Stock Prediction Using CNN and LSTM from Fundamentals

Abstract

Recently, future stock price prediction using machine learning models has gained a great deal of attention in the financial sector. However, predicting the future behavior of stock prices with a high degree of accuracy remains a challenge. The goal of this project is to compare and analyze the performance of well-known and optimal deep learning models - CNNs (Convolutional Neural Network) and LSTMs (Long and Short Term Memory Networks) - in predicting future stock prices. These models can be used to forecast stock prices due to their ability to learn from and dynamically adapt to complex, noisy, nonlinear, and volatile data. We will train and test these models on New York Stock Exchange historical stock price data for Google.

Introduction

Stock prices are influenced by an ill-defined abundance of variables, making them extremely difficult to predict. Fortunately, a tremendous amount of market data is publicly available, enabling empirical research geared towards deriving methods for forecasting future stock prices. Given the dynamic nature of market data and the large number of variables stock prices depend on, it is nearly impossible to accurately compute the probabilities of different price movements and account for all these factors. Hence, an error driven approach is favorable and will be the center of our research. This paper analyzes and compares two price prediction models: a convolutional neural network (CNN) and a long-short term memory neural network (LSTM). Prior works have shown that these models particularly achieve high accuracy in this task and outperform other modern time series prediction models (such as the ARIMA model). Furthermore, LSTM models are extremely attractive for stock price forecasting heir ability to remember information over periods of time and also process entire sequences instead of just single data points. To narrow our focus, this paper will analyze and compare the performances of the models in predicting the future closing stock prices of the stock GOOGL, corresponding to the Alphabet Inc., listed on NASDAQ.

LSTM Experimental Design

I. Preprocessing

The data preparation stage involved four main components: removing non-stationarity, transforming the data to be labelled, normalizing the data, and formatting the data for the LSTM model. LSTM models achieve greater accuracy when the input data is stationary, meaning constant mean and constant variance across the time series. This is commonly accomplished by differencing to make the mean constant and taking the log of the data values to make the variance constant.

LSTM models are trained via supervised learning. Hence, the model requires labelled data associating inputs with their correct outputs. To transform our time series data to labelled data, for every

closing price data at timestep i ≥ 11 , a data point will be created in the form <X_i-10, X_i-9,...,X_i-1, Y> where Y is the closing price at timestep i and the X's are the preceding price information.

Since different features have different value ranges, we rescale the feature-values to all be between 0 and 1 to make the data consistent.

Finally, we format the data to the form <Samples, Timesteps, Features>, a three dimensional array taken as the input by all recurrent neural networks.

II. Build Model

The architecture of the LSTM network in this experiment contains one hidden layer with 128 LSTM neurons, using the tanh activation function, and one output layer containing three output neurons, which output the linear combination of the weighted inputs.

To train the model, the mean squared error loss function and the efficient ADAM optimization algorithm are utilized.

III. Hyperparameter Tuning

The LSTM network's hyperparameters, the batch size and the number of epochs, were tuned through empirical testing using a validation set. To construct the training, validation, and test set, the New York Stock Exchange dataset on Kaggle was filtered to only contain data associated with GOOGL. Then, we split the data set 80/20 into a training set and a test set, respectively. Finally, we split the training data set, 80/20 to create the final training set and validation set.

After learning the parameters using the training set, the LSTM network was run on the validation set with several different batch sizes and number of epochs. Specifically, we tested the model on the validation set using the following batch sizes: 50, 75, 100, 125, 150, 200. For each batch size, we varied the number of epochs from 1-200. This allowed us to find a suitable configuration such that the batch size allowed for efficient training time and for the model to converge, and the number of epochs did not lead to overfitting.

IV. Test Model

After tuning the hyperparameters, the final LSTM neural network learned from the data with a batch size of 150 and 200 epochs. After learning its parameters, the network ran on the test data, and achieved an extremely impressive average (over 10 trials) prediction accuracy of 94.9%. The high accuracy reflected the LSTM network's strength in forecasting time series data involving patterns. Since LSTM networks have a memory capacity, they are able to remember information and leverage medium, or long term, patterns, unlike convolutional neural networks and most others.

Convolutional Neural Networks (CNNs)

CNN's are specialized types of neural networks that are designed to work with 2-dimensional image data. A CNN consists of two processing layers: the convolutional layers and the pooling layers

(Mehtab, 2020). The convolutional layer reads the input in the form of 2-dimensional images and captures the dependencies in an image by using relevant filter mapping. These filter maps reduce the number of parameters without losing any relevant features. The pooling layer uses the feature mapping to capture the relevant image features. By filtering the image data, the pooling layer reduces the dimension of the image data, thus reducing the computational power needed to process the data (Saha, 2018).

