Transactions on Computer Science and Intelligent Systems Research
ISSN: 2960-1800, eISSN: 2960-2238 | Volume 5
AIDML 2024

Stacked Block Analysis Based on LSTM for Stock Price Prediction

Shuaijie Shao*

Shijiazhuang No.2 High School, Shijiazhuang, China

* Corresponding Author Email: shawn@shawnshao.org

Abstract. Long Short-Term Memory (LSTM) networks and their modified versions, Stacked LSTM, are often used for time series prediction due to their powerful ability to model long-range dependencies. However, few papers have delved into the relationship between the number of layers of LSTM and model performance. For this purpose, this study used LSTM and stacked LSTM at different levels to predict the univariate time series dataset of Apple Inc. stock prices. The goal is to study the relationship between stacking and performance in stock price prediction. The time span in the data is from 2000 to 2022, to discuss whether this relationship and overfitting will damage the model. The results indicate that as long as the model complexity is suitable for the complexity of the dataset, the number of layers is positively correlated with the model performance. It can provide basic knowledge of selecting layers when training other stock market prediction models, which will significantly save time and effort.

Keywords: Long Short-Term Memory; Long-Range Dependencies; Stock Market Prediction Models.

1. Introduction

Stock market prediction has long been a hot topic in machine learning. This is due to its unpredictable and huge financial meanings. A good prediction can earn money for investors, reduce the loss of a company, or even prevent a financial crisis [1, 2]. Meanwhile, the advantages of machine learning are that it can deal with a mass of data, and reveal nonlinear complicated models, making it the exact choice to help solve stock price prediction tasks. However, there are multiple choices in machine learning, and any tiniest parameter can completely change the model performance [3].

Introducing Long short-term memory (LSTM) model in stock market prediction has long been a hot topic [4]. As an advanced form of Recurrent Neural Networks (RNN), LSTM was utilized to remember patterns over time, making this model a reasonable choice when solving time series data, for example, stock market prediction [5]. To show the performance, three LSTM models and two Convolutional Neural Networks (CNN) models have been applied rolling cross-validation, ending up with LSTM having a better performance [6]. Another research combined CNN model and LSTM model, using a sequence array containing historical data and several related indicators. CNN uses this array as an input, and the processed data is then used as LSTM model's input vector, forming a new framework structure named Stock Sequence Array Convolutional LSTM (SACLSTM) [7]. Along with LSTM, other traditional models, Autoregressive Integrated Moving Average (ARIMA), and advanced model, Stacked LSTM or Attentions LSTM, were often introduced in stock price prediction [8]. In general, stacking multiple LSTM layers together allows the model to have greater complexity. Since its hidden layers add depth to the model, each hidden layer can process and recombine the data from its prior layer and return a sequence to the next layer [9]. As an RNN-based model, it has been proved that deep RNN, Stacked LSTM in this case, performs better than shallower ones in several tasks. Some also reported a 4 layers deep model performs better in an encoder-decoder framework [10]. However, increasing layers does not necessarily lead to overfitting, and the model construction should keep well balance between the model complexity and overfitting risk [9].

The main objective of this study is to find the relation between the number of layers of stacked LSTM versus model performance in stock prediction. Specifically, first, vanilla LSTM is constructed and tested as a control group. Second, different layers of stacked LSTM matched with different dropout layers percentages models were set up and evaluated. Third, the Mean Squared Error (MSE), Mean

Absolute Error (MAE), Explained Variance (EV), and Coefficient of Determination (R^2) of all models are analyzed and compared. In addition, using the effect adding dropout layers brings, whether the models have overfitting problems is speculated. The experiment results demonstrate that the number of stacked layers positively correlates with model performance as long as dropout layers are correctly considered and used. This can provide a strong base for choosing the number of layers when training stacked LSTM models in future research, saving considerable time and effort.

2. Methodology

2.1. Dataset Description and Preprocessing

The dataset is from Yahoo Finance, it's the stock price of Apple Inc. from 2000 to 2022 [11]. This dataset contains date, time, open price, highest price, lowest price, close price, volume, dividends, and stock splits. After achieving all data using the 'yfinance' library, the close price is extracted and normalized to 0 to 1 using 'MinMaxScaler'. Then it is put into sequence and reshaped to 80% training data and 20% test data. A hyperparameter, sequence length, is required to be specified, which represents the number of days the model would consider before the analyzing day. In this case, Autocorrelation Function (ACF) graph and Partial Autocorrelation Function (PACF) graph should be drawn, if the result shows that it is not a random walk, the sequence length can be directly determined. Otherwise, testing the MSE of multiple sequence lengths is needed.

