Aashrith sai Tadkamadla UIN:929004194

Deep Learning Report Week 3

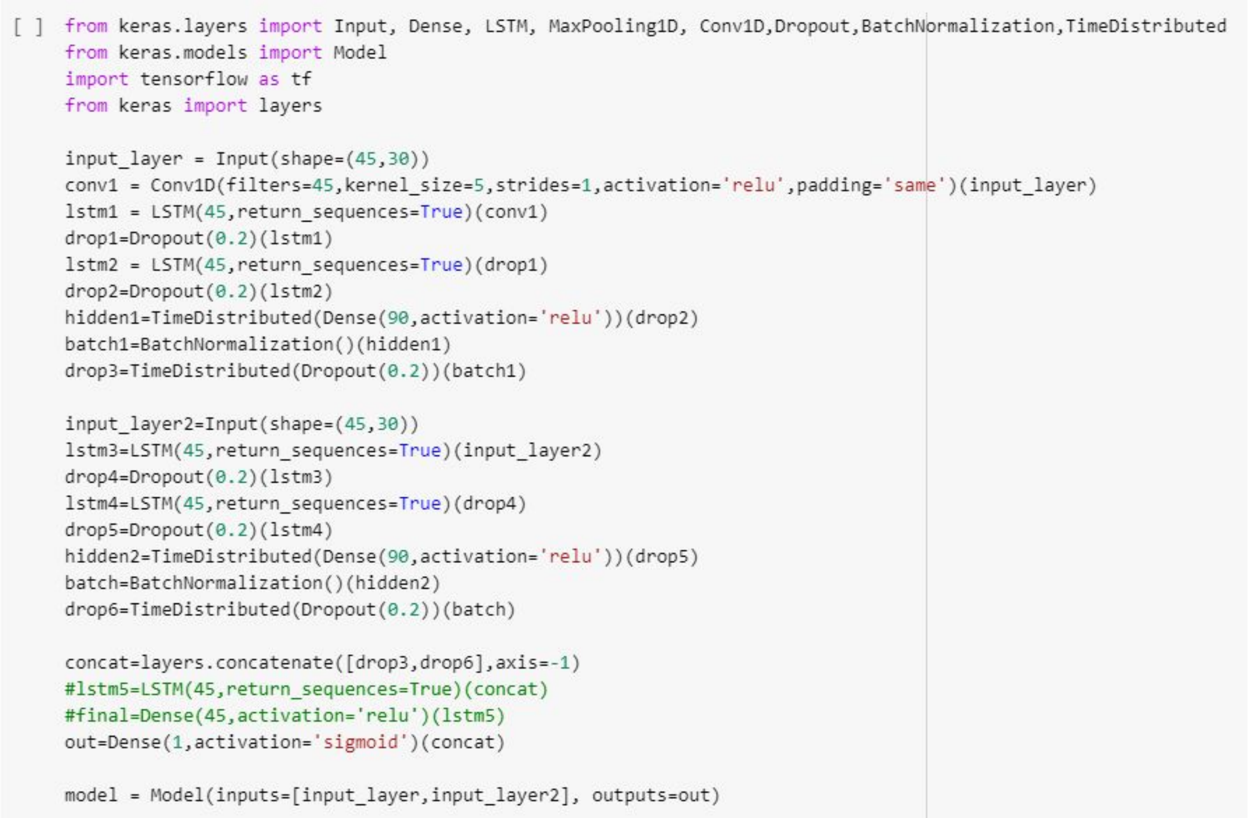
1.Topic In this project I have built a Neural Network to detect Tremor/Shaking of Upper or Lower Body of people. I have used a deep neural network with Long-Short-Term memory cells

2. Experiments and Results.

2.1 Network Architectures.

2.1.1 1D convolutional + LSTM network using keras API: I have concatenated two networks using keras APIs to get a better classification accuracy. But I have got the same validation accuracy as my previous network which is approximately ~80%.

Code:



