NUS CS-3235: Computer Security April 20, 2020

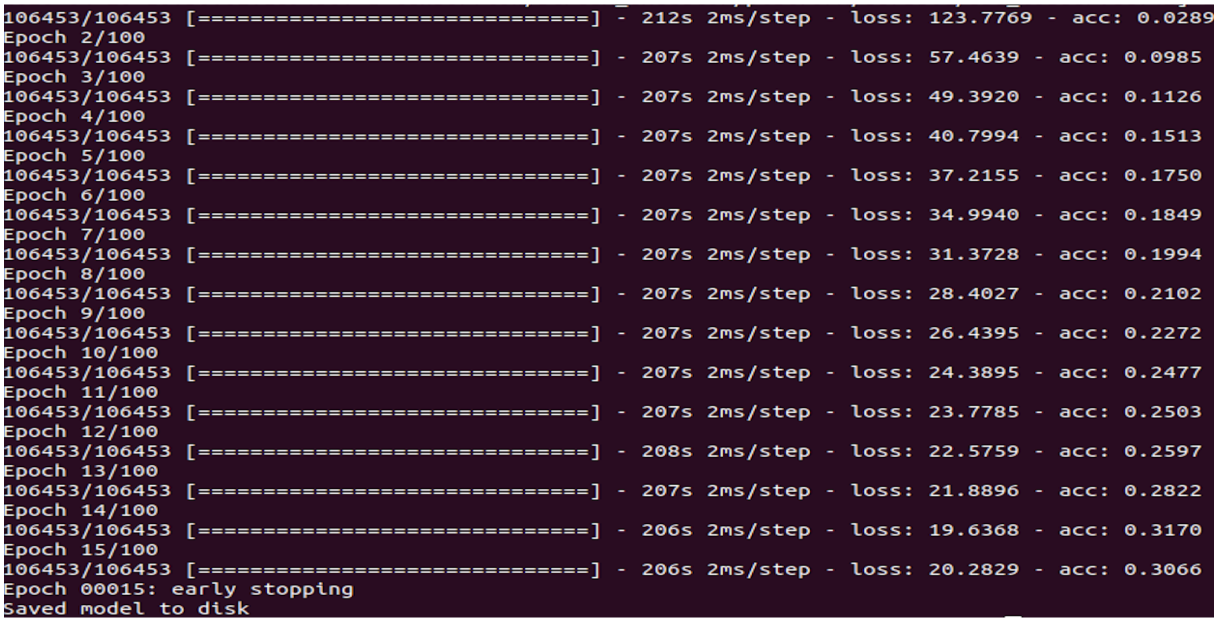
Assignment 3 Report

Lecturer: Reza Shokri *Student: Rachit Bansal A0214350W*

I experimented alot of things in this assignment. Firstly, I designed a model by extracting features like data transfer speed, combination of the direction of flow and data size by multiplying the data size by 1 if it is inwards else multiplied it by -1. Then using these extracted features I compared test data(which I obtained by dividing the given data into training and test data) by calculating the mean squared error between the features of the test data and the finally extracted features for each URL. This method gave quite less accuracy(less than 10 percent). It was probably because it would be difficult to extract the information about the specific time series using vanilla ML.

So, finally I decided to design a LSTM model using Keras library of python which turned out to be a mammoth project but quite interesting and I finally got quite appreciable results. In this model, I used 2 layers, one LSTM layer and the other dense layer which gives the final output. The LSTM layer takes in as input a training example of size (blocksize, numberoffeatures) and gives as output a vector of a specific size. The number of memory units in LSTM is set as 512 after trying a range of values. The idea of blocksize which I thought about is that given a trace for a specific URL in the training data I will be dividing that trace into overlapping blocks of size blocksize with the length of step for sliding window as 1. This will help in increasing the training data and also help in keeping the input size the same for the model while taking care of the sequencing. I initially chose the blocksize to be 32 but when I increased the block size I found out that the result became better but it took more time to train and finally I found out blocksize of 256 is the sweet spot. I initially chose the number of features to be 3 which are the time difference between two consecutive timestamps in seconds, the data size, +1 for in and -1 for out. But when I combined the 2nd and 3rd feature I got better results. I also tried combining 1st and 3rd, 1st and 2nd to get speed, all 3 but the results are not better. So finally I have 2 features which are the time difference and the data size with a sign representing direction of flow.

The number of epochs which I set is till when the loss decreases which is known as early stopping technique. I also tried normalization as well as batch normalization but that seems to don't work well with time series data. I used adam optimizer and mean squared error loss function. A quite interesting find which increased the accuracy is that when a trace file for a specific URL is small than the blocksize then I repeated the elements in the same order till the size of the block becomes the chosen blocksize. In some of the cases it took more than hours to train but overall it was a great experience. It will take about 50 seconds to get the result after running "./test observation1 observation2" using Quadro P600 GPU. I used a machine with Nvidia GPU and the prediction process requires tensorflow-gpu version 1.14. I believe that the training pipeline will give more accuracy given more data. The present training accuracy on blocks is around 0.31 for which I think after much research is quite good for this much data and these many URL classes:-



```
106453/106453 [==============================] - 212s 2ms/step - loss: 123.7769 - acc: 0.0289
Epoch 2/100
106453/106453 [==============================] - 207s 2ms/step - loss: 57.4639 - acc: 0.0985
Epoch 3/100
106453/106453 [==============================] - 207s 2ms/step - loss: 49.3920 - acc: 0.1126
Epoch 4/100
106453/106453 [==============================] - 207s 2ms/step - loss: 40.7994 - acc: 0.1513
Epoch 5/100
106453/106453 [==============================] - 207s 2ms/step - loss: 37.2155 - acc: 0.1750
Epoch 6/100
106453/106453 [==============================] - 207s 2ms/step - loss: 34.9940 - acc: 0.1849
Epoch 7/100
106453/106453 [==============================] - 207s 2ms/step - loss: 31.3728 - acc: 0.1994
Epoch 8/100
106453/106453 [==============================] - 207s 2ms/step - loss: 28.4027 - acc: 0.2102
Epoch 9/100
106453/106453 [==============================] - 207s 2ms/step - loss: 26.4395 - acc: 0.2272
Epoch 10/100
106453/106453 [==============================] - 207s 2ms/step - loss: 24.3895 - acc: 0.2477
Epoch 11/100
106453/106453 [==============================] - 207s 2ms/step - loss: 23.7785 - acc: 0.2503
Epoch 12/100
106453/106453 [==============================] - 208s 2ms/step - loss: 22.5759 - acc: 0.2597
Epoch 13/100
106453/106453 [==============================] - 207s 2ms/step - loss: 21.8896 - acc: 0.2822
Epoch 14/100
106453/106453 [==============================] - 206s 2ms/step - loss: 19.6368 - acc: 0.3170
Epoch 15/100
106453/106453 [==============================] - 206s 2ms/step - loss: 20.2829 - acc: 0.3066
Epoch 00015: early stopping
Saved model to disk
```