DEEP LEARNING PROJECT WEEK-4

1.Topic

In this project I have built a Deep Neural Network to detect Tremor/Shaking of Upper body of people. I have used a deep neural network with Long-Short-Term memory cells and several Dense layers to optimize the performance.

2.Data

For this application I have created my own data set with several videos of positive and negative samples. I initially started with as few as 20 videos, but eventually I have increased my dataset to a huge extent. I have made videos that are completely positive, completely negative and videos that contain both positive and negative actions to know the limitations of my model and further improve it.

The videos are taken at a rate of 30 frames per second. I have acquired the body landmarks of the person in the video using openpose.I have taken the pose_keypoints for this model. Open-pose gives 50 key points of human pose but I have taken only 30 key points as the rest are not useful for the prediction purposes.The program converts the video into json format, this json file is converted into a dataframe and is further fed into the network.

As weeks passed I kept adding data to my model, but as the videos were manually generated, there were high fluctuations in the action as there can be many ways to shiver. This week I have cleaned all the redundant data and added more data that is more relevant to the action. This process has improved my model performance to a huge extent.

My network gives frame wise output but the final result is the mean of every 30 frames as it makes a second. I have reshaped my data into 90 time steps and with a batch size of 30 my model gives highly accurate results.

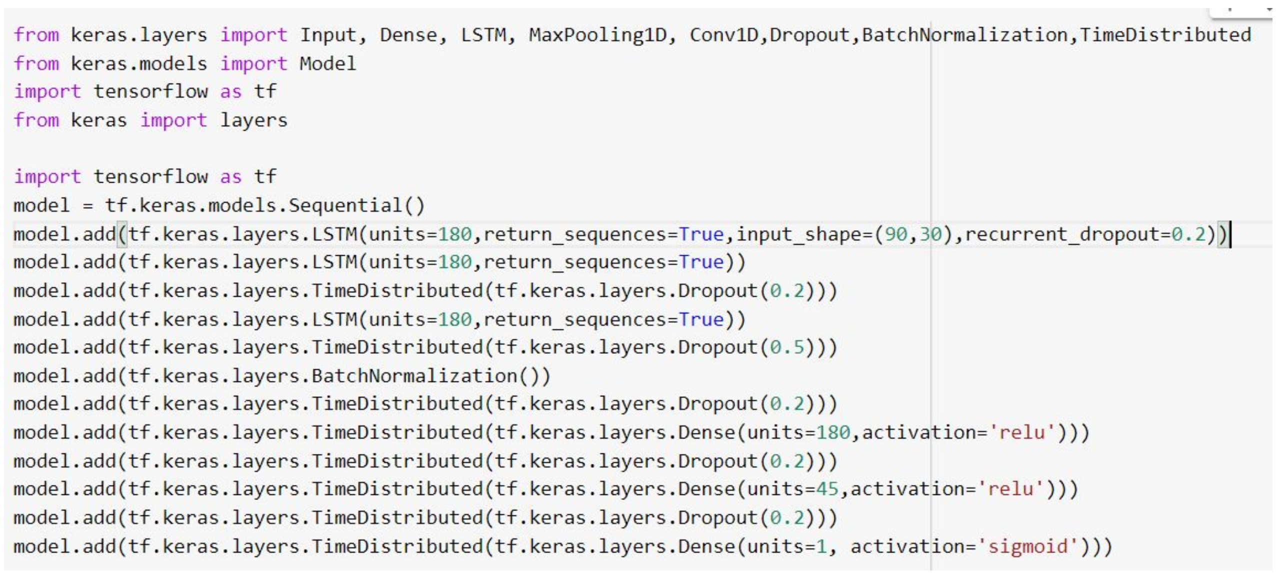
3.DNN Model

I have tried several sequential models and choose to ensemble a sequential LSTM model and a sequential Bidirectional LSTM model with model weights of 0.9 and 0.1 respectively.

3.1 Model 1

The model is built with a linear stack of LSTM layers with each having 180 units and 90 timesteps. I have used dropout layers in both LSTM and Dense layers to generalize the training well and improve accuracies with unseen videos. After experimenting with a lots of models I have finalized this model as it gives very high accuracies.

3.1.1 Architecture



```
from keras.layers import Input, Dense, LSTM, MaxPooling1D, Conv1D,Dropout,BatchNormalization,TimeDistributed
from keras.models import Model
import tensorflow as tf
from keras import layers

import tensorflow as tf
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.LSTM(units=180,return_sequences=True,input_shape=(90,30),recurrent_dropout=0.2))
model.add(tf.keras.layers.LSTM(units=180,return_sequences=True))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dropout(0.2)))
model.add(tf.keras.layers.LSTM(units=180,return_sequences=True))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dropout(0.5)))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dropout(0.2)))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(units=180,activation='relu')))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dropout(0.2)))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(units=45,activation='relu')))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dropout(0.2)))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(units=1, activation='sigmoid')))
```

3.1.2 Input: Shape of tensor

Before reshaping:

X_train shape is (80640,30):(Json files,features)

X_test shape is (20160,30):(Json files,features)

After reshaping:

X_train shape is (896,90,30):(samples,Timesteps,features)