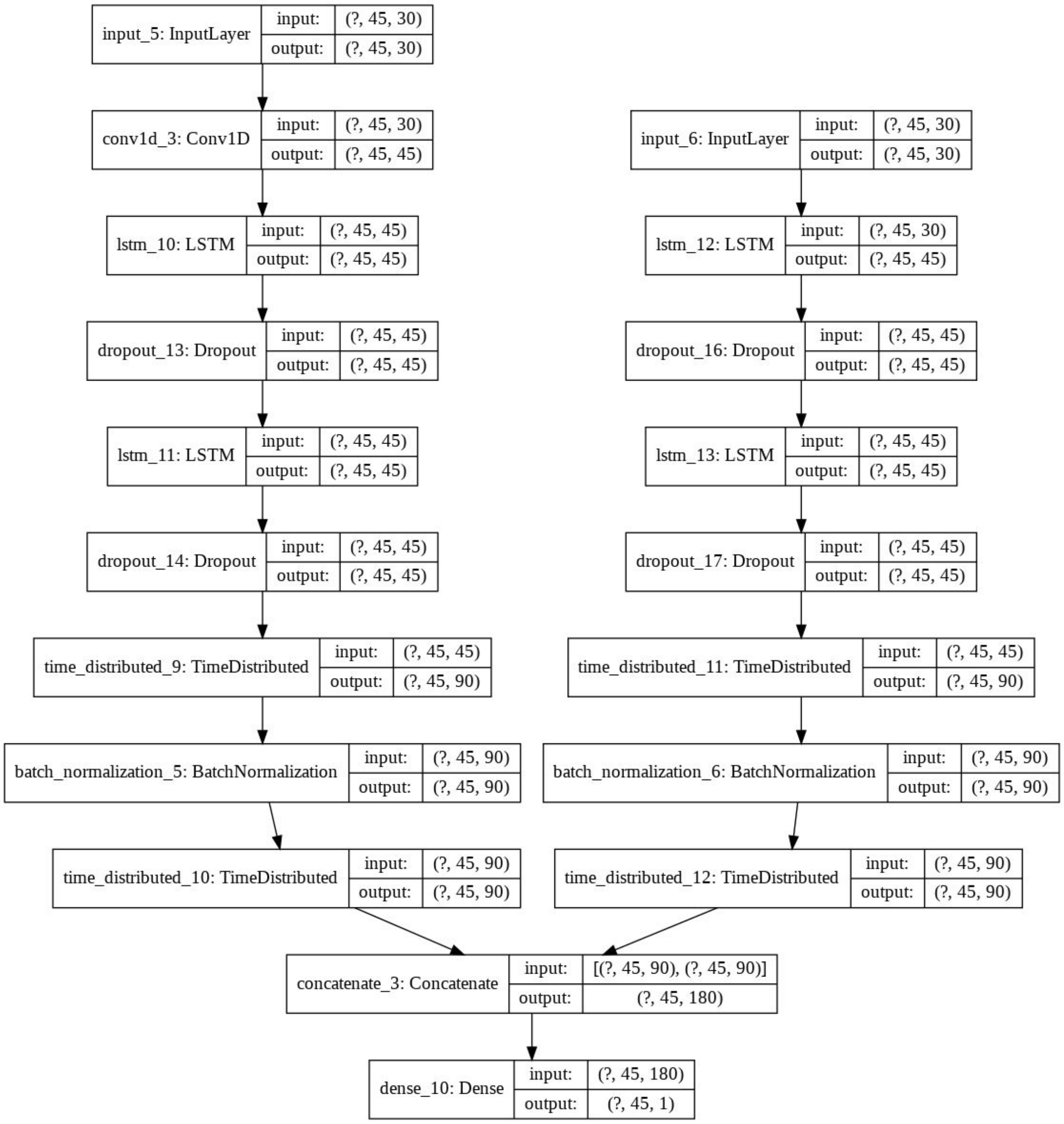
```
[ ] from keras.layers import Input, Dense, LSTM, MaxPooling1D, Conv1D,Dropout,BatchNormalization,TimeDistributed
    from keras.models import Model
    import tensorflow as tf
    from keras import layers

    input_layer = Input(shape=(45,30))
    conv1 = Conv1D(filters=45,kernel_size=5,strides=1,activation='relu',padding='same')(input_layer)
    lstm1 = LSTM(45,return_sequences=True)(conv1)
    drop1=Dropout(0.2)(lstm1)
    lstm2 = LSTM(45,return_sequences=True)(drop1)
    drop2=Dropout(0.2)(lstm2)
    hidden1=TimeDistributed(Dense(90,activation='relu'))(drop2)
    batch1=BatchNormalization()(hidden1)
    drop3=TimeDistributed(Dropout(0.2))(batch1)

    input_layer2=Input(shape=(45,30))
    lstm3=LSTM(45,return_sequences=True)(input_layer2)
    drop4=Dropout(0.2)(lstm3)
    lstm4=LSTM(45,return_sequences=True)(drop4)
    drop5=Dropout(0.2)(lstm4)
    hidden2=TimeDistributed(Dense(90,activation='relu'))(drop5)
    batch=BatchNormalization()(hidden2)
    drop6=TimeDistributed(Dropout(0.2))(batch)

    concat=layers.concatenate([drop3,drop6],axis=-1)
    #lstm5=LSTM(45,return_sequences=True)(concat)
    #final=Dense(45,activation='relu')(lstm5)
    out=Dense(1,activation='sigmoid')(concat)

    model = Model(inputs=[input_layer,input_layer2], outputs=out)
```

Hyperparameter Tuning:

	Tried	Best
Learning rate	0.1, 0.01, 0.001, 0.0001	Changed according to the loss function in between. But,0.0001 worked the best

Steps_per_epoch	1,5,10,20,30	20
epochs	10,20,30,100,500	Due to the complex architecture the model started overfitting after 30 epochs.
Sample weights	{0:1,1:3},{0:1,1:5},{0:1,1:50}, None	None worked the best as validation accuracy kept decreasing with sample weights on.

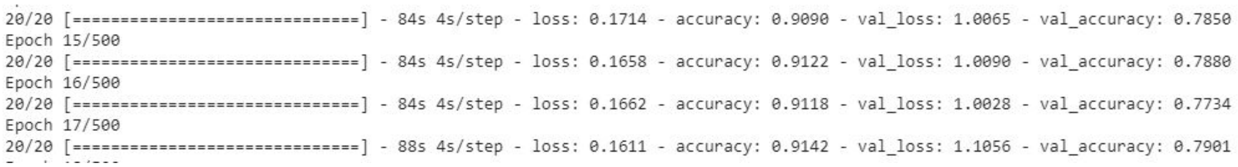
Input/Output flow chart:


