**CSC685 Advanced Machine Learning**

**Spring 2020**

**Project 7**

**Team 2**

**Jiajie Zhu & Getong Liu & Abby**

**Date: 05/04/2020**

**Part 1. Requirements**

**Part I: RNN/LSTM/GRU**

In this exercise, you are to work in your previous group on stock market forecast. Collect performance data for a period of 12 months for a favorite stock (Apple, Google, etc.) and perform the following tasks:

1) Use the most accurate model (LSTM or GRU) that best predicts the collected data pertaining to the past 12 months. Pay special attention to the selected #units, activation functions, dropouts, dense layer configuration, etc.

2) Use your model from part 1 to make predictions for the next month, next 5 months, and the next 10 months.

3) Make sure to plot your results in both parts 1 and 2.

4) Fully discuss and justify your results.

**Part II: CNN (Extra Credit 30 Points)**

Coil100 contains 7200 color images of 100 objects (72 images per object). The objects have a wide variety of complex geometric and reflectance characteristics. The objects were placed on a motorized turntable against a black background. The turntable was rotated through 360 degrees to vary object pose with respect to a fixed color camera. Images of the objects were taken at pose intervals of 5 degrees.This corresponds to 72 poses per object

1) Train and test the best model you can design to predict picture-specific normative ratings for each of the objects.

2) Show how you can build a pipeline for preprocessing operations and adding data augmentations, if applicable.

2) Graph your results from parts 1-2 and discuss what can be done to further improve the classification accuracy of your system.

**Part 2. Details**

**Part I: RNN/LSTM/GRU**

**Introduction**

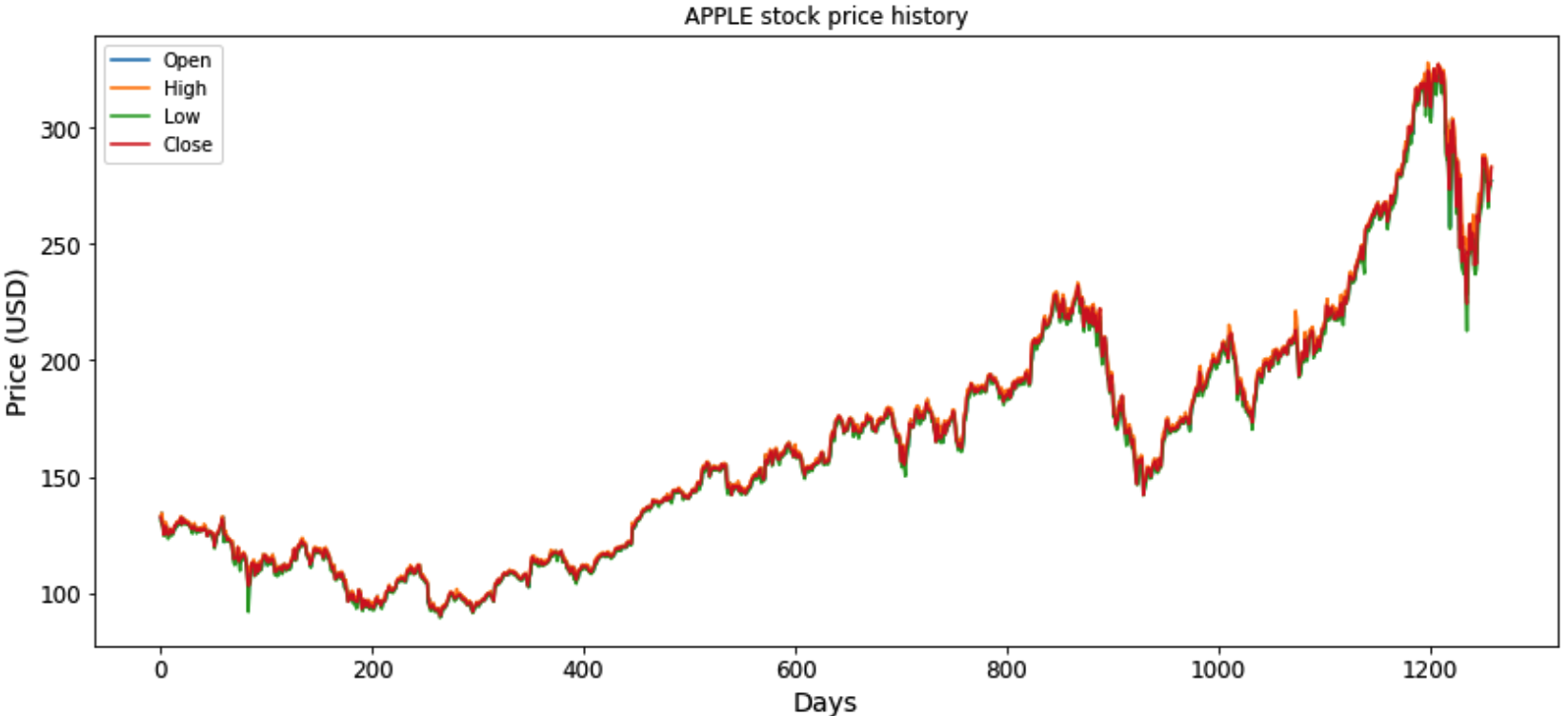
Technical indicators are available at large numbers and used mostly by stock traders. Most indicators have user-defined variables which allow traders to adapt key inputs such as look-back period which tell us how much historical data will be used to form the calculations to suit their needs. We will use some of the indicators to create features in the existing data set. Applying these features, we will see if we can predict the future price of a particular stock.