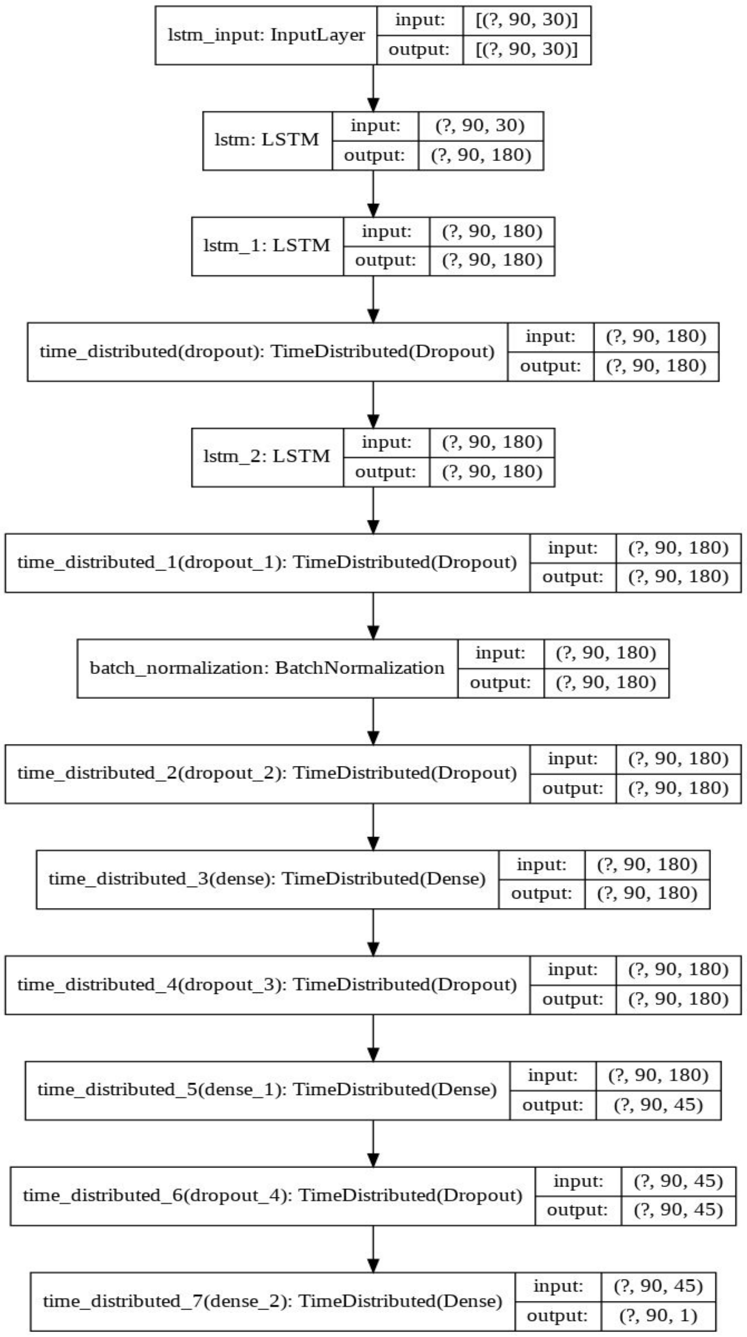
X_test shape is (224,90,30):(samples,Timesteps,features)

3.1.3 Output: Shape of tensor

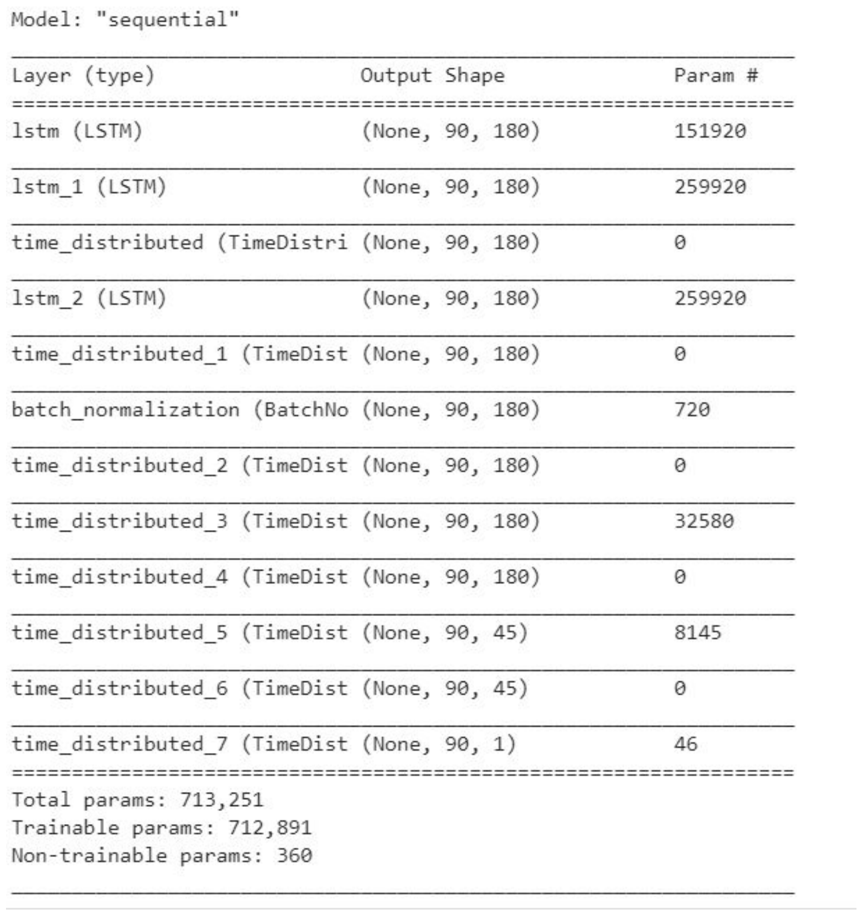
Y_train is (80640,1)

Y_test is (20160,1)

After reshaping:
Y_train is (896,90,1)
Y_test is (224,90,1)



