DEEP LEARNING PROJECT WEEK-1

1.Topic

In this project I have built a Neural Network to detect Tremor/Shaking of Upper or Lower Body of people. I have used a deep neural network with Long-Short-Term memory cells.

2. Dataset

In this project I have created my own data set of positive and negative samples. I have taken 30 videos of positive samples and 20 videos of negative samples, each being 1 minute long. I have taken them in 30 fps and hence for every video I got 1800 frames.

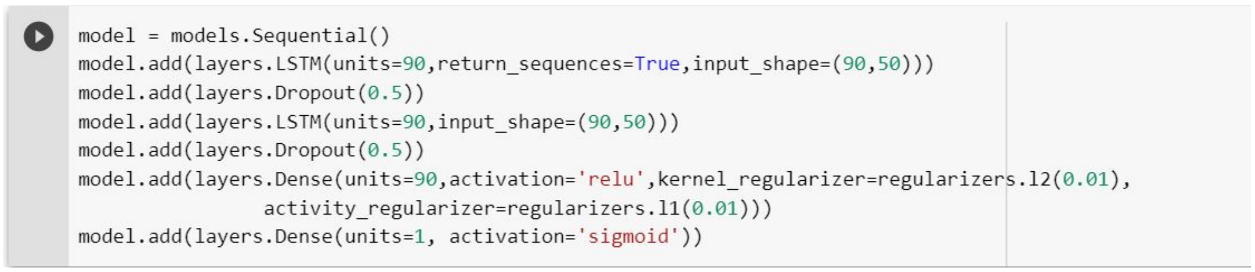
I have converted this video into a Json file containing the coordinates of the human body using Open-pose. I have taken the pose_keypoints for this model. Open-pose gives 50 key points of human pose and hence my neural network had 50 features.

I have reshaped the data into 90 time steps and hence my network gives output after every 3 seconds. As 90 frames are formed after 3 seconds when captured at a rate of 30fps. I have generated 49,770 json files from open pose and converted into 530 samples. I have used 370 samples to train and 183 samples to validate, with a train_test_split ratio of 0.33.

