Portfolio Analysis: Backtesting, LSTM Supervised Learning, and DQN Reinforcement Learning Strategies for Coca Cola Stocks

Counselor: Diana Elisabeta Aldea Mendes

Student : Pavlo Vanat 108089

Lisboa 2023

Abstract

Predictions on stock market prices are a great challenge because it is an immensely complex, chaotic, and dynamic environment. In the era of big data, predicting stock prices using machine learning has become popular among financial analysts since the accuracy of predictions can be improved using these techniques. Reinforcement learning offers distinct advantages in portfolio management compared to traditional machine learning approaches because it excels in dynamic and evolving environments, adapting its strategies over time to maximize returns in response to changing market conditions. In this paper, an optimized trading strategy with moving averages, Long Short-Term Memory networks (LSTM), and Deep Q learning has been employed to analyze portfolio returns by trading Coca Cola company shares in both long and short directions, utilizing data from 12-21-2018 to 23-12-2023 based on price history and technical indicators. For this goal, a prediction model was built, and a series of experiments were executed, and their results analyzed against several metrics to assess the performance of these algorithms. The results indicate that the Deep Q Network (DQN) demonstrates superior capacity in managing the portfolio, outperforming the buy-and-hold strategy by nearly 20%. However, it is important to note that these results may be misleading, and further in-depth research on the data and suitable models needs to be investigated.

Introduction

Predicting stock market outcomes can be highly lucrative if successful, yet it is renowned for its inherent difficulty. The multitude of variables and information sources involved, coupled with a significant signal-to-noise ratio, makes forecasting future stock prices a challenging endeavor. Nevertheless, numerous scientists are actively engaged in addressing this challenge, presenting a diverse array of approaches in pursuit of this goal.

This work endeavors to apply simple backtesting with moving averages, alongside Machine Learning algorithms like Long Short-Term Memory networks (LSTM) and Deep Q Learning, to learn from historical data and formulate a trading strategy for shares of Coca Cola companies. The LSTM network, known for its success in distinguishing between recent and earlier examples, is employed due to its ability to assign varying weights to each, thereby discerning relevant information for predicting the next output. Additionally, stock market prediction, involving sequential decision-making based on historical data and market conditions, aligns with the capabilities of the Deep Q Network (DQN), designed for such dynamic financial market modeling.

Historical closing prices of USA equity Coca Cola will serve as the primary data source for the models. Technical indicators will complement this dataset, enriching the network's features. The models will undergo training, evaluation, and optimization processes to enhance portfolio returns.

The project's objective is to investigate the applicability of Reinforcement Learning in comparison to backtesting and supervised learning. The focus is on assessing its performance in terms of portfolio valuation.

Background and related work

When it comes to trading with any speculative instrument, there are constant debates on the predictability of returns. The Efficient-Market Hypothesis (Malkiel, 1970) and the Random Walk Hypothesis (Malkiel, 1973) posit that stock price changes independently of its history. In other words, tomorrow's price will only depend on tomorrow's information, regardless of today's price. On the other hand, some authors claim that stock prices can be predicted to some degree (MacKinlay, 1990). It's worth mentioning that, in 2012, approximately 85% of trades within the United States' stock markets were performed by algorithms (Kissell, 2013).

Although there are multiple ways to predict stock markets, the most common ones use technical indicators, which focus on historical trading data; fundamental data, which focuses on financial statements such as a company's revenue; sentimental data, which focuses on investor sentiment; or breadth data, which reflects the number of stocks participating in each move in an index.

Technical analysis uses past market data to predict price directions and is based on statistical methods to identify patterns. They can be categorized into four major types: trend, momentum, volume, and volatility (Dahlquist, 2006). Moving averages (MAs) are the oldest indicators among various momentum indicators. Research indicates that trend-chasing MA strategies are more profitable for firms in the introduction and shakeout/decline stages characterized by greater information uncertainty and, thus, longer underreaction-driven price continuation (Kuan-Hau Chen, 2020).

In terms of computational intelligence, many studies have combined financial transactions with machine learning to estimate stock prices or predict market movements. To enhance prediction accuracy, artificial neural networks and deep learning have been employed in financial markets. Due to the time series characteristics in the financial market, where each price is affected by the long-term or short-term, the Long Short-Term Memory (LSTM) algorithm has found diverse applications and has demonstrated greater accuracy than traditional machine-learning methods (Jimmy Ming-Tai Wu, 2020). Therefore, the decision was made to utilize LSTM to estimate the winning rate of our trading strategy.