3.1.4 Shape of output tensor in each layer



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm (LSTM)                  (None, 90, 180)           151920    
_________________________________________________________________
lstm_1 (LSTM)                (None, 90, 180)           259920    
_________________________________________________________________
time_distributed (TimeDistri (None, 90, 180)           0         
_________________________________________________________________
lstm_2 (LSTM)                (None, 90, 180)           259920    
_________________________________________________________________
time_distributed_1 (TimeDist (None, 90, 180)           0         
_________________________________________________________________
batch_normalization (BatchNo (None, 90, 180)           720       
_________________________________________________________________
time_distributed_2 (TimeDist (None, 90, 180)           0         
_________________________________________________________________
time_distributed_3 (TimeDist (None, 90, 180)           32580     
_________________________________________________________________
time_distributed_4 (TimeDist (None, 90, 180)           0         
_________________________________________________________________
time_distributed_5 (TimeDist (None, 90, 45)            8145      
_________________________________________________________________
time_distributed_6 (TimeDist (None, 90, 45)            0         
_________________________________________________________________
time_distributed_7 (TimeDist (None, 90, 1)             46        
=================================================================
Total params: 713,251
Trainable params: 712,891
Non-trainable params: 360
_________________________________________________________________
```

3.1.5 Hyperparameter Tuning

Range of Hyperparameters tried

Batch size	1,3,5,10,15,20,30,50
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
LSTM units	45,90,180,240,360

Optimal Hyperparameters found

Batch size	30
Epochs	500
Learning rate	0.0001
Dropout	0.2,0.5
Depth of LSTM layers	3
LSTM units	180

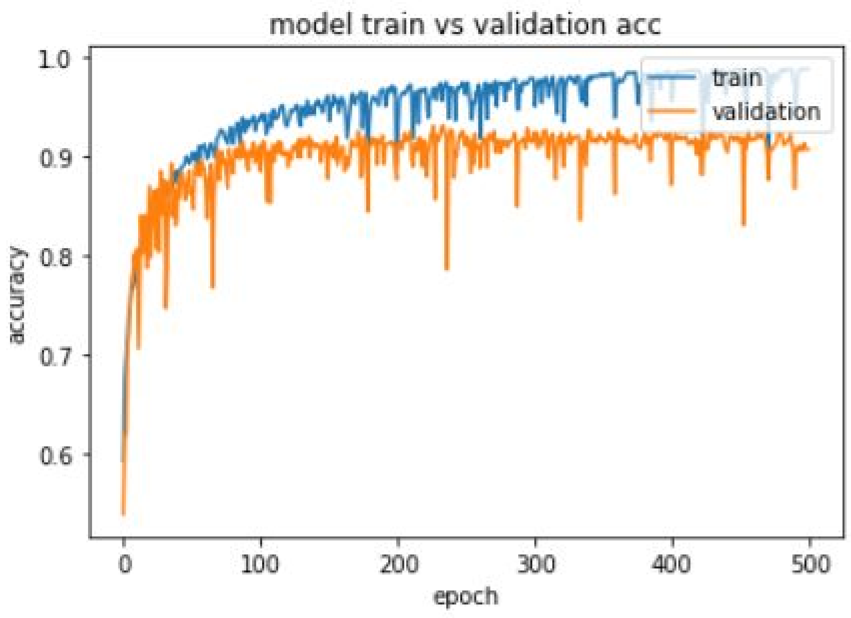
3.1.6 Performance

Best model found at 340th epoch

3.1.6.1 Accuracies

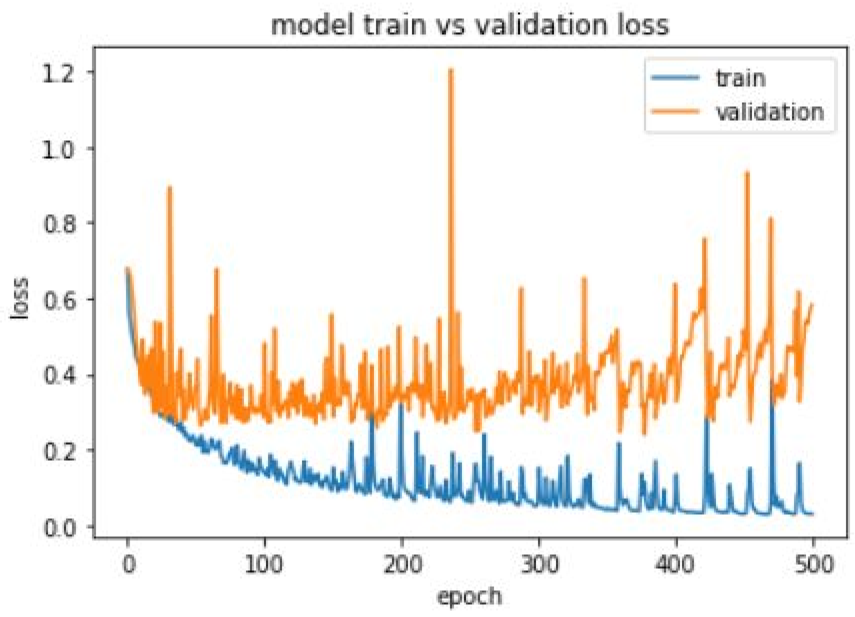
The below graph shows how the training and validation accuracies change for each epoch

Training acc: 96.23%   validation acc: 93.02%



3.1.6.2 Losses

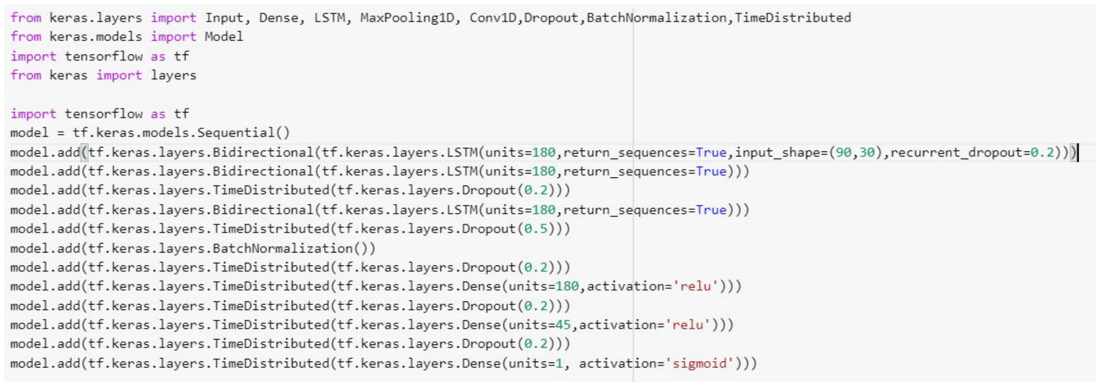
The below graph shows how the training and validation Losses change for each epoch
Training loss: 0.086 validation loss: 0.283



3.2 Model 2

This model was built in the similar way as model 1 but I have used Bidirectional LSTM units to reduce the number of false positives. I have used a batch size of 15 for this model.

3.2.1 Architecture

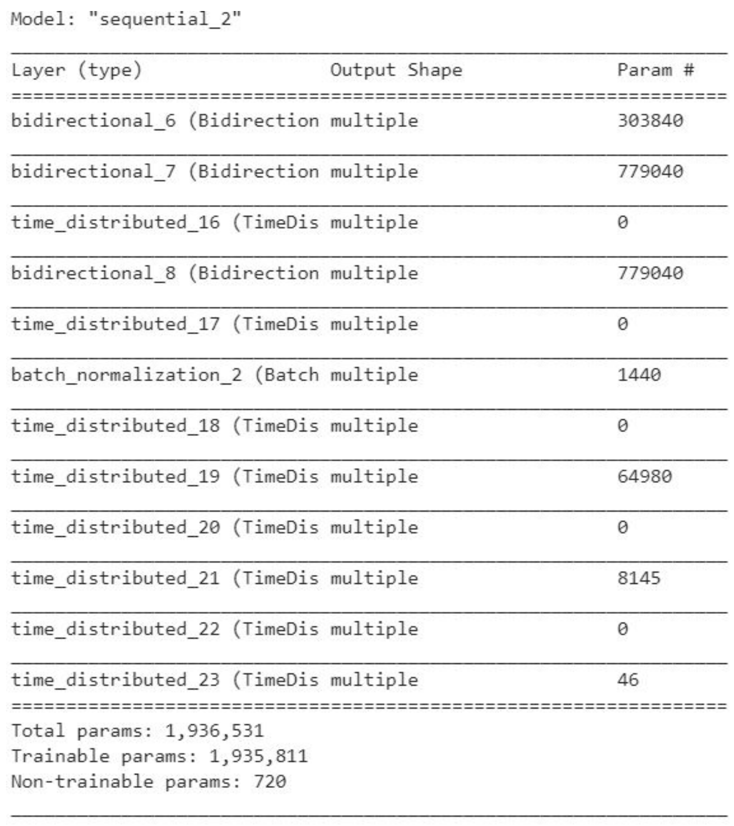