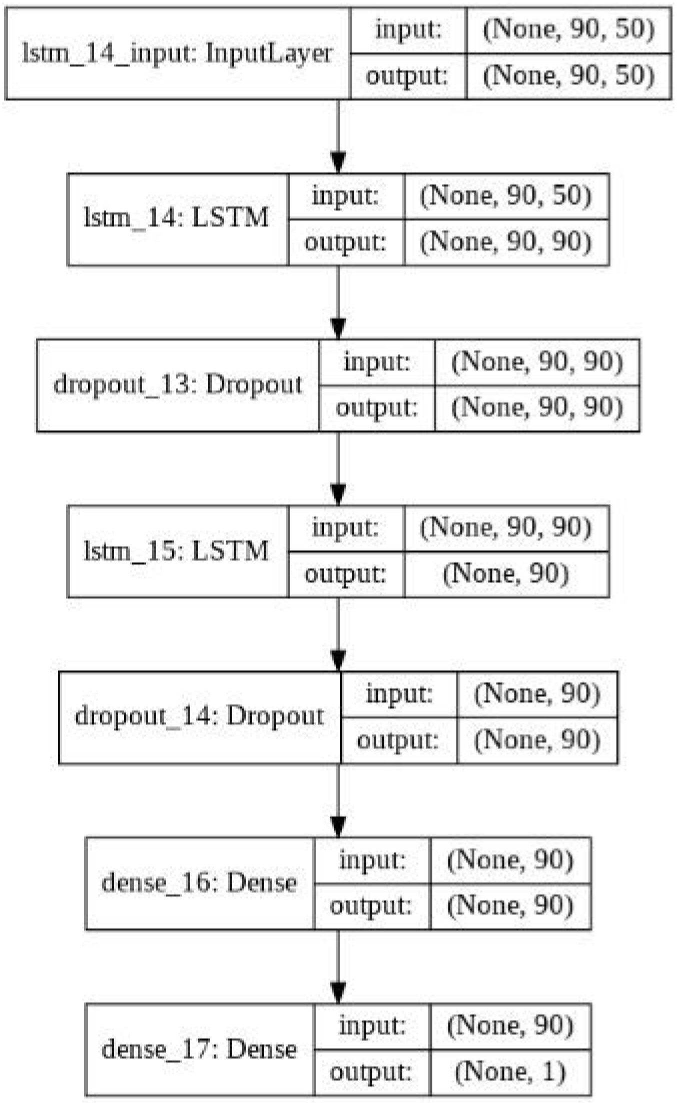
3. DNN Model

3.1 Architecture

The model is built with a linear stack of layers with the Sequential model. I have used two LSTM layers with a dropout layer on the top and then a dense layer with regularization and the output layer. The code snippet and the model architecture are shown below



```
model = models.Sequential()
model.add(layers.LSTM(units=90,return_sequences=True,input_shape=(90,50)))
model.add(layers.Dropout(0.5))
model.add(layers.LSTM(units=90,input_shape=(90,50)))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(units=90,activation='relu',kernel_regularizer=regularizers.l2(0.01),
                activation_regularizer=regularizers.l1(0.01)))
model.add(layers.Dense(units=1, activation='sigmoid'))
```

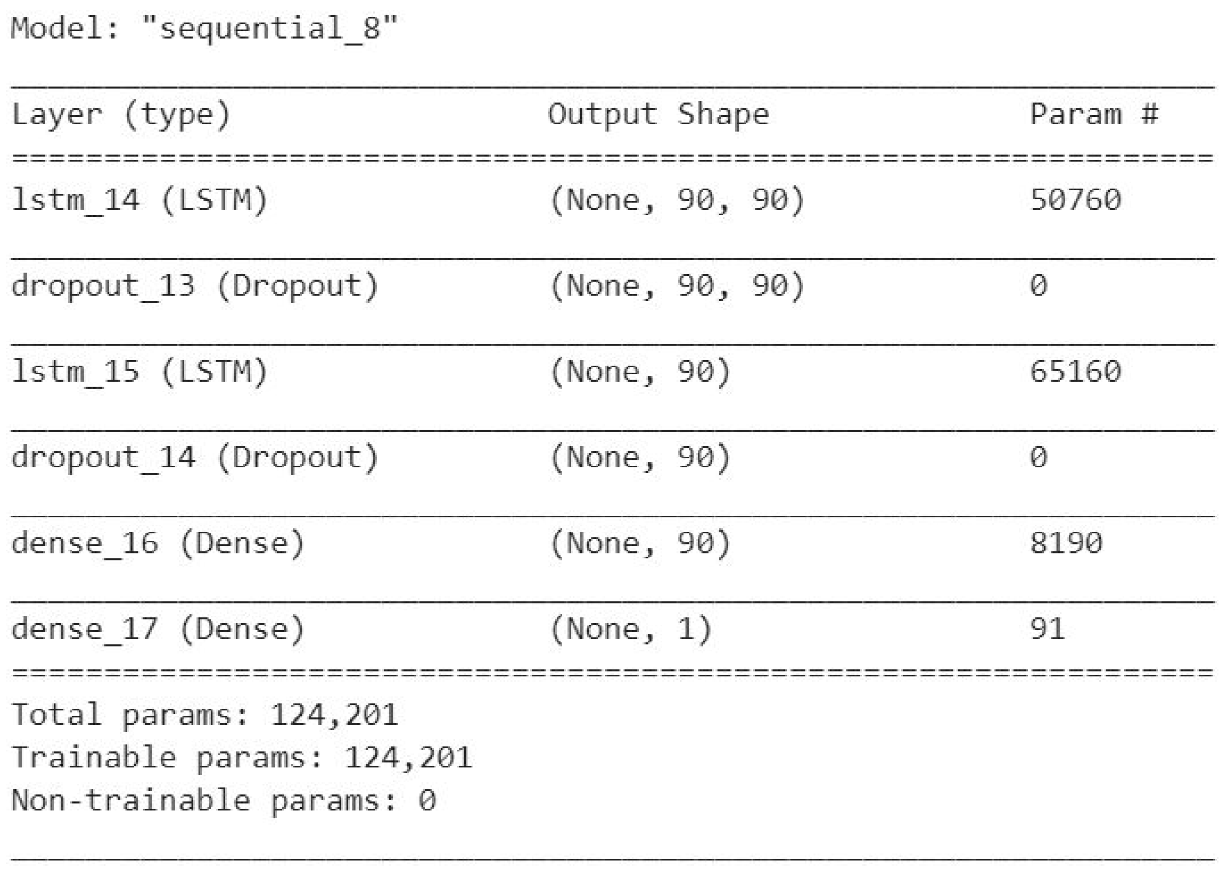


The above graph is the architecture of the model used in the project with the input and output vectors mentioned.