input_5: InputLayer | input: (?, 45, 30) | output: (?, 45, 30)
conv1d_3: Conv1D | input: (?, 45, 30) | output: (?, 45, 45)
lstm_10: LSTM | input: (?, 45, 45) | output: (?, 45, 45)
dropout_13: Dropout | input: (?, 45, 45) | output: (?, 45, 45)
lstm_11: LSTM | input: (?, 45, 45) | output: (?, 45, 45)
dropout_14: Dropout | input: (?, 45, 45) | output: (?, 45, 45)
time_distributed_9: TimeDistributed | input: (?, 45, 45) | output: (?, 45, 90)
batch_normalization_5: BatchNormalization | input: (?, 45, 90) | output: (?, 45, 90)
time_distributed_10: TimeDistributed | input: (?, 45, 90) | output: (?, 45, 90)
input_6: InputLayer | input: (?, 45, 30) | output: (?, 45, 30)
lstm_12: LSTM | input: (?, 45, 30) | output: (?, 45, 45)
dropout_16: Dropout | input: (?, 45, 45) | output: (?, 45, 45)
lstm_13: LSTM | input: (?, 45, 45) | output: (?, 45, 45)
dropout_17: Dropout | input: (?, 45, 45) | output: (?, 45, 45)
time_distributed_11: TimeDistributed | input: (?, 45, 45) | output: (?, 45, 90)
batch_normalization_6: BatchNormalization | input: (?, 45, 90) | output: (?, 45, 90)
time_distributed_12: TimeDistributed | input: (?, 45, 90) | output: (?, 45, 90)
concatenate_3: Concatenate | input: [(?, 45, 90), (?, 45, 90)] | output: (?, 45, 180)
dense_10: Dense | input: (?, 45, 180) | output: (?, 45, 1)

Training and Performance:

Aashrith sai Tadkamadla UIN:929004194

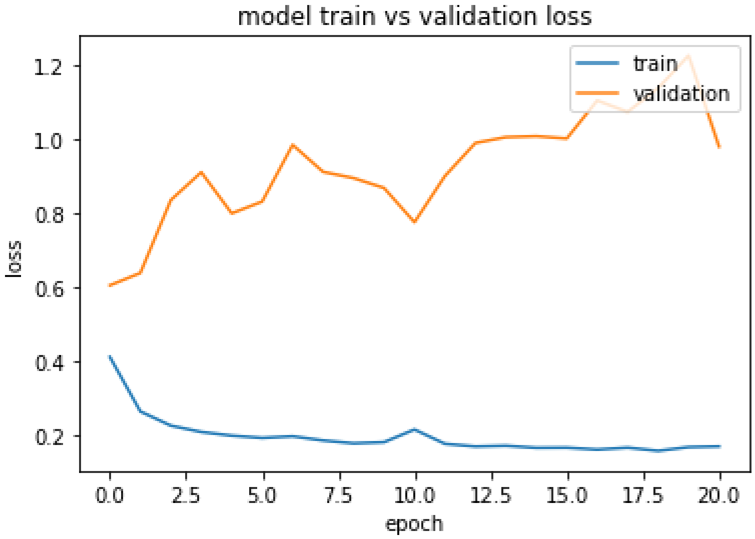
I have chosen the best weights where the validation accuracy is around 79.01% and training accuracy is around 91.42%



```
20/20 [==============================] - 84s 4s/step - loss: 0.1714 - accuracy: 0.9090 - val_loss: 1.0065 - val_accuracy: 0.7850
Epoch 15/500
20/20 [==============================] - 84s 4s/step - loss: 0.1658 - accuracy: 0.9122 - val_loss: 1.0090 - val_accuracy: 0.7880
Epoch 16/500
20/20 [==============================] - 84s 4s/step - loss: 0.1662 - accuracy: 0.9118 - val_loss: 1.0028 - val_accuracy: 0.7734
Epoch 17/500
20/20 [==============================] - 88s 4s/step - loss: 0.1611 - accuracy: 0.9142 - val_loss: 1.1056 - val_accuracy: 0.7901
```

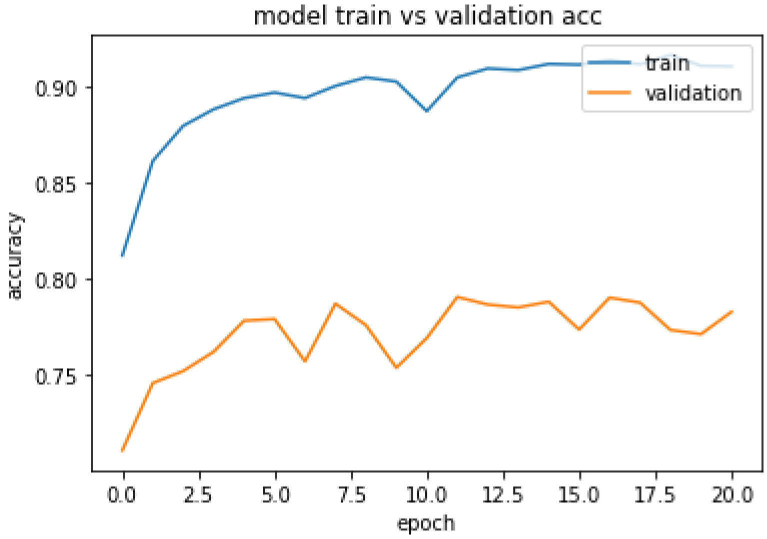
Training Loss vs Validation Loss:

Training loss:0.1611 Validation loss: 1.1056



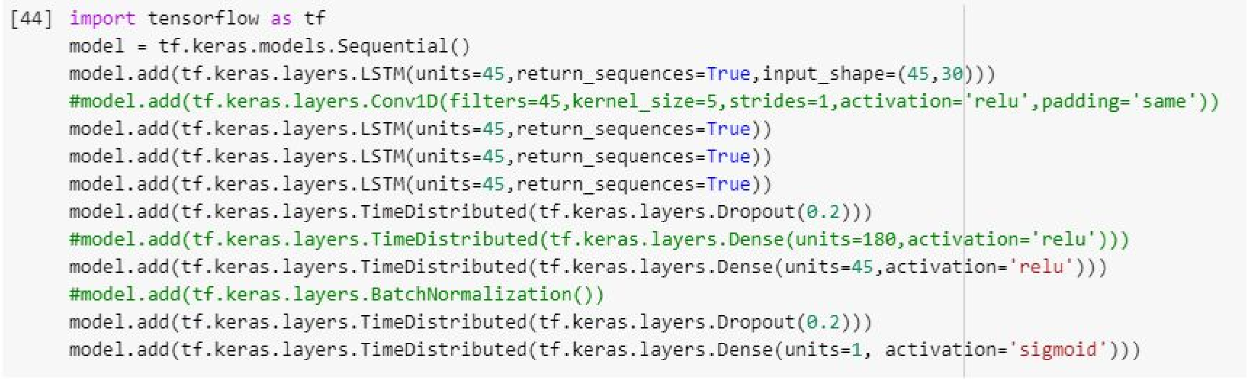
Training accuracy vs Validation accuracy:

Training accuracy: 91.42 Validation accuracy: 79.01



2.1.2 Sequential Network with several LSTM layers: I have added several LSTM layers from 2 to 5 in my previous model and compared the results.

