American Journal of Applied Mathematics and Statistics, 2024, Vol. 12, No. 1, 15-23
Available online at http://pubs.sciepub.com/ajams/12/1/3
Published by Science and Education Publishing
DOI:10.12691/ajams-12-1-3

Optimized Investment Strategy Based on Long Short-Term Memory Networks (LSTMs)

Qingyun Wang[1], Yayuan Xiao[2,*]

[1]College of Mathematics and Computer Science, Gannan Normal University, Ganzhou, China, 341000
[2]Department of Mathematical Sciences, Ball State University, Muncie, IN, USA, 47396
*Corresponding author: yxiao3@bsu.edu

Received January 12, 2024; Revised February 15, 2024; Accepted February 22, 2024

Abstract In recent decades, Long Short-Term Memory networks (LSTMs), an enhanced version of Recurrent Neural Networks (RNNs), have made significant contributions across various domains. Particularly in the study of time series data, they have offered promising capabilities in capturing temporal dependencies and patterns. This paper delves into the application of LSTMs in market forecasting, aiming to use historical price data to construct predictive models and optimize investment allocations for improved portfolio performance. The investigation includes a detailed examination of hyperparameters tailored for Invesco QQQ Trust (QQQ), SPDR Gold Trust (GLD), and Bitcoin (BTC) LSTM models, employing them for price prediction and the development of high-return trading strategies. Following this, an analysis is carried out on portfolio holdings, return rates, and risk enhancements for each investment asset within the testing set under this trading strategy.

Keywords: *RNN, LSTM, QQQ, GLD, BTC*

Cite This Article: Qingyun Wang, and Yayuan Xiao, "Optimized Investment Strategy Based on Long Short-Term Memory Networks (LSTMs)." *American Journal of Applied Mathematics and Statistics*, vol. 12, no. 1 (2024): 15-23. doi: 10.12691/ajams-12-1-3.

1. Introduction

In the realm of artificial intelligence and deep learning, Recurrent Neural Networks (RNNs) have emerged as indispensable tools for handling sequential data. RNNs maintain information states in time series data through recursive connections. However, they encounter challenges such as the vanishing or exploding gradient problem, making it difficult to learn long-term dependencies.

In 1997, Hochreiter and Schmidhuber [1] introduced Long Short-Term Memory networks (LSTMs), an improved version of RNNs specifically designed to address the issue of long-term dependencies. LSTMs enhance their ability to capture long-term information in sequences by introducing memory cells and gating mechanisms. In the field of Natural Language Processing (NLP), Hochreiter and Schmidhuber's seminal work in 1997 laid the foundation for the application of LSTMs in tasks such as machine translation, sentiment analysis, and named entity recognition. Following this, LSTMs have been widely applied in various fields, such as speech recognition [2], medical image processing [3], intelligent transportation systems [4], time series forecasting [5], and financial market [6,7]. Inspired by these studies, this paper applies LSTM to predict market trends for Invesco QQQ Trust (QQQ), SPDR Gold Trust (GLD), and Bitcoin (BTC), thereby discussing potential optimizations for investment portfolios. GLD is often considered a safe-haven asset, BTC serves as a global digital asset and is typically viewed as a higher-risk appreciating asset, while QQQ encompasses major companies in the technology sector, offering relatively stable long-term growth potential. Therefore, the selection of these three investment assets takes into account risk diversification, hedging and appreciation, as well as global market coverage.

2. LSTM Neural Network-Based Price Prediction Model

2.1. The Structure of Long Short-Term Memory (LSTM)

It is well known that Recurrent Neural Networks (RNNs) are a type of deep learning model designed for handling sequential or time-based data. Their unique feature lies in the inclusion of cyclic connections, allowing information to be passed within the network and utilized for processing different time steps.

The basic structure of an RNN (Figure 1) consists of input, hidden, and output layers. The neurons in the hidden layer not only receive input from the current time step but also incorporate output from the hidden layer of the previous time step.