```
from keras.layers import Input, Dense, LSTM, MaxPooling1D, Conv1D,Dropout,BatchNormalization,TimeDistributed
from keras.models import Model
import tensorflow as tf
from keras import layers

import tensorflow as tf
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=180,return_sequences=True,input_shape=(90,30),recurrent_dropout=0.2)))
model.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=180,return_sequences=True)))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dropout(0.2)))
model.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=180,return_sequences=True)))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dropout(0.5)))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dropout(0.2)))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(units=180,activation='relu')))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dropout(0.2)))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(units=45,activation='relu')))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dropout(0.2)))
model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(units=1, activation='sigmoid')))
```

3.2.2 Input: Shape of tensor
Before reshaping:
X_train shape is (80640,30):(Json files,features)
X_test shape is (20160,30):(Json files,features)
After reshaping:
X_train shape is (896,90,30):(samples,Timesteps,features)
X_test shape is (224,90,30):(samples,Timesteps,features)

3.2.3 Output: Shape of tensor
Y_train is (80640,1)
Y_test is (20160,1)
After reshaping:
Y_train is (896,90,1)
Y_test is (224,90,1)



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_6 (Bidirection multiple                  303840
_________________________________________________________________
bidirectional_7 (Bidirection multiple                  779040
_________________________________________________________________
time_distributed_16 (TimeDis multiple                  0
_________________________________________________________________
bidirectional_8 (Bidirection multiple                  779040
_________________________________________________________________
time_distributed_17 (TimeDis multiple                  0
_________________________________________________________________
batch_normalization_2 (Batch multiple                  1440
_________________________________________________________________
time_distributed_18 (TimeDis multiple                  0
_________________________________________________________________
time_distributed_19 (TimeDis multiple                  64980
_________________________________________________________________
time_distributed_20 (TimeDis multiple                  0
_________________________________________________________________
time_distributed_21 (TimeDis multiple                  8145
_________________________________________________________________
time_distributed_22 (TimeDis multiple                  0
_________________________________________________________________
time_distributed_23 (TimeDis multiple                  46
=================================================================
Total params: 1,936,531
Trainable params: 1,935,811
Non-trainable params: 720
_________________________________________________________________
```

3.2.5 Hyperparameter Tuning

Range of Hyperparameters tried

Batch size	1,3,5,10,15,20,30,50
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
LSTM units	45,90,180,240,360

Optimal Hyperparameters found

Batch size	30
Epochs	500
Learning rate	0.0001
Dropout	0.2,0.5
Depth of LSTM layers	3
LSTM units	180