Figure 1 is the ACF and PACF graph of the dataset, which is exactly a random walk set. In this case, the MSE of sequence length from 1 to 20 was tested among Vanilla LSTM, 4 layers stacked LSTM with 50% dropout and without dropout layers, 5 layers stacked LSTM with 50% dropout and without dropout layers. The result is in Figure 2, Figure 3 and Figure 4 respectively, which shows that for this random walk, the best performance occurs when sequence length Equals 1.

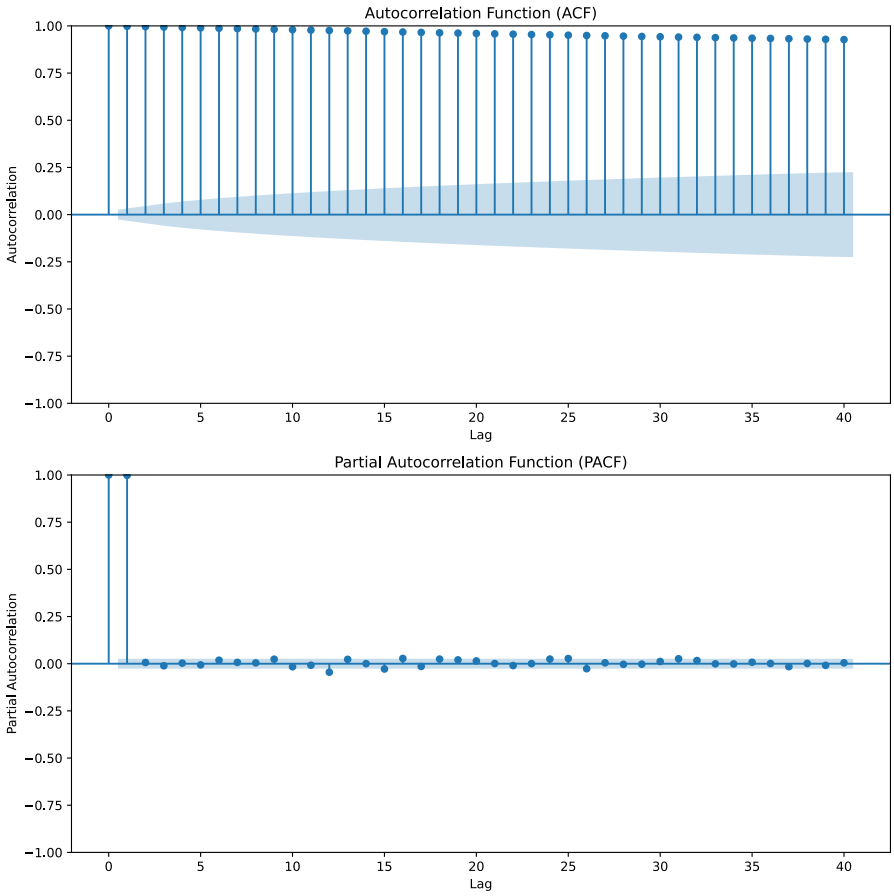

Figure 1. ACF and PACF graph

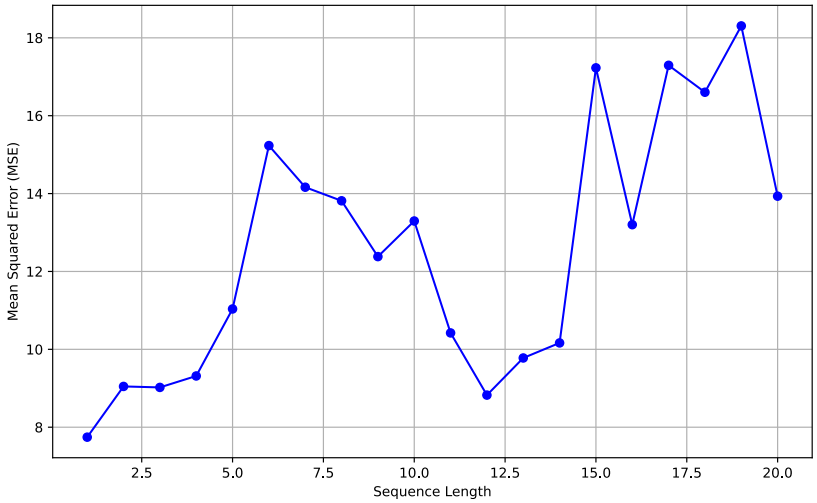

Figure 2. Vanilla LSTM MSE versus sequence length

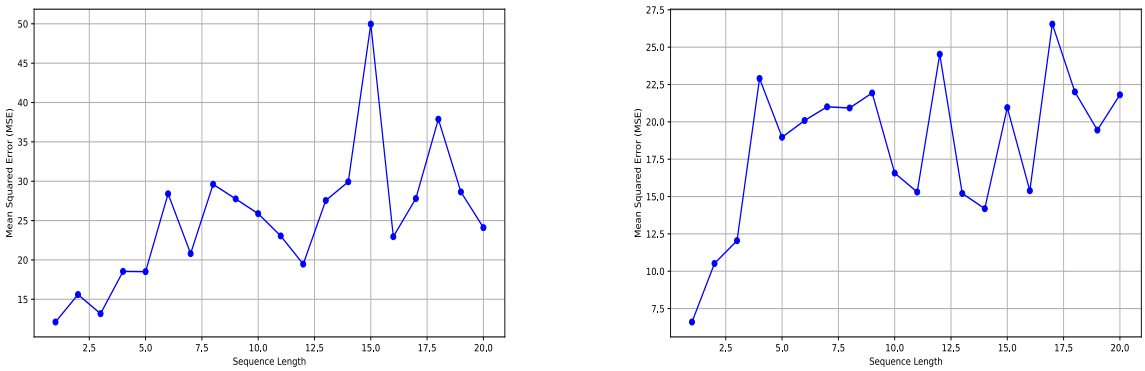