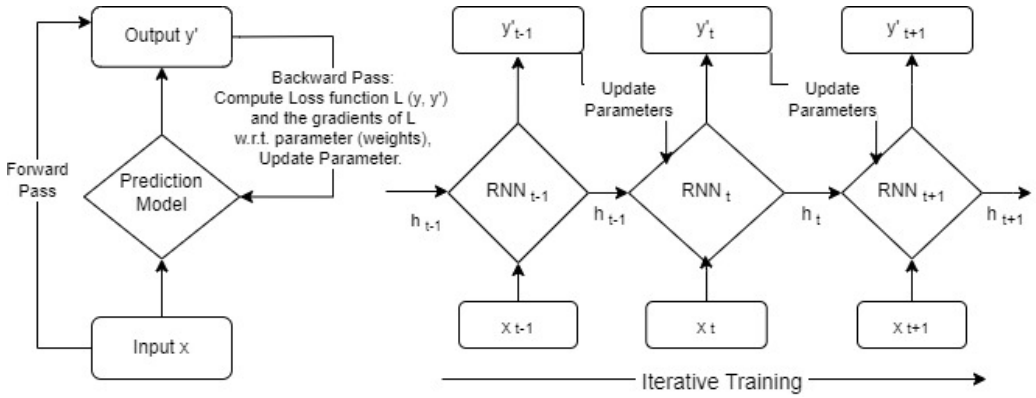


Figure 1. The basic structure of RNN

At each time step, an RNN receives input and performs a forward pass. The input includes the data for the current time step x_t and the hidden state from the previous time step h_{t-1} The RNN computes the output y'_t and the hidden state for each time step h_t . By comparing the model's output y'_t with the actual target values y_t , the loss function L is computed. Next, the backward pass algorithm is employed to calculate the gradients of the loss function with respect to the network parameters (weights), where the gradients represent the rate of change of the loss function with respect to the parameters. Using gradient descent or other optimization algorithms, the gradient information is applied to the network parameters to update their values. This process ensures that the next forward pass of the network produces predictions closer to the actual targets. The above steps are iterated, moving through multiple time steps. In each iteration, the model adjusts its parameters to minimize the loss function. The hidden state is passed from each time step to the next, creating a recurrent structure that enables the network to consider contextual information in the sequence. When handling long sequences with traditional RNNs, it's important to note that during backward pass, gradient information can become extremely small (vanishing) or exceptionally large (exploding). The former hinders effective learning of long-term dependencies, while the latter causes unstable network parameters and uncontrollable training, impacting overall model performance. To address these issues, an improved structure was introduced, known as Long Short-Term Memory (LSTM), which uses memory cells and gate mechanisms, effectively alleviating the impact of gradient vanishing and exploding, and overcoming the challenges of long-term dependencies.

In an LSTM, the memory cell C_t is responsible for storing information and can retain or update this information over extended periods, enabling more effective handling of long sequential dependencies. The recurrent unit in LSTM consists of three main gates: the input gate, the forget gate, and the output gate(see Figure 2). These gates control the flow of information, allowing the LSTM to selectively store, forget, or output information at different time steps.

The forget gate decides which information will be removed from the previous memory cell C_{t-1}. Let h_t and x_t be the hidden state and input data in the current timestep t , and f_t be the weight of information to be forgotten in C_{t-1} , Then the forget gate uses a sigmoid activation function $\sigma(x)=\dfrac{1}{1+e^{-x}}$ to decide how much information to retain or forget based on input data x_t and the previous time step's hidden state h_{t-1}. $f_t=\sigma\left(W_f \cdot \left[h_{t-1},x_t\right]+b_f\right)$, where W_f and b_f are the weight vector (or matrix) and bias parameter of the forget gate. Then the output of f_t ranges between 0 and 1, where 0 indicates a preference to forget and discard, and 1 indicates a preference to remember and retain information from the past.

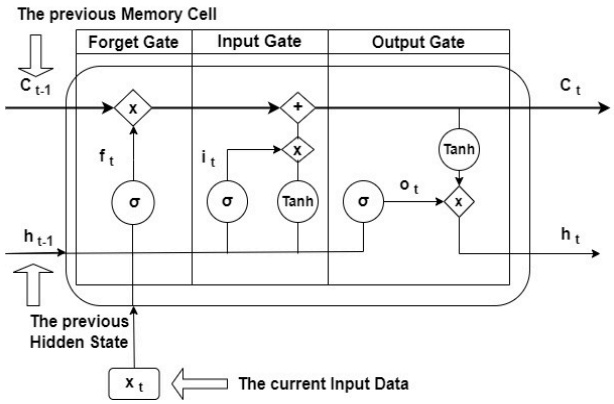


Figure 2. The basic structure of a LSTM Recurrent Unit

The input gate calculates the current moment candidate memory cell state value $\tilde{C}_t$ and the input gate value i_t between 0 and 1 to control the proportion of input information x_t to be stored into the current memory cell C_t , and then use $\tilde{C}_t$ and i_t to update the memory cell.

$$\tilde{C}_t = tanh\left(W_c \cdot \left[h_{t-1},x_t\right]+b_c\right),$$

$$i_t = \sigma\left(W_t \cdot \left[h_{t-1},x_t\right]+b_i\right),$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

where W_c , W_t , b_c , and b_i are the weight vectors (or matrices) and bias parameters of the input gate.

The output Gate determines the output hidden state h_t using the output gate value O_t .

$O_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right)$, and $h_t = O_t * tanh\, tanh\left(C_t\right)$.

It controls how much information is output, utilizing the sigmoid and hyperbolic tangent (tanh) activation functions. To be more precise, it produces an output between 0 and 1 based on input data and the current time step's hidden state, and outputs a portion of the information processed by the tanh function on the memory cell. Through the combination of these gates, LSTM can more effectively handle long sequences, aiding the network in learning and retaining dependencies on past information. This capability makes it well-suited for various tasks.

2.2. Critical Hyperparameters in LSTM

When building a Long Short-Term Memory (LSTM) network, a crucial aspect involves the selection of hyperparameters. Unlike parameters, which are internal variables adjusted during training, hyperparameters are external configuration settings that play a pivotal role in shaping the learning process. These higher-level structural choices significantly impact the model's performance and ability to generalize new data. In the context of LSTMs, careful consideration of hyperparameters is essential for achieving optimal results. This section will delve into the key hyperparameters associated with LSTMs, exploring their roles, significance, and the considerations involved in selecting appropriate values for a given task. From learning rates to the number of hidden units, each hyperparameter contributes to the overall architecture and behavior of the LSTM, making their understanding and fine-tuning crucial for successful model training.