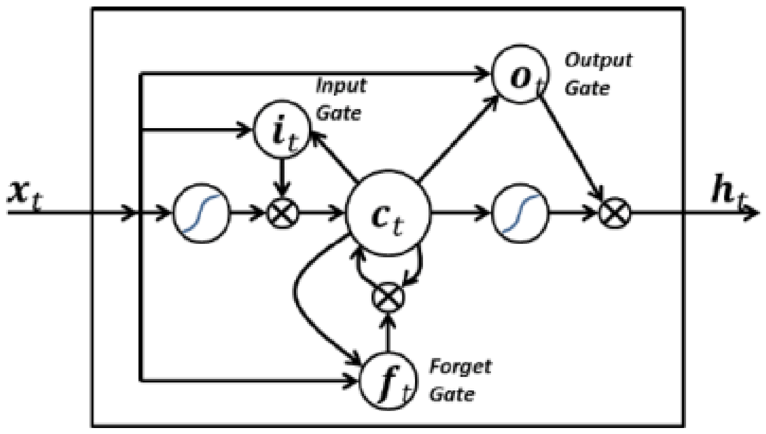


Figure 1. Long short-term memory neural network (K. Greff, 2015)

Long Short-Term Memory (LSTM) networks (see Figure 1), utilized in this project, represent a type of deep and recurrent neural network model. Unlike traditional feed-forward networks, recurrent networks feature neural connections that extend beyond a single direction, allowing neurons to transmit data to the previous or the same layer. In this setup, data doesn't follow a unidirectional flow, resulting in the practical effect of incorporating short-term memory, complementing the long-term memory inherent in neural networks because of training. LSTMs were introduced by Schmidhuber (Schmidhuber, 1997) with the aim of addressing the vanishing gradient issue encountered by recurrent networks when handling long sequences of data. This challenge arises due to the diminishing influence of gradients as they propagate backward through the network during training. LSTMs mitigate this issue by maintaining a constant error flow through specialized units known as "gates." These gates facilitate weight adjustments and enable the truncation of the gradient when its information is deemed unnecessary (Nelson, Pereira, & Oliveira, 2017).

Reinforcement learning (RL) is a subfield of deep learning that is distinct from other fields such as statistical data analysis and supervised learning. It is a strategy that seeks to maximize profits while adapting constantly to changes in the environment in which it operates. Supervised learning's objective is to condense the mapping from input to output, which frequently results in the omission of critical information. Due to the frequency and duration of stock price fluctuations, reinforcement learning (RL) is a more suitable predictive tool for statistical analysis of data than supervised learning.

RL can be classified into five categories based on its sample efficiency: (1) model-based, (2) off-policy, (3) actor-critic, (4) on-policy, and (5) evolutionary gradient-free. In this case, 1 is the most efficient sample size option, while 5 is the least efficient. Among them, model-based and non-policy methods, such as TD-learning and *Q*-learning, are referred to as “the critic-only methods” and are used to solve discrete area optimization problems. (Junhao Zhang, 2022)

In RL, the agent chooses which action to take in order to maximize his or her reward. This means that throughout the learning process, the agent maximizes the accumulated rewards and thus develops optimal strategies for a variety of problems.

This article will analyze and predict data using one machine learning frameworks: “critic-only DQN”. DQN converts a single-layer network to a Recurrent network with multiple layers. It not only enables experience replay but also enables the network to self-train using its memorized history, which more closely may match the time autocorrelation and intertemporal correlation that have emerged in the American market (Junhao Zhang, 2022)

Methodology

All methods were employed to assess the return on trading Coca Cola's stocks from December 21, 2018, to December 23, 2023. Each strategy involved either longing or shorting its shares, executed with a full account.

For backtesting, a straightforward strategy was devised utilizing two moving averages. This strategy generated sell or buy signals based on varying sets of moving average lengths for sell and buy decisions. The process included parameter optimization, ultimately leading to the selection of the most profitable strategy.

For the machine learning strategy, several regression models based on Long Short-Term Memory (LSTM) were formulated to predict stock prices, and the most effective model was selected. The objective was to determine whether the stock's trend would rise or fall the next day, and actions were chosen based on the predicted trend daily. The models were trained on the following features: volume, 5-10-20-50-100-200-365 moving and exponential moving averages, relative strength index, Bollinger Bands, On-Balance Volume, William Oscillator, MACD, VIX, trend slope of 2, 4, 6, 12, 24, 48-day periods, and yearly, monthly, and weekly time cycles encoded with sinusoidal functions. Also, varying amounts of price lags were considered. The model was tasked with predicting the closing price for the next five days, and the binary trend was subsequently calculated.

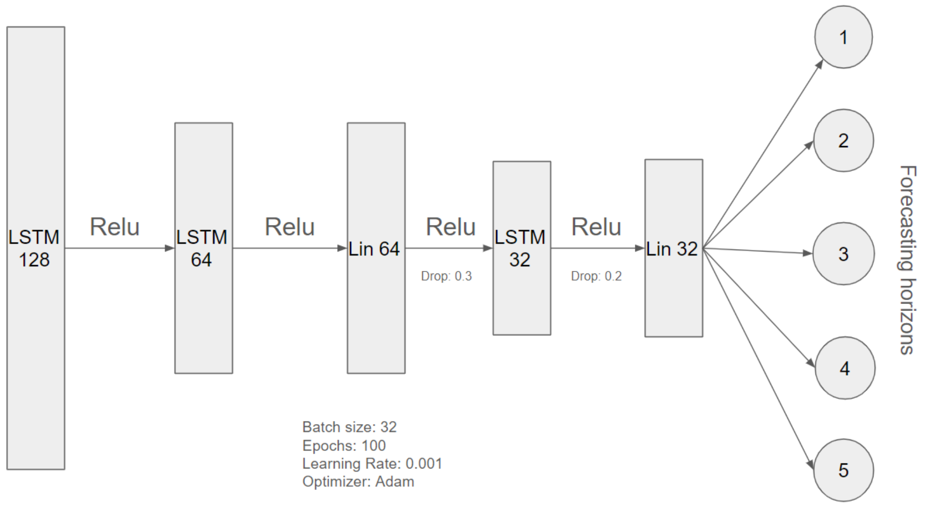


Figure 2. Long short-term memory neural network (Vanat, 2023)