In this project, our CNN model (as shown in figure 1) consists of one 2-dimensional convolution layer that extracts 8 feature maps. The input that layer reads is a time series forecasting from the Google stock price dataset that has been converted to a 2-dimensional image vector. The model uses a Max Pooling layer, a type of pooling layer that returns the maximum values of pixels of the image covered by the filter map. This step helps in reducing the dimension of the image data and also removes noise from data. The output image data is then converted into a 1 dimensional vector for the dense layers. The dense layer is a fully connected hidden layer used to learn non-linear combinations of data. A dropout layer is used to regularize data and remove overfitting. The performance of the layers is optimized by using the adam optimizer, a version of the stochastic gradient descent algorithm, with a learning rate of 0.01. We trained the model using 100 epochs and a batch containing 64 input values (Mehtab, 2020). Finally, the accuracy of the model is tested using the root mean square error (RMSE).

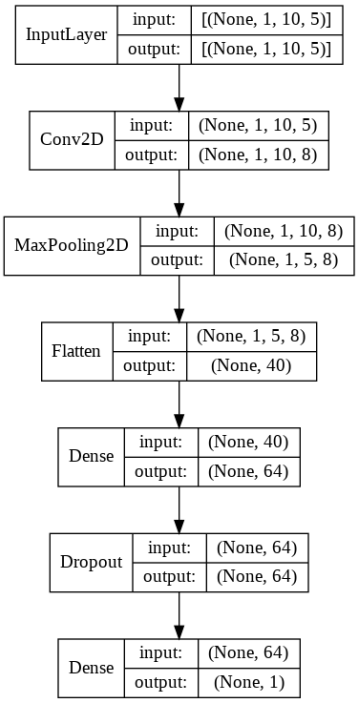
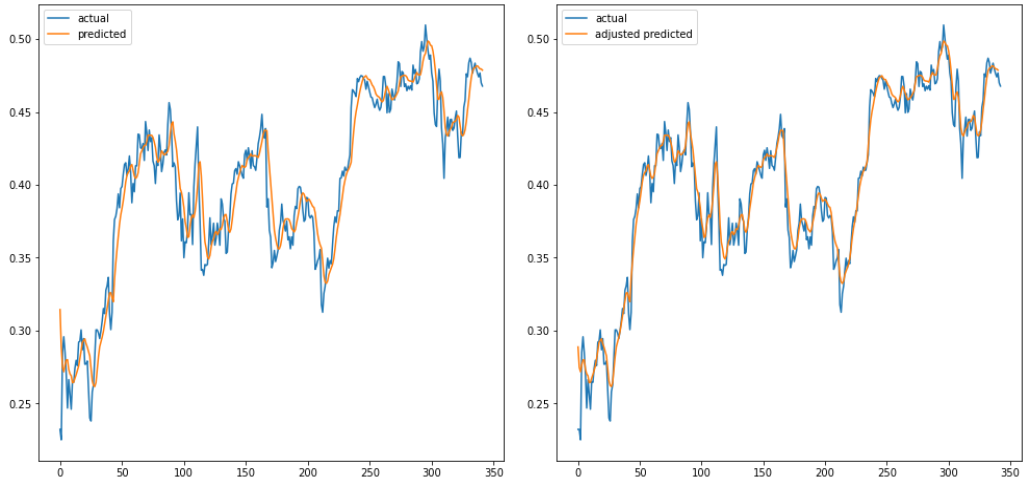


Figure 1: Architecture of CNN model.

Model Comparison and Optimization

Both the LSTM and CNN performed well on our dataset with both being able to easily reach at least 90% with just an increase in epochs. However, lowering the epochs and optimizing other parameters served to be much easier for the LSTM. This makes sense because the parameters available to the single LSTM in Keras are easier to simply increase and receive a better result because they are applied to just the first LSTM layer. Initially, we believed that the CNN allows better configuration in Keras, but after realizing how the parameter changes propagate through the five more layers in the CNN compared to the LSTM model, we realized why we could achieve easier results with the LSTM. Because of the approach we took to the CNN and the additional layers we noticed a lack of consistency between runs even at high epochs from low 80% to 95%.

When first comparing the LSTM and CNN results, we first noticed how much smoother the LSTM graph appears even when closely matched to the expected data. Initially, we believed the smoothness can just be thought of as the limitation of the configured time steps in Keras, but after increasing the units or neurons in the model layer the model very accurately matches the small price differences between days. Although the CNN more easily matches these small changes, the LSTM matches the greater trend much better.