Initial Learning Rate α The initial learning rate is the starting value for the step size or the rate at which the model's parameters are updated during training. A higher α may lead to faster convergence but may risk overshooting the optimal values. A lower α may result in slower convergence but might provide more stable updates.

Input Size N_{input} The input size refers to the dimensionality of the input data fed into the LSTM network. In natural language processing, for example, it could be the length of a sequence or the number of features in each time step. Larger N_{input} may require more computational resources but can capture more complex patterns. Smaller N_{input} may lead to faster training but might result in a loss of information.

Number of Hidden Units N_{hidden} The number of hidden units (also called hidden neurons) represents the dimensionality of the hidden state and cell state in the LSTM. It determines the capacity of the LSTM to learn and represent complex relationships. A higher N_{hidden} increases the model's capacity to capture intricate patterns but may also increase the risk of overfitting, especially with limited data. A lower N_{hidden} may lead to underfitting, where the model might struggle to capture important patterns.

Number of Epochs T_{epochs} The number of epochs is the count of times the entire training dataset is passed through the LSTM during training. Training for too few epochs may result in an underfit model, while training for too many epochs may lead to overfitting. It is essential to find a balance to achieve the best generalization for unseen data.

In short, in a LSTM, Initial Learning Rate α Influences the speed and stability of convergence. Input Size N_{input} affects the model's ability to capture information from the input. Number of Hidden Units N_{hidden} determines the model's capacity to learn complex patterns. Number of Epochs T_{epochs} Influences the balance between underfitting and overfitting. Choosing appropriate values for these parameters involves a trade-off between model complexity, training speed, and generalization to new data. Hyperparameter tuning is often required to find the optimal configuration for a specific task. In the following sections, we will delve into price prediction models based on LSTM neural networks, focusing specifically on QQQ, GLD, and BTC. Our emphasis will be on investigating more effective input sizes and the number of hidden units.

2.3. Data Processing

We collected the daily closing prices for QQQ, GLD, and BTC from January 2010 to July 2023. As QQQ and GLD ETF do not trade on weekends and holidays, we use the closing price from the previous trading day for those days. For the daily price data of these three investment assets, we selected the first 70% (from August 1, 2010, to September 8, 2018) as the training set, and the remaining 30% as the test set (from September 9, 2019, to July 31, 2023).

Given that neural network learning inherently involves capturing the distribution of the data, it becomes imperative to normalize the data to maintain consistency. Without normalization, each batch of training data may exhibit a different distribution. As the neural network strives to strike a balance among these multiple distributions, the input data for each layer undergoes constant changes, complicating the search for an optimal equilibrium and potentially hindering the convergence of the constructed neural network model. To expedite model convergence, mitigate the risk of gradient explosion, and enhance overall training speed and efficiency, data normalization is applied.

The normalization formula is expressed as follows.

$x'_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}$, where x_i is the price on day i, x_{min} is the minimum value in the price sample and x_{max} is the maximum value in the price sample.

2.4. Model Effectiveness Evaluation

Set the initial learning rate $\alpha =$ 0.005 and choose $T_{epochs} = 120$, we use the LSTM network with different values for N_{input} and N_{hidden} to predict prices for QQQ, GLD, and BTC. Subsequently, we assess the model performance using the average absolute error (MAE), mean absolute percentage error (MAPE), and root mean square error (RMSE).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i' - y_i|,$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i' - y_i|}{y'_i}\times 100\%,$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i' - y_i)^2},$$

Where n is the size of the sample, y'_i is the model prediction price, and y_i is the real price. A smaller MAE, MAPE, and RMSE indicate a closer alignment between the predicted and true values.

We selected input sizes of 5, 10, 20, and 60 based on the common practice in daily stock trading, where market participants often rely on moving averages calculated over these specific periods (5 days, 10 days, 20 days, and 60 days) for making assessments. This choice aligns with established strategies in the financial industry, leveraging these specific timeframes to gauge trends and make informed decisions in stock trading.

In Table 1, it is evident that all of the three model evaluation metrics reach their minimum values when $N_{input} = 5$ and the $N_{hidden} = 150$ (corresponding to Forecast 1 in Figure 3) in the testing set. Slightly higher values are observed when $N_{input} = 5$ and the $N_{hidden} = 200$ (resulting in Forecast 2 in Figure 3). As a conclusion drawn from this analysis, for QQQ, utilizing the daily prices of the preceding five days as input in the LSTM network seems more suitable for accurate price estimation.

In the case of LSTM networks for GLD, we observe similar outcomes. Specifically, for GLD, using the daily prices from the previous 20 days as input in the LSTM network appears more suitable for precise price estimation. This observation is based on the fact that all three model evaluation metrics reach their minimum values when $N_{input} = 20$ and the $N_{hidden} = 200$, corresponding to Forecast 1 within the testing set shown in Figure 4. The second-best performing LSTM network, characterized by the second-to-last smallest values for all three evaluation metrics, generates the Forecast 2 in Figure 4. It shares the same N_{input} as the top-performing model, which is 20; and its $N_{hidden} = 100$.