Figure 3. 4 Layers Stacked LSTM MSE versus sequence length

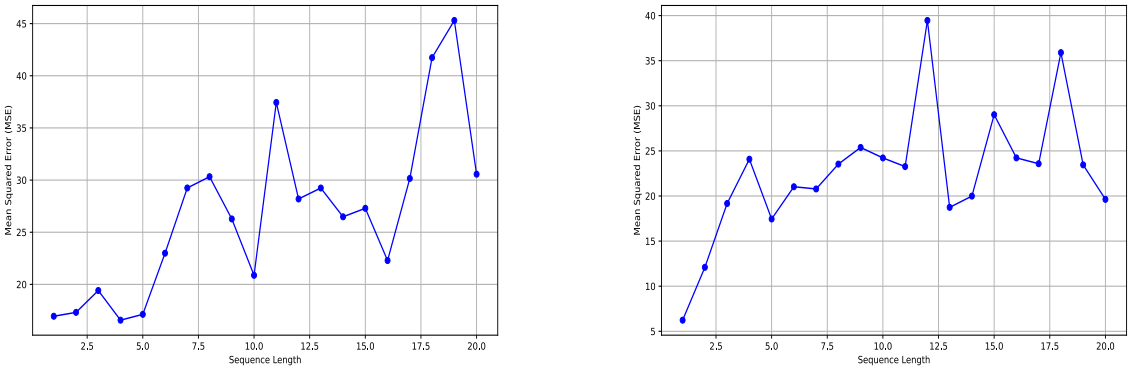

Figure 4. 5 Layers Stacked LSTM MSE versus sequence length

2.2. Proposed Approach

The focus of this research is discovering the relationship between stacked LSTM layers and model performance. In this case, vanilla LSTM model, 2 to 7 layers stacked LSTM models are tested. To determine if overfitting is affecting model performances, different percentages (0%, 20%, 30%, 40%, 50%, 60%) of dropout layers were added to each stacked LSTM model. And calculated the MSE, MAE, EV, and R^2 of all 25 models, and analyzed the relation and overfitting effect. The pipeline of this research is in Figure 5.

