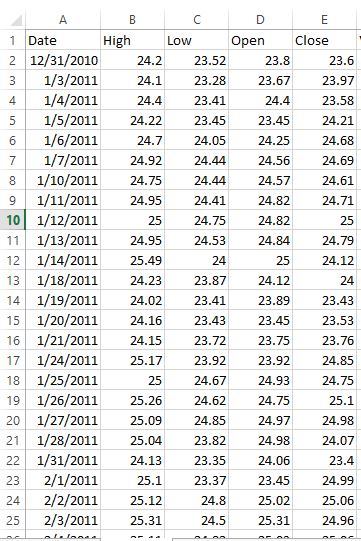
**Chapter 5**

**Experimental Results and Analysis:**

This chapter contains experimental results and analysis about our project. We analyzed everything with proper graphs and explained everything in details.

Real Time Data Collection:

Our program can collect real time data from Yahoo! Finance website. After given input of starting date, end date and company ticker name data will save into CSV file.



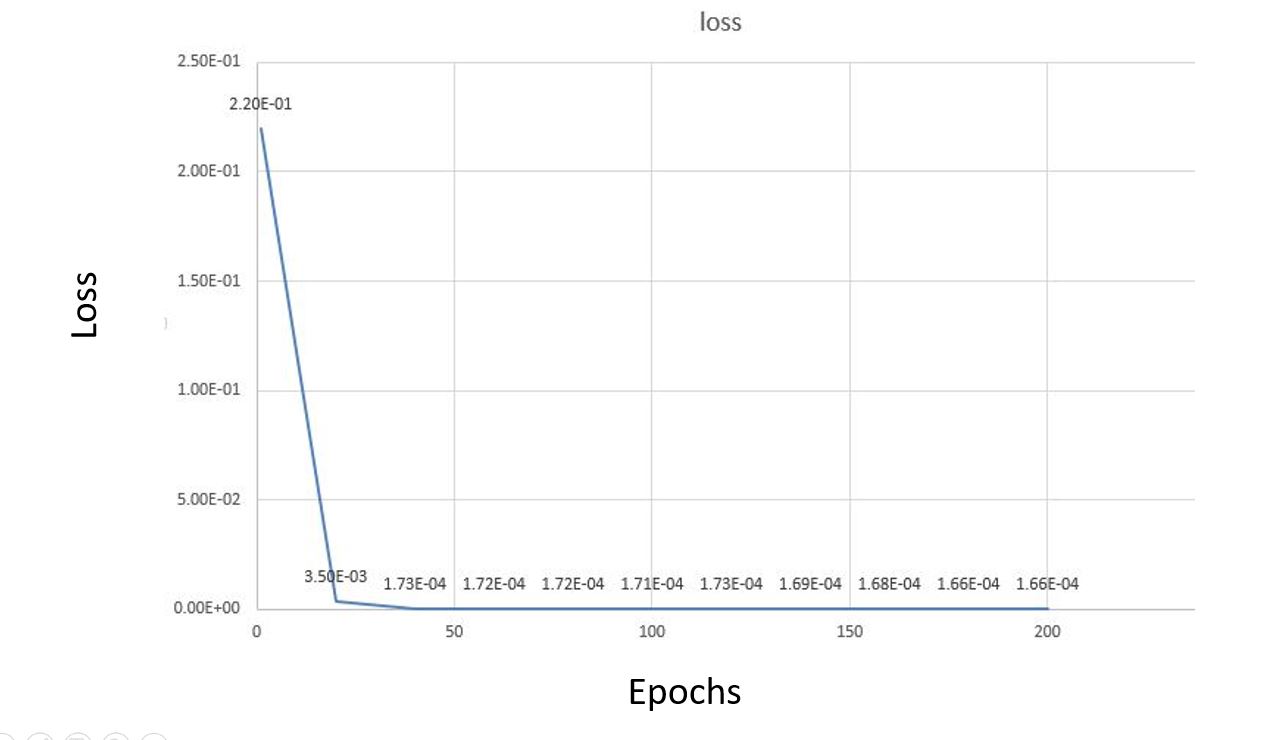
Effect of Epochs to Learn the Patterns:

An epoch is when an entire dataset is passed forward and backward through the neural network only once. Training a deep neural network involves optimizing a large set of parameters which are heavily interdependent. As for this, it can take a lot of scaled training examples before the network even settles into an area of the solution space which is close to the optimal solution or at least, the optimal solution for this training data set. This is exacerbated by the stochastic nature of batch gradient descent as the optimization algorithm is very data-hungry. To summarize, batch gradient descent requires more iterations to converge than one pass over the data set will allow.