We will use Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) of recurrent units to compare their performance. LSTM is well established on sequence-based tasks with long-term dependencies, and GRU is a new addition in the field of machine learning which is an improvised version of Recurrent Neural Network(RNN).

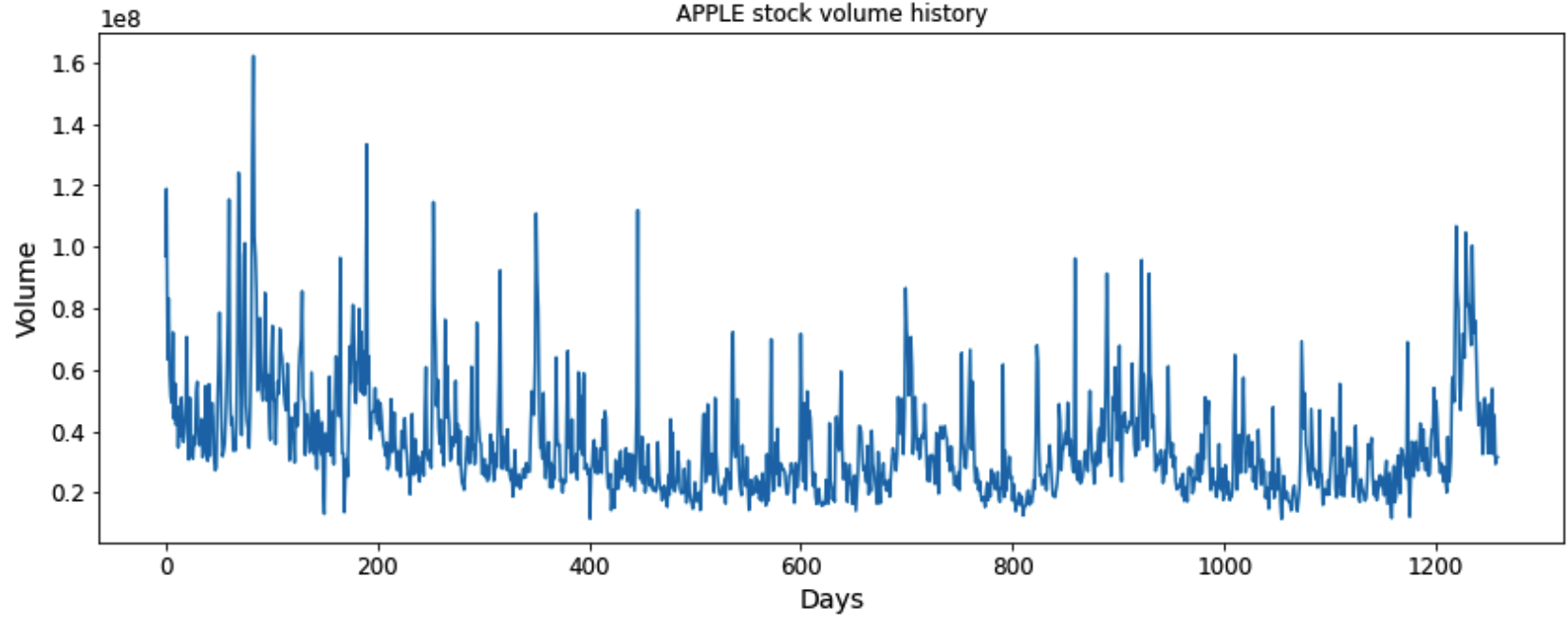
We use LSTM and GRU to predict the collected data pertaining to the past 12 months first. Then we use this model to make predictions for the next month, next 5 months, and the next 10 months. Finally we compared the results at the end of this project.

**(1)Prepare the Data**

There are 1259 instances in the dataset, while all of them have non-null values.All attributes are numerical —— stock prices each day (high, low, open, close) and stock volume —— except the Date attribute, which is Object type. Now let's see how does it look on a plot:



**stock prices**



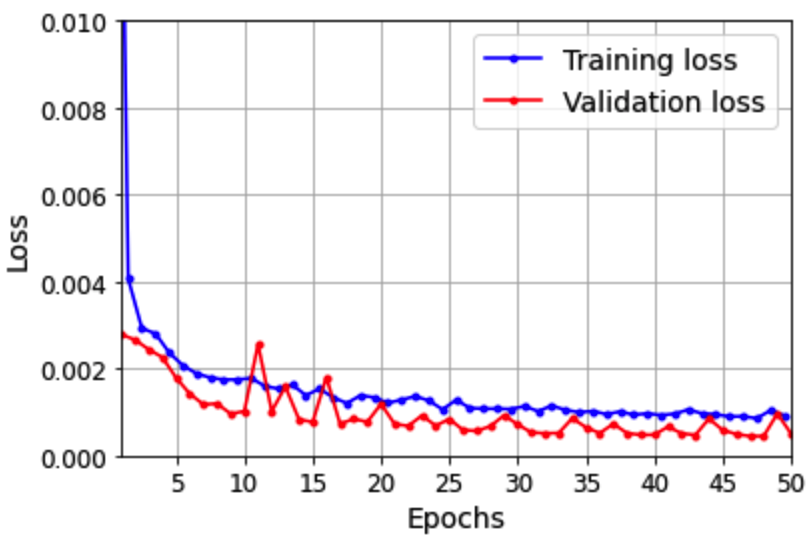
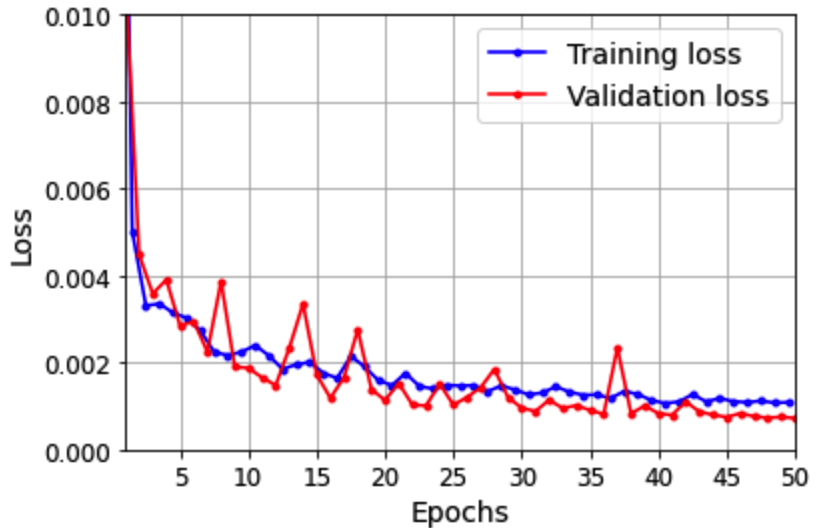
**stock volume**

**(2)Building the LSTM and GRU layers and some Dropout regularization**

We use the learning curve to value these two models.

If validation loss > training loss you can call it some overfitting.

If validation loss < training loss you can call it some underfitting.

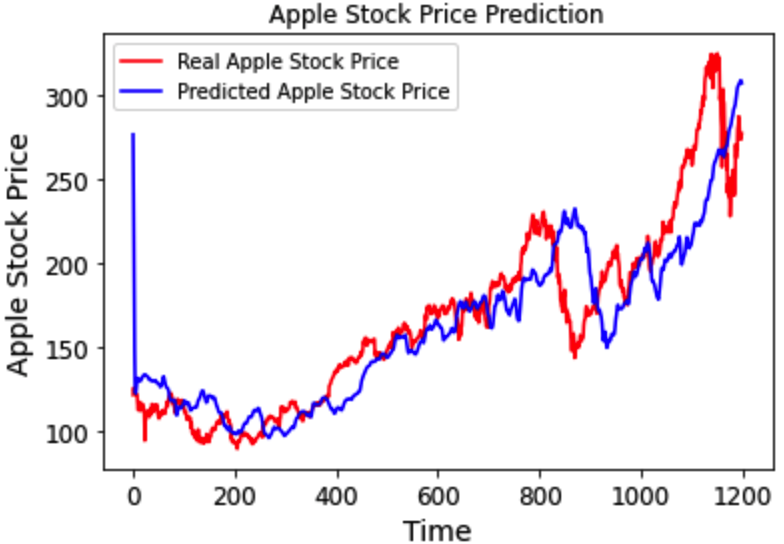
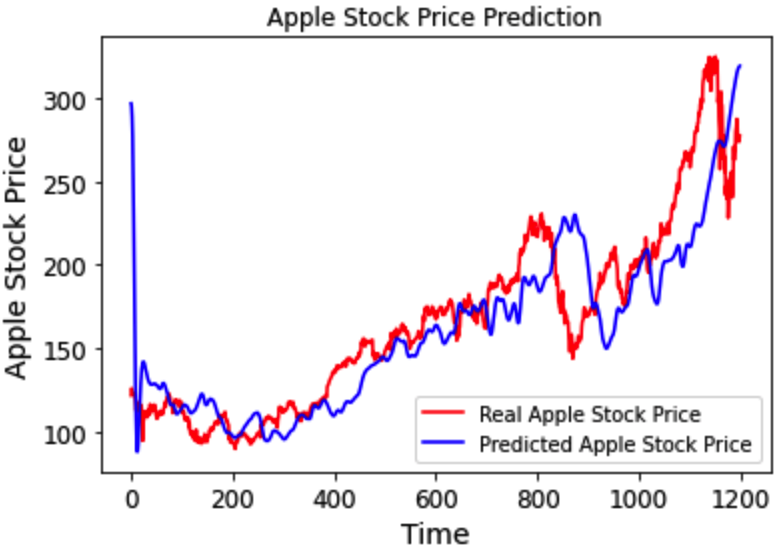


**LMTS**  **GRU**

Here we can see validation Loss is a little bit lower than training loss, thus LMTS model is a little bit underfitting. And the GRU model is somewhat underfitting as well, but a bit better than the LSTM model.