3.2 Input: Shape of tensor
Before reshaping:
X_train shape is (33,300,50):(Json files,features)
X_test shape is (16,470,50):(Json files,features)
After reshaping:
X_train shape is (370,90,50):(samples,Timesteps,features)
X_test shape is (183,90,50):(samples,Timesteps,features)

3.3 Output: Shape of tensor
Y_train is (370,1)
Y_test is (183,1)

3.4 Shape of output tensor in each layer



```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_14 (LSTM)               (None, 90, 90)            50760
_________________________________________________________________
dropout_13 (Dropout)         (None, 90, 90)            0
_________________________________________________________________
lstm_15 (LSTM)               (None, 90)                65160
_________________________________________________________________
dropout_14 (Dropout)         (None, 90)                0
_________________________________________________________________
dense_16 (Dense)             (None, 90)                8190
_________________________________________________________________
dense_17 (Dense)             (None, 1)                 91
=================================================================
Total params: 124,201
Trainable params: 124,201
Non-trainable params: 0
_________________________________________________________________
```

4.Hyperparameters

4.1 List of Hyperparameters used

In this model the hyperparameters I have used are: epochs, batch size, learning rate, number of LSTM layers and depth of the network, L1 and L2 regularizers, dropout rate.

4.2 Range of Hyperparameters tried

Batch size	1,3,5,10,15,20,32
Epochs	10,50,100,200,500,700
Learning rate	0.1,0.01,0.001,0.0001,0.00001
Dropout	0.1,0.2,0.4,0.5,0.7
Depth of LSTM layers	1,2,3,4,5,6
L1 and L2 regularizers	0.1,0.01,0.001

4.3 Optimal Hyperparameters found

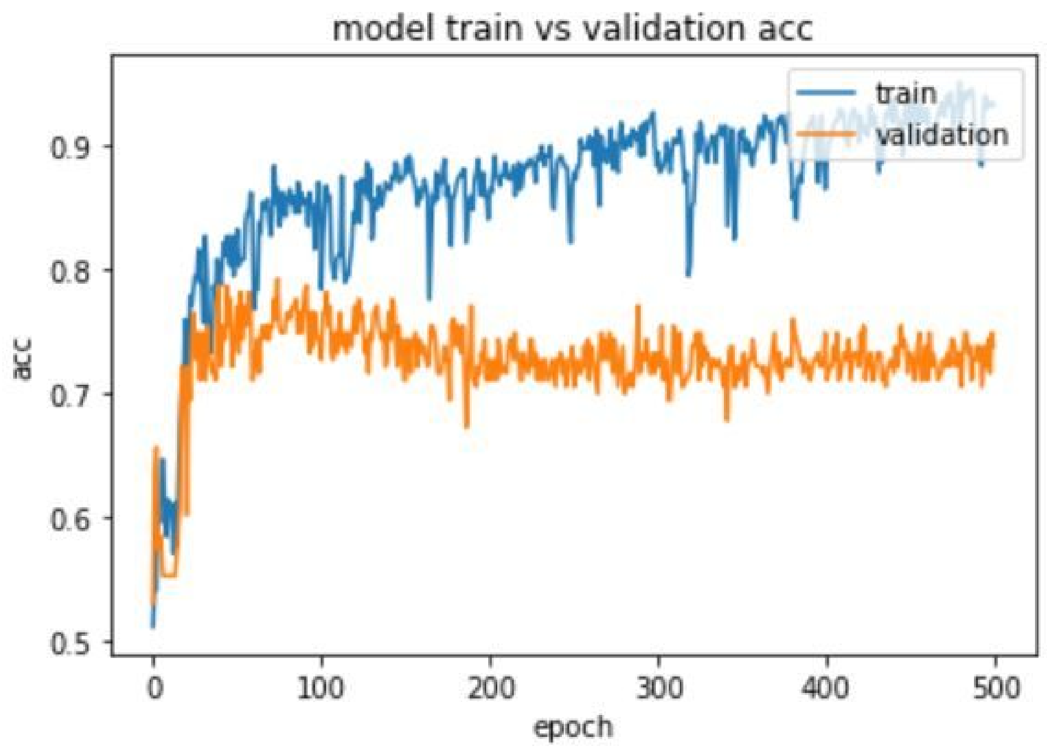
Batch size	5
Epochs	500
Learning rate	0.0001
Dropout	0.5
Depth of LSTM layers	2
L1 and L2 regularizers	0.01