To learn the pattern our loss function is is mean squared error (MSE) which is an estimator measures the average of the squares of the errors.

If is a vector of predictions generated from a sample of *n* data points on all variables, and is a vector of observed values of the variable being predicted, then the with in sample mean squared error (MSE) of the predictor is computed as

The less the value of the loss functions, the better our data will fit onto the model and the better our model will learn the patterns.



From the figure, we can see that at less epochs we have more loss. But after 150 epochs our loss functions get saturated.

We have a pretty low loss function values approximately 1.66E-04. From this we can conclude that our model will predict the outputs perfectly.

Effect of Mini-batch Gradient Descent:

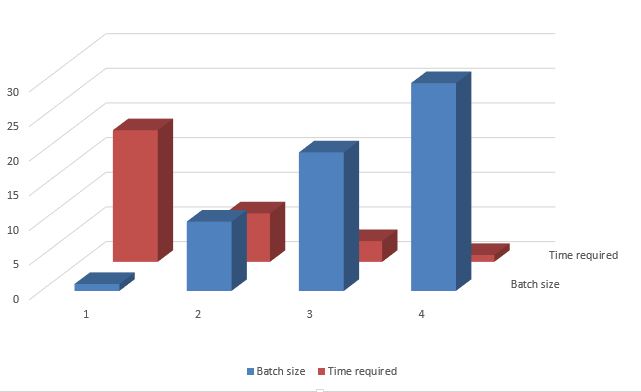
At complex structure, gradient descent can detect a gradient which might be the gradient of a smaller sub part but that is not the optimal gradient which is known as local minima problem.

Stochastic gradient descent updates the weight matrix after evaluation the cost/loss function after each sample.  That is, rather than summing up the cost/loss function values for all the sample then taking the mean, stochastic gradient descent (SGD) updates the weights after every training sample is analyzed. It can solve local minimum problem but it takes much computation time. It responds to the effects of each and every sample, and the samples themselves will contain an element of noisiness that will make the result noisy moreover it takes much computation time.

Mini-batch gradient descent is a good trade-off between stochastic gradient descent and batch gradient descent. In this techniques, the cost function (and therefore gradient) is averaged over a small number of samples, from around 10-500.  This is opposed to the stochastic gradient descent batch size of 1 sample, and the batch gradient descent size of all the training samples.  It looks like this:

Where are the weights, *α* is the learning rate and ∇ is the gradient of the cost function with respect to changes in the weights and is the mini-batch size.

From the figure we can see when our batch size is one which is basically stochastic gradient descent, is taking lots of time to compute the networks. A good trade-off between stochastic gradient descent and batch gradient descent is mini batch gradient descent which is 30.



Effect of Training Periods:

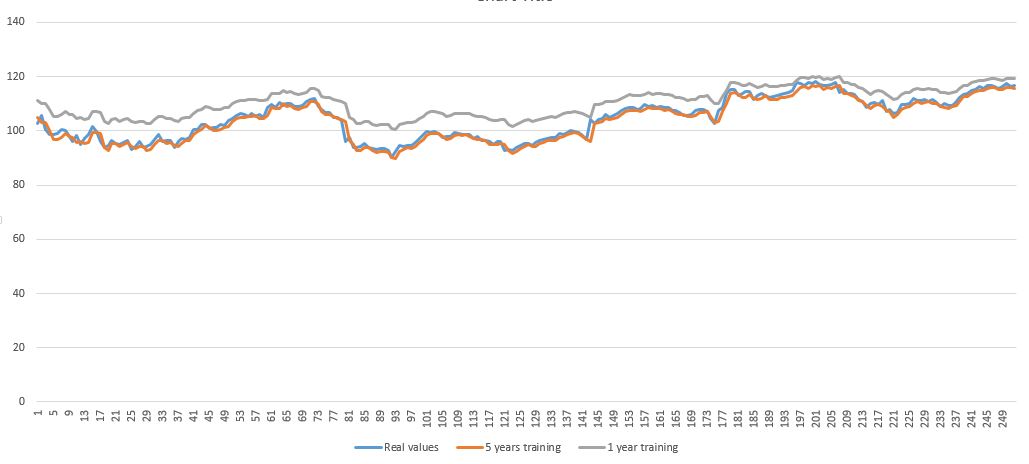
When the training data set is small, deep neural networks do not perform well. Algorithms of statistical learning cannot learn well from a few examples because of the fundamental principles in their design.

