Project Report : CS 7643
Should LSTMs be used for stock prediction?

Alexander Sutherland
Georgia Institute of Technology
801 Atlantic Drive Atlanta, GA 30332-0280
asutherland9@gatech.edu

Feng Lin
Georgia Institute of Technology
801 Atlantic Drive Atlanta, GA 30332-0280
flin304@gatech.edu

August 1, 2024

Abstract

Recent breakthroughs in machine learning have sparked significant interest in various algorithms, including neural network models such as Recurrent Neural Networks (RNN) and its more popular variant, Long Short-Term Memory (LSTM). Researchers and individuals are eager to apply these innovative models to traditional challenges, such as the longstanding quest to develop accurate stock prediction algorithms. Many published papers on stock prediction models using LSTM are based on simple premises and do not discuss their shortcomings. This paper will explore some of these findings through experiments and observations. Finally, it will propose future ideas for improving stock prediction models but in reality it'll prove LSTMs should not be used.

1. Introduction

Since the stock market's inception, individuals and institutions have sought ways to achieve superior returns. Today, this quest includes not only traditional investors but also large hedge funds and mutual funds, all aiming to outperform the market. Recent advancements in deep learning have led researchers and bloggers to highlight the potential of Long Short-Term Memory (LSTM) networks for predicting stock prices. Inspired by these findings, our paper aims to replicate the results presented in existing literature.

If successful, replicating these results could slightly beat the market, potentially yielding significant gains over time, especially with compounding interest in a 401(k). Consistently outperforming the market, even by a small margin, could lead to substantial long-term benefits. For individual investors, this might enhance retirement savings and financial security, while for institutional investors, it could improve fund performance, attracting more investors and capital.

Common practices in LSTM stock prediction involve using recent closing prices and comparing them to previous days' technical indicators. These indicators can range from simple trading data, such as opening and closing prices, high/low prices, and trading volumes, to more complex ones like moving averages, Bollinger bands, or RSI.

Our paper not only replicates these results but also seeks to enhance current methodologies, offering a detailed explanation of our improvements. Thankfully, stock technical data is readily available, and we sourced it from one of the common areas, Yahoo Finance [3]. The data we pulled from the website primarily focuses on multiple previous days' closing prices to identify future closing price trends. We selected the VanEck Semiconductor ETF (SMH), a stock related to deep learning, for our analysis.

We chose a challenging time frame (January 1, 2014, to September 31, 2016) for our models to predict the test closing prices, where the closing price varied within and beyond the range seen in the training set. This was done to rigorously test if the models had learned effectively. Additionally, we'll demonstrate how to use daily returns instead of closing prices. We preprocessed the data using techniques such as Min-Max scaling. The experiments produced results comparable to those documented in the literature.

Despite these comparable results, our findings reveal a critical insight: the models, while appearing impressive as presented, are unfortunately impractical for real-world application. Our paper discusses the limitations and challenges that hinder the reliable use of these models in actual stock trading scenarios. We aim to clarify these issues to prevent others from adopting similar LSTM-based approaches with technical analysis for stock market predictions. Many software bloggers continue to develop similar models, often repeating past mistakes and misrepresenting their effectiveness.

For future research, we propose exploring transformer models that predict stock price trends based on news and community sentiment toward companies. Such models could offer a better approach for short-term market gains. By incorporating real-time data from news and social media, these models could capture market sentiment and respond more swiftly than traditional technical analysis, providing investors with a more robust tool for making informed trading decisions and potentially achieving higher returns and a competitive edge in the market.

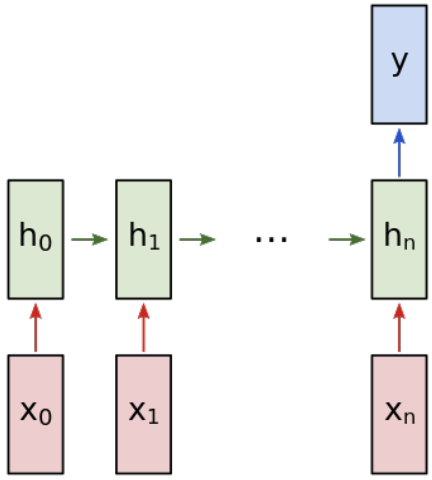

Figure 1. Graphic representation of a many to one LSTM Model [7]

2. Approach

Our first goal was to investigate how deep learning techniques have been applied to stock market prediction. The paper "Stock Market Prediction via Deep Learning Techniques: A Survey" [5] provides an overview of the most common methods used in stock market research over the past 25 years. This survey highlights that LSTM-based models, which take price as an input and predict the target price (typically the next day's closing price), are among the predominant approaches.