Table 1. LSTM network with different hyperparameters for QQQ

N_{hidden}	60	100	150	200	N_{hidden}	60	100	150	200
N_{input}	**MAE**				N_{input}	**MAPE (%)**			
5	25.52	26.48	**13.36**	**17.55**	5	7.84	8.21	**4.07**	**5.56**
10	32.50	31.78	16.54	30.41	10	10.10	9.91	5.13	9.89
20	45.77	46.69	21.22	33.64	20	14.23	14.26	6.45	10.47
60	75.40	59.90	58.51	83.36	60	23.19	18.33	17.97	25.97
N_{input}	**RMSE**								
5	30.41	30.85	**17.53**	**19.84**					
10	37.57	36.68	20.02	32.49					
20	52.36	55.27	26.41	38.44					
60	86.50	69.82	67.62	93.27					

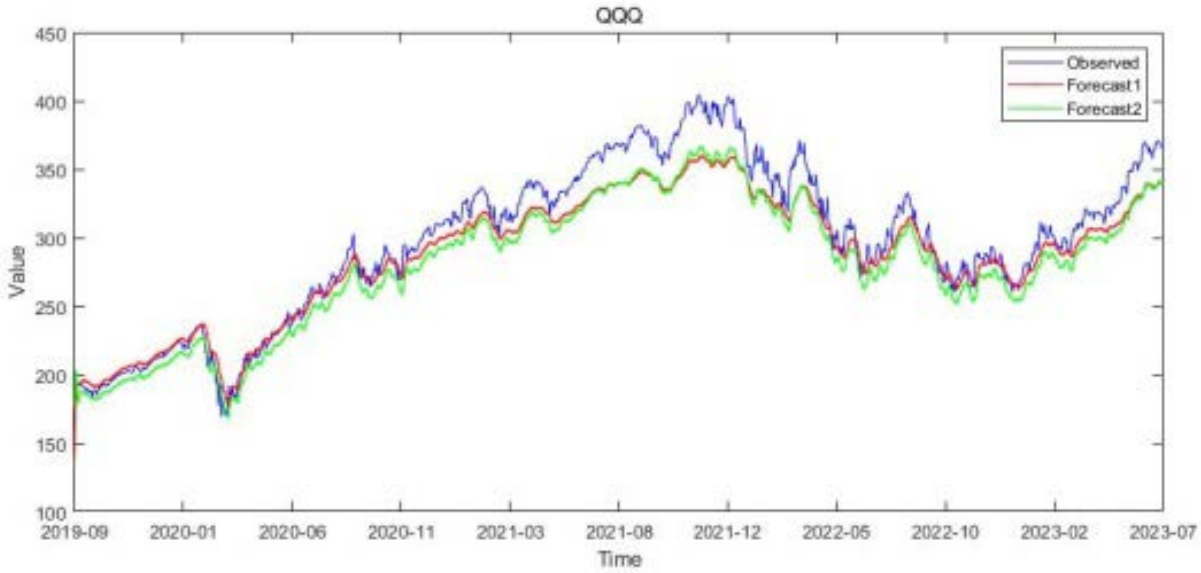


Figure 3. QQQ Daily Prices - Actual vs. LSTM Predictions

Table 2. LSTM network with different hyperparameters for GLD

N_{hidden}	60	100	150	200	250	N_{hidden}	60	100	150	200	250
N_{input}	**MAE**					N_{input}	**MAPE(%)**				
5	2.37	1.67	3.96	4.09	2.02	5	1.40	1.01	2.36	2.44	1.21
10	1.60	5.67	3.42	1.58	1.96	10	0.95	3.38	2.01	0.94	1.18
20	5.81	**1.54**	1.71	**1.14**	3.64	20	3.39	**0.93**	1.04	**0.68**	2.19
60	4.93	4.50	5.85	10.64	6.04	60	2.88	2.63	3.42	6.22	3.51
N_{input}	**RMSE**										
5	2.78	2.16	4.31	4.48	2.50						
10	2.14	6.01	3.82	2.10	2.47						
20	6.56	**2.12**	2.20	**1.70**	4.08						
60	5.62	5.19	6.53	11.44	6.94						