The neural network architecture depicted in Figure 2 was adapted from my previous work, where it achieved an 80% accuracy in predicting the SPX trend. However, given the omission of crucial indicators in this study, its performance was not anticipated to be robust. Instead, it serves as a comparative example to assess against the DQN model. The best model was selected using MAPE and trend accuracy metrics and from there portfolio was simulated. Also LSTM model uses 80% of data for training, 10% of data for validation and 10% of data for testing and many trials different features, unistep and multistep prediction.

The main idea behind Q-learning is that if we had a function $Q*:State \times Action \rightarrow \mathbb{R}$. that could tell us what our return would be, if we were to take an action in a given state, then we could easily construct a policy that maximizes the reward:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \ Q^*(s,a)$$

Equation 1. Policy formulation in Q-learning

And then, we can design a deep neural network as a universal function approximator. And we can train it to resemble Q∗. The following steps outline our methodology:

- **Environment Setup:** A trading environment was established using historical market data, featuring key attributes such as volume; percentage price differences from 10, 20, 50, and 100 moving and exponential moving averages; Bollinger Bands; On-Balance Volume (OBV); William Oscillator; Moving Average

Convergence Divergence (MACD); VIX; trend slope for periods 2, 4, 6, 12, 24, and 48; and weekly, monthly, quarterly, and yearly time cycles encoded with sinusoidal functions. Additionally, cumulative returns over periods from 2 to 20 were incorporated. In contrast to LSTM networks, the goal was not to predict closing prices but rather the logarithmic return of the next day. Consequently, all features were transformed into stationary. The data input into the network was multivariate by N length which was hyperparameter for us to tune.

- **Action and State Representation:** Actions were defined as buying, selling, or holding shares, and the state representation included features from the environment setup. Additionally, it incorporated the previously one-hot encoded action taken in the environment.
- **Deep Q-Network Architecture:** A neural network architecture for the Q-function approximation was adapted from an external work that showed some promising results (Wang 2021). The network comprised layers with specified neurons and activation functions that can be consulted on Figure 3.
- **Experience Replay:** Experience replay was implemented to store and randomly sample past experiences, mitigating temporal correlations in the training data.

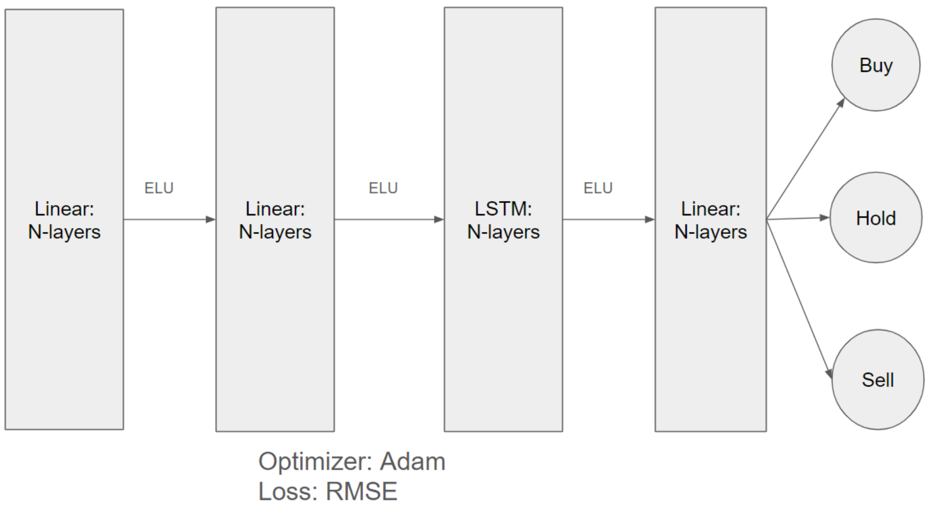


Figure 3. DQN Architechture

- **Target Q-Network:** To enhance training stability, a target Q-network was incorporated, periodically updating target Q-values. This auxiliary network, separate from the primary Q-network, mitigates issues associated with the moving target problem in reinforcement learning. In addition to the target Q-network, a policy network played a crucial role. The policy network determined the agent's actions based on the current state and the learned Q-values. Its purpose was to facilitate effective decision-making during both training and inference, contributing to the overall reinforcement learning framework. The policy network's parameters were updated iteratively, aligning with the reinforcement learning process to optimize the agent's behavior over time. This combination of a target Q-network and a policy network worked in tandem to improve the stability and performance of the Deep Q-Learning (DQN) algorithm. Number of layers was the hyperparameter to tune. Both networks were configured with combination of quantile losses: 0.95, 0.5, 0.05