Code:



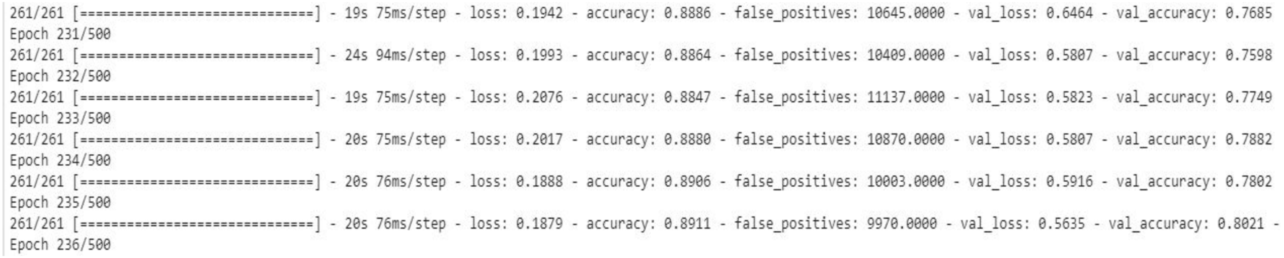
```
[44] import tensorflow as tf
     model = tf.keras.models.Sequential()
     model.add(tf.keras.layers.LSTM(units=45,return_sequences=True,input_shape=(45,30)))
     #model.add(tf.keras.layers.Conv1D(filters=45,kernel_size=5,strides=1,activation='relu',padding='same'))
     model.add(tf.keras.layers.LSTM(units=45,return_sequences=True))
     model.add(tf.keras.layers.LSTM(units=45,return_sequences=True))
     model.add(tf.keras.layers.LSTM(units=45,return_sequences=True))
     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.Dense(units=45,activation='relu')))
     #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=1, activation='sigmoid')))
```

Hyperparameter Tuning

	Tried	Best
Learning rate	0.1, 0.01, 0.002, 0.0002	0.0002
batch_size	5,10,20	10
epochs	10,20,30,100,500	I have run the model for 500 epochs as I have used callbacks to save the nest model.
No of LSTM layers	1,2,3,4,5	2 worked the best with 45 units in each.

Training and Performance

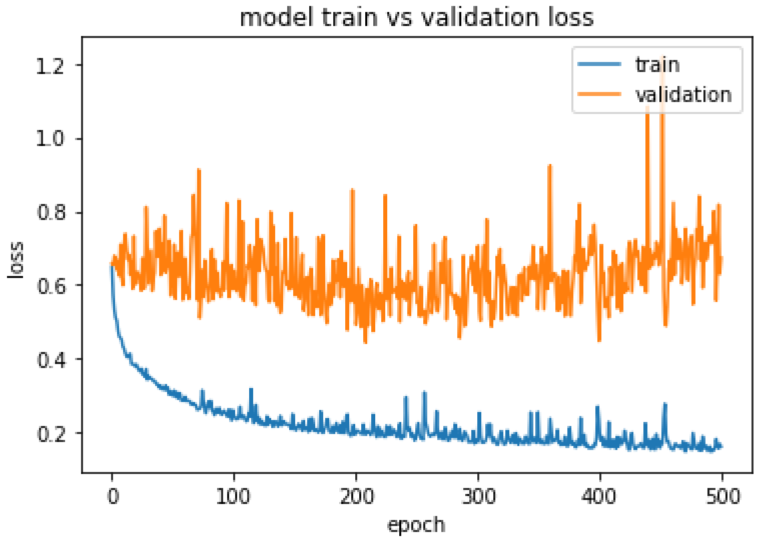
I have used call backs to get the best model of validation accuracy of 80.21% as shown in the below figure at 235 epochs.



```
261/261 [==============================] - 19s 75ms/step - loss: 0.1942 - accuracy: 0.8886 - false_positives: 10645.0000 - val_loss: 0.6464 - val_accuracy: 0.7685
Epoch 231/500
261/261 [==============================] - 24s 94ms/step - loss: 0.1993 - accuracy: 0.8864 - false_positives: 10409.0000 - val_loss: 0.5807 - val_accuracy: 0.7598
Epoch 232/500
261/261 [==============================] - 19s 75ms/step - loss: 0.2076 - accuracy: 0.8847 - false_positives: 11137.0000 - val_loss: 0.5823 - val_accuracy: 0.7749
Epoch 233/500
261/261 [==============================] - 20s 75ms/step - loss: 0.2017 - accuracy: 0.8880 - false_positives: 10870.0000 - val_loss: 0.5807 - val_accuracy: 0.7882
Epoch 234/500
261/261 [==============================] - 20s 76ms/step - loss: 0.1888 - accuracy: 0.8906 - false_positives: 10003.0000 - val_loss: 0.5916 - val_accuracy: 0.7802
Epoch 235/500
261/261 [==============================] - 20s 76ms/step - loss: 0.1879 - accuracy: 0.8911 - false_positives: 9970.0000 - val_loss: 0.5635 - val_accuracy: 0.8021 -
Epoch 236/500
```