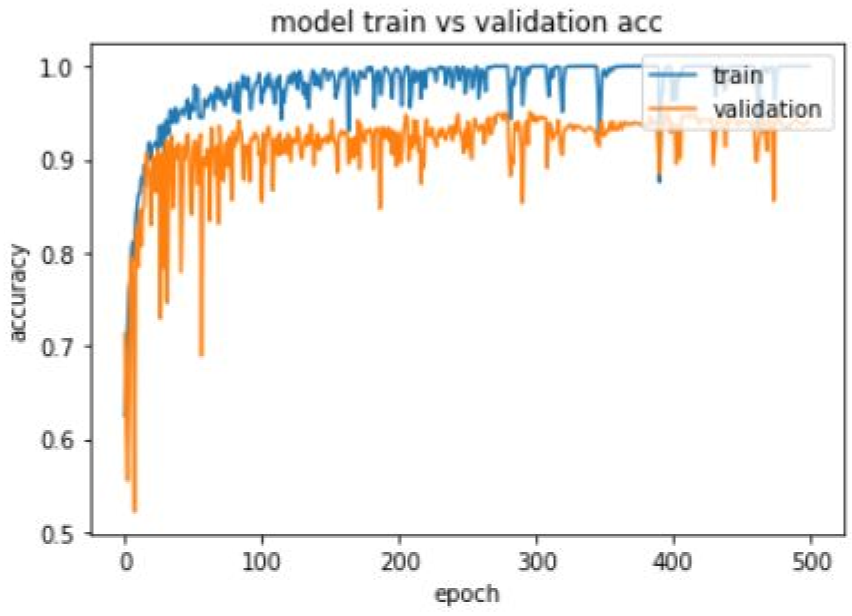
3.2.6 Performance

Best model found at 296th epoch

3.2.6.1 Accuracies

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

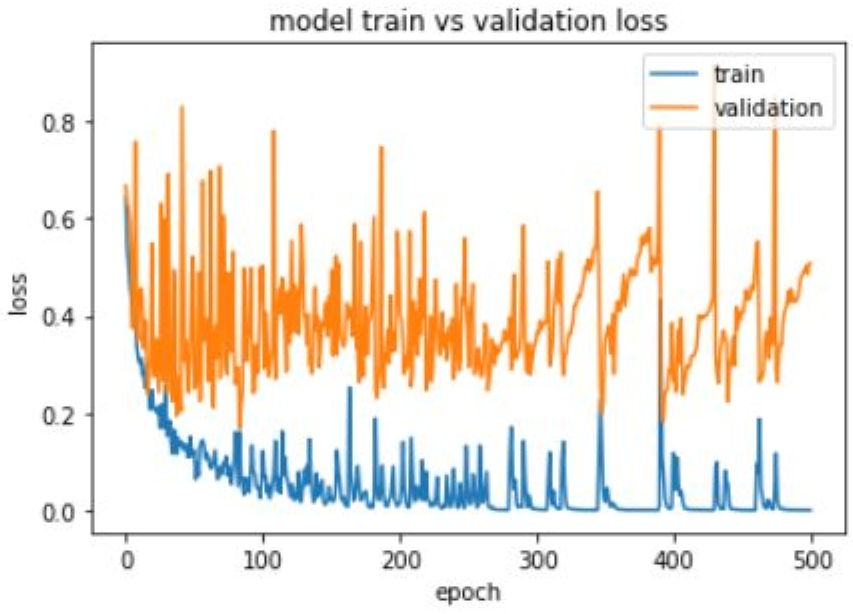
Training acc: 98.63%   validation acc: 95.05%



3.2.6.2 Losses

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

Training loss: 0.0165 validation loss: 0.2788



Final model= 0.8*model-1 + 0.2*model-2

4.MPROVEMENTS

- I have cleaned the entire dataset and re-labelled them so that more and more accurate results can be produced. I have removed irrelevant positive actions and finetuned them.
- The model detects shivering from most of the angles and when the person is sitting as well. I have included several videos from different angles in the training data.
- I have improved my model accuracy from 80% to 95% approximately, by changing the Data, hyperparameters of the network, network architectures and time stamps.
- In my model I have included videos from several angles and my model detects the shivering while sitting and in the side angles as well with higher accuracies.
- Many negative actions such as jumping jacks, standing, walking were being detected as positive in my previous models, I have overcome this by adding these videos as negative samples to the training. By these attempts I have improved the models performance in several edge cases.

5.EXPERIMENTS

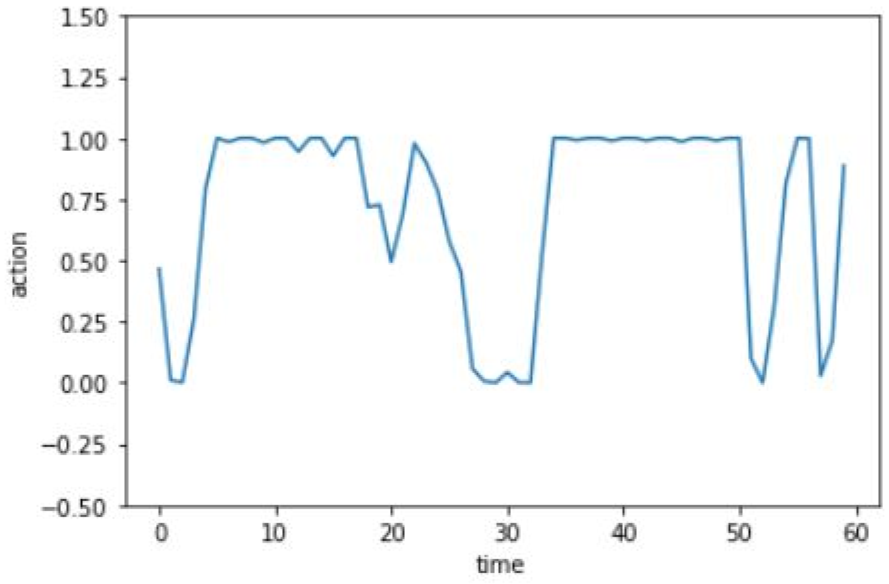
1. I have used several architectures including sending the video input directly to the network using VGG16 pretrained network, but my actions were slight movements in the body which were not able to get detected by VGG16 and produced very bad results.
2. I have changed several layers, tried 1 bidirectional LSTM and 2 LSTM layers, 2 Bidirectional LSTM and 1 LSTM layers and many more combinations to get the accurate results.
3. I have been getting bad results when a video has both positive and negative actions but successfully after several experiments and hyperparameter trials, I have ended up in solving the problem. I have checked all these networks for 15, 30, 45 and 90 timestamps. I have created several more videos to improve the validation accuracy.
4. I have tried several regularization techniques for all the layers including L1 and L2 norms and batch normalization apart from dropout. I have tried batch normalization after the LSTM layers and after the Dense layers as well.
5. I have followed the iterative data science approach where I trained the model, found out the limitations of the model, tried all the permutations to get the nest fit and then created more videos where the model does not perform well and then again trained the network. I continued this cycle throughout the project.

6.RESULTS

Video 1:
0-15 seconds : positive action
15-30 seconds: negative action
30-45 seconds: positive action
45-60 seconds: negative action

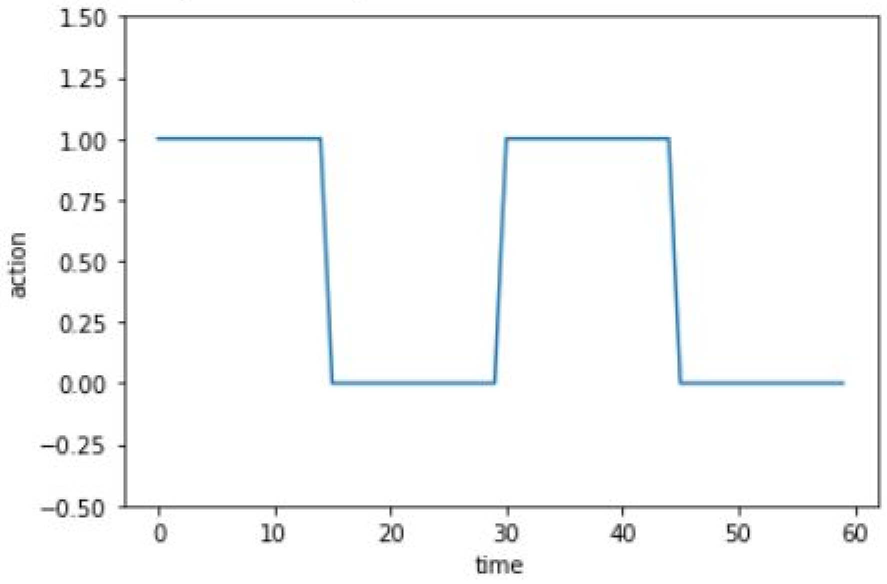
Prediction

```
Found:  1820 json keypoint frame files
```