To delve deeper, we reviewed the paper "A CNN-BiLSTM-AM Method for Stock Price Prediction" [6]. This study explores the combination of LSTM with CNN, Bidirectional-LSTM, and an attention mechanism. The evaluation metrics included Mean Absolute Percentage Error, Root Mean Squared Error, and R-Squared Error, comparing various models—LSTM, CNN-LSTM, CNN-LSTM Attention, and CNN-BiLSTM Attention—against actual closing prices.

Our initial approach was to replicate the results from "A CNN-BiLSTM-AM Method for Stock Price Prediction" [6]. We began by constructing an LSTM-based model with a many-to-one architecture, as depicted in Figure 1. This architecture was selected because it is suited for predicting the next day's closing price based on several previous days of data, which helps in identifying potential patterns.

We opted for the LSTM model due to its capability to capture both long-term and short-term dependencies effectively, and its relative simplicity compared to more complex models like CNN-LSTMs. LSTMs manage these dependencies using a cell state, a hidden state, and specialized activation gates, as illustrated in Figure 2.

The most challenging aspect of the project was preprocessing the SMH stock data, which involved transitioning from a CSV file to Pandas, then to NumPy, and finally to PyTorch. Learning to use Pandas was more difficult than anticipated. The preprocessing included several stages: loading the data, cleaning it, feature engineering, scaling, reshaping, and converting it into tensors. We started by loading the stock data into a Pandas DataFrame, addressing issues like data cleaning, handling missing values, and ensuring chronological order. Feature engineering involved creating new features from existing data, such as daily returns and technical indicators, though we ultimately focused only on daily returns. After applying Min-Max scaling for normalization, we reshaped the data for the LSTM model and converted it into PyTorch tensors. This process, implemented in the DataUtil class in data_util.py, required a thorough understanding of both the data and the tools used, with Pandas proving particularly challenging.

In the next phase, we built a modular LSTM architecture to experiment with various configurations, including input size, hidden size, number of layers, and batching of the SMH data. To enhance the learning capacity of the LSTM, we added four linear layers to its output. The first linear layer matched the LSTM's hidden state size and had an output size of 10. The second layer transformed an input size of 10 into an output of 50. The third layer converted an input of 50 to an output of 25. The final layer produced the predicted closing price from an input of 25. This design aimed to prevent overfitting by ensuring the model did not rely solely on the LSTM output. By using a common encoder-decoder technique, where the encoder output is smaller than the decoder's output, we intended to force the model to better learn and retain key features. This model is implemented in the LSTM class in lstm.py.

Finally, we developed a notebook and train_model.py. The notebook is used for running the models, while train_model.py contains functions for training and testing the models, including support for batching. Feeding the data into the model required extensive manipulation, which was extremely time-consuming. We decided to go with a MSE loss function and optimzer using Adam. Adam has

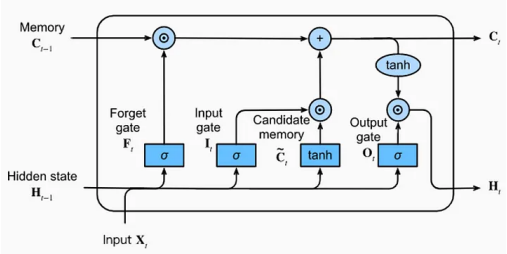

Figure 2. Graphic representation of base building block of a LSTM Model and cited source is a good reference for any beginner [2]