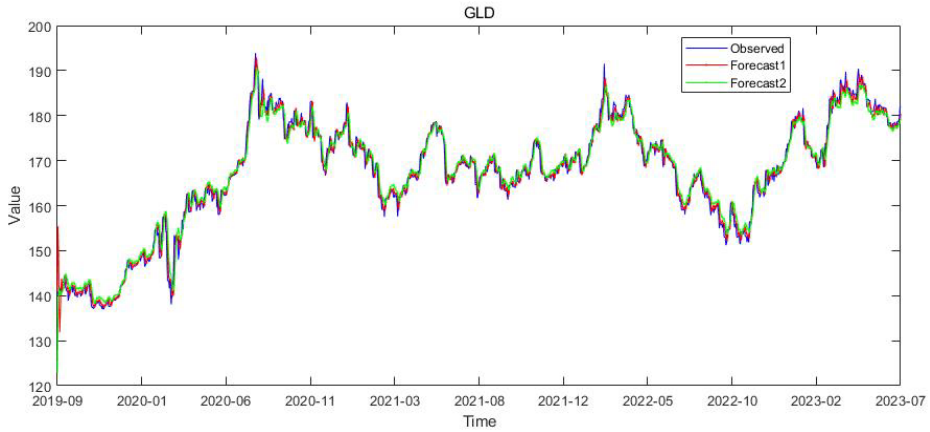


Figure 4. GLD Daily Prices - Actual vs. LSTM Predictions

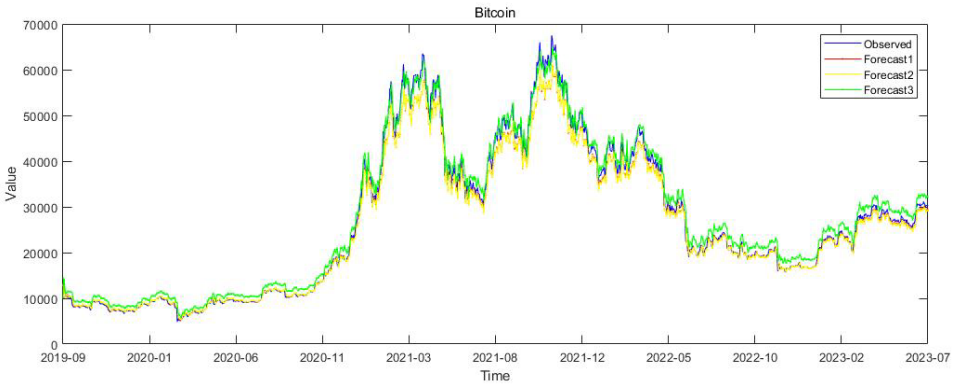


Figure 5. BTC Daily Prices - Actual vs. LSTM Predictions

Table 3. LSTM network with different hyperparameters for BTC

N_{hidden}	60	100	150	200
N_{input}	**MAE**			
5	1545.49	**1404.64**	**1344.03**	1765.85
10	3166.65	2966.30	1609.26	2088.86
20	3884.93	2834.63	2433.34	1914.92
60	12926.62	12222.18	10298.25	8856.76
N_{input}	**MAPE(%)**			
5	4.98	**4.24**	**4.04**	5.51
10	8.22	8.78	8.26	6.23
20	10.74	8.08	6.76	5.72
60	35.27	32.85	27.60	23.40
N_{input}	**RMSE**			
5	2221.18	2219.41	**2170.74**	2565.78
10	5082.76	4201.00	**1871.44**	3055.74
20	6530.79	4643.09	3924.60	3011.47
60	18048.67	17347.09	14910.88	12886.14

When comparing these BTC LSTM networks, the models with the minimum and second minimum values for MAE and MAPE are the same, corresponding to forecast 1 and 2 in Figure 5. The hyperparameters for forecast 1 are $N_{input} = 5$ and the $N_{hidden} = 150$, while for forecast 2, the hyperparameters are $N_{input} = 5$ and the $N_{hidden} = 100$. However, the networks with the minimum and second minimum values for RMSE differ from forecast 1 and 2. They align with forecast 2 and forecast 3 in Figure 5, where forecast 3 represents the LSTM network with $N_{input} = 10$ and the $N_{hidden} = 150$.

In this context, directly comparing forecast 1 (minimizing MAE and MAPE) and forecast 3 (minimizing RMSE) for predicting BTC prices is challenging due to their unique features. Forecast 1 is better suited for periods of lower BTC price volatility, as shown in Figure 6.

Conversely, forecast 3 is considered more effective during times of increased BTC price fluctuations, as depicted in Figure 7. This difference in suitability stems from the distinct characteristics of the evaluation metrics and the underlying patterns captured by each forecast in response to different levels of BTC price volatility.

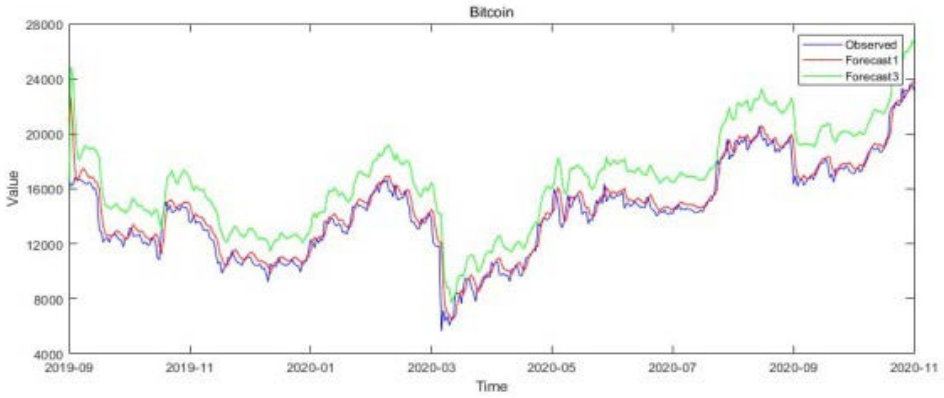


Figure 6. Extracting Time Periods of Low BTC Price Volatility from Figure 5

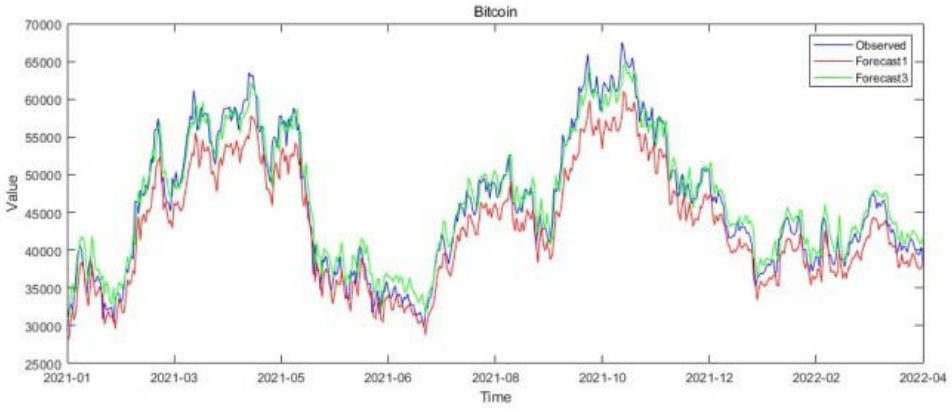


Figure 7. Extracted Time Periods of Elevated BTC Price Volatility from Figure 5

Summarizing the analysis and discussion in this subsection, we obtain the following intriguing findings: When employing LSTM for price evaluation, QQQ is best suited for using the prices of the previous five days; GLD is best suited for utilizing the prices of the previous 20 days. As for BTC, it is optimal to use the prices of the previous five days when its price volatility is low and the prices of the previous ten days when volatility is high.

3. Trading Strategies Based on Daily Price Predictions

Building upon the analysis in subsection 2.4, we select LSTM models with the following crucial hyperparameters for QQQ, GLD, and BTC to generate forecasts for the next day's prices.

Table 4. Key LSTM Hyperparameters Chosen for QQQ, GLD, BTC Price Predictions

	QQQ	GLD	BTC
N_{hidden}	150	200	150
N_{input}	5	20	5

Let $P_t{}'$ denote the predicted price on day t, and P_t denote the actual price on day t, then the expected daily return rate is $R_t^{'} = \frac{P_t^{'} - P_{t-1}}{P_{t-1}}$. We compute and compare the daily $R_t^{'}$ for QQQ, GLD, and BTC, utilizing the comparison results to inform our trading strategies. On regular trading days, we prioritize the investment target with the highest expected daily return among QQQ, GLD, and BTC. We sell the other two assets and use the proceeds to buy more of the highest-return assets. If all assets have negative expected daily returns, we sell everything and hold cash. On holidays (non-trading days), QQQ and GLD remain untouched, while BTC is tradable. If the expected daily return for BTC is negative, we sell it; if positive, we use all available cash to buy BTC.

Let C_t denote the cumulative return rate on day t, then $C_t = (1-r_1) \cdot (1-r_2) \ldots (1-r_t)$, where $r_t = \frac{B_t - B_{t-1}}{B_{t-1}}$ is the real daily return rate, and B_t is overall balance of all assets on day t. Applying the above trading strategy to our test set, covering from September 2019 to July 2023, resulted in a remarkable return rate of 317.54%, with an annual average return rate of 81.91%.

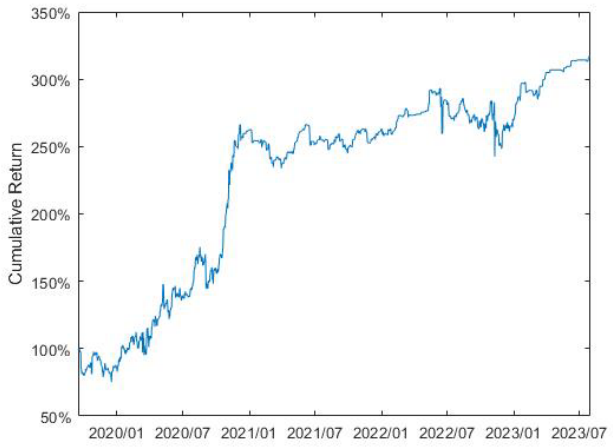
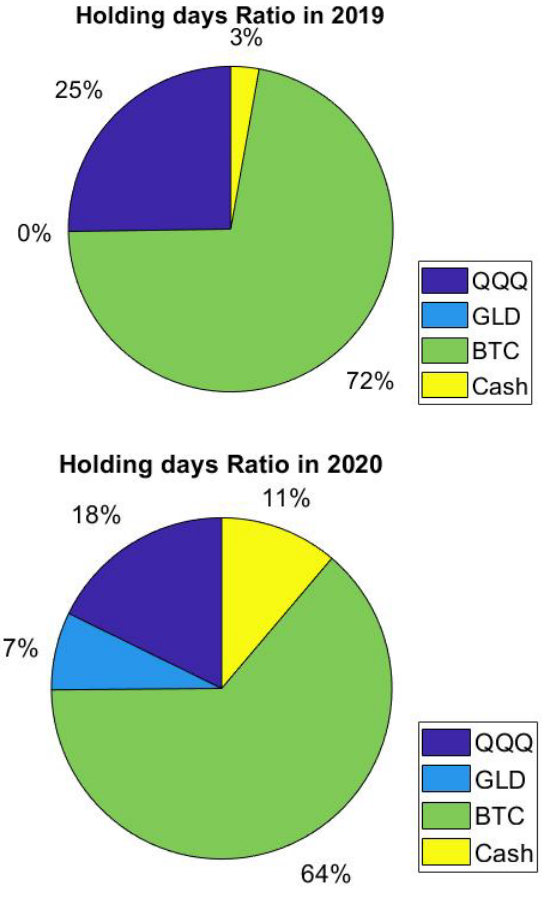