Actual Output

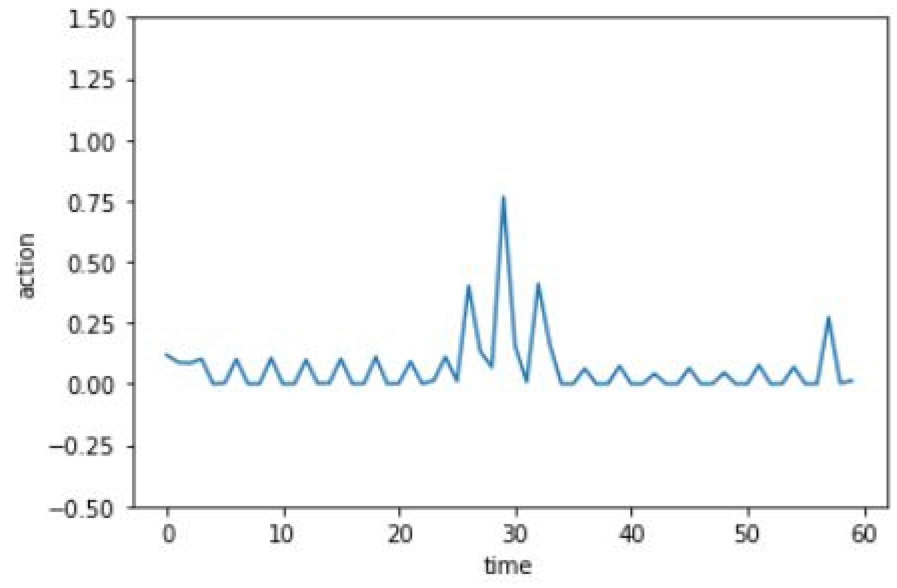
```
Text(0, 0.5, 'action')
```



Video 2:
0-60 seconds: negative action

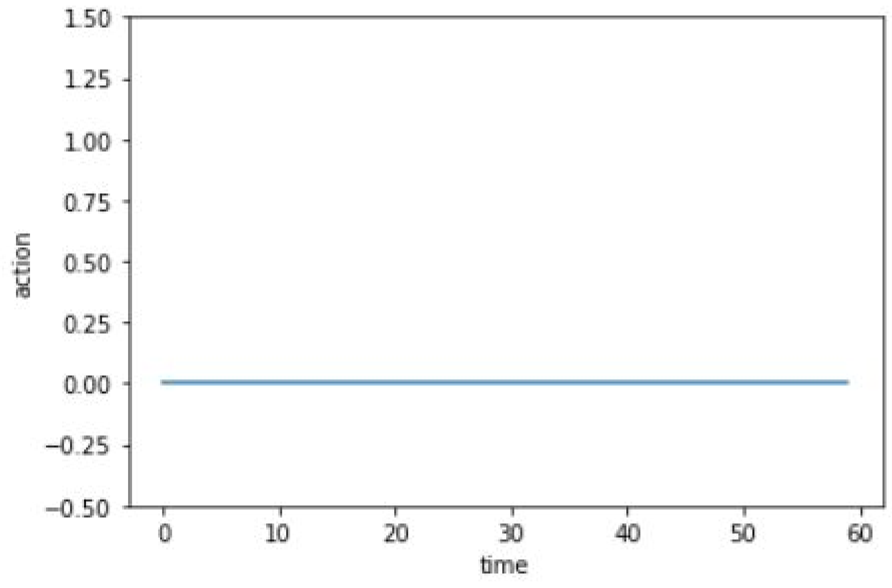
Prediction:

```
Found:  3624 json keypoint frame files
```



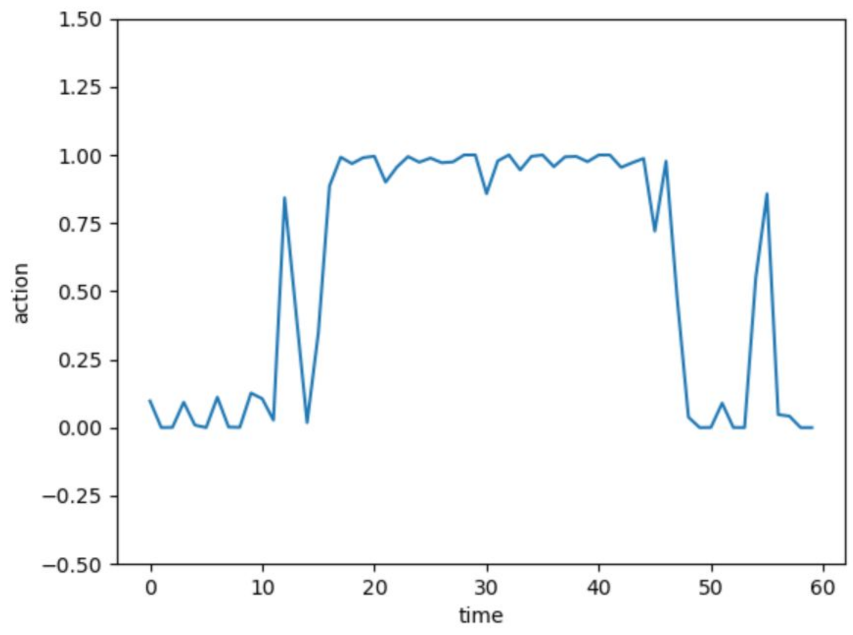
Actual Output:

```
Text(0, 0.5, 'action')
```