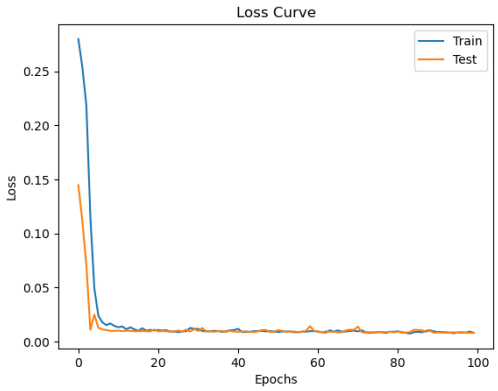

Figure 3. Captures the loss curve for training and testing of the LSTM Model for closing price

shown to be one of the best starting optimizers, as it easily updates the model well but not perfectly. Initially, we tested the models using actual closing prices but faced issues with accurately predicting test data. Revisiting "A CNN-BiLSTM-AM Method for Stock Price Prediction" [6], we noted the suggestion to apply Min-Max normalization to the closing prices, which significantly improved our data processing. Despite investing considerable time in tweaking parameters such as learning rate, look-back period, batch size, and optimizer, we achieved results similar to those reported in the paper. However, these results proved less meaningful than anticipated.

We realized that a CNN-LSTM model might not offer substantial improvements, as it would only produce higher accuracy for results that were not particularly useful. A large issue was that the Min-Max scaler did not account for the stock market's historical upward trend. Once prices exceeded the predicted range, the model struggled, as it was trained on a specific segment of the market environment that had shifted. Consequently, we decided to train the model on percent changes instead of absolute stock prices, believing that this approach would better handle the constant range of percentage changes. The results and further details are discussed in the experimental section of the report.

3. Experiments and Results

(10 points) How did you measure success? What experiments were used? What were the results, both quantitative and qualitative? Did you succeed? Did you fail? Why? Justify your reasons with arguments supported by evidence and data.

According to the paper "Stock Market Prediction via Deep Learning Techniques" [5], there are typically three ways to evaluate the performance of a model: Accuracy-based, Error-based, and Return-based. Accuracy-based evaluation measures how often the model's predictions are correct. Sometimes a baseline is provided for the evaluation, e.g., if the return is greater than 1. Error-based evaluation measures the difference between the predicted value and the ground truth value. A lower error value indicates better performance. Error-based evaluation is widely popular, leveraging various regression calculations such as Mean Squared Error, Mean Absolute Error, etc. Return-based evaluation measures the total return of model predictions. Two common methodology is to compute the return ratio or Sharpe ratio. For longer period of time, annual rate of return is calculated.

We consider error-based evaluation to be the most straightforward metric for our experimental design. Additionally, we plot the computed predictions against the training and test data to gain a more intuitive sense of the overall quality of the predictions. Figures 4 and 5 show using n+ prior day "adjusted close" return to predict the current day return.

We consider error-based evaluation to be the most straightforward metric for our experiments. The best overall results we got from tuning our LSTM were found using a learning rate of 0.001, batch size of 32, and a look back of 7. Loss curve graph can be seen in Figure 3.

However, relying solely on the loss curve can be misleading, as the Mean Squared Error (MSE) does not account for data distribution and may not accurately reflect model performance over the experiment period. To gain a more intuitive understanding of prediction quality, we also plot the computed predictions against both the training and test data. The following section presents Experiment One, which uses the adjusted close return from the previous n+ days to predict the return for the current day.

Initially, we felt optimistic about the early experimental outcomes. While we observed overfitting in the training data, we believed our methodology of using consecutive day returns had potential, as it produced reasonable predictions. However, we discovered a design flaw in our approach when we conducted tests over a different period show in Figures 6 & 7.

Knowing this, we decided to modify our approach by

Figure 4. Captures the trading day period between 01/01/2014 to 12/31/2015 for SMH training data set and predicting closing price

Figure 6. Captures the trading day period between 01/01/2014 to 9/31/2016 for SMH training data set and predicting closing price

changing the model's output from closing prices to the percentage change of closing prices, also known as daily returns, to see if it would improve our results. The outcomes of this method can be seen in Figures 8 and 9.