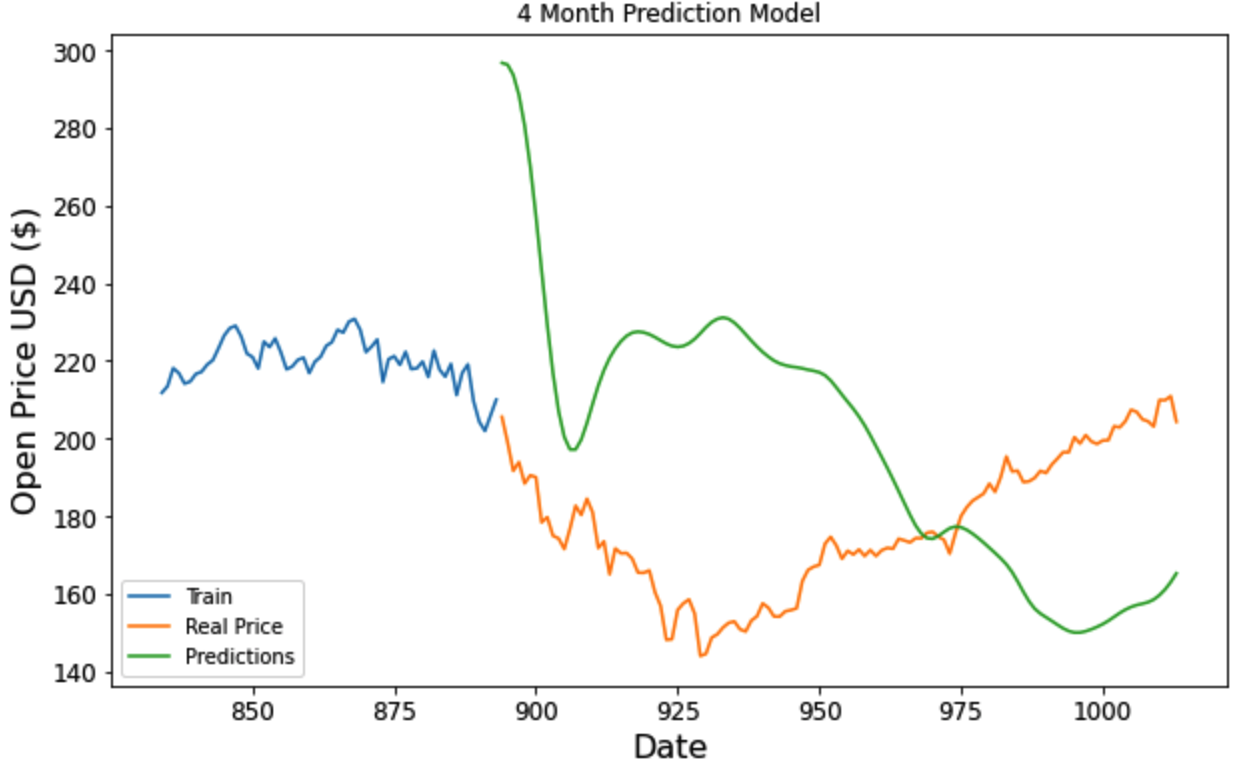
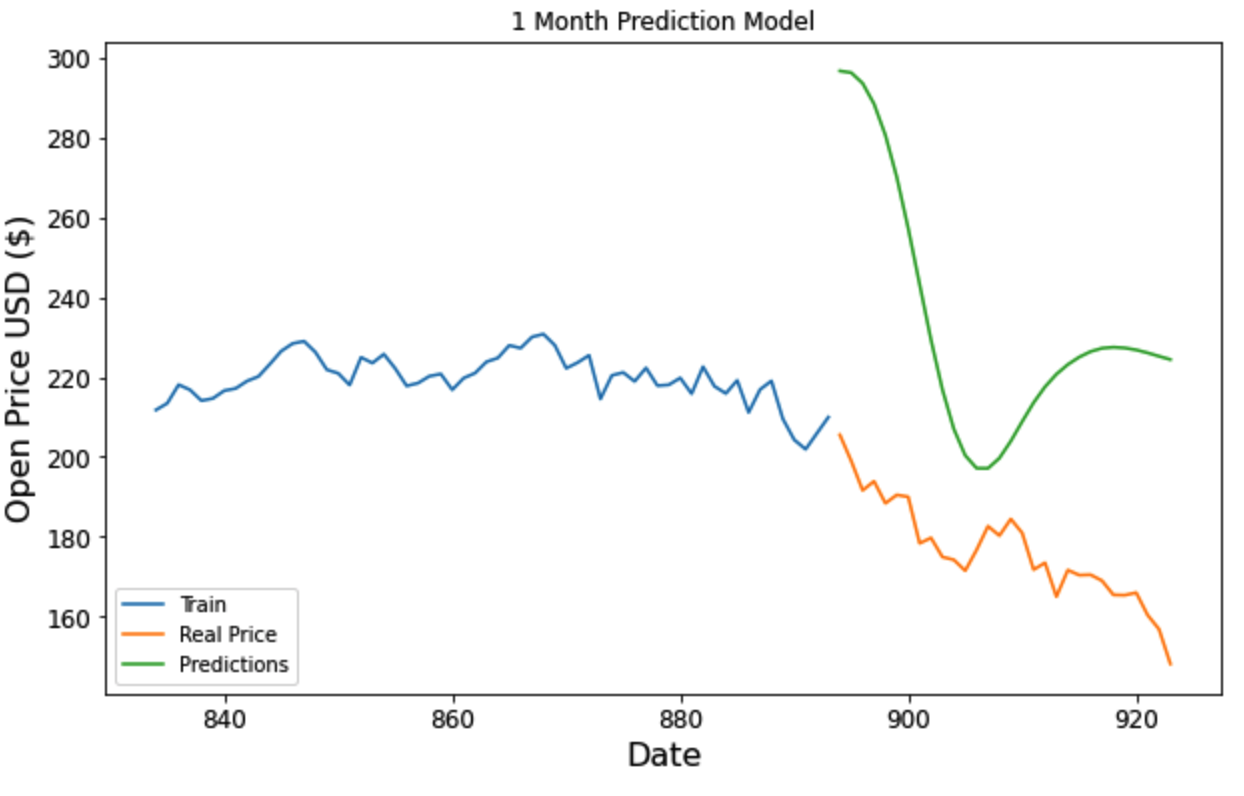
**(3)Using LSTM and GRU model to predict**