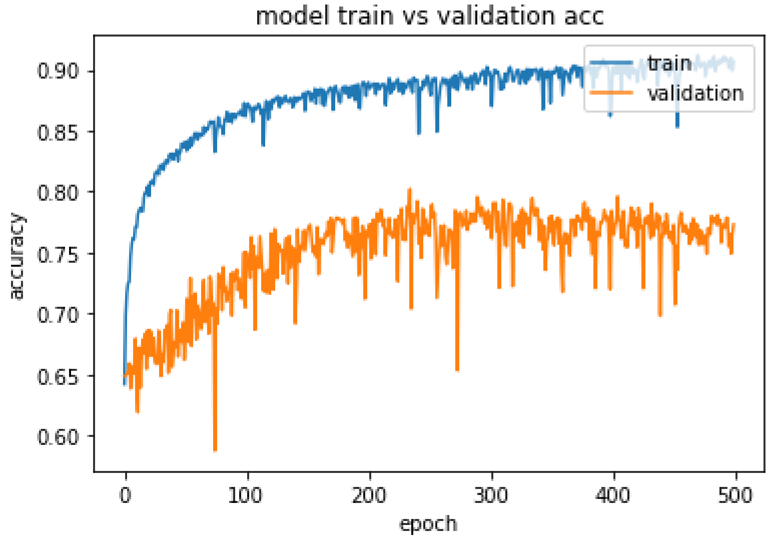
Training Loss vs Validation Loss:

Training loss:0.1879 Validation loss: 0.5635

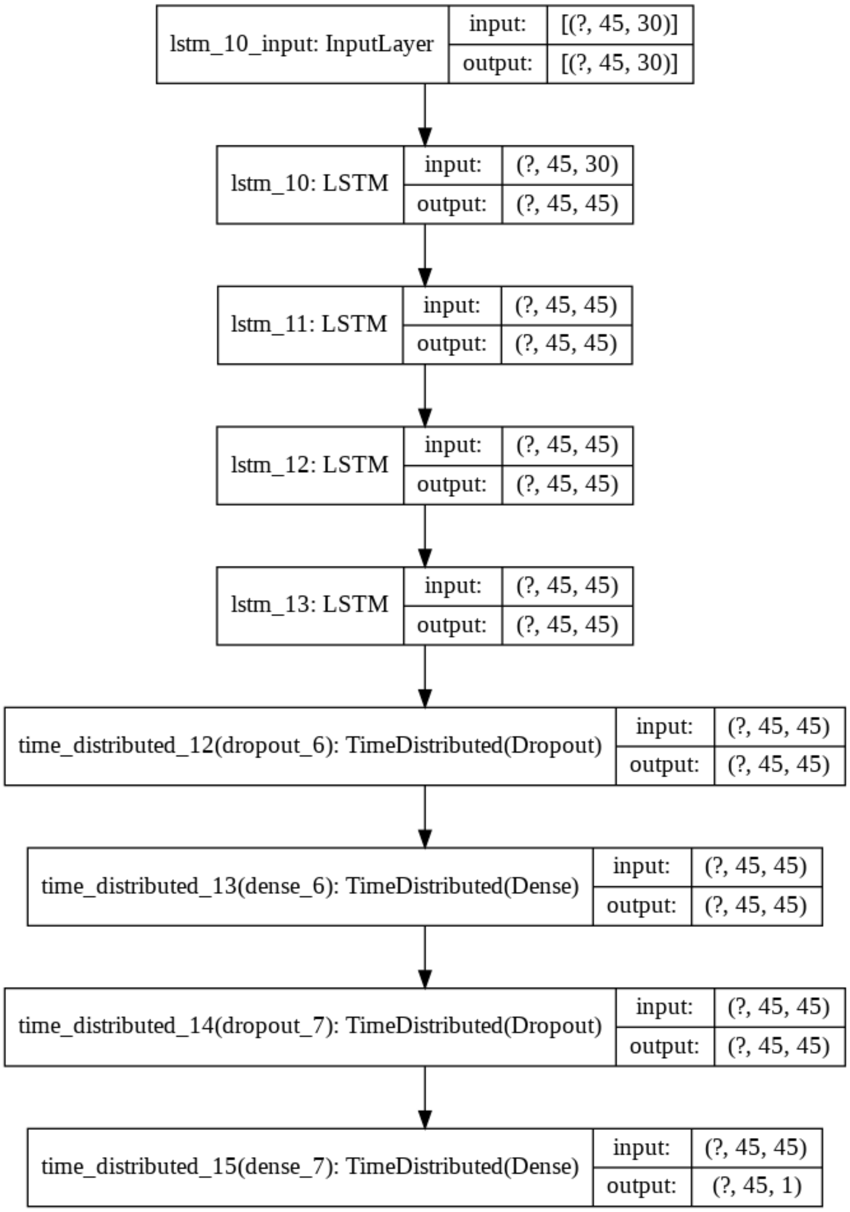


Training accuracy vs Validation accuracy:

Training accuracy: 89.11 Validation accuracy: 80.21

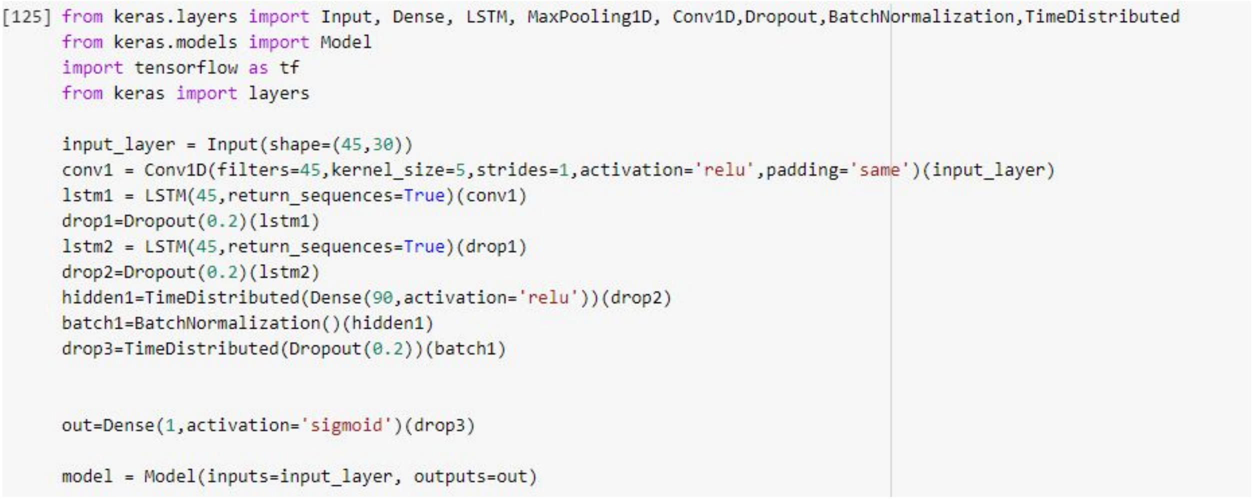


Input/Output flow chart:



2.1.3 sequential Conv1D+LSTM:

Code



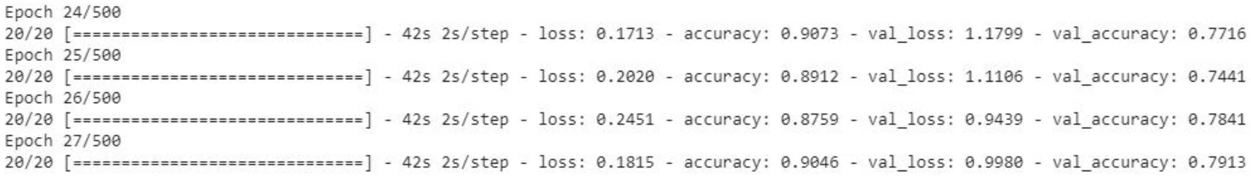
```
[125] from keras.layers import Input, Dense, LSTM, MaxPooling1D, Conv1D,Dropout,BatchNormalization,TimeDistributed
      from keras.models import Model
      import tensorflow as tf
      from keras import layers

      input_layer = Input(shape=(45,30))
      conv1 = Conv1D(filters=45,kernel_size=5,strides=1,activation='relu',padding='same')(input_layer)
      lstm1 = LSTM(45,return_sequences=True)(conv1)
      drop1=Dropout(0.2)(lstm1)
      lstm2 = LSTM(45,return_sequences=True)(drop1)
      drop2=Dropout(0.2)(lstm2)
      hidden1=TimeDistributed(Dense(90,activation='relu'))(drop2)
      batch1=BatchNormalization()(hidden1)
      drop3=TimeDistributed(Dropout(0.2))(batch1)


      out=Dense(1,activation='sigmoid')(drop3)

      model = Model(inputs=input_layer, outputs=out)
```

Hyperparameter Tuning

	Tried	Best
Learning rate	0.1, 0.01, 0.002, 0.0001	0.002
batch_size	5,10,20	10
epochs	10,20,30,100,500	I have used early stopping at patience level 30

Training and Performance

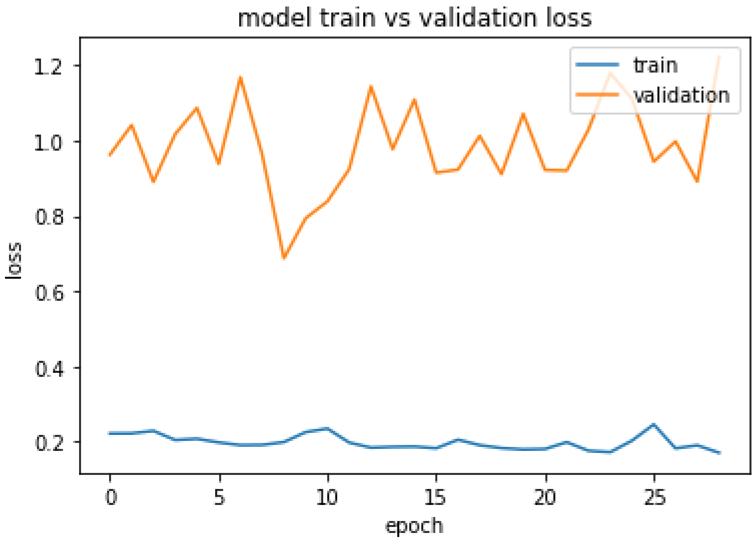
I have used call backs and early stopping to get the best weights of the model as it started converging at 24 epochs.



```
Epoch 24/500
20/20 [==============================] - 42s 2s/step - loss: 0.1713 - accuracy: 0.9073 - val_loss: 1.1799 - val_accuracy: 0.7716
Epoch 25/500
20/20 [==============================] - 42s 2s/step - loss: 0.2020 - accuracy: 0.8912 - val_loss: 1.1106 - val_accuracy: 0.7441
Epoch 26/500
20/20 [==============================] - 42s 2s/step - loss: 0.2451 - accuracy: 0.8759 - val_loss: 0.9439 - val_accuracy: 0.7841
Epoch 27/500
20/20 [==============================] - 42s 2s/step - loss: 0.1815 - accuracy: 0.9046 - val_loss: 0.9980 - val_accuracy: 0.7913
```