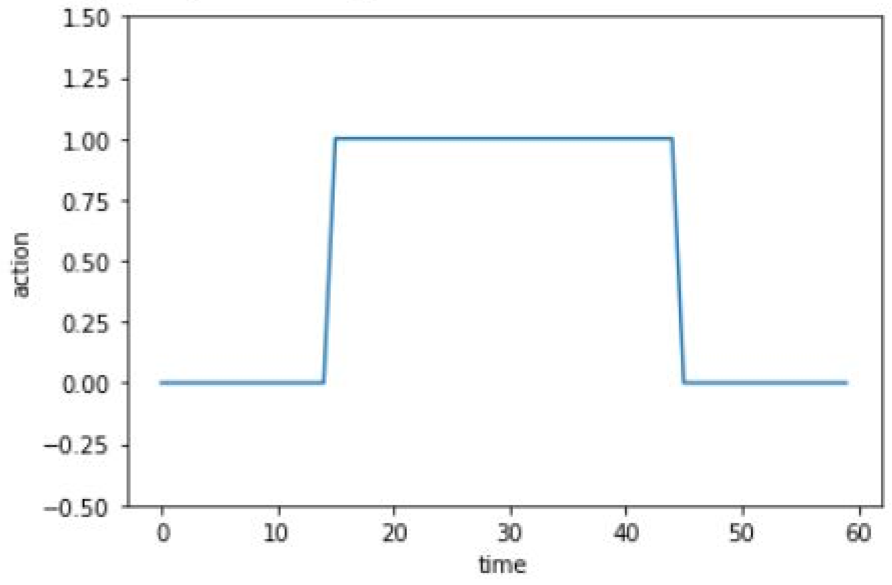
Video-3:
0-15 seconds: negative action
15-45 seconds: positive action
45-60 seconds: negative action

Prediction:



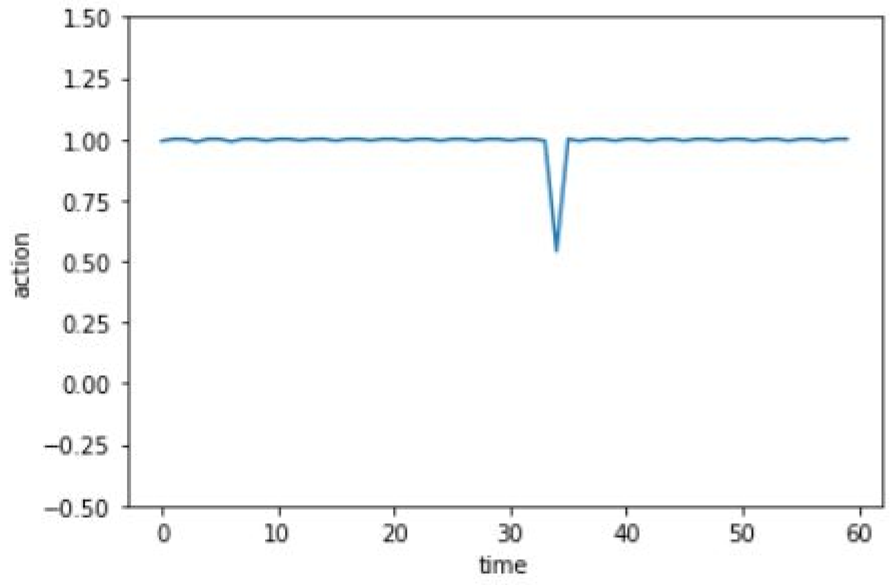
Actual Output:

```
Text(0, 0.5, 'action')
```



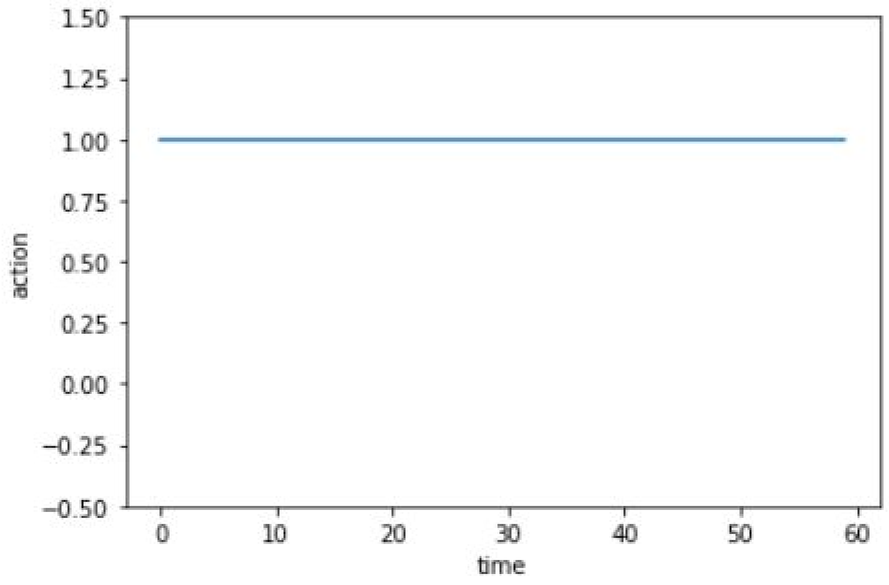
Video 4:
0-60 seconds: positive action
Predicted:

```
Found:  1817 json keypoint frame files
```



Actual Output:

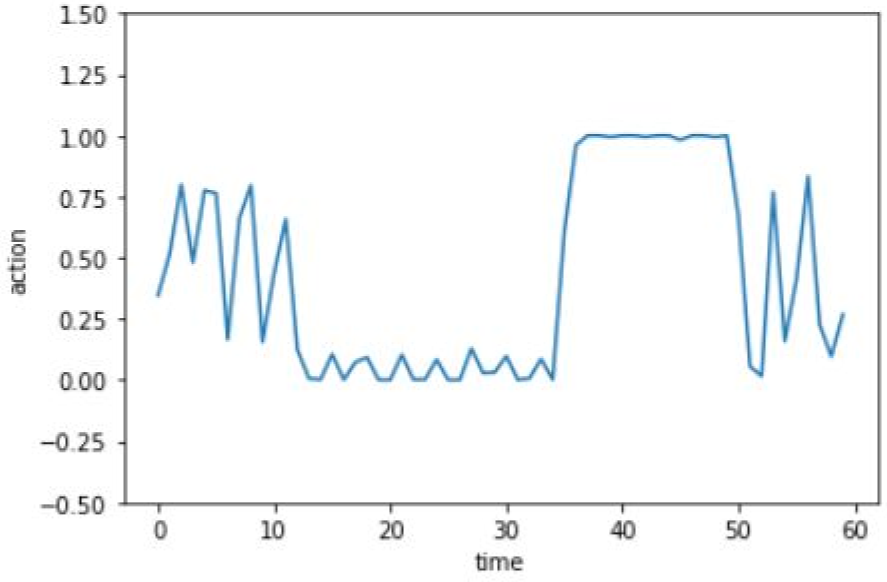
```
Text(0, 0.5, 'action')
```



Video 5:
0-15 seconds: positive action
15-30 seconds: negative action
30-60 seconds: positive action