5. Training and Testing performances

5.1 Accuracies

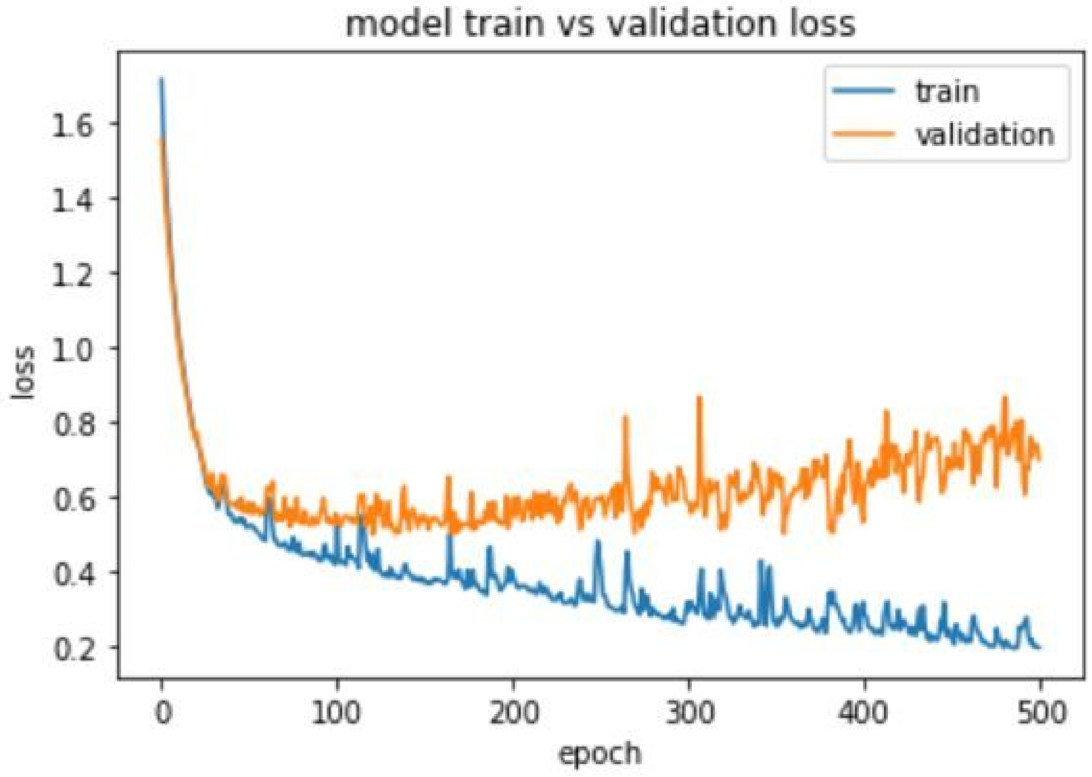
The below graph shows how the training and validation accuracies change for each epoch

Training acc: 91.89% validation acc: 73.77%



5.2 Losses

The below graph shows how the training and validation Losses change for each epoch
Training loss: 0.2849 validation loss: 0.7312



5.3 Testing performances
The testing performances have been submitted as graphs and json files have been created in the testing.ipynb file. There are a few cases where the outputs were to be less than 0.5 but they turned out to be more than 0.5. In the coming weeks I will be working on improving these errors.

5.4 Json files
In the code as my model outputs a value after every 90 frames i.e. 3 seconds I have printed the json file output in the interval of 3 seconds.

6.Instruction on how to test the trained DNN

6.1 Install Dependencies
• Python 3
• Keras
• Tensorflow
• Numpy,Scipy

6.2 Path

Path of the input data:
In the beginning of the code please change the path to access the Json files accordingly. Rest of the code takes care of converting the Json files into a dataframe and reshaping the input.

Path of the Model:
In the code please change the path from where the model needs to be loaded.

6.3 Code
Please find the code on github

6.4 Video Link
Video of how to use the test.ipynb is in the below link.
https://youtu.be/CDwjklctJ6A