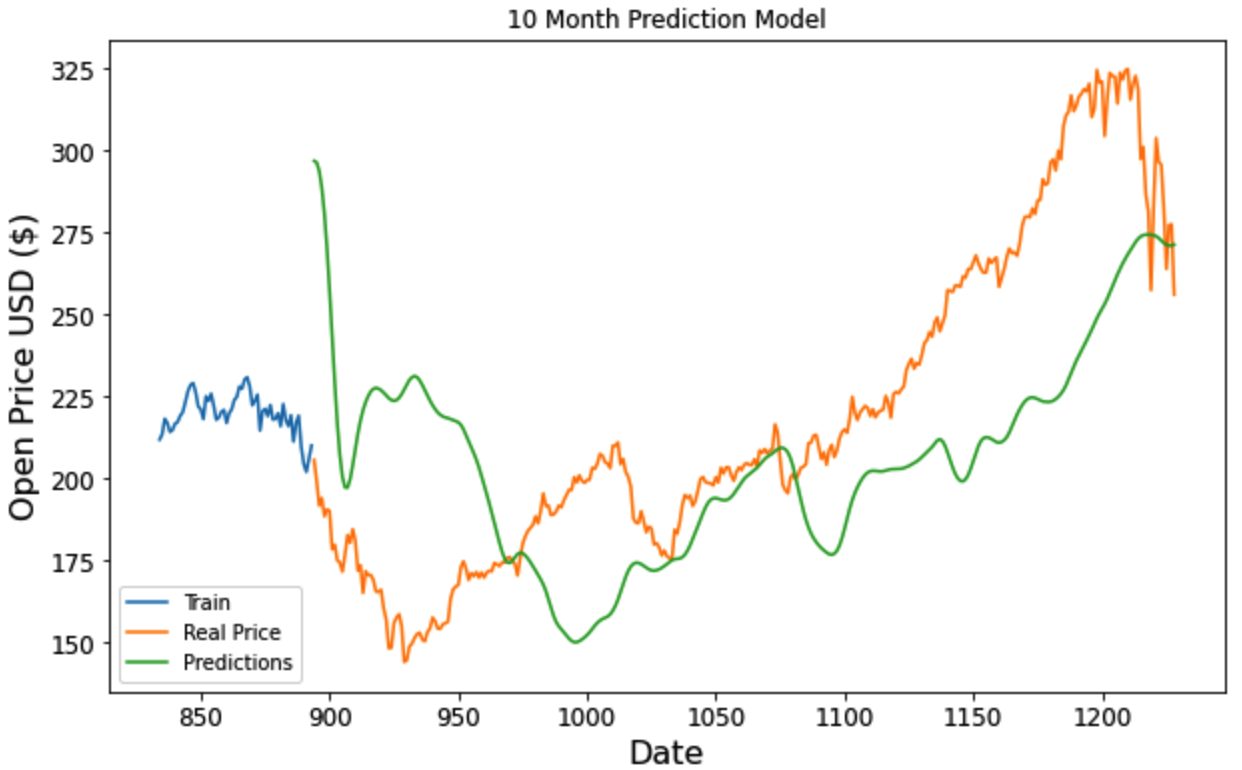
We use the Open stock price to make predictions. As we can see in this prediction plot, the trends of the predicted and real prices are pretty much the same. The lines have the same peaks and troughs. This is probably because of LSTM's ability to remember sequenced data. A traditional feedforward neural network would not have been able to forecast this result. This is the true power of LSTM and RNNs. As we also can see from this GRU plot, the trends of the predicted and real prices are pretty much the same as well, not too much different than the LSTM model: the lines still have the same peaks and troughs.