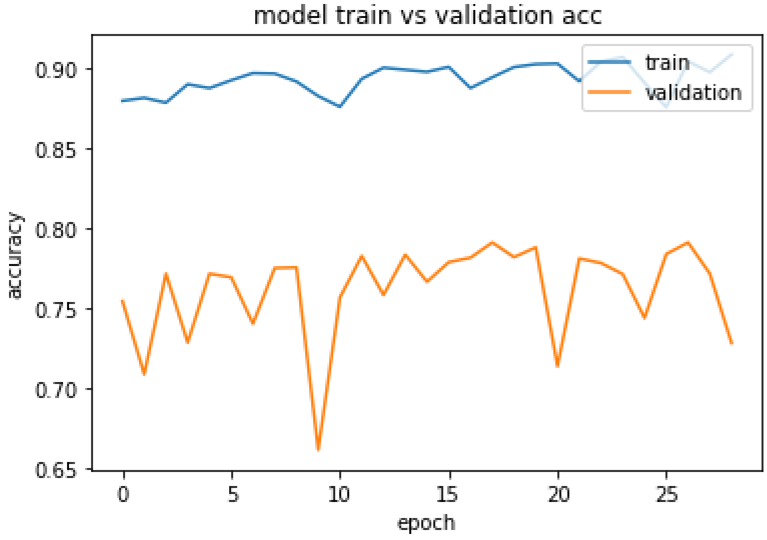
Training Loss vs Validation Loss:

Training loss:0.1815 Validation loss: 0.9980

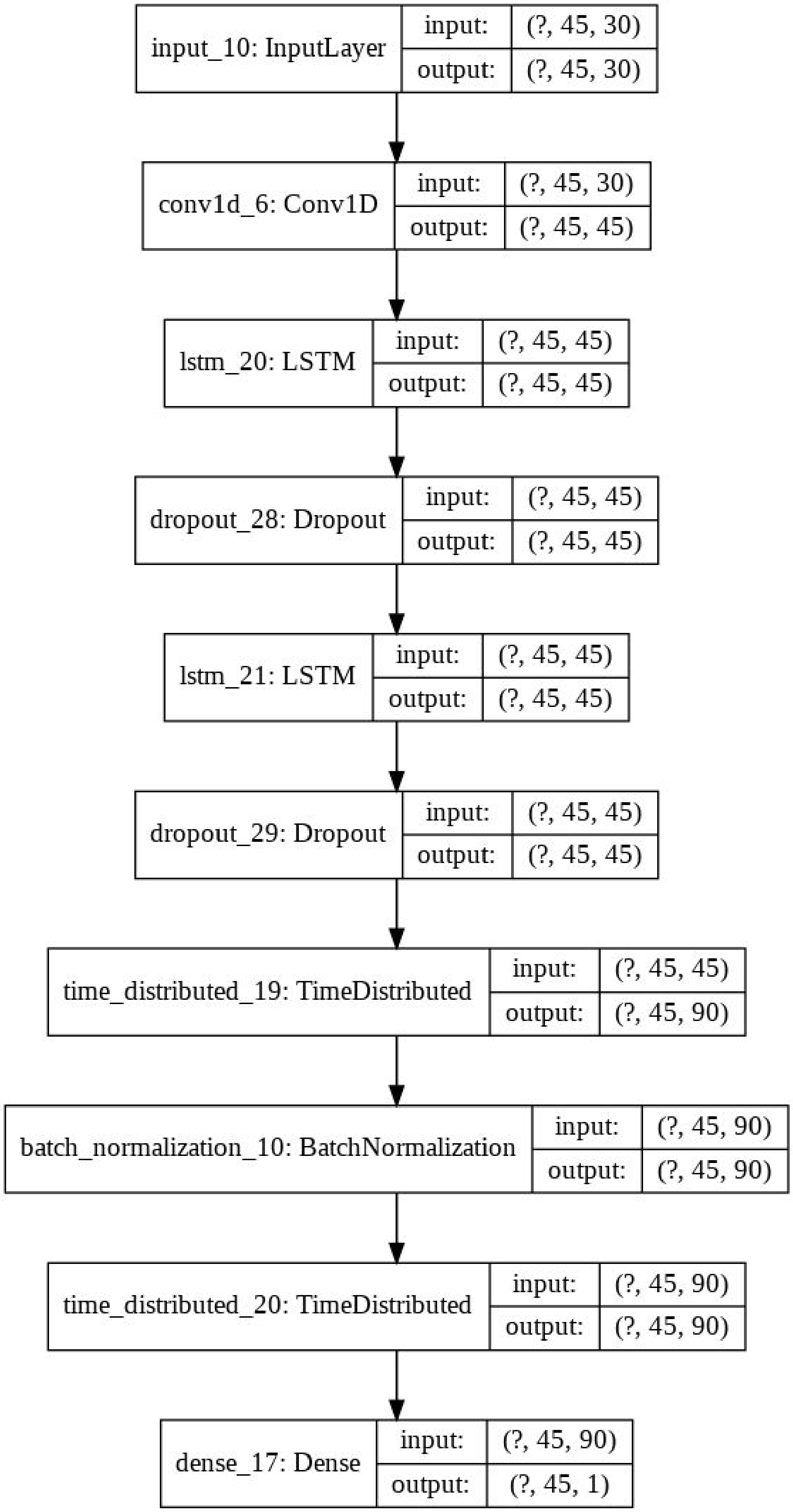


Training accuracy vs Validation accuracy:

Training accuracy: 90.46 Validation accuracy: 79.13



Input/Output flowchart



2.2 Improvements

2.2.1 Data: I have made 20 more videos of each 2 minutes approximately which has increased the data set to around 1.3 lakh frames. I have had the same validation data set as it was diverse and had several edge cases from before itself. I have figured out several actions which have been predicted incorrectly and made mode videos containing these actions, eg: My model used to detect standing as a positive output and hence I have given more standing videos labelling them as negative.

