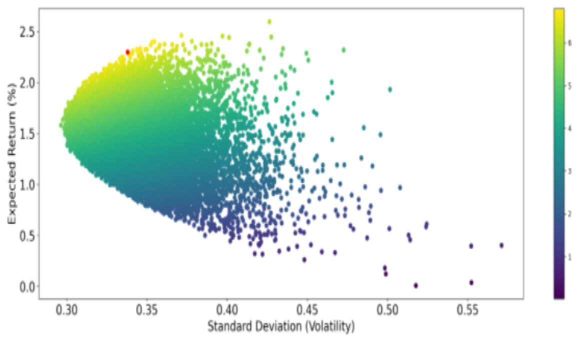


Figure 7: Simple Monte Carlo simulations on training set. This shows the Expected Return and Volatility of the predictions, and locates the highest Sharpe Ratio set of weights (red dot). The colorbar corresponds to the Sharpe Ratio, the more yellow it is, the highest the Sharpe Ratio.

were derived. These weights were tested on the actual testing set. This way, the performance of each strategy was seen. Figure 8 captures the min-to-min return of the strategies on the testing set (10 hours). On table 2, the anual Expected Return, Volatility and Sharpe Ratio were calculated. The best performance appears in bold.

By looking at Table 2, it is seen that there is a clear winner in all metrics (indicated in bold), which is the $1/N$ portfolio. Indeed, the equally weighted portfolio achieves an anual expected return of 3136%, which is nearly double as the second best-performing strategy. It also manages to have the lowest anual expected volatility (41.9 %), which combined, yields the highest Sharpe Ratio of the five portfolios: 74.9.

The Deep Learning based models are the following best-performing strategies out of all, followed by the stacked LSTM and then the single LSTM. For the stacked LSTM, a Sharpe Ratio of 39.5 was achieved, which was calculated by dividing the anual Expect Return of 1804 % by the Volatility, 45.7 %. The single LSTM model on the other hand, achieved a smaller Sharpe Ratio (24.4), a smaller Expected Return (1120 %), but a bigger Volatility (45.8 %).

As for the Monte Carlo simulations, a Sharpe Ratio of 14 was achieved, with the Expected Returns being 652 % and Volatility 46.5%, which ranks the second highest. The worst performing portfolio in terms of Expected Return, Volatility and Sharpe Ratio was the random portfolio. A Sharpe Ratio of 3.7 was achieved, while a 186 % Expect Return and 50.1 % Volatility were expected. Note that these are the lowest returns and Sharpe Ratio, and the highest Volatility.

4 Discussion

To compare performances we consider the following perspectives on the portfolios: Sharpe Ratio, Expect Returns and Volatility on the testing set. The metrics are then transformed to anual rates, thus giving the reader a better understanding of the performances on a long-term horizon.

Table 2 presents the results of the three metrics shown above for the different portfolios. The performance ranking in terms of Sharpe Ratio is:

Portfolio allocation technique	**ETH**	**BTC**	**SHIB**	**DOGE**	**ADA**	**DOT**	**SOL**
Monte Carlo	18.5 %	16.3 %	0.7 %	16.9 %	40.6%	0.3%	6.5%
Single LSTM Neural Network	23.5%	19.6 %	1.3 %	15.1%	29.7%	0.9%	9.9%
Stacked LSTM Neural Network	17.2%	13.9%	2.0%	23.6%	34.6%	1.7%	6.8%
1/N	14.3%	14.3%	14.3%	14.3%	14.3%	14.3%	14.3%
Random	23.2%	11.0%	13.6%	26.4%	0.9%	24.4%	0.4%

Table 3: The weights (as percents of portfolio) of each allocation strategy. It is seen that the three first strategy allocate big proportions of the portfolio to ETH, BTC, DOGE and ADA.

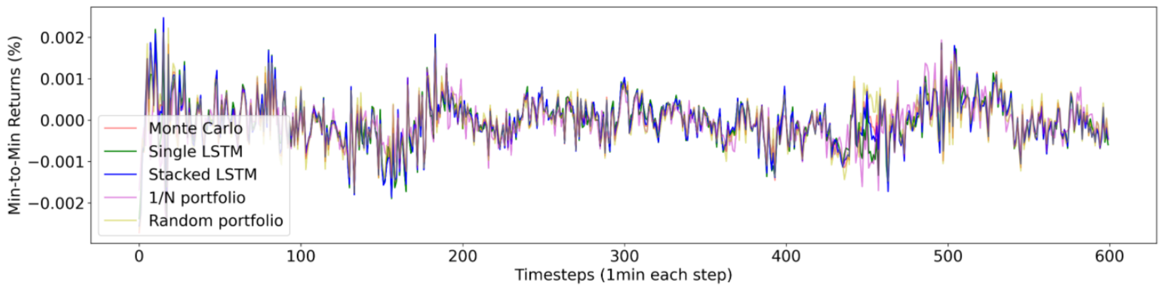


Figure 8: Portfolio min-to-min returns (performance) on the testing dataset. The types of strategy to allocate correspond to Monte Carlo random sampling, Single-LSTM, Stacked-LSTM, 1/N portfolio and Random portfolio

- 1/N portfolio
- Stacked LSTM portfolio
- Single LSTM portfolio
- Monte Carlo sampling portfolio
- Random portfolio

An interesting point to note from the allocation weights is that Monte Carlo, single LSTM and stacked LSTM have a similar behaviour. All three models agree on having a bigger % of portfolio located on ETH, BTC, DOGE and ADA, while having single digit percentages on the other assets.