**LSTM**   **GRU**

**(4)Forecasting One Month/Four Month/Ten Month**

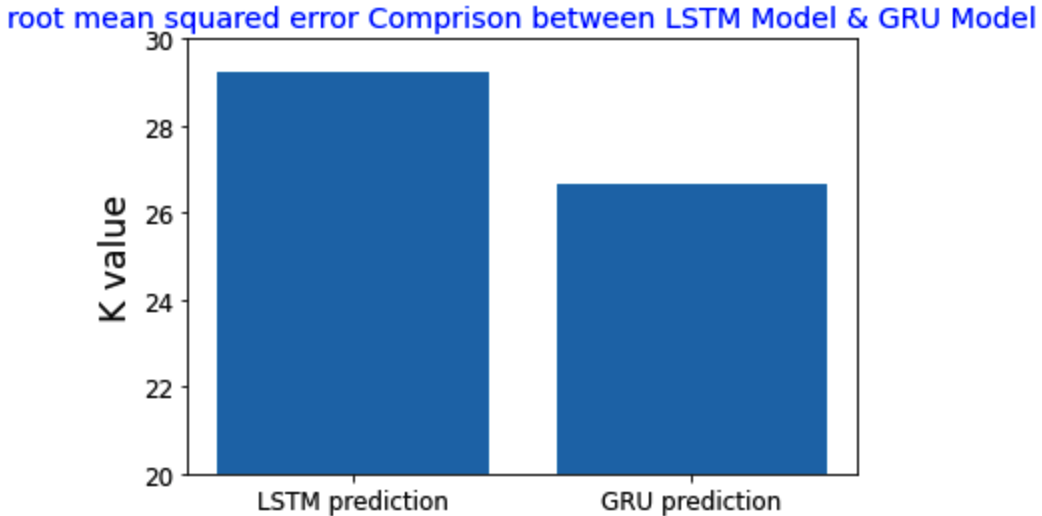
****

****

# **(5)Graph Results of Our Experiments and Conclusion**

Both GRU and LSTM funcion fairly well when using the previous 60 days to predict the next day's open price. As we can see from previous plots, the trends of the predicted and real prices in both models are pretty much the same. Meaning both LSTM and GRU have the ability to remember sequenced data. A traditional feedforward neural network would not have been able to forecast such good results. This is the true power of LSTM, GRU and RNNs.

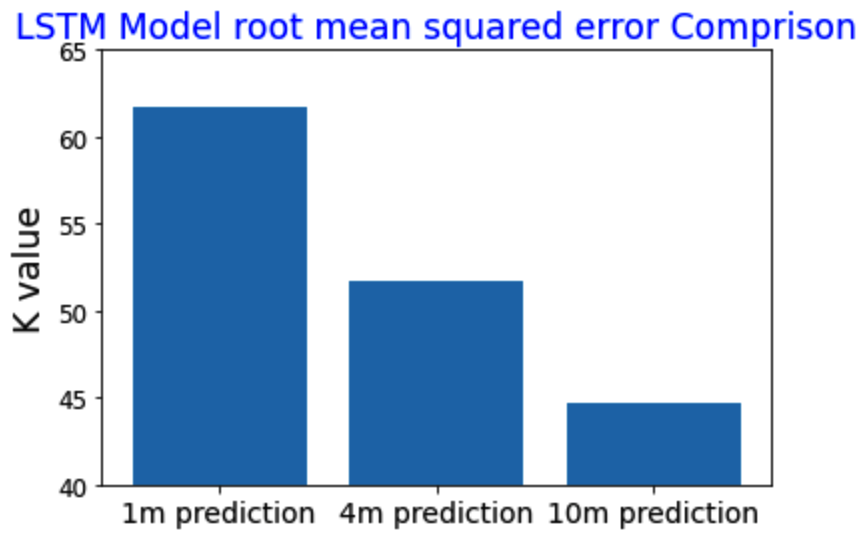
The key difference between a GRU and an LSTM is that a GRU has two gates (reset and update gates) whereas an LSTM has three gates (namely input, output and forget gates). GRU network is simpler and thus easier to modify, and takes less time to train. However, if the sequence is large or accuracy is very critical, LSTM is better.

****

As we've learned during lectures the prediction for 1 month should have better accuracy than the predictions for 4 months and 10 months later, since the errors might accumulate.

But our practice data shows that the 1 month prediction has much higher deviation from the really price since we only use the previous 60 days for training. Its performance gets better when the model has more instances involved.