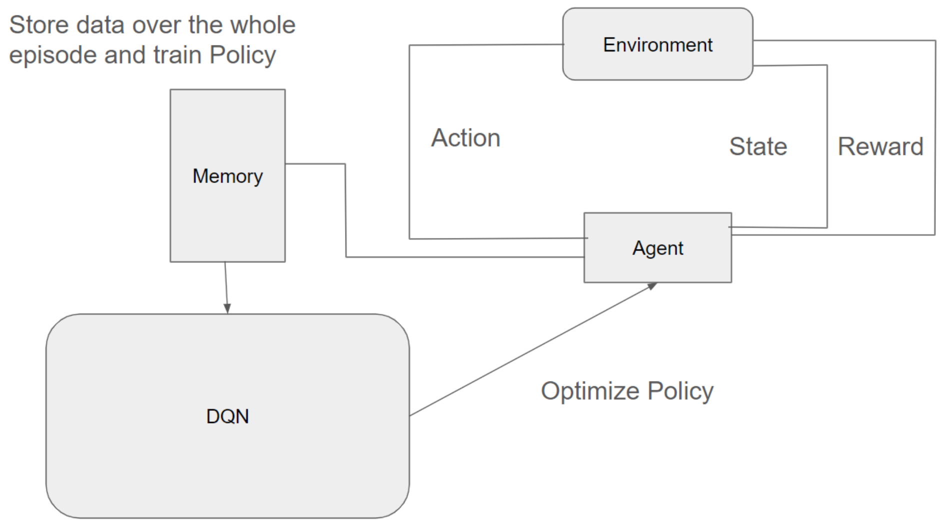


Figure 4. DQN training process

- **Training Process:** The Q-network was initialized and trained over multiple episodes. For each episode, the agent gave the whole cycle of interaction with environment which selected actions based on exploration-exploitation strategy. Hyperparameters were fine-tuned with Bayesian optimization to increase performance.
- **Exploration-Exploitation Strategy:** The epsilon-greedy strategy was employed to manage the exploration-exploitation tradeoff during training. It was logarithmically decaying based on the number of episodes that was previously selected. The number of episodes was another hyperparameter for us to tune.
- **Reward Function:** A logarithmic reward function was defined to evaluate the agent's performance, encouraging the learning of profitable trading strategies.
- **Soft Update: a** manual soft update for the Q-networks was implemented every n step which was the hyperparameter to tune. The purpose of a soft update is to stabilize the training process by gradually adjusting the parameters of the target Q-network. Instead of directly copying the parameters from the policy (online)

network to the target network, a fraction (gamma) of the target network's parameters is updated softly, preventing abrupt changes.
- **Training and Evaluation Episodes:** The model underwent training for a specified number of episodes, with convergence determined through evaluation metrics. Another validation environment was employed to assess the model's performance.
- **Performance Metrics:** final portfolio return was a metric to evaluate the performance
- **Implementation Tools:** The implementation was carried out in Python using PyTorch and Optuna.

This methodology outlines the steps taken to implement and train the DQN model for predicting and trading Coca Cola's stocks. The approach considered both the architectural aspects of the neural network and the reinforcement learning techniques employed. The performance was evaluated with portfolio returns over given number of states.

Data Description

Data for the S&P 500 stock was downloaded from "Yahoo! Finance" covering the period from December 21, 2018, to December 23, 2023, to conduct experiments using the features mentioned above. Unfortunately, I didn't have much time or patience to dedicate toward data processing for the following reasons:

- It's highly unlikely to make decent predictions by solely considering price actions, especially given that our work focused on working with technical indicators, based on my experience and previous projects (Vanat, 2023)
- Every feature needed to be analyzed for causality, which I have no experience with, but I will certainly incorporate it in future research. The tests for causality include Granger Causality, Correlation Analysis, Directed Acyclic Graphs, Instrumental Variables Analysis, and the use of machine learning algorithms specifically designed to search for causality in time series. Additionally, I plan to employ Counterfactual analysis, quantile observation, and extreme event analysis. In the context of financial markets, quantile observation and extreme event analysis involve examining how prices or indicators behave during extreme conditions. This can be valuable for understanding risk, developing trading strategies for extreme events, and identifying potential opportunities in markets characterized by extreme behavior.

- Ideally, predictions should be supported by sentimental, breadth, and fundamental data.
- Another crucial consideration is that the data should guide the determination of neural network architectures. If we identify patterns through the aforementioned analysis, we must ensure that these patterns are effectively captured by the machine learning algorithm. Unfortunately, I currently lack the necessary expertise in this area.

For both LSTM and DQN, the data was logarithmized, normalized, and scaled. In the context of financial analysis, presented below are the formulas for the specified financial indicators and data transformations:

Relative Strength Index (RSI)

$$RSI = 100 - \frac{100}{1 + RS};\ where\ RS = \frac{Average\ Gain}{Average\ Loss}$$

Equation 2. Relative Strength Index (RSI)

Bollinger Bands

$$Upper\ Band = SMA + (2 \cdot S\tan d\,ard\ Deviation)$$

$$Lower\ Band = SMA - (2 \cdot S\tan d\,ard\ Deviation);\ where\ SMA\ is\ the\ Simple\ Moving\ Average$$

Equation 3. Bollinger Bands (BB)

Moving Average (SMA)

$$SMA = \frac{Sum\ of\ prices\ over\ N\ periods}{N}$$

Equation 4. Simple Moving Average (SMA)

Exponential Moving Average (EMA)

$$EMA = \alpha \cdot Close_t + (1 - \alpha) \times EMA_{t-1};\ where\ \alpha = \frac{2}{Number\ of\ periods + 1}$$