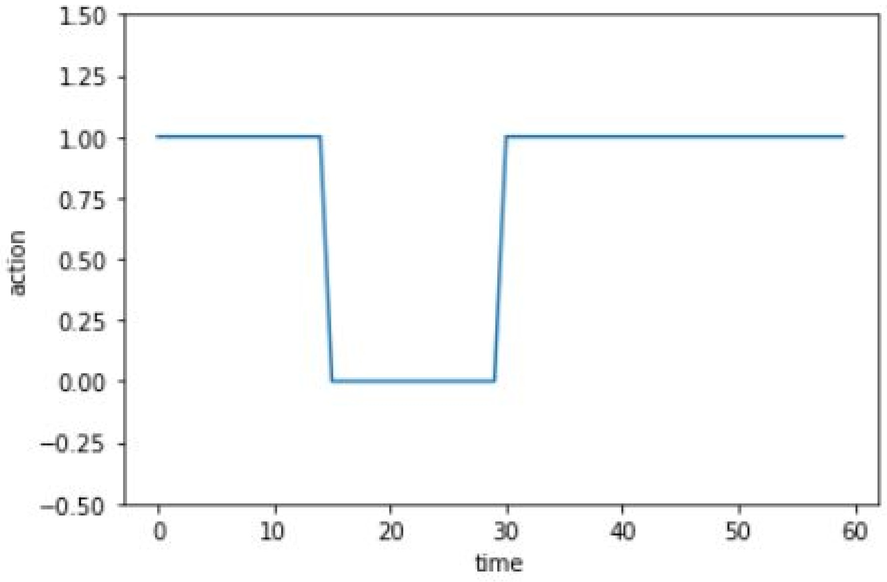
Predicted:

```
Found:  1830 json keypoint frame files
```



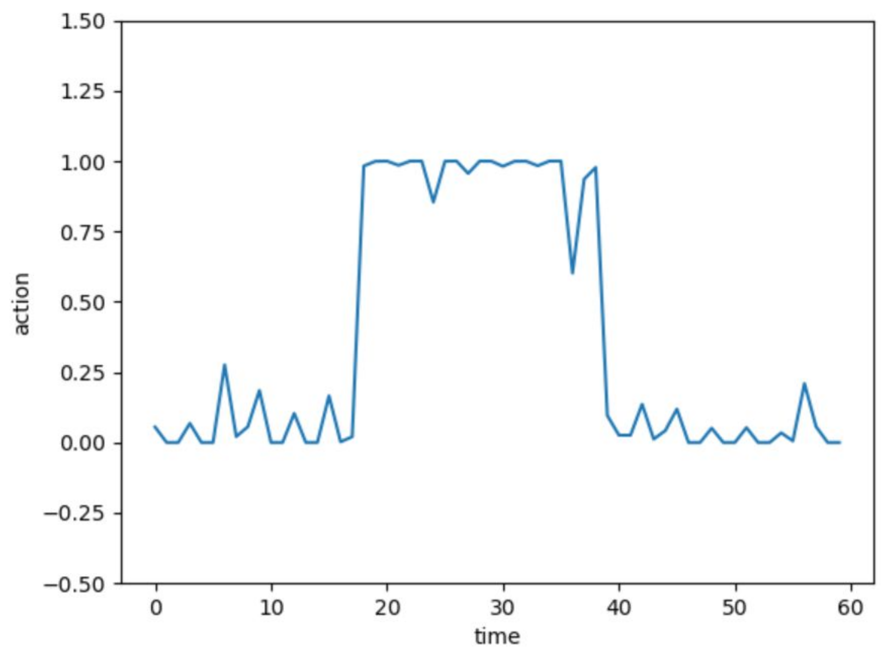
Actual Output:

```
Text(0, 0.5, 'action')
```



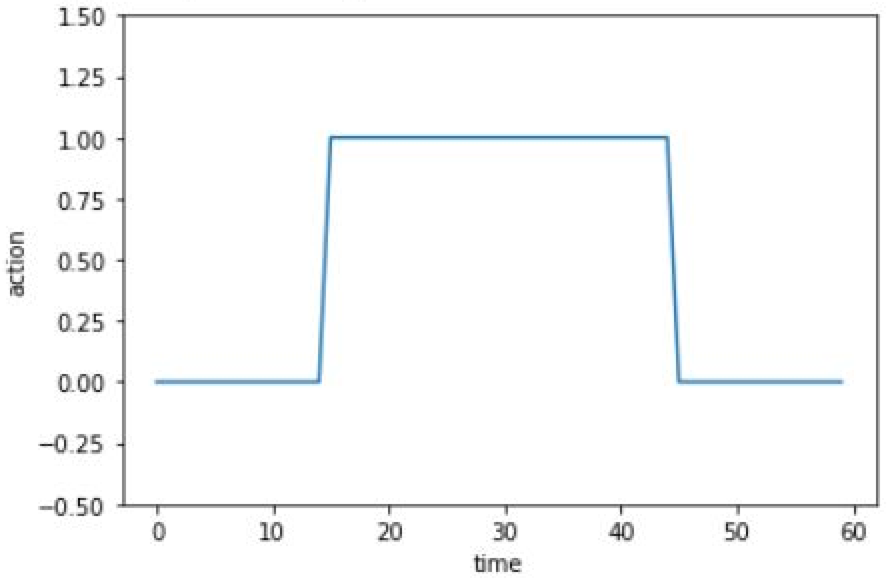
Video 6:
0-15 seconds: negative action
15-45 seconds: positive action
45-60 seconds: negative action

Prediction:



Actual Output:

```
Text(0, 0.5, 'action')
```



7.STEPS TO RUN THE PROGRAM

Step1: Install the entire folder from the drive link.
Step2: Open command prompt in that folder.
Step3: Run test.py file in the folder.

Inputs: Inputs of the network i.e the json files are present in the input json folder.
Results: output graphs and time label json files will be stored in the respective folders named in the file.