For future practice, it is reasonable to feed more previous data( e.g. 365 days, 5 years, or even more) into the model, therefore an error increasing curve is estimated to happen in the long period prediction.

****

**Part II: CNN (Extra Credit 30 Points)**

**Introduction**

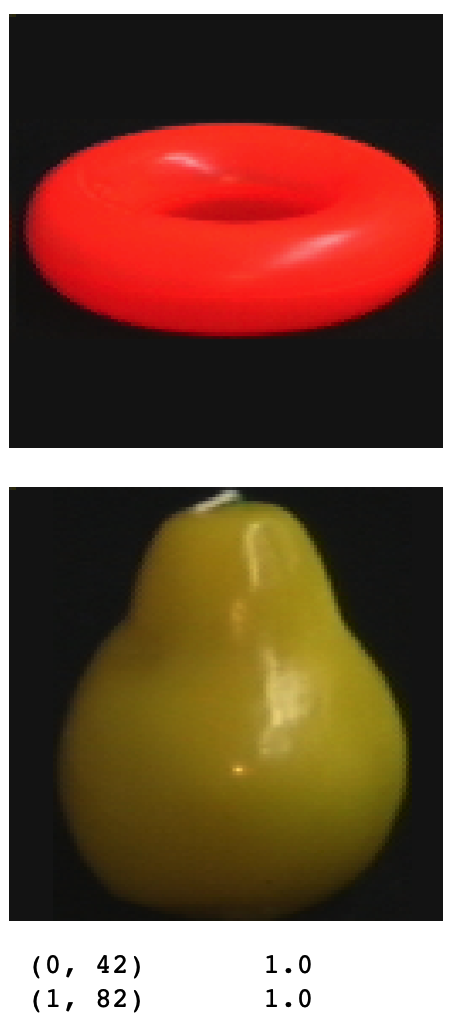
The dataset contains 7200 color images of 100 objects (72 images per object). The objects have a wide variety of complex geometric and reflectance characteristics. The objects were placed on a motorized turntable against a black background. The turntable was rotated through 360 degrees to vary object pose with respect to a fixed color camera. Images of the objects were taken at pose intervals of 5 degrees.This corresponds to 72 poses per object.

CNNs are [regularized](https://en.wikipedia.org/wiki/Regularization_(mathematics)) versions of [multilayer perceptrons](https://en.wikipedia.org/wiki/Multilayer_perceptron). Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to [overfitting](https://en.wikipedia.org/wiki/Overfitting) data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

In this part we are going to apply this kind of neural networks to classify the dataset. Also we are going to apply some generalization and optimization techniques to improve the performance of the model.

**(1)Prepare the Data**

At the section, We load the data first, and then because the prefix of the image name corresponds to the object name/ class so we need to extract this tag to construct a labeled data set to train our model. Here be created a label list that using the prefix of the image name. For example: Filename: obj47\_\_70 belongs to object '47', there that is what we need substrate and target as the label. Finally, we scale the Date from 0-255 to 0-1. We can see some of the images examples:



### **(2)Define the model**

Then we will define our model using conv layers that allow us to input tensor of shape (n,n,d) dimensional with instead of (n,n) dimensional arrays. This will help us to input each colour layer of the image and deal with the problem of the colour variation.

**Convolutional Layer:** This layer tries to find patterns in the images applying filters and then will generate feature maps, this first layer is composed by 30 filters with a size of 5x5 and applies a stride of 1 also uses the relu function as the activation function.

**Max pooling layer:** This layer will select the most important features from the conv layer generated feature maps that use a pool size of 2 and stride equal to 1.

**Convolutional Layer:** This layer tries to find patterns in the images applying filters and then will generate feature maps, this first layer is composed by 15 filters with a size of 3x3 and applies a stride of 1 also uses the relu function as the activation function.

**Max pooling layer:** This layer will select the most important features from the conv layer generated feature maps that use a pool size of 2 and stride equal to 1.

**Dropout layer:** This layer is used to improve the generalization of the model, in this case, this drops 20% of the neurons from the previous layer to force the weights to be distributed equivalently.

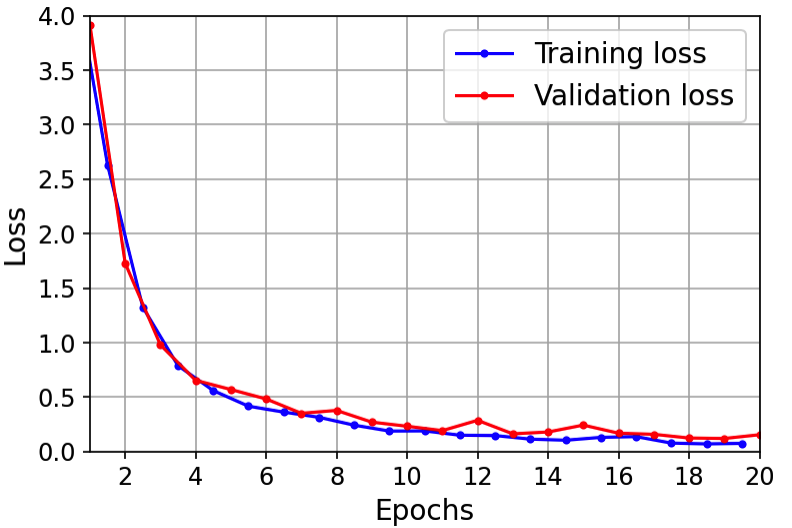
**Flatten layer:** this layer flattens the input tensor to create a single long feature vector to be used by the dense layer for the final classification.