We conclude that the best portfolio for the given set of data is the $1/N$ portfolio. In many cases, as previously mentioned, this portfolio is indeed the best performing one, and beats fancy optimization techniques. This naive diversification is possibly the best because it does not depend on any previous data, and does not fall into the overfitting problem. On the other hand, a reason for the poor performance of the Deep Learning models can be due to overfitting the data.

Indeed, one of the issues of this analysis was the data, which was not abundant. The datapoints that were taken into account in all of the experiments were around 16,000 per cryptocurrency, which is not a big quantity when comparing to some other Deep Learning models that require Terrabytes of data to outperfom other models. Thus, one potential explanation for the poor performance of the models is this. With a bigger set of datapoints, the predictions would have been more accurate and a longer time horizon could have been considered.

This leads to another of the issues of this research. The time horizon that is considered is too short. By this, it is meant that it's not practically feasible to reconstruct a portfolio based on a forecast of 10 hours, as the trading fees are non-zero, and the more one trades, the more the fees are. If the forecasts at a certain time are positive for a given set of assets, and at the next time, other totally different assets are positive, one would have to sell and buy every 10 hours, which could lead to more trading fees than profit. One possibility of tackling this issue would have been treating the Min-to-Min data as End-of-Day data, and analyse the portfolios based on this modification. This way, the 10 hours horizon could have been more than a year long.

In general, the main limitation of the research was the data, which was not abundant. This lead to, most probably, overfitted Deep Learning models, and to portfolio construction techniques. With more data, the NNets would have generalised better, and longer time horizon could have been considered, with more difference in the portfolio performances. The assumptions taken, such as no-fee trades and no market impact were needed in order to model this.

On the other hand, the LSTM + Monte Carlo sampling proposes a novel approach to portfolio construction, to the best of my knowledge. This leads to several topics that could be explore in the future:

Firstly, there is room for improvement in the Neural Networks architectures. In this paper, only two models have been explored, which built on top of LSTM layers. The recent advances in time series analysis with Deep Learning could be explored and applied to this setting, thus seeking to add complexity to the models and better capture the temporal and sequential dynamics of the input data. As it could be seen, the more complex model outperformed the other ML model, so a similar outperformance could be noted by adding complexity to the model.

Furthermore, one could explore adding more inputs to the Networks. This paper only focuses on predicting returns based on input returns, but other variables such as size of order or time of the day could be included in the inputs.

Another interesting topic to study would be to include more assets in the analysis. In this case, only cryptocurrencies were taken into account. As seen previously, these were highly correlated and lead to similar performances. One could include U.S. Treasury Bonds, which would bring stability (less volatility) to the portfolio, and is not correlated to the universe of cryptocurrencies. It would be an interesting analysis to perform this with a different range of asset class and to see when the ML algorithms prefer crypto investing, as opposed to more traditional stock investing.

Finally, this paper is the initial work done for the Dissertation of the MSc Machine Learning at University College London. The work will explore Heuristic Optimization methods for portfolio construction in Decentralized Finance. If time permitting, Reinforcement Learning agents will be explored in this setting. As the Dissertation will be carried out in partnership with a company, emphasis will be given to the practicality of the solution.

References

John Beasley. 2013. Portfolio optimisation: Models and solution approaches. pages 201–221.

Victor DeMiguel, Lorenzo Garlappi, and Raman Uppal. 2007. Optimal Versus Naive Diversification: How Inefficient is the 1/N Portfolio Strategy? *The Review of Financial Studies*, 22(5):1915–1953.

Fabio D. Freitas, Alberto F. De Souza, and Ailson R. de Almeida. 2009. Prediction-based portfolio optimization model using neural networks. *Neurocomput.*, 72(10–12):2155–2170.

Christopher Krauss, Xuan Anh Do, and Nicolas Huck. 2017. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the sp 500. *European Journal of Operational Research*, 259(2):689–702.

Nikolaos A. Kyriazis. 2019. A survey on efficiency and profitable trading opportunities in cryptocurrency markets. *Journal of Risk and Financial Management*, 12(2).

Harry Markowitz. 1952. Portfolio selection. *The Journal of Finance*, 7(1):77–91.

Richard O. Michaud. 1989. The markowitz optimization enigma: Is 'optimized' optimal? *Financial Analysts Journal*, 45(1):31–42.

Danilo Milhomem and Maria Dantas. 2020. Analysis of new approaches used in portfolio optimization: a systematic literature review. *Production*, 30.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602.

Sarah Perrin and Thierry Roncalli. 2019. Machine learning optimization algorithms amp; portfolio allocation.

Francesco Rundo, Francesca Trenta, Agatino Luigi di Stallo, and Sebastiano Battiato. 2019. Machine learning for quantitative finance applications: A survey. *Applied Sciences*, 9(24).

Fengmei Yang, Zhiwen Chen, Jingjing Li, and Ling Tang. 2019. A novel hybrid stock selection method with stock prediction. *Applied Soft Computing*, 80:820–831.