Equation 5. Exponential Moving Average (SMA)

Moving Average Convergence Divergence (MACD)

$$MACD\ Line\ =\ ShortTerm\ EMA\ -\ LongTerm\ EMA$$

$$Signal\ Line\ =\ x\ day\ EMA\ of\ MACD\ Line$$

$$MACD\ Histogram\ =\ MACD\ Line\ -\ Signal\ Line$$

Equation 6. Moving Average Crossover divergence (MACD)

Volatility Index (VIX)

$$VIX\ =\ rollingStaNdardDeviation(Logarithmic\ returns,\ window\ =\ 7)\ \times \sqrt{7}$$

Equation 7. VIX formula. Rolling standard deviation of logarithmic returns with a window of 7 periods

Trend Slope

$$Trend\ Slope\ =\ \frac{Close_t - \ Close_0}{t\ -\ 0}$$

Equation 8. Trend slope formula

Sinusoidal Encoding of Time Cycles

$$Sinusoidal\ Encoding\ =\ \sin\left(\frac{2\pi\ \times t}{Cycle\ period}\right)$$

Equation 9. Sinusoidal encoding

Logarithmization

$$Logarithmization(x)\ =\ \log_{10}(x)$$

Equation 10. Logarithmization

Normalization (Z-Score)

$$ZScore\ =\ \frac{Value\ -\ Mean}{Stadart\ Deviation}$$

Equation 11. Normalization

Min-Max Scaling

$$MinMaxScaling\ =\ \frac{Value - MinValue}{MaxValue - MinValue}$$

Equation 12. MinMaxScalling

Results and Interpretation

Figure 5 shows the portfolio return of the optimized backtesting model. The best entry parameters were defined by the cross of the 30-day and 70-day moving averages, while the exit parameters were found using the 15-day and 160-day moving averages. The model underperformed the buy-and-hold strategy by 20%.

Figure 6 illustrates the portfolio return of the optimal LSTM model, where each action was taken based on the model's predictions for each observation. Unfortunately, the portfolio incurred a 50% loss over the specified period. Interestingly, as depicted in Figure 7, a counterintuitive observation emerges: by adopting the opposite strategy to the algorithm's predictions, a 5% outperformance against the buy-and-hold strategy is achieved. However, it is crucial to note that this does not imply identical future returns, as discussed previously.