From the figure, we can see when we have 1 year of training data, our model cannot learn properly.

The predicted values curve cannot follow the real values curve.

From the figure, we can see when we have 5 year of training data, our model can learn properly.

The predicted values curve follow the real values curve.



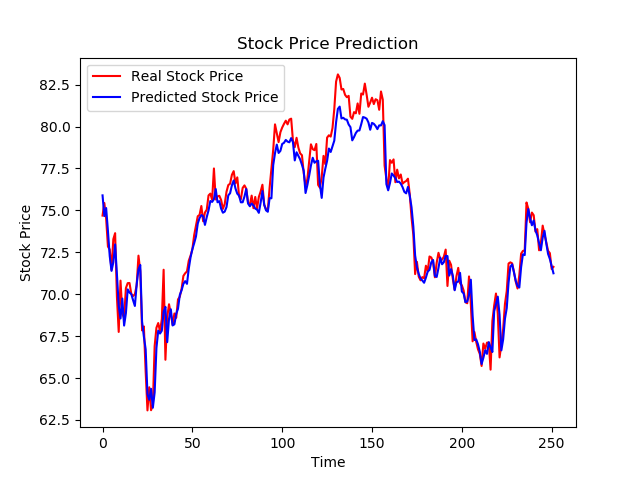
Observing efficiency for several companies:

Our model is a regression model and regression model is generally evaluated by calculating root mean square error (RMSE). The root-mean-square error (RMSE) (or sometimes root-mean-squared error) is a frequently used measure of the differences between values predicted by a neural networks and the values observed.

If is a vector of predictions ganerated from a sample of *n* data points on all variables, and is a vector of real values of the variable being predicted, then the with in sample root mean squared error (RMSE) of the predictor is computed as

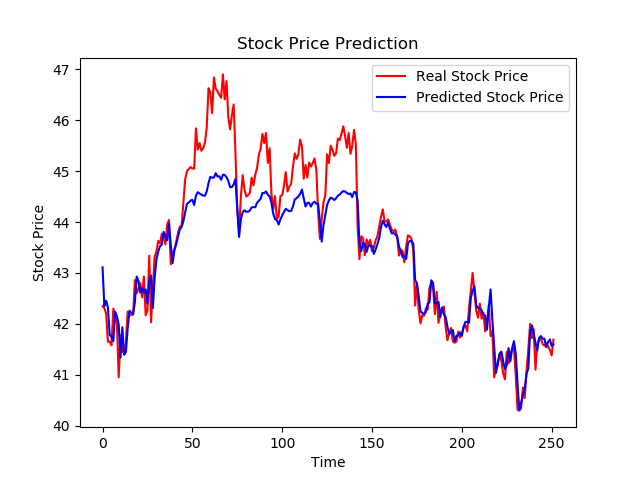
The comparison of real stock prices and predicted stock prices for Lowe's Companies, Inc. (LOW)

is given where the percentage of root mean square error is 0.0127464.



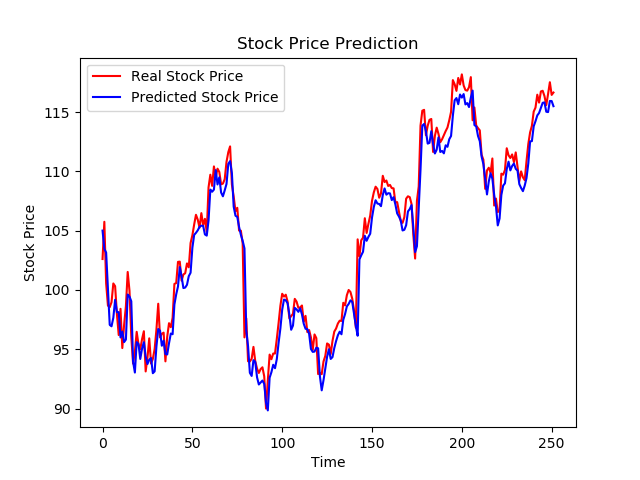
The comparison of real stock prices and predicted stock prices for The Coca-Cola Company (KO)

is given where the percentage of root mean square error is 0.0144508.



The comparison of real stock prices and predicted stock prices for Apple Inc. (AAPL)

is given where the percentage of root mean square error is 0.013996.



Conclusion:

In this chapter we will discuss about experimental results and analysis. In the next chapter we will discuss about conclusion and recommendations of this project.