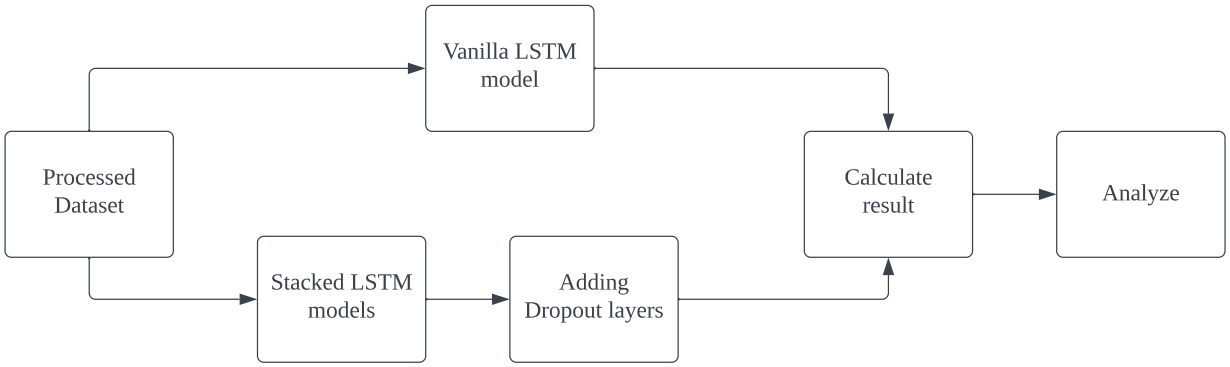

Figure 5. The pipeline of the research

2.2.1. RNN

A simple Feedforward Neural Network (FNN) only uses data for time t (x_t) when deciding the output (h_t) of the current time. This may perform well in tasks like image classification or so, but there will be issues when processing text generation or stock prediction since they require analysis of former data. This is why RNN is needed. A simple RNN neuron takes in the data, calls itself recursively, and transmits the processed data to a future neuron. In this case, every neuron has data not only from time t, but from time 1 to time t. Which is in fact a recursion function as Figure 6 shows. This makes the model suitable for many short-term time sequence problems.

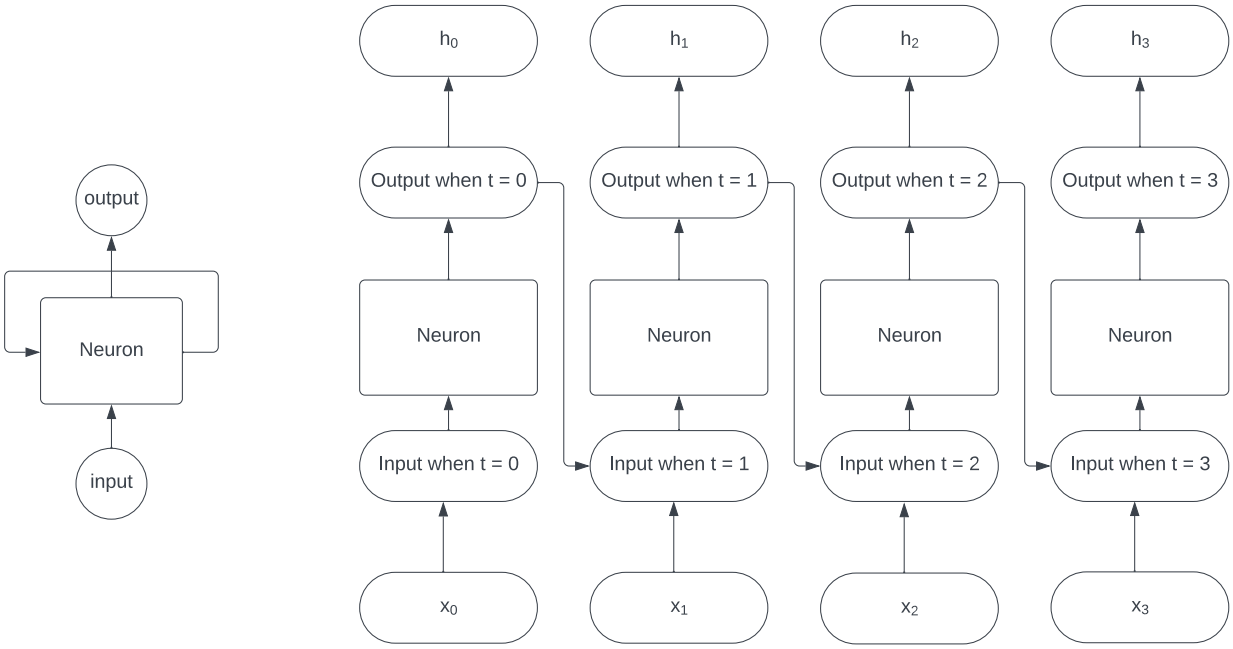

Figure 6. An overall scheme of RNN (left) and an expanded example when t=3 (right)

2.2.2. LSTM

The reason why RNN can solve short-term time sequence problems is due to its cost of memorizing history data. Vanilla RNN remembers exactly all of the data transmitted by the previous neuron and processes them in the current one, this will cause huge computational complexity since you don't know whether you want data from nearby neurons (For example yesterday's stock price might affect today's stock price), or data from neurons far away (For example there is an economic crisis occurring every 30 years which affect stock price greatly). In this case, LSTM was introduced to solve these problems.