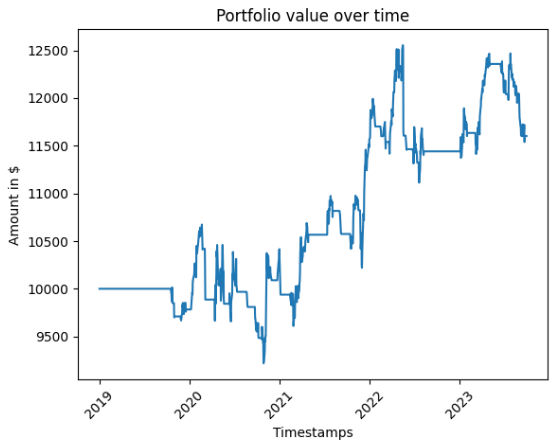


Figure 5. Portfolio Valuation of optimized backtesting model

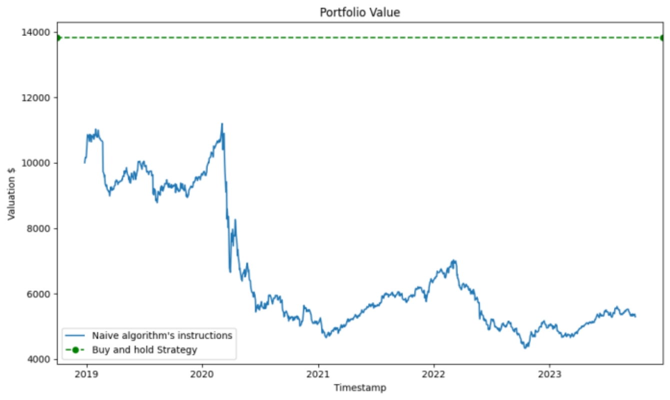


Figure 6. Portfolio Valuation of LSTM model

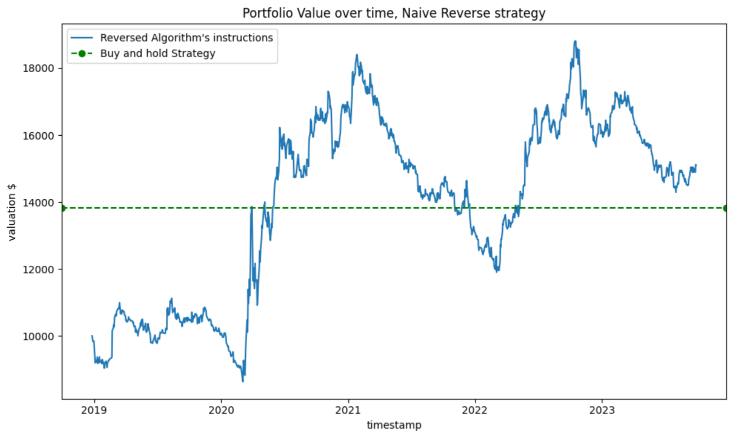


Figure 7. Portfolio Valuation: LSTM Model - Reverse Strategy

The LSTM model underwent optimization with respect to the optimal number of lags. It is noteworthy that all models exhibited a shared characteristic—significant fluctuations and jumps in the validation loss. This phenomenon is likely attributed to the limited size of the dataset, impeding the model's ability to generalize effectively. Furthermore, the influence of technical indicators on prediction may be limited. It is plausible that the dataset contains inherent noise, underscoring the imperative need for thorough preprocessing and data cleaning to ensure data quality. Notably, each model displayed sensitivity to random initialization, emphasizing the importance of running the training process multiple times with varying initializations to discern its impact. In summary, a comprehensive tuning of the model's hyperparameters is imperative for drawing more nuanced and conclusive insights.

Nevertheless, there were time spans during which the model demonstrated consistent correctness and also periods where its predictions were consistently inaccurate, as depicted in Figure 7. It would be naive to assert that the model lacks predictive power; rather, for specific time intervals, its performance is noteworthy. Perhaps, with further tuning, it could be integrated into a larger model where it specifically predicts certain defined periods of time.

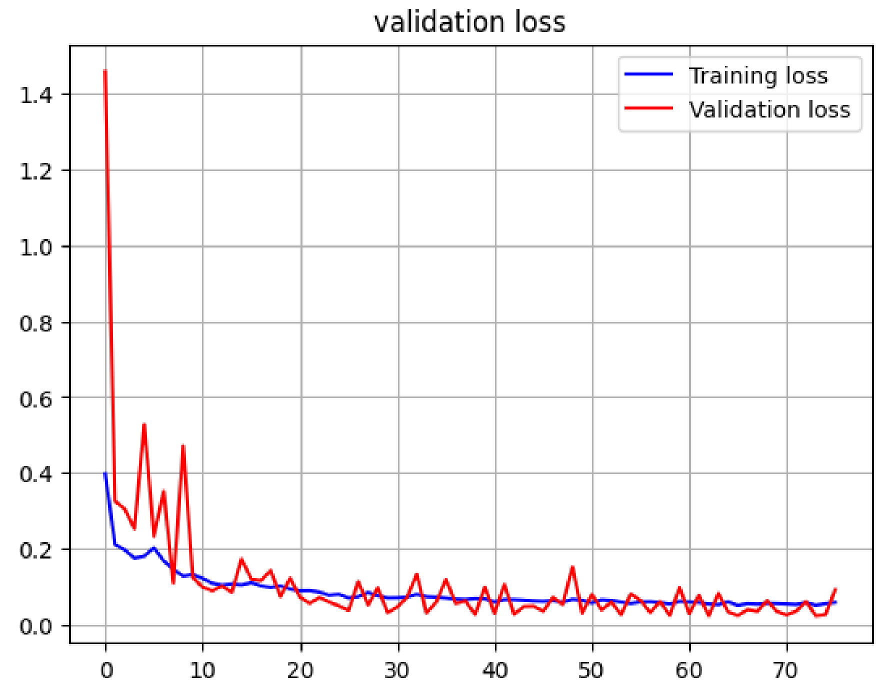
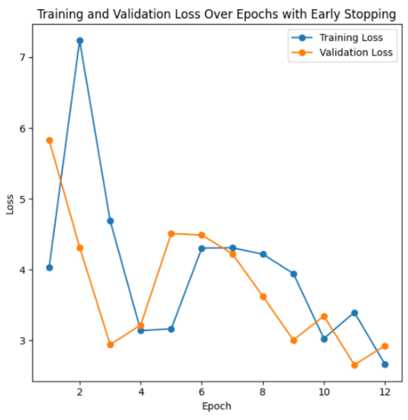
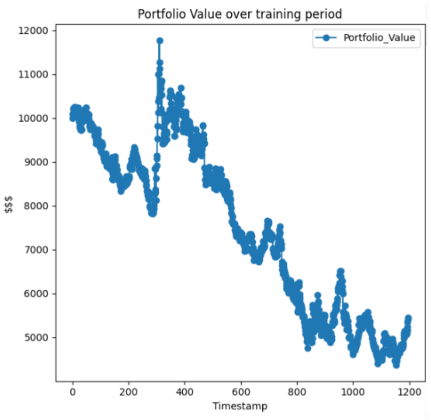


Figure 8. Training and Validation loss of the best LSTM model

The figures presented below depict the training-validation losses and portfolio valuation based on the specified set of hyperparameters, which resulted from the iterative application of Bayesian optimization for DQN, conducted over 10 iterations.

Upon initial inspection, it is evident that this DQN model exhibits a pronounced sensitivity to hyperparameters, even for training epochs, necessitating extensive tuning, a task that ideally should be performed on hundreds of occasions. Due to time constraints, I conducted 10 iterations only; however, given these outcomes, thorough tuning is imperative for future projects. Examining the train-validation losses, none of the models, except the last one, appears to convey meaningful information, suggesting that the former models may generate random predictions or lack confidence in their predictions. The final model exhibited three abrupt jumps in training validation loss, potentially stemming from interactions between the agent and the environment. I configured the replay memory to encompass half of the data from the previous episode and the entirety of the data from the new episode. Consequently, future considerations should include the replay memory size as a critical hyperparameter. Despite these challenges, the model outperformed the buy-and-hold strategy by a little over 10%. Had it not been for a sudden downturn coinciding with the onset of the COVID-19 lockdown, the portfolio would have demonstrated a consistently upward trend. This underscores the significance of fundamental data, particularly in anticipating unforeseen events such as black swan occurrences, an aspect currently absent in our model. Remarkably, the model left a positive impression, warranting further exploration in subsequent work.





