# Extra Credit project: Portfolio Management

# Performance Analysis of LSTM RNN model by varying its parameters

Leena Dighole

## 1. Introduction

The main aim of this paper is performing tunning of the parameters of LSTM RNN model and analyze their performance. When we are using any deep learning methods and tries to build and train a neural network for specific application then it's hard to choose the optimal parameters for getting sufficiently better results. This problem can be solved by choosing some set of parameters needed to be tuned while building a model and perform some initial trial and errors in order to avoid the model overfitting problem. Overfitting of the model gives less training error but increases the testing error. As the test data set error increases, we can say that the model is predicting results poorly. The problem of overfitting may arise due to multiple parameters. In order to have a basic reference for building any LSTM RNN model, we must perform some parameter tunning trials and try to find the range of parameters suitable for our specific application.

In this paper, we selected 8 parameters and tried to analyze the performances by varying them within specific range of values. The 8 parameters are,

- Varying No of LSTM layers
- Varying dropout rates
- Different batch size
- Different epoch size
- Varying rolling window sizes
- Different optimizing techniques
- Different loss functions for training the model

- Different no of nodes in hidden layers

It’s hard to comment on the optimal no of nodes in hidden layers. It varies for different applications. The false assumption that large no of hidden nodes will give the better results. Similarly, it’s a false assumption that more the no of layers, more accurate will be predictions. In practical, we need to select the no of parameters wisely while building a model for a specific application. The performance analysis is done based upon training and testing errors. SP500 index is used as the benchmark for this paper. The top 5 stocks are selected using their market capitalization as a selection criteria. The daily data from yahoo finance from 2013 to 2020 is used for the analysis.

## 2. Literature Review

LSTM RNN model captures the patterns of time series data using short term memory which increases its prediction accuracy. LSTM is the special architecture of RNN which remembers the values of either long or short time durations[3]. Jia [4] investigated the effectiveness of usage of LSTM RNN for stock predictions. Tingwei [5] showed the performances of LSTM networks and SVR using SP500 as a benchmark. The models are compared using RMSE, MAE, MAPE, AMAPE errors supporting the fact that LSTM has the lowest error. Van in his research paper supports the same observations [7] about outperformance of LSTM RNN over SVR and Linear Regression. He used Adam optimizer for training LSTM RNN model.

Yilin [2] in his research paper compared the performances of 3 machine learning and 3 deep learning algorithms. He used LSTM RNN, Convolutional Neural Network (CNN), Deep Multilayer Perceptrons (DMLP) as deep learning algorithms and Random Forest (RF) , Support Vector Regressor(SVR) and ARIMA models as machine learning algorithms. The ascending order as per the performances of these models is : RF,LSTM RNN, DMLP, SVR, ARIMA, CNN.

While using LSTM RNN model for stock predictions, its wise to have a reference of the range of the parameters tuning to build a model with better prediction accuracy. This project tries to find the optimal range and values of the parameters for building an LSTM RNN model for stock predictions.

## 3. LSTM RNN model

Long Short Term Memory Recurrent Neural Network is a special type of RNN models. The important features of memory extension makes it more suitable for time series predictions. Each neuron in LSTM has a memory cell that links current data with the previous ones.

LSTM has 3 gates and they are Input gate(It), forget gate(Ft), output gate (Ot). Writing a certain neural network into memory involves inputting it to memory cell. Forget get allows us to decide whether previous data is important to be involved for prediction at current state or not. LSTM RNN is used to overcome the limitations of RNN to retain the long term information property. These memory cells are present in each hidden layer. This paper uses ADAM optimizer for training the NN. Adam is better than stochastic gradient descent algorithms and its widely used for stock predictions supported by this paper references.

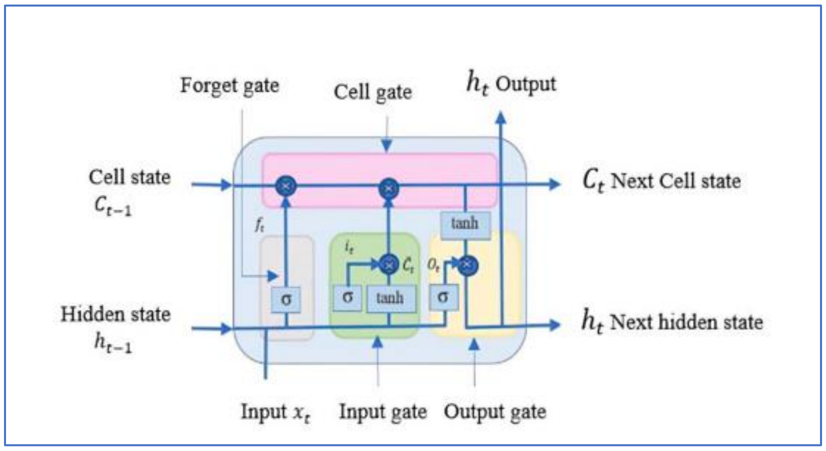


*Figure 1: LSTM structure at time step t*

Memory cell makes decisions about which information to be stored. It also decides when to allow reading, forgetting and writing into the memory. The activation functions are used which computes values between 0 and 1. Commonly used activation functions are sigmoid, tanh and Relu functions.

The activation functions of LSTM RNN networks for a memory cell are:

$$
\begin{aligned}
i_t &= \sigma_1(W_i x_t + U_i c_{t-1} + b_i)\\
f_t &= \sigma_1(W_f x_t + U_f c_{t-1} + b_f)\\
o_t &= \sigma_1(W_o x_t + U_o c_{t-1} + b_o)\\
c_t &= f_t \bullet c_{t-1} + i_t \bullet \sigma_1(W_c x_t + b_c)\\
h_t &= o_t \bullet \sigma_2(c_t)
\end{aligned}
$$

*Figure 2: LSTM memory cell equations*

Where, $x_t$ is the input vector at time t

$h_t$ is the output vector at time t

$c_t$ is the memory cell state at time t

$i_t$ is the input gate vector at time t

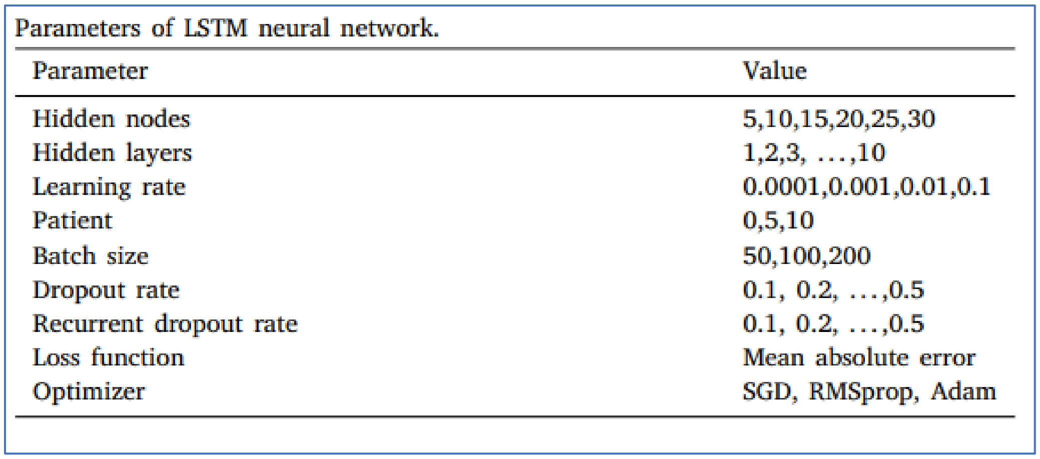
$f_t$ is the forget gate vector at time t

$o_t$ is output gate vector at time t

W and U are weight matrices and b are biases

Sigma represents activation functions

Various parameters for selecting LSTM RNN networks can be viewed in below picture:



Parameters of LSTM neural network.

| Parameter | Value |
|---|---|
| Hidden nodes | 5,10,15,20,25,30 |
| Hidden layers | 1,2,3, …,10 |
| Learning rate | 0.0001,0.001,0.01,0.1 |
| Patient | 0,5,10 |
| Batch size | 50,100,200 |
| Dropout rate | 0.1, 0.2, …,0.5 |
| Recurrent dropout rate | 0.1, 0.2, …,0.5 |
| Loss function | Mean absolute error |
| Optimizer | SGD, RMSprop, Adam |

*Figure 3: LSTM Parameters*

## 4. Parameter Tunning and Performance analysis

We will discuss the performance of LSTM RNN model by tunning its various parameters. In this analysis, we will tune 8 different parameters of the model.

Lets analyze the performances of different parameter tunning and try to get a reference of model parameter range while constructing a LSTM RNN model for stock predictions. We used top 5 stocks daily returns from 01/01/2013 to 12/25/2020 in this paper for analysis. We used RMSE and MSE errors for the comparison of performances of different parameters in different cases.

### Case 1:

**Motivation:** Performance analysis with different No of LSTM Layers

We will use 3 different LSTM models. Model 1 consists of 4 layers with one input, one output and 2 LSTM hidden layers. As our aim is to check whether the prediction accuracy of the model increases by increasing no of hidden layers of LSTM RNN. We will keep our basic specifications listed below and vary the no of hidden layers.

**Basic specifications:**

Rolling window =100

Batch size =10

epoch=5

dropout =0

**Performance table:**

| **Parameters** | **LSTM RNN(4 layers)** | **LSTM RNN(5 layers)** | **LSTM RNN(6 layers)** |
|---|---|---|---|
| Hidden layers | 2 | 3 | 4 |
| RMSE | 0.02650304 | 0.02618564 | 0.02595340 |
| MAE | 0.01704586 | 0.01663435 | 0.01652306 |

*Table 3: LSTM RNN model performances with different layers*

**Observations:**

- As the no of Hidden NN layers in LSTM RNN models increases, the test RMSE and MAE errors decreases
- The prediction accuracy increases as the no of hidden layers in model increases.
- Increasing the no of hidden layers in NN adds up the computational cost for training the NN.
- We have to choose the no of hidden layers based upon our time and computational capacity of the available resources.

---

**<u>Case 2:</u>**

**Motivation:** Performance analysis with different dropout rates

We will use 3 different dropout rates. We will dropout rates equals to 0,0.2,0.5 for our performance analysis. As our aim is to check whether the prediction accuracy of the model increases by increasing dropout rate of LSTM RNN model. We will keep our basic specifications listed below and vary the dropout rates.

The dropout percentage (probability value between range [0,1]) randomly drops the input data at each node. The random dropping avoids problem of overfitting the data into LSTM RNN model giving better performance.

**Basic specifications:**

Rolling window =100

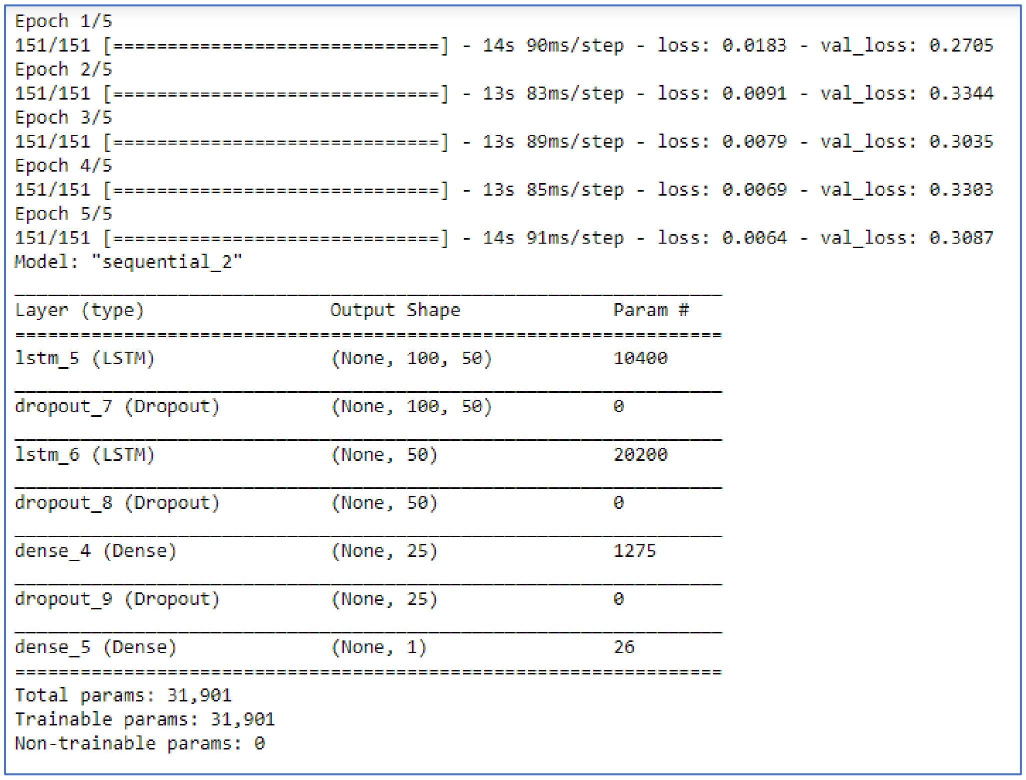
Batch size =10

epoch=5

LSTM model layers =4

For dropout rate =0,

The model summary and no of parameters tuned by model can be seen in below figure,



```
Epoch 1/5
151/151 [==============================] - 14s 90ms/step - loss: 0.0183 - val_loss: 0.2705
Epoch 2/5
151/151 [==============================] - 13s 83ms/step - loss: 0.0091 - val_loss: 0.3344
Epoch 3/5
151/151 [==============================] - 13s 89ms/step - loss: 0.0079 - val_loss: 0.3035
Epoch 4/5
151/151 [==============================] - 13s 85ms/step - loss: 0.0069 - val_loss: 0.3303
Epoch 5/5
151/151 [==============================] - 14s 91ms/step - loss: 0.0064 - val_loss: 0.3087
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_5 (LSTM)                (None, 100, 50)           10400     
_________________________________________________________________
dropout_7 (Dropout)          (None, 100, 50)           0         
_________________________________________________________________
lstm_6 (LSTM)                (None, 50)                20200     
_________________________________________________________________
dropout_8 (Dropout)          (None, 50)                0         
_________________________________________________________________
dense_4 (Dense)              (None, 25)                1275      
_________________________________________________________________
dropout_9 (Dropout)          (None, 25)                0         
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 26        
=================================================================
Total params: 31,901
Trainable params: 31,901
Non-trainable params: 0
```

*Figure 4: Model summary with Dropout rate =0*

**Performance table:**

| **Parameters** | **Model 1** | **Model 2** | **Model 3** |
|---|---|---|---|
| Dropout rate | 0 | 0.2 | 0.5 |
| RMSE | 0.02650304 | 0.02618564 | 0.02501940 |
| MAE | 0.01704586 | 0.01700873 | 0.01697757 |

*Table 2: LSTM RNN model performances with different dropout rates*

**Observations:**

- As the dropout rate in LSTM RNN models increases, the test RMSE and MAE errors decreases.

- The prediction accuracy increases as the dropout rate in model increases as the RMSE and MAE error decreases.
- Dropout probability of 0.5 gives us the optimal results as increasing the dropout rate beyond 0.5 will force model to be trained using less than 50% of the data.
- Optimal range of drop out rate is between 0 to 0.5

---

**Case 3:**

**Motivation:** Performance analysis with different batch size for training model

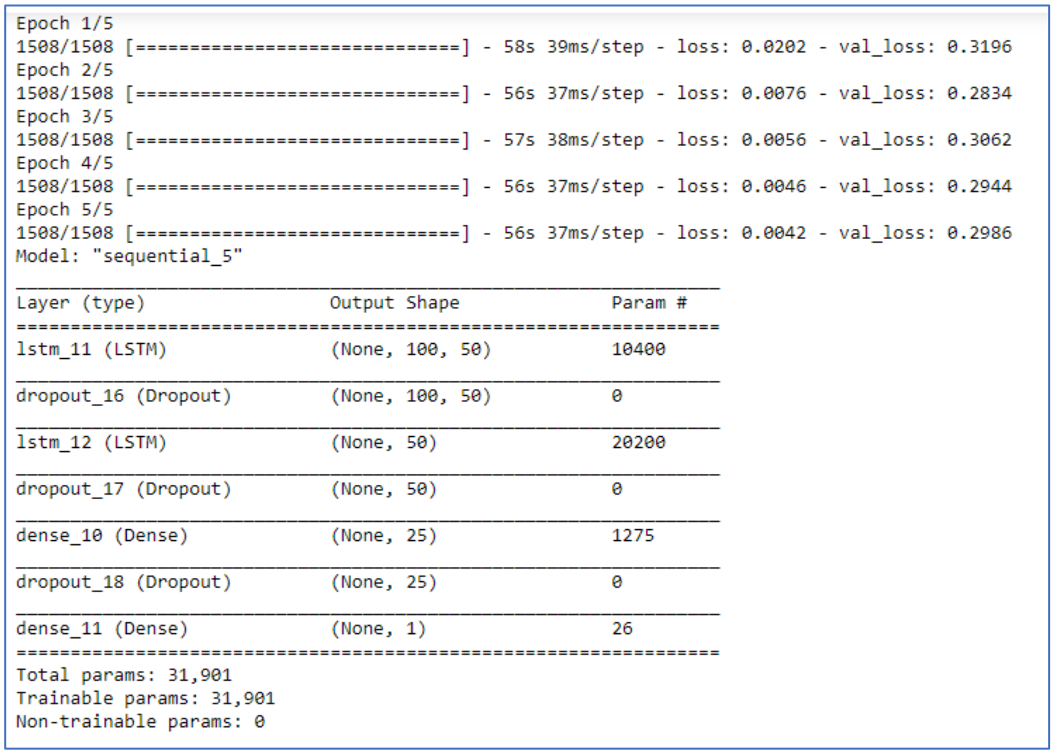
We will use 3 different LSTM models. Model 1 consists of 4 layers with one input, one output and 2 LSTM hidden layers. As our aim is to check whether the prediction accuracy of the model increases by increasing the batch size of LSTM RNN. We will keep our basic specifications listed below and vary the no of batch sizes.

Batch size of training data is used for Stochastic Gradient descent optimization. Batch size of 1 takes all the training data to train the model. By passing batch size greater than one, we will divide the entire dataset into given batch size and pass one batch at a time to train the model.

For 100 training dataset, batch size = 10 will make small batches of size 100/10 =10 to train the model.

For batch size =1,

The model summary and no of parameters tuned by model can be seen in below figure,



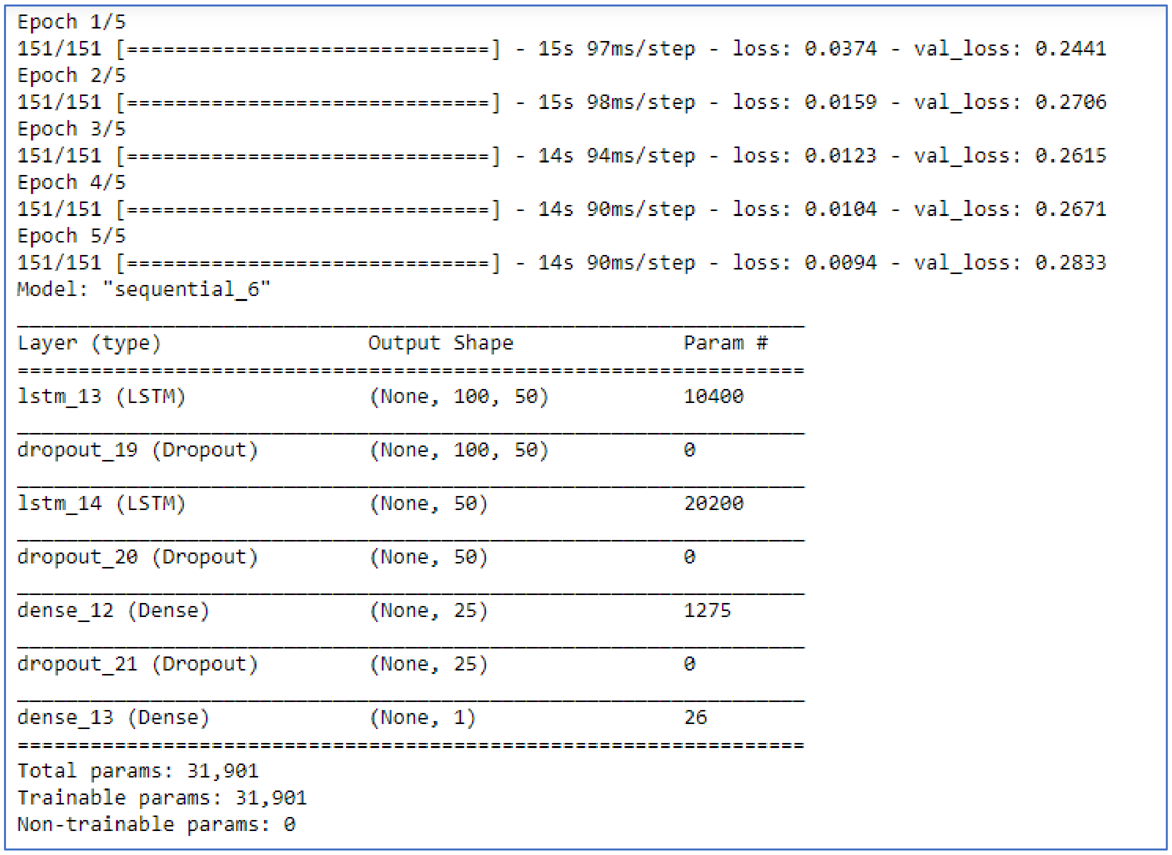
```
Epoch 1/5
1508/1508 [==============================] - 58s 39ms/step - loss: 0.0202 - val_loss: 0.3196
Epoch 2/5
1508/1508 [==============================] - 56s 37ms/step - loss: 0.0076 - val_loss: 0.2834
Epoch 3/5
1508/1508 [==============================] - 57s 38ms/step - loss: 0.0056 - val_loss: 0.3062
Epoch 4/5
1508/1508 [==============================] - 56s 37ms/step - loss: 0.0046 - val_loss: 0.2944
Epoch 5/5
1508/1508 [==============================] - 56s 37ms/step - loss: 0.0042 - val_loss: 0.2986
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_11 (LSTM)               (None, 100, 50)           10400
_________________________________________________________________
dropout_16 (Dropout)         (None, 100, 50)           0
_________________________________________________________________
lstm_12 (LSTM)               (None, 50)                20200
_________________________________________________________________
dropout_17 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_10 (Dense)             (None, 25)                1275
_________________________________________________________________
dropout_18 (Dropout)         (None, 25)                0
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 26
=================================================================
Total params: 31,901
Trainable params: 31,901
Non-trainable params: 0
```

*Figure 5: Model summary with Batch size =1*

From above figure, we can notice that the batch size = 1 takes all the training parameter and due to epoch size =5, it will train the model 5 times by passing all the 1508 data at each iteration. The training error loss and validation loss decreases from iteration 1 to 5 which shows that the LSTM RNN model weights of hidden layers are getting tuned to give more accurate predictions.

For batch size = 10,

The batch of randomly chosen 151 datapoints is passed as a training data to the model.



```
Epoch 1/5
151/151 [==============================] - 15s 97ms/step - loss: 0.0374 - val_loss: 0.2441
Epoch 2/5
151/151 [==============================] - 15s 98ms/step - loss: 0.0159 - val_loss: 0.2706
Epoch 3/5
151/151 [==============================] - 14s 94ms/step - loss: 0.0123 - val_loss: 0.2615
Epoch 4/5
151/151 [==============================] - 14s 90ms/step - loss: 0.0104 - val_loss: 0.2671
Epoch 5/5
151/151 [==============================] - 14s 90ms/step - loss: 0.0094 - val_loss: 0.2833
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_13 (LSTM)               (None, 100, 50)           10400     
_________________________________________________________________
dropout_19 (Dropout)         (None, 100, 50)           0         
_________________________________________________________________
lstm_14 (LSTM)               (None, 50)                20200     
_________________________________________________________________
dropout_20 (Dropout)         (None, 50)                0         
_________________________________________________________________
dense_12 (Dense)             (None, 25)                1275      
_________________________________________________________________
dropout_21 (Dropout)         (None, 25)                0         
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 26        
=================================================================
Total params: 31,901
Trainable params: 31,901
Non-trainable params: 0
```

*Figure 6: Model summary with Batch size =10*

**Basic specifications:**

Rolling window =100

epoch=5

dropout =0.5

LSTM model = 4 layers

**Performance table:**

| **Parameters** | **Model 1** | **Model 2** | **Model 3** |
|---|---|---|---|
| Batch size | 1 | 10 | 20 |
| RMSE | 0.0249314 | 0.0255783 | 0.02541987 |
| MAE | 0.0168839 | 0.0177530 | 0.01753955 |

*Table 3: LSTM RNN model performances with different batch sizes*

**Observations:**

- As the batch size for training LSTM RNN models increases, the test RMSE and MAE errors increases
- The prediction accuracy decreases as the no of batch size in model increases.
- More the no of training dataset, less will be the RMSE and MAE error.
- As the batch size divides original training data set into batches randomly, the computational time for training the model decreases as we increase the no of batch size for training the model.

---

<u>**Case 4:**</u>

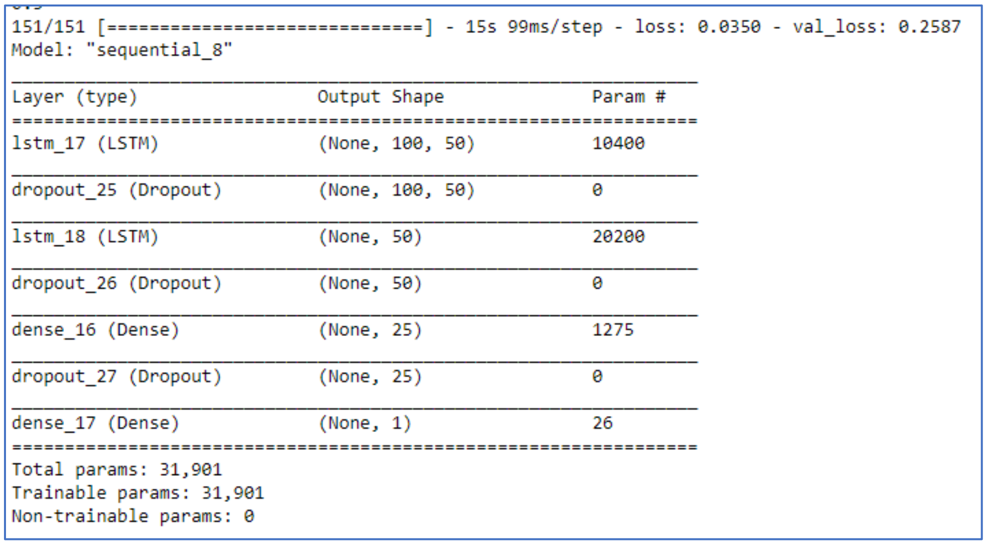
**Motivation:** Performance analysis with different epoch size for training model

We will use 3 different LSTM models. Model 1 consists of 4 layers with one input, one output and 2 LSTM hidden layers. As our aim is to check whether the prediction accuracy of the model increases by increasing the epoch size of LSTM RNN. We will keep our basic specifications listed below and vary the no of epoch sizes.

Epoch size is the no of times the model will be passed through the given training data set. Epoch size of will train the model once. By passing epoch size greater than one, we will increase the training iterations of the model. For 500 training dataset, epoch size = 10 will train the model 10 times. This will increase the prediction accuracy.

For Epoch size =1,

The model summary and no of parameters tuned by model can be seen in below figure,



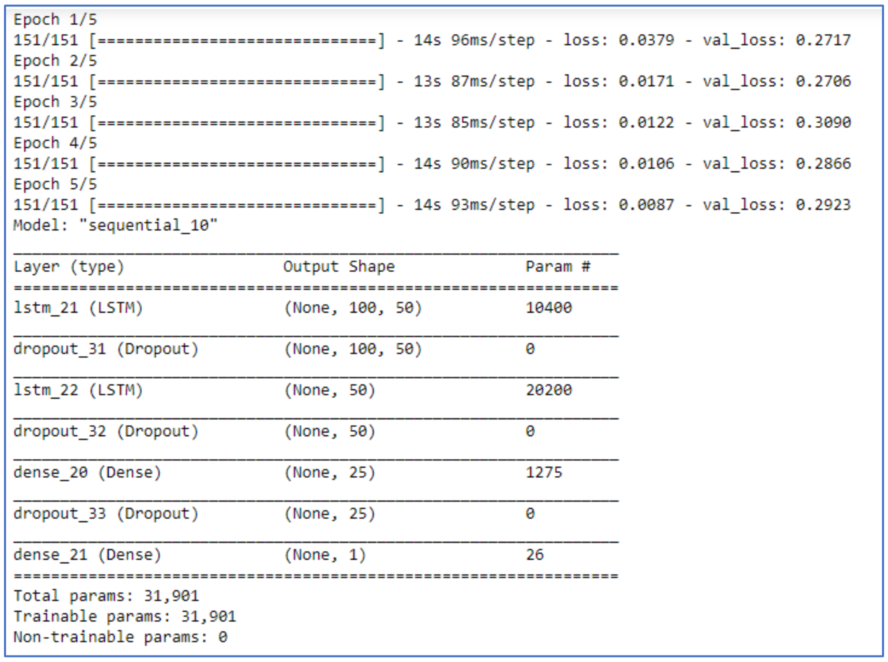
```
151/151 [==============================] - 15s 99ms/step - loss: 0.0350 - val_loss: 0.2587
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_17 (LSTM)               (None, 100, 50)           10400
_________________________________________________________________
dropout_25 (Dropout)         (None, 100, 50)           0
_________________________________________________________________
lstm_18 (LSTM)               (None, 50)                20200
_________________________________________________________________
dropout_26 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_16 (Dense)             (None, 25)                1275
_________________________________________________________________
dropout_27 (Dropout)         (None, 25)                0
_________________________________________________________________
dense_17 (Dense)             (None, 1)                 26
=================================================================
Total params: 31,901
Trainable params: 31,901
Non-trainable params: 0
```

*Figure 7: Model summary with Epoch size =1*

From above figure, we can notice that the epoch size = 1 takes all the training parameter generated with batch size =10 and it will train the model once. The batch size of 10 will be trained once with epoch size of 1.

For epoch size = 5,

The model will get trained 5 times in order to increase the prediction accuracy.



```
Epoch 1/5
151/151 [==============================] - 14s 96ms/step - loss: 0.0379 - val_loss: 0.2717
Epoch 2/5
151/151 [==============================] - 13s 87ms/step - loss: 0.0171 - val_loss: 0.2706
Epoch 3/5
151/151 [==============================] - 13s 85ms/step - loss: 0.0122 - val_loss: 0.3090
Epoch 4/5
151/151 [==============================] - 14s 90ms/step - loss: 0.0106 - val_loss: 0.2866
Epoch 5/5
151/151 [==============================] - 14s 93ms/step - loss: 0.0087 - val_loss: 0.2923
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_21 (LSTM)               (None, 100, 50)           10400
_________________________________________________________________
dropout_31 (Dropout)         (None, 100, 50)           0
_________________________________________________________________
lstm_22 (LSTM)               (None, 50)                20200
_________________________________________________________________
dropout_32 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_20 (Dense)             (None, 25)                1275
_________________________________________________________________
dropout_33 (Dropout)         (None, 25)                0
_________________________________________________________________
dense_21 (Dense)             (None, 1)                 26
=================================================================
Total params: 31,901
Trainable params: 31,901
Non-trainable params: 0
```

*Figure 8: Model summary with Epoch size =5*

**Basic specifications:**

Rolling window =100

Batch size=10

dropout =0.5

LSTM model = 4 layers

**Performance table:**

| **Parameters** | **Model 1** | **Model 2** | **Model 3** |
|---|---|---|---|
| Epoch | 1 | 5 | 10 |
| RMSE | 0.0276880 | 0.02513784 | 0.02524649 |
| MAE | 0.0205788 | 0.01713565 | 0.01787873 |

*Table 4: LSTM RNN model performances with different epochs*

**Observations:**

- As the epoch size for training LSTM RNN models increases, the test RMSE and MAE errors decreases.
- The prediction accuracy increases as the no of epoch size of model increases.
- More the no of training dataset is iterated for training the model, less will be the RMSE and MAE error.
- The testing error for epoch = 10 is higher due to overfitting of the data.
- Epoch size of 5 is giving us optimal accuracy with comparatively at a lower computational time cost.
- The proper selection of a batch size and no of epochs depends upon the users willingness about choosing an effective values by managing a tradeoff between accuracy and computational cost.

**<u>Case 5:</u>**

**Motivation:** Performance analysis with different rolling window size

We will use 3 different LSTM models. Model 1 consists of 4 layers with one input, one output and 2 LSTM hidden layers. As our aim is to check whether the prediction accuracy of the model increases by increasing the rolling window size of training data for LSTM RNN model. We will keep our basic specifications listed below and vary the rolling window sizes.

The rolling window size groups the training data which serves as the input to each node of the neural Network. More the rolling window size, more data will be passed as the input data for training the LSTM RNN model.

Note that these rolling window sizes are compared using the daily returns of the data.

For monthly returns the testing size will vary from daily returns dataset.

**Basic specifications:**

Epoch =5

Batch size=10

dropout =0.5

LSTM model = 4 layers

**Performance table:**

| **Parameters** | **Model 1** | **Model 2** | **Model 3** |
|---|---|---|---|
| Rolling window | 10 | 100 | 200 |
| RMSE | 0.02584991 | 0.0251378 | 0.025357 |
| MAE | 0.01811273 | 0.0171356 | 0.017446 |

*Table 5: LSTM RNN model performances with different rolling windows*

**Observations:**

- As the rolling window size for training LSTM RNN models increases, the test RMSE and MAE errors decreases.
- The prediction accuracy increases as the input dataset of model increases.
- The testing error for rolling window = 200 is higher due to overfitting of the data.
- Rolling window size of 100 seems more optimal size to avoid overfitting.
- Just increasing the no of training input data doesn't improve NN predictions. It may lead to overfitting the model.
- As we increase the rolling window size of the data, the training input for particular NN node increases. This increases the computational time of training the LSTM RNN model.

---

**<u>Case 6:</u>**

**Motivation:** Performance analysis with different optimizers for training the model.

As our aim is to check whether the prediction accuracy of the model increases by using different optimizing functions for training LSTM RNN model. We will keep our basic specifications listed below and vary optimizing technique.

We will train our algorithm using Adam optimizer, Stochastic Gradient descent optimizer and Adamax optimizer. The Adam optimizer works better than other optimizer.

Adamax is the extension of the adam optimizer and using infinity norm. Stochastic Gradient descent method is the extension of gradient descent method by calculating gradient of single randomly chosen datapoint than gradient of all datapoints. Adam is a stochastic optimization technique using only first order gradient of the datapoints.

**Basic specifications:**

Epoch =5

Batch size=10

dropout =0.5

LSTM model = 4 layers

Rolling window =100

**Performance table:**

| **Parameters** | **Model 1** | **Model 2** | **Model 3** |
|---|---|---|---|
| Optimiser | adam | SGD | Adamax |
| RMSE | 0.0251378 | 0.0256917 | 0.0252039 |
| MAE | 0.0171356 | 0.0178970 | 0.0172209 |

*Table 6: LSTM RNN model performances with different optimizing technique*

**Observations:**

- Adam optimizer works better than Stochastic Gradient Descent optimizer and Adamax optimizer as it has minimum training error.
- According to literature survey, Adam optimizer outperforms other optimizing technique. Our findings are inline with our literature survey.
- The important feature of adam optimizer is that it naturally chooses the optimal stepsize suitable for minimizing loss function error.
- The computational time for all optimizing techniques has no significance difference while training our LSTM RNN model.

---

<u>**Case 7:**</u>

**Motivation:** Performance analysis with different loss functions for training the model.

As our aim is to check whether the prediction accuracy of the model changes by using different loss functions for training LSTM RNN model. We will keep our basic specifications listed below and vary loss functions for adam optimizing technique.

We will train our algorithm using mean squared error, mean absolute error, mean absolute percentage error.

The formula for Mean squared error is,

**loss = square(y_true - y_pred)**

The Formula for Mean Absolute error is,

**loss = abs(y_true - y_pred)**

The formula for Mean Absolute percentage error is,

**loss = 100 * abs(y_true - y_pred) / y_true**

The optimizer tries to lower the loss function error to tune the weights and bias values of the LSTM RNN model at each iteration while training the model.

Lets see the results of using these different loss functions used for training our model.

**Basic specifications:**

Epoch =5

Batch size=10

dropout =0.5

LSTM model = 4 layers

Rolling window =100

Optimizer = Adam

**Performance table:**

| **Parameters** | **Model 1** | **Model 2** | **Model 3** |
|---|---|---|---|
| Loss functions | Mean square error | Mean Absolute Error | Percent mean absolute error |
| RMSE | 0.0251378 | 0.02477145 | 0.02589414 |
| MAE | 0.0171356 | 0.01674728 | 0.01818977 |

*Table 7: LSTM RNN model performances with different loss functions*

**Observations:**

- The Mean Absolute Error loss function is having minimum test RMSE and MAE errors.
- The percent mean absolute error performs worst than other loss functions.
- Generally, we use mean squared error as a training loss function which has comparatively minimum testing error.
- Different loss functions will give almost similar results and their computational cost is also same.
- When we round off the RMSE and MAE errors to 3 decimal digits, we wont get any difference in errors which shows that all loss functions behaves similar and their results has no significance difference.

---

<u>**Case 8:**</u>

**Motivation:** Performance analysis with different no of nodes of hidden layers.

As our aim is to check whether the prediction accuracy of the model changes by using different no of nodes for building the LSTM RNN model. We can build different models by changing the total no of hidden layers and corresponding no of nodes for each hidden layer. We will use LSTM model with 4 layers and we will vary the no of nodes of each hidden layer and try to analyze the performances of them.

We will use 3 models with different no of hidden nodes.

**For Model 1**,

Total no of layers =4

Input layer nodes = 50

Output layer node =1

First Hidden layer node = 50

Second Hidden layer node = 25

**For Model 2,**

Total no of layers =4

Input layer nodes = 100

Output layer node =1

First Hidden layer node = 50

Second Hidden layer node = 25

**For Model 3,**

Total no of layers =4

Input layer nodes = 100

Output layer node =1

First Hidden layer node = 100

Second Hidden layer node = 50

The other parameters used for building above 3 models are,

**Basic specifications:**

Epoch =5

Batch size=10

dropout =0.5

LSTM model = 4 layers

Rolling window =100

Optimizer = Adam

Loss function=mean square error

**Performance table:**

| **Parameters** | **Model 1** | **Model 2** | **Model 3** |
|---|---|---|---|
| Hidden layer nodes | (50,50,25,1) | (100,50,25,1) | (100,100,50,1) |
| RMSE | 0.0251378 | 0.0249225 | 0.0255813 |
| MAE | 0.0171356 | 0.0168684 | 0.0177622 |

*Table 8: LSTM RNN model performances with different no of hidden layers*

**Observations:**

- Total no of hidden layers and the no of nodes of each layer is hard to choose unlike optimizing function or size of epoch.
- The different no of layers and no of nodes adds the variations into the output.
- Its hard to comment on the optimal no of nodes in hidden layers. It varies for different applications.
- The false assumption that large the no of hidden nodes will give the better results.
- Similarly its a false belief that more the no of layers more accurate will be predictions.
- In practical, its always wise to use more no of layers but it varies completely depending upon different applications.
- For a particular application, we can perform different parameter tuning with available time and computational cost and should try to choose the parameters which gives us approximately better results which are expected.
- As a data scientist, one must try to manage the tradeoff between accuracy and available time and computational cost resources for a particular application to make better predictions.

## 5. Conclusion

We performed parameter tuning of the LSTM model and tried to find out the optimal parameter values for stock prediction application. We used 8 different cases and analyzed its output using quantitative analysis of RMSE and MAE values. For our stock prediction application, we will choose below 8 parameters while building and LSTM RNN model to get better predictions results.

- Total no of Hidden layers = depending upon computational cost and time (we choose 4)
- Optimal dropout rate =0.5
- Optimal batch size = 1 or 5
- Optimal epoch= 5
- Rolling window size = between 100 and 200
- Optimizer = Adam
- Loss function = Mean Squared Error
- No of nodes in hidden layers = (100,50,25,1) for 4 layered LSTM RNN model.

We can't comment confidently on the optimal results using above parameter values but can use above analysis as a reference while building and selecting our LSTM RNN models for different applications. For a particular application, we can perform different parameter tuning with available time and computational cost and should try to choose the parameters which gives us approximately better results which are expected. As a data scientist, one must try to manage the tradeoff between accuracy and available time and computational cost resources for a particular application.

## 6. Future Scope

We can use LSTM model with encoder-decoder to perform its performance. The motivation behind using encoder-decoder is taking all the features of inputs and encode them into latent space to reduce the dimensionality which is also called as non-linear PCA and then decode it to check the original features. We can train our model using adam optimiser and mean squared loss error function to be able to predict future stock prices more accurately.

Varying the input parameters of the LSTM model. We can vary the number of assets in our portfolio (top 5/10/30/50 stocks by Market capitalization) and we can check the performances of these variations and try to improve the performance of the LSTM RNN models. The parameter tuning for different applications will be different. One must perform above 8 cases and try to find the suitable parameters for better predictions.

## REFERENCES

[1] Hao, C. Y., Wang, J. Q., & Xu, W. (2013). Prediction-based portfolio selection model using support vector machines. In Proceedings of sixth international conference on business intelligence and financial engineering (pp. 567–571).

[2] Yilin Ma, Ruizhu Han , Weizhong Wang.(2020). Portfolio optimization with return prediction using deep learning and machine learning.

[3] Jain L C, Medsker L R. Recurrent Neural Networks: Design and Applications. Joint Conference on Neural Networks. 1999:1537 – 1541

[4] Jia H. Investigation Into The Effectiveness Of Long Short Term Memory Networks For Stock Price Prediction. 2016.

[5] Tingwei Gao, Yueting Chai, Yi Liu.(2020) Applying Long Short Term Memory Neural Networks for Predicting Stock Closing Price.

[6] Fischer, T.; Krauss, C. Deep learning with long short-term memory networks for financial market predictions. Eur. J. Oper. Res. 2018, 270, 654–669.

[7] Van-Dai Ta , CHUAN-MING Liu and Direselign Addis Tadesse,2020. Portfolio Optimization-Based Stock Prediction Using Long-Short Term Memory Network in Quantitative Trading.

[8] S. Selvin, R. Vinayakumar, E.-A. Gopalakrishnan, V. K. Menon, and K.-P. Soman, "Stock Price Prediction Using LSTM, RNN and CNNSliding Window Model," in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI 2017), 2017, pp. 1643–1647.