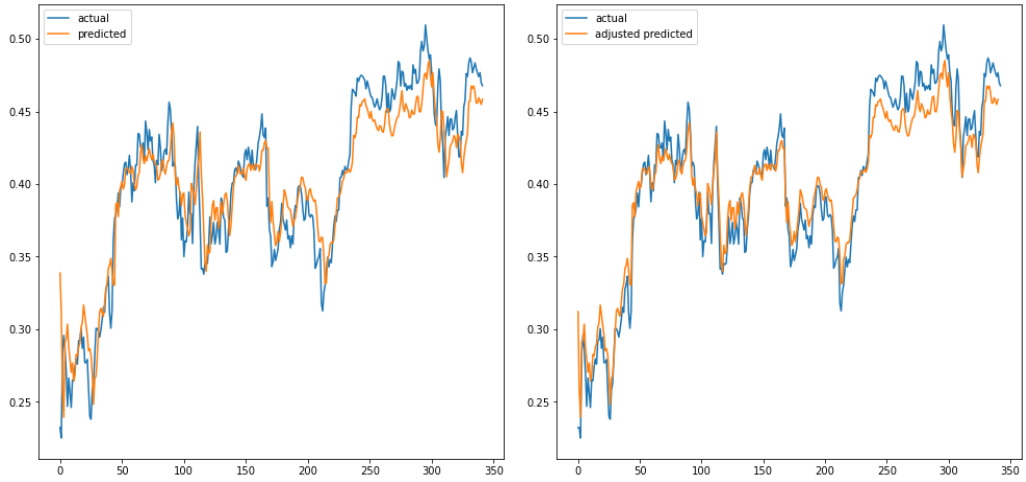


91% *94.6%*

Figure 7: Example(not final) run of original accuracy of lstm vs one day adjusted accuracy

After the initial optimization of the LSTM and increasing the neurons in the layer, the model would compile very slowly at the same epochs as the CNN. After reducing the epochs to a very low amount of 10, it still performed at a higher degree of accuracy even though the calculation per epoch was more expensive(Figure 7).

One thing that we noticed once a high accuracy was reached was an apparent shift of the data to the right. After a one-day-adjusted shift back to the left there was around 4% increase of accuracy for both models(Figure 7,8). Because of the nature of our data and the limitations of our models, both the LSTM and CNN effectively just predict the previous days' closing values. However, this is not a problem especially considering our low feature set of simple stock fundamentals. We could add additional datasets in future work to minimize this issue.



86.5% *89.8%*

Figure 8: Example(not final) run of original accuracy of CNN vs one day adjusted

Prior works

The research paper, *Comparative Analysis of Time-Series Forecasting Algorithms for Stock Price Prediction* (Joosery, Deepa, 2019), analyzes and compares the performance of an LSTM neural network with that of an autoregressive integrated moving average (ARIMA) model. This paper considered a specialized LSTM model with an attention mechanism that served "...to orient the way in which inputs are processed by the model and memory access" (Joosery, et al., p. 2). The mechanism enabled the model to better filter out noisy, useless data, leading to greater performance. While we did not explore this feature, the mechanism should be further researched to determine for which tasks it makes a worthwhile impact.

Conclusion

Our contributions to the project included Naman working on the CNN, Jason on the LSTM, and Lucas on model comparison and optimization, with everyone doing bug fixes. Both the LSTM and the CNN are effective models for the dataset and were able to reach a high degree of accuracy without tuning. Additional tuning led to much greater consistency between runs for both models and allowed the tuning to get easier the more that we did. Reflecting on the choice of dataset, there would likely be a higher degree of accuracy and less delay in prediction if an additional feature set was added from a dataset of news

headlines or other public information; we listed this in the proposal, however the dataset at that point only included top headlines and not a varied enough source. Although academically the five stock fundamentals that we used were perfectly acceptable for reaching a high degree of accuracy, the usability of our models for trading do not necessarily add to the strategy of simply trading on the previous day. In the future, additional public information text datasets could be added to encourage richer results.

Work cited

Saha, Sumit. "A Comprehensive Guide to Convolutional Neural Networks - the ELI5 Way." *Medium*, Towards Data Science, 17 Dec. 2018, towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

Mehtab, Sidra, and Jaydip Sen. "Stock Price Prediction Using CNN and LSTM-Based Deep Learning Models." *2020 International Conference on Decision Aid Sciences and Application (DASA)*, 2020, doi:10.1109/dasa51403.2020.9317207.

Joosery, B., & Deepa, G. (2019). Comparative analysis of time-series forecasting algorithms for stock price prediction. *Proceedings of the International Conference on Advanced Information Science and System*. https://doi.org/10.1145/3373477.3373699

Sayavong, L., Wu, Z., & Chalita, S. (2019). Research on Stock Price Prediction Method Based on Convolutional Neural Network. *2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*. https://doi.org/10.1109/icvris.2019.00050