	Week1	Week2	Week3
Training Frames	50,000	90,000	1,30,000
Validation Frames	10,000	18,000	18,000

2.2.2 Performance: I have trained several models and used Ensembling Technique to get better results. Though the validation accuracy remained the same, there is a huge improvement in the graph outputs as you can see below. The sequential layer with convolutional was detecting negative samples really well and hence by ensembling these networks I have got better graphs and better results.

$Y=0.5*Y1+0.5*Y2$
Y1: predictions of convolutional and LSTM network
Y2: predictions of deep LSTM network.

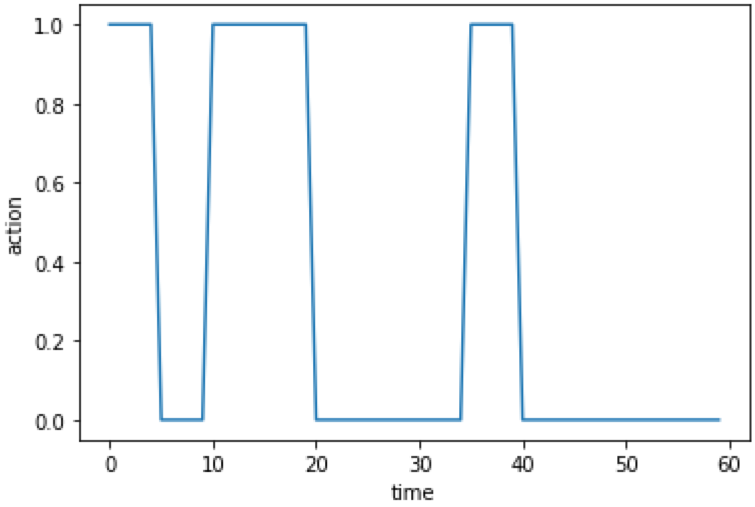
2.2.3 Networks: I have implemented several networks and have observed the results, I have chosen the two best networks and have ensembled them.

2.2.4 Sample weights: I have used sample weights to compensate for the imbalanced dataset with less number or positive samples.

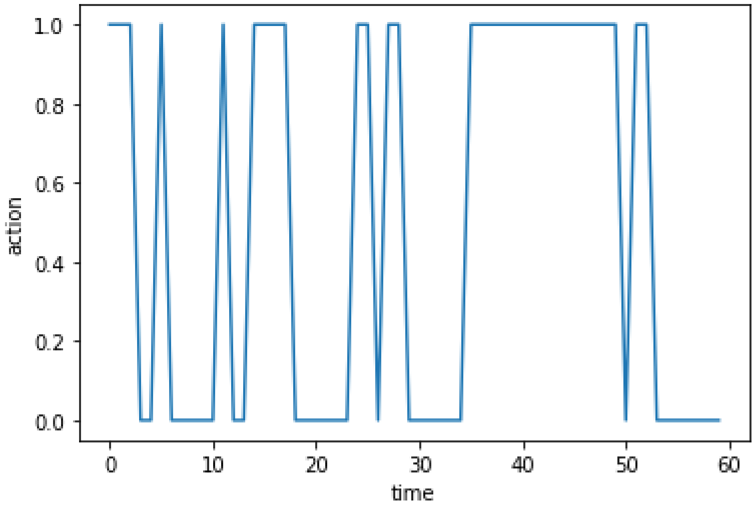
2.3 Results and Graphs

2.3.1 Video1

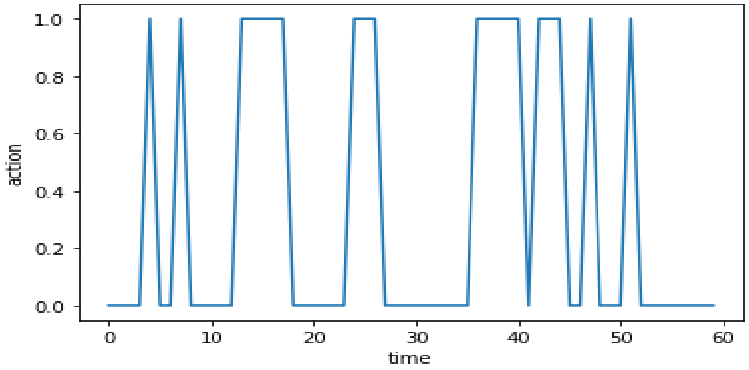
Original Graph



Week 2 (Deep LSTM network)



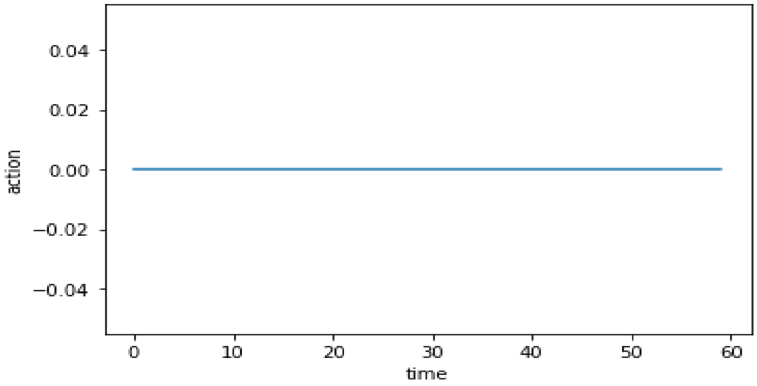
Week 3 (Ensemble Convolutional LSTM + Deep LSTM)