Unlike basic RNN which regards every data as equally valuable, LSTM uses a hyperbolic tangent function to decide the weight of the historical data. In this case, some useless data can be removed, and spare space for further calculation. A complete LSTM contains four interaction layers. One of them uses tanh, and the other three use sigmoid activation function.

The overall equations of LSTM and the structure (Figure 7) of it are shown below. Where f_t, i_t, o_t represent the result of forget gate, input gate and output gate respectively. $\widetilde{c_t}$ is the candidate cell state, c_t is the updated cell state, h_t is the hidden state. x_t is the input at time t and h_{t-1} is the time state of time $t-1$. $W_t, W_i, W_c, W_o, U_f, U_i, U_c, U_o, b_f, b_i, b_c, b_o$ are the weight matrices and the bias of different states.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{1}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{2}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{3}$$

$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{5}$$

$$h_t = o_t \odot tanh(c_t) \tag{6}$$

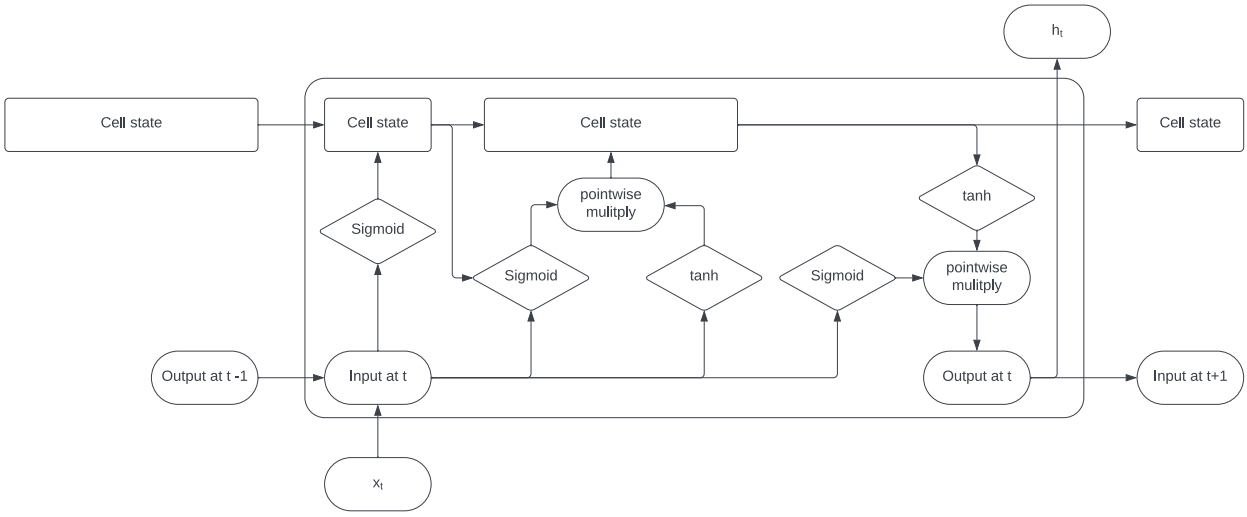

Figure 7. Scheme of Vanilla LSTM

The first gate of this model is the forget gate, this gate takes the previous step output as its input, deciding the weight distributed to the previous given data. Since it uses sigmoid function as an activation function, it generates a number ranging from 0 to 1, and multiplies this number pointwise to the input vector. For example, when the number is 0, the current layer completely forgets all the data provided before and completely remembers everything if the number is 1. So, this gate completes the task of deciding which data is useless and weaken or completely remove those useless data. In the advanced version of LSTM, the input data of this layer also contains the previous cell state. This is shown in Formula (1).

Next is the memory gate, this gate combines two layers: The layer using tanh activation function reads and extracts the useful data from the previous data (shown in formula 4), then the layer with sigmoid activation function generates a number from 0 to 1, and decide the weight arrangement of these useful data (shown in Formula 2). In the advanced version, the sigmoid function layer also acquires data containing the previous cell state, provided by the forget gate results. This is shown in

Formula (5). Finally, is the output gate, it first obtains the vector containing the current input and the previous output and uses a sigmoid function to identify and extract the useful information, which is what Formula (3) does. Then it uses hyperbolic tangent function to compress mapping it to the interval (-1, 1). By consolidating the results and multiplying them point-wise, the final output is the result of this LSTM layer (shown in Formula 6).

2.2.3. Stacked LSTM

Stacked LSTM is based on the previously mentioned vanilla LSTM, which has no difference compared with normal stacked RNNs. The following equations demonstrate a 2 layers LSTM as an example.[8].

Let x_t be the input of layer 1, then the equations of the first layer LSTM will be like this.

$$f_t^{[1]} = \sigma_g(W_f^{[1]}x_t + U_f^{[1]}h_{t-1}^{[1]} + b_f^{[1]}) \tag{7}$$

$$i_t^{[1]} = \sigma_g(W_i^{[1]}x_t + U_i^{[1]}h_{t-1}^{[1]} + b_i^{[1]}) \tag{8}$$

$$o_t{}^{[1]} = \sigma_g(W_o{}^{[1]}x_t + U_o{}^{[1]}h_{t-1}{}^{[1]} + b_o{}^{[1]}) \tag{9}$$

$$\tilde{c}_t{}^{[1]} = tanh(\, W_c{}^{[1]}x_t + U_c{}^{[1]}h_{t-1}{}^{[1]} + b_c{}^{[1]}) \tag{10}$$

$$c_t{}^{[1]} = f_t{}^{[1]} \odot c_{t-1}{}^{[1]} + i_t{}^{[1]} \odot \tilde{c}_t{}^{[1]} \tag{11}$$

$$h_t{}^{[1]} = o_t{}^{[1]} \odot tanh(\, c_t{}^{[1]}) \tag{12}$$

In this case, the input of the second layer LSTM will be $h_t^{[1]}$, and the equation will be like this.

$$f_t^{[2]} = \sigma_g(W_f^{[2]}x_t + U_f^{[2]}h_{t-1}^{[2]} + b_f^{[2]}) \tag{13}$$

$$i_t^{[2]} = \sigma_g(W_i^{[2]}x_t + U_i^{[2]}h_{t-1}^{[2]} + b_i^{[2]}) \tag{14}$$

$$o_t{}^{[2]} = \sigma_g(W_o{}^{[2]}x_t + U_o{}^{[2]}h_{t-1}{}^{[2]} + b_o{}^{[2]}) \tag{15}$$

$$\tilde{c}_t{}^{[2]} = tanh(\, W_c{}^{[2]}x_t + U_c{}^{[2]}h_{t-1}{}^{[2]} + b_c{}^{[2]}) \tag{16}$$

$$c_t{}^{[2]} = f_t{}^{[2]} \odot c_{t-1}{}^{[2]} + i_t{}^{[2]} \odot \tilde{c}_t{}^{[2]} \tag{17}$$

$$h_t{}^{[2]} = o_t{}^{[2]} \odot tanh(\, c_t{}^{[2]}) \tag{18}$$

In the formulas above, $f_t^{[l]}, i_t^{[l]}, o_t^{[l]}$ represent the result of forget gate, input gate, and output gate at layer l respectively. $\tilde{c}_t^{[l]}$ is the candidate cell state at layer l, $c_t^{[l]}$ is the updated cell state at layer l, $h_t^{[l]}$ is the hidden state at layer l. x_t is the input at time t and $h_{t-1}^{[l]}$ is the time state of time t-1 at layer l. $W_t^{[l]}, W_i^{[l]}, W_c^{[l]}, W_o^{[l]}, U_f^{[l]}, U_i^{[l]}, U_c^{[l]}, U_o^{[l]}, b_f^{[l]}, b_i^{[l]}, b_c^{[l]}, b_o^{[l]}$ are the weight matrices and the bias of different states in layer l. Even if there are more layers, the structure is exactly the same, and Figure 8 is a scheme of stacked LSTM.

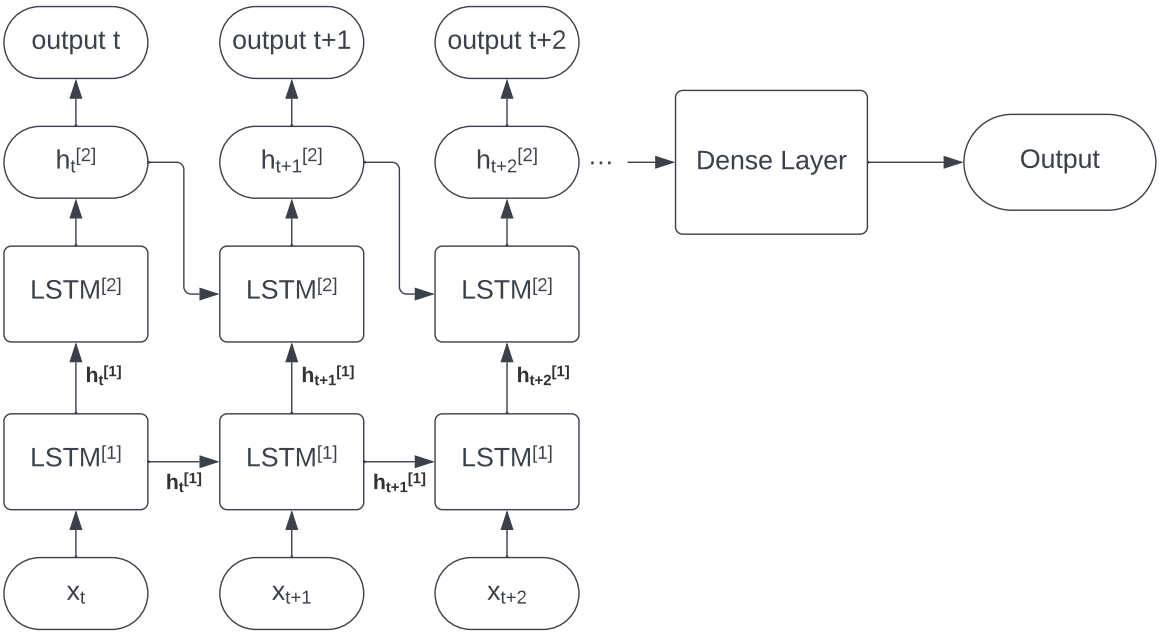

Figure 8. Scheme of stacked LSTM

2.2.4. Loss function

Since the task requires future price prediction, it is a typical regression task, which makes mean squared error a suitable evaluation method. It measures how well the predicted price aligns with the actual prices, in this case, it can show the training progress and model performance.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{19}$$

The above formula denotes the Mean Squared Error loss, in which n is the number of samples or data points, y_i is the actual value of the target variable for the i-th sample. $\hat{y}_i$ is the predicted value of the target variable for the i-th sample. The summation is taken over all n samples.

2.3. Implementation Details

The research used 64-bit python 3.10, libraries are 2.15.0 version Keras, 3.7.2 version Matplotlib, 1.24.3 version Numpy, 2.0.3 version Pandas and 1.3.0 version Scikit learn. All models are trained on a Windows 11 platform, with GPU MX450, having a GPU memory of 2GB and a memory clock of 1.65 GHz, released in 2020. The CPU contains 4 cores and 16 GB RAM. All models are trained with 100 epochs, 50 output units, a single dense hidden layer, hyperbolic tangent activation function, Adam optimization, and MSE loss.

3. Results and Discussion

This section compiled the results of all models and compared all models without dropout layers to discover the relation between the number of layers of stacked LSTM versus model performance. And then analyze the effect of adding different percentages of dropout layers to see if overfitting is changing the model performances.

3.1. Stacked layers versus performance

The accuracy of different layers stacked models without dropout layers is placed in Table 1, by comparing the MSE of each model, it can be seen that Vanilla model and two or three layers stacked models perform significantly better. However, when the number of the stacked layers comes to four, the model performance decreases and slowly becomes better with layers added, but this rate was

relatively slow. This can be explained due to the fewer layers model has a smaller model complexity while making it suitable enough to deal with this amount of data. With the number of layers increasing, the models are allowed to break down the data hence process more complicated data, then performing slightly better during prediction. However, since this dataset is quite simple, stacked LSTM did not show a significant boost in model performance. However, it can still be convinced that for multiple layers stacked LSTM, the number of stacked layers positively correlates with model performance as long as the model complexity suits the data complexity well.