Learning Rate (lr): 0.092

Discount Factor (gamma): 0.96

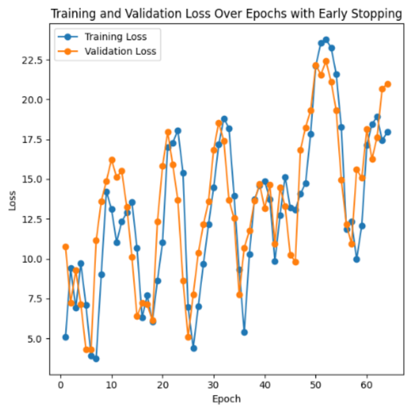
Batch Size: 73

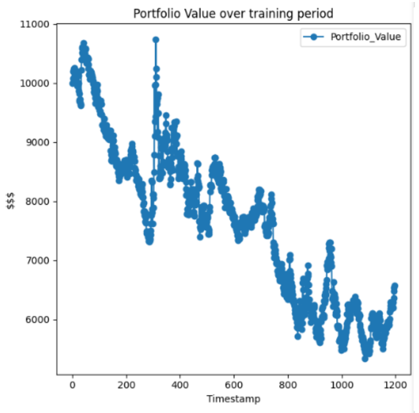
Window Length (T): 5

Episodes: 12

Number of Layers: 69

Soft Update Interval: 50





Learning Rate (lr): 0.002

Discount Factor (gamma): 0.95

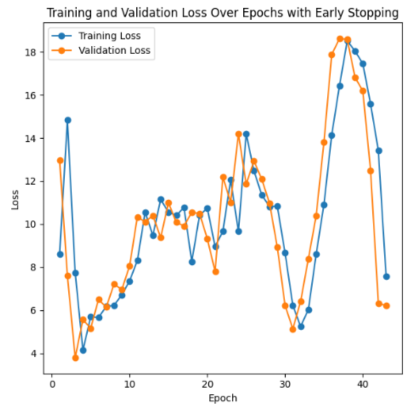
Batch Size: 140

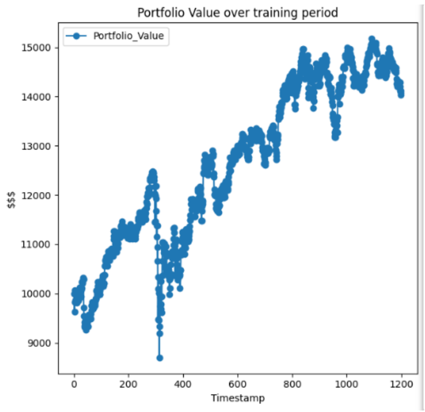
Window Length(T): 8

Episodes: 87

Number of Layers: 85

Soft Update Interval: 40





Learning Rate (lr): 0.092

Discount Factor (gamma): 0.96

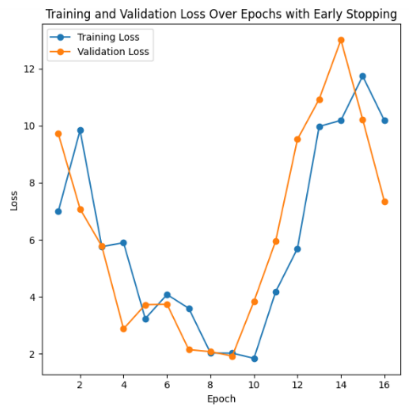
Batch Size: 73

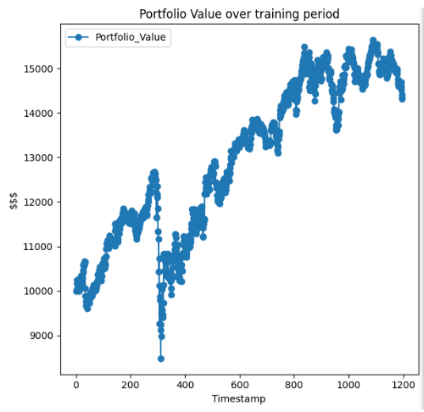
Window Length(T): 5

Episodes: 12

Number of Layers: 69

Soft Update Interval: 50





Learning Rate (lr): 0.092

Discount Factor (gamma): 0.96

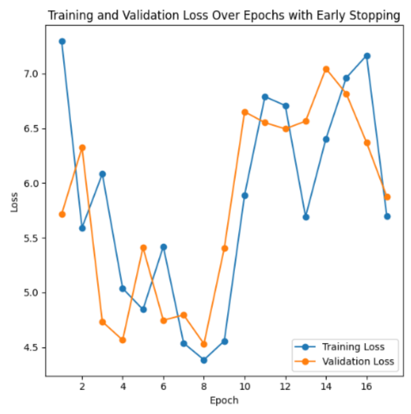
Batch Size: 73

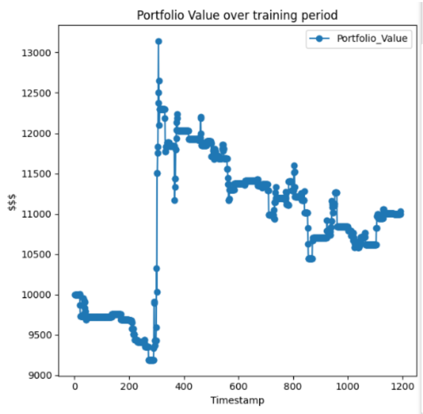
Window Length(T): 5

Episodes: 12

Number of Layers: 69

Soft Update Interval: 50





Learning Rate (lr): 0.0088

Discount Factor (gamma): 0.94

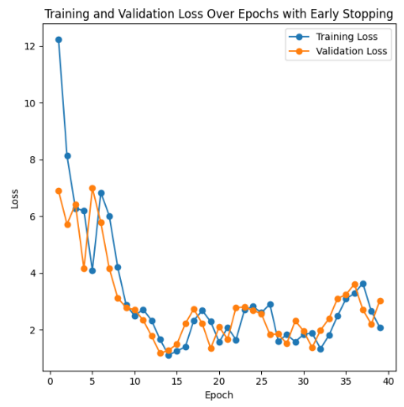
Batch Size: 228

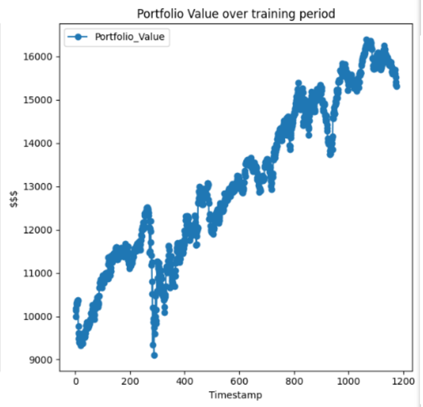
Window Length(T): 8

Episodes: 17

Number of Layers: 318

Soft Update Interval: 14





Learning Rate (lr): 0.024

Discount Factor (gamma): 0.91

Batch Size: 216

Window Length(T): 27

Episodes: 39

Number of Layers: 261

Soft Update Interval: No

Figure 9. Training-validation loss in conjunction with portfolio valuation for Deep Q Networks (DQNs) characterized by distinct hyperparameters.

Conclusion

It is evident that Reinforcement Learning surpasses Supervised machine learning methods and backtesting. Although we refrained from comprehensive backtesting, a task that could be facilitated by libraries like vectorBT, its computational cost would exponentially increase as more features are introduced. Regrettably, the model's suitability for deployment is hindered by substantial jumps in the training loss.

As mentioned earlier, there were extended periods during which the LSTM model exhibited commendable predictive capabilities. Considering that DQN predicts rewards while LSTM predicts the next close, a promising approach would involve enhancing DQN with LSTM predictors, thereby creating an imitation learning model.