**Dense layer:** this layer also tries to find some relationship between the features coming from the flattening layer.

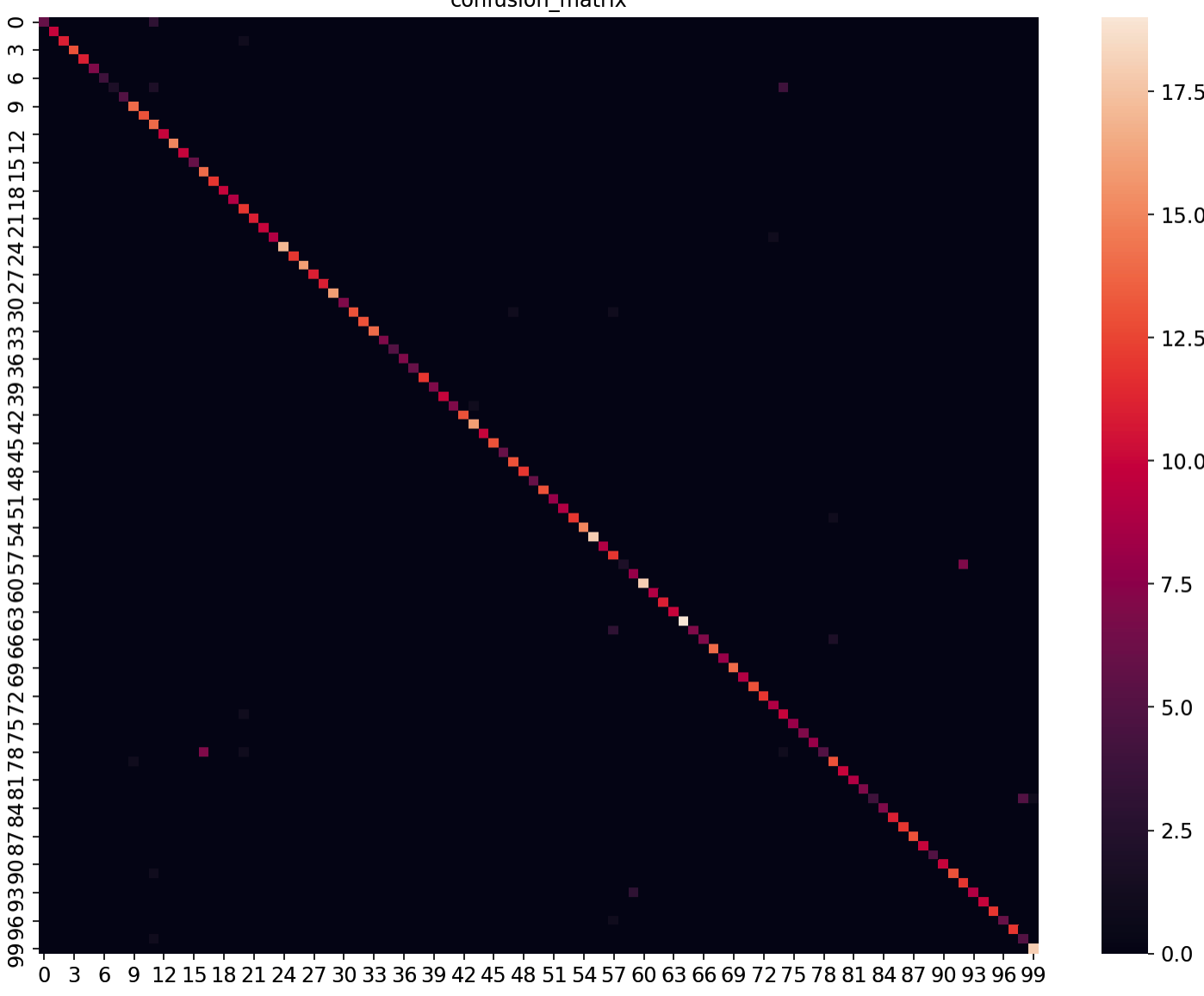
**Output Layer(dense):** The final layer has 100 neurons that correspond to each class, in this case, we used the softmax function to map a probability for each class.

**(3)Test prediction accuracy**

We calculate the test loss and accuracy, and we have achieved 95.37% accuracy which is pretty good so far.



Finally we wil plot the confusion matrix for prediction:



# **(4)Conclusion**

With the limitation of time and dataset size, data argumentation was not implemented. With this technique added and more epochs added, it is prominent to achieve higher accuracy in the future tests.