Table 1. Performance of Models without Dropout layers

Model	MSE	MAE	EV	R^2
Vanilla	0.913294	0.370142	0.999237	0.999233
Two Layers	0.917532	0.361706	0.999230	0.999229
Three Layers	1.150721	0.646070	0.999232	0.999034
Four Layers	1.789993	0.539813	0.998654	0.998497
Five Layers	1.602416	0.844119	0.999114	0.998655
Six Layers	1.620915	0.644075	0.998652	0.998639
Seven Layers	1.466351	0.546381	0.998753	0.998698
Eight Layers	1.200970	0.469251	0.998992	0.998992

3.2. Effect of Overfitting

Figure 9 shows the MSE of different layers models with different dropout percentages. Through which showing when dropout layers are added, the MSE increases overall, meaning the model performance decreases with dropout layers added. Based on the fact that dropout layers eliminate data and prevent overfitting problems, it can be inferred that in this case, the model performance was not seriously affected by overfitting. This is due to the simplicity of the dataset and the tested models do not have extreme numbers of layers. This result is compatible with the former study showing that stacked models do not necessarily lead to overfitting, but this conclusion is only in this specific task, meaning that overfitting problems can occur in other conditions.

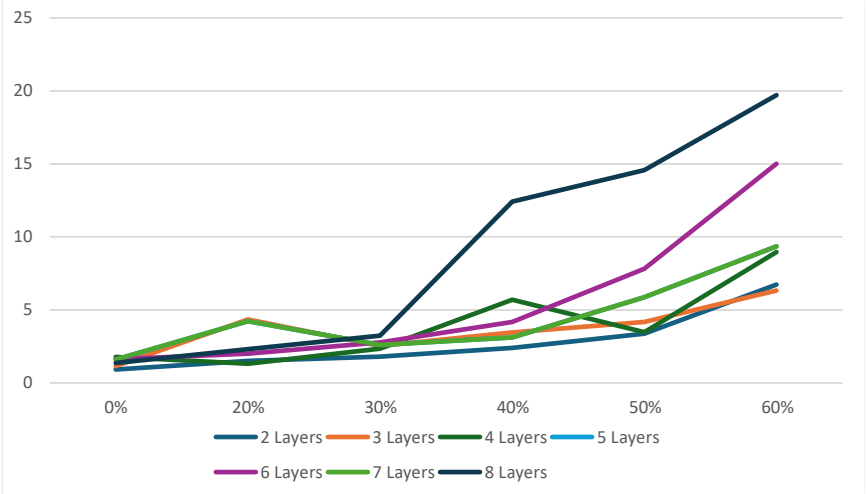

Figure 9. MSE of different layers models and different dropout percentages

4. Conclusion

The core purpose of this study is to investigate the relationship between the number of layers in stacked LSTM models and model performance. Secondly, detailed experiments are introduced to

examine the impact of overfitting. Firstly, Vanilla LSTM and multi-layer LSTM are used as basic models. LSTM has efficient long-distance sequence modeling capabilities. Stock price prediction was used as the basic scenario for this study. The introduction of MSE values is used to check model performance and discard layers to identify any overfitting issues. A large number of experiments were conducted to evaluate the proposed method. The experimental results indicate that there is a positive correlation between the number of layers and model performance when processing appropriate datasets, and no overfitting occurred in the tested model. In future work, more datasets and layers will be introduced in testing to provide stronger support for conclusions and facilitate the training progress of more models.

References

[1] S.P. Chatzis, V. Siakoulis, A. Petropoulos, et al. Forecasting stock market crisis events using deep and statistical machine learning techniques. Expert Systems with Applications, 2018, pp. 353-371.

[2] G. Sonkavde, D.S. Dharrao, A.M. Bongale, et al. Forecasting Stock Market Prices Using Machine Learning and Deep Learning Models: A Systematic Review, Performance Analysis and Discussion of Implications. 2023.

[3] K.E. Hoque and H. Aljamaan, Impact of Hyperparameter Tuning on Machine Learning Models in Stock Price Forecasting. 9, 2021, pp. 163815-163830.

[4] S. Hochreiter, and J. Schmidhuber. Long short-term memory. Neural Computation 9(8), 1997, pp. 1735–1780.

[5] U. Sharma, S. Seniaray, Stock Prediction and Forecasting using Stacked LSTM-Recurrent Neural Network Model, 2021.

[6] H. Hamoudi, A. Mohamed, Stock Market Prediction using CNN and LSTM, Winter 2018.

[7] W. MingTai, N. Herencsar, et al. A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. Multimedia systems, 2023.

[8] Z. Zhichao, Q. Zihao, Using LSTM in Stock prediction and Quantitative Trading, Winter 2020.

[9] C.R. Staudemeyer, E.R. Morris, Understanding LSTM a tutorial into Long Short-Term Memory Recurrent Neural Networks, 2019.

[10] G. Van Houdt, C. Mosquera, & G. Nápoles, A review on the long short-term memory model, 2020.

[11] Information on: https://finance.yahoo.com/.