Nevertheless, numerous ideas for future enhancements have emerged from these experiments. Implementing Actor-Critic, N-step temporal difference, optimizing neural network architecture, and incorporating additional supervised machine learning methods are imperative for ongoing improvement.

Bibliography

Malkiel, E. F. (1970). Efficient capital markets: A review of theory and empirical work, The Journal of Finance, vol. 25, no. 2, pp. 383-417.

Malkie, B. (1973). A Random Walk Down Wall Street, New York:Norton.

MacKinlay, A. L. (s.d.). A non-random walk down Wall Street., Princeton, NJ [u.a.]:Princeton Univ. Press, 1999, [online] Available: http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+249613484&sourceid=fbw_bibsonomy. . 1990.

Kissell, M. G. (2013). Multi-asset Risk Modeling, Academic Press

Dahlquist, C. D. (2006). echnical Analysis: The Complete Resource for Financial Market Technicians. Financial Times Press. Wiley.

Kuan-Hau Chen a, X.-Q. S.-F.-C. (2020). Profitability of moving-average technical analysis over the firm life cycle: Evidence from Taiwan https://www.sciencedirect.com/science/article/abs/pii/S0927538X21001402.

Jimmy Ming-Tai Wu, M.-E. W.-J. (2020). Convert index trading to option strategies via LSTM architecture https://link.springer.com/article/10.1007/s00521-020-05377-6.

Schmidhuber, S. H. (1997). Long short-term memory", Neural Comput., vol. 9, no. 8, pp. 1735-1780, Nov.

Nelson, D. M., Pereira, A. C., & Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks https://ieeexplore.ieee.org/abstract/document/7966019.

Junhao Zhang (2022) Deep Reinforcement Learning for Stock Prediction https://www.hindawi.com/journals/sp/2022/5812546/

Vanat Pavlo, 2023, 30-day SPX 500 index forecast using hybrid ArimaLSTM Model based on its price lags, moving averages, US market breadth and proprietary sentiment indicator

Wang Lixu, Jing Jiang, 2021, Financial Trading as a game: A deep reinforcement learning approach. https://github.com/conditionWang/DRQN_Stock_Trading/tree/main (sorry, couldn't find the scientific paper, but they have video presentation and code)