We observed that the test predictions never exceeded 0.2 of the scaled stock prices. Once the testing close price surpassed this threshold, the model began to undercut its estimations. This issue arose because the stock market has historically shown continuous growth. Our approach only worked effectively when the range of stock returns was consistent for both training and testing periods, regardless of the application of Min-Max scaling. This observation makes sense as Min-Max scaling normalizes data within a specific range, which is disrupted if the testing data deviates significantly from the training data's range.

We would like to mention that even though we produced similar results when bounded between Min and Max, the result was essentially a simple moving average. Knowing this, we question the efficacy of using an LSTM for stock prediction with closing prices. Achieving this goal seems too straightforward, and the loss function does not adequately predict closing prices. This could be due to multiple factors such as insufficient data or incorrect evaluation metrics. Based on these results, we believe that building LSTMs or similar models won't yield reliable predictions if using closing prices as the target output.

After these findings, we searched the internet to verify if we had incorrectly implemented the model. To our surprise, many articles presented incorrect LSTM models claiming to predict closing prices. We believe these claims are misleading. Examples include "Stock Price Prediction using LSTM and its Implementation"[4] and "Machine Learning to Predict Stock Prices"[1]. These articles build simple LSTM models without fully utilizing their recurrent power and use closing price as the target. Such articles could mislead individuals new to this field, leading them to implement LSTMs incorrectly and use the wrong target for stock prediction.

Given these insights, we decided to modify our approach

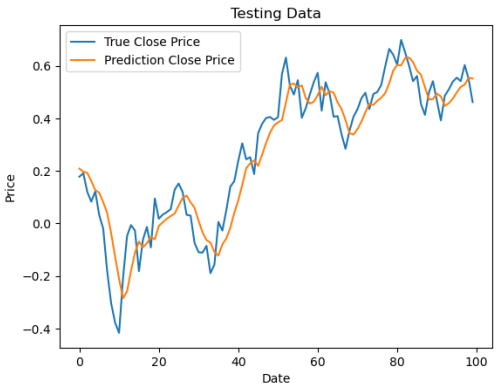

Figure 5. Captures the trading day period between 01/01/2014 to 12/31/2015 for SMH testing data set and predicting closing price

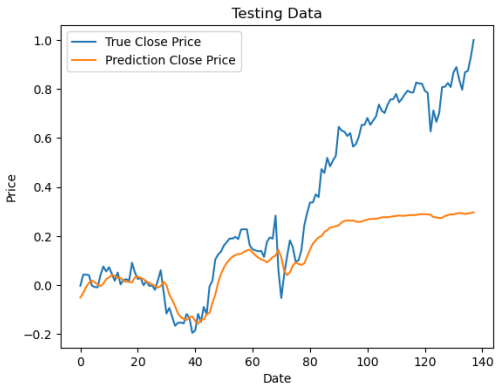

Figure 7. Captures the trading day period between 01/01/2014 to 9/31/2016 for SMH testing data set and predicting closing price

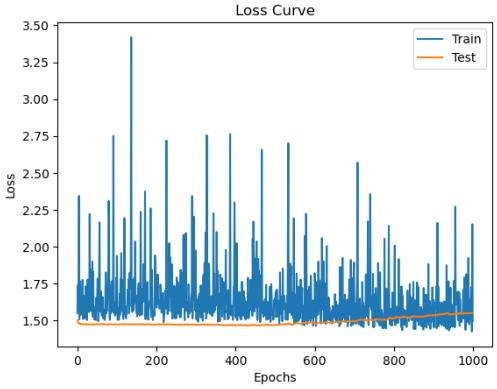

Figure 8. Captures the loss curve for training and testing of the LSTM Model for Daily Returns

Figure 9. Captures the trading day period between 01/01/2014 to 12/31/2015 for SMH training data set and predicting daily return

by changing the model's output from closing prices to the percentage change of closing prices, also known as daily returns. This adjustment aimed to see if it would improve our results. Our loss results can be seen in figure 8.