Figure 8. Cumulative Return Rate in the Testing Set (2019/09-2023/07)

In addition to the high return rate in this trading strategy, another noteworthy outcome is the variability in the number of days each distinct asset is held in the testing set. In 2019 and 2020, our trading strategy involved holding BTC for the longest duration, while the period of holding gold was the shortest, with no gold holdings in 2019. However, in the subsequent two years, gold emerged as the most frequently chosen investment among the three, dominating in 2021 and slightly decreasing thereafter. There were no days of holding BTC in 2021 and 2023, and the holding period in 2022 was also the shortest. In contrast, the days of holding gold constituted a significant proportion over these three years.



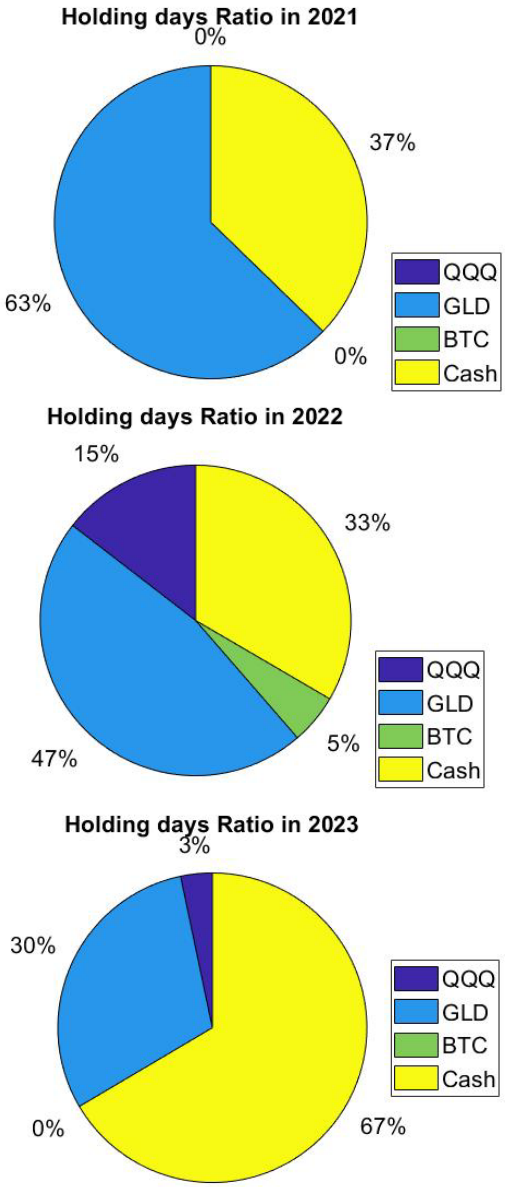


Figure 9. Investment Portfolio: Holding Days Ratio Over 2019-2023 (QQQ, GLD, BTC)

The investment strategy discussed above focuses on high returns without taking into consideration the associated risks. However, it's important to note that there are significant differences in the risk profiles of these three investment assets. Let R_i represent the actual daily return rate of an asset, calculated as $R_i = \frac{P_i - P_{i-1}}{P_{i-1}}$ with P_i is the actual price of the asset on day i. The risk standard deviation RSD is given by $RSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(R_i - \bar{R}_i\right)^2}$, where N is the number of historical return rates, and $\bar{R}_i$ denotes the average historical return value. Subsequently, considering N as 30 days, we generated the following Risk-Return graph based on the actual daily prices of QQQ, GLD, and BTC for the testing set.

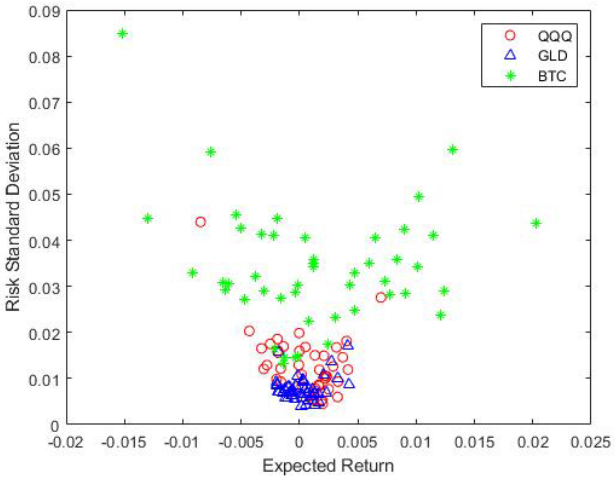


Figure 10. Actual Risk-Return for QQQ, GLD, and BTC (2019-2023)

In Figure 10, it is evident that among the three investment targets, GLD stands out as the most stable in terms of both returns and risks, followed by QQQ. Conversely, BTC exhibits the highest risk, but it also encompasses a wider range of returns. A natural follow-up question is how the risk-return profiles of these three investment targets will evolve within the trading strategy outlined in this article. Let r_i' denote the daily trading return rate for the asset. We define $r_i^{'} = R_i$ if we choose to hold the asset on day i, i.e., buy or retain the asset on day $i-1$, and $r_i^{'} = 0$ if we choose not to hold the asset on day i, i.e., sell or refrain from buying the asset on day $i-1$. The corresponding trading RSD is calculated by $\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(r_i'-\overline{r}_{i'}\right)^2}$.

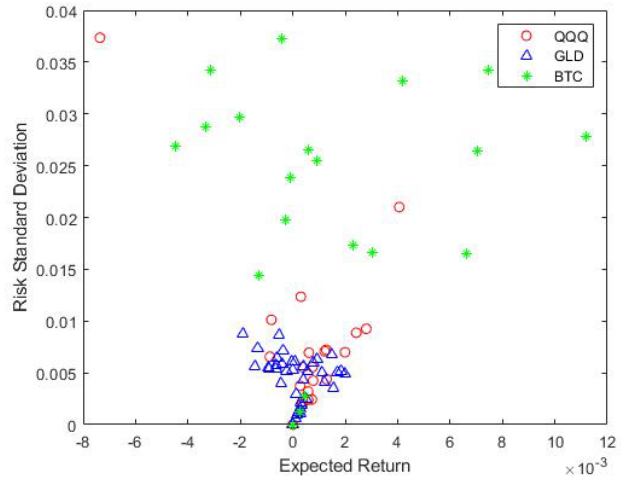


Figure 11. Trading Risk-Return for QQQ, GLD, and BTC (2019-2023)

Upon comparing Figure 10 and Figure 11, it becomes evident that, when applying our trading strategy, both QQQ and BTC experience a notable reduction in risk along with a substantial increase in returns compared to their actual daily risk-return profiles. Although the risk range for GLD remains relatively stable, there is an enhancement in its return rate as well.

4. Future Work

In this article, we delved into the application of LSTM for predicting the prices of investment assets, facilitating the creation of a high-return trading strategy. Our focus revolved around the selection of two crucial hyperparameters for LSTM: N_{hidden} and N_{input}. There is potential for further optimizing the model by incorporating techniques such as Dropout [8,9], and batch normalization [10], among others. Moreover, when crafting trading strategies based on predicted prices, we can broaden our analysis to encompass factors like transaction costs and the inflation rate of cash. In addition, our current trading strategy involves daily transactions. In the future, we may investigate strategies with weekly or monthly trading frequencies and compare the optimal trading frequency for various investment assets.

References

[1] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780.

[2] Graves, Alex & Mohamed, Abdel-rahman & Hinton, Geoffrey. (2013). Speech Recognition with Deep Recurrent Neural Networks. ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings. 38. 10.1109/ICASSP.2013.6638947.

[3] Prasoon, A., Petersen, K., Igel, C., Lauze, F., Dam, E., Nielsen, M. (2013). Deep Feature Learning for Knee Cartilage Segmentation Using a Triplanar Convolutional Neural Network. In: Mori, K., Sakuma, I., Sato, Y., Barillot, C., Navab, N. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013. MICCAI 2013. Lecture Notes in Computer Science, vol 8150. Springer, Berlin, Heidelberg.

[4] Y. Lv, Y. Duan, W. Kang, Z. Li and F.-Y. Wang (2015), "Traffic Flow Prediction With Big Data: A Deep Learning Approach," in IEEE Transactions on Intelligent Transportation Systems, 16(2), 865-873.

[5] Hansika Hewamalage, Christoph Bergmeir, Kasun Bandara (2021), Recurrent Neural Networks for Time Series Forecasting: Current status and future directions, International Journal of Forecasting, 37(1), 388-427.

[6] Thomas Fischer, Christopher Krauss (2018), Deep learning with long short-term memory networks for financial market predictions, European Journal of Operational Research, 270(2), 654-669.

[7] Qiu J, Wang B, Zhou C (2020). Forecasting stock prices with long-short term memory neural network based on attention mechanism. PLoS One. 15(1), e0227222. PMID: 31899770; PMCID: PMC6941898.

[8] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929-1958.

[9] Pascanu Razvan, Mikolov Tomas, and Bengio Yoshua (2013). On the difficulty of training recurrent neural networks. In International Conference on Machine Learning. 1310–1318.

[10] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning. Vol. 37, 448-456.