Sadly, no matter how much we tried tweaking the date range, data size, learning rate, look-back period, batch size, or optimizer, the loss for the test data was significantly off to the training data. We found that the model was unable to learn effectively from the given data, likely because stock technical data is sporadic and often influenced by external factors such as news and CEO approval ratings. The outcomes of this method can be seen in Figures 9 and 10. The outcomes of this method can be seen in Figures 9 and 10 show similar issues.

It is worth noting that the range of percentage change is more stable, especially for ETFs, as it ranges from -6% to 4% during this period. This stability ensures that the output remains within the maximum and minimum bounds between the training and testing periods. Additional tests over different periods confirmed the stability of the model. How-

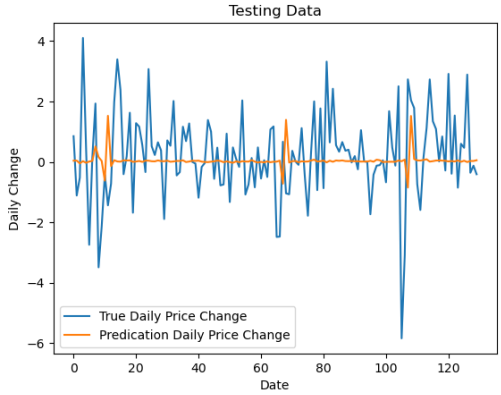

Figure 10. Captures the trading day period between 01/01/2014 to 12/31/2015 for SMH testing data set and predicting daily return

ever, based on the percent change experiment, we observed that the model learns very little, if anything, when using technical indicators to predict daily returns.

We had hoped that using daily returns would resolve the issues associated with using closing prices, but it presented its own challenges when relying solely on technical data. These experiments led us to believe that combining daily returns with news related to the sector could yield better results. It is also worth noting that these techniques may work significantly better when including stock data throughout each trading day.

4. Conclusion

We have gained a better understanding of the LSTM architecture and achieved success by building a working LSTM model. We also identified the limitations and issues related to manipulating data inputs and outputs. We do not recommend individuals from attempting to using LSTMs when the available technical information. Given additional time and resources, we would like to explore ways to incorporate additional features and technical indicators. Beyond that, we aim to include additional data from non-traditional stock market sources, such as news media reports, earnings releases, or social media platform sentiments. In hindsight, if something is readily available across the internet and there's not much news on it especially in improve gaining in the market, it likely doesn't work.

Student Name	Contributed Aspects	Details
Alex Sutherland	Research, Implementation Analysis , and Report Writing	Both members contributed to all parts of the project but focused more on the coding, building and tuning side
Feng Lin	Research, Implementation Analysis, and Report Writing	Both members contributed to all parts of the project but focused more on the research side

Table 1. Contributions of team members.

5. Work Division

Please add a section on the delegation of work among team members at the end of the report, in the form of a table and paragraph description. This and references do **NOT** count towards your page limit. An example has been provided in Table 1.

References

[1] Roshan Adusumilli. Machine learning to predict stock prices, 2024. https://towardsdatascience.com/predicting-stock-prices-using-a-keras-lstm-model-4225457f0233. 4

[2] Ottavio Calzone. An intuitive explanation of lstm, 2022. https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c. 3

[3] Yahoo Finance. Vaneck semiconductor etf (smh), 2024. https://finance.yahoo.com/. 1

[4] Siddharth M. Stock price prediction using lstm and its implementation, 2024. https://www.analyticsvidhya.com/blog/2021/12/stock-price-prediction-using-lstm/. 4

[5] Jinan Zou Qingying Zhao Yang Jiao Haiyao Cao Yanxi Liu Qingsen Yan Ehsan Abbasnejad Lingqiao Liu Javen Qinfeng Shi. Stock market prediction via deep learning techniques: A survey, 2022. Existing surveys on stock market prediction often focus on traditional machine learning methods instead of deep learning methods. 2, 3

[6] Wenjie Lu Jiazheng Li Jingyang Wang and Lele Qin. A cnn-bilstm-am method for stock price prediction". neural computing and applications 33, 10 (2021), 4741–4753., 2014. Supplied as additional material `tr.pdf`. 2, 3

[7] Victor Zhou. An introduction to recurrent neural networks for beginners, 2024. https://victorzhou.com/blog/intro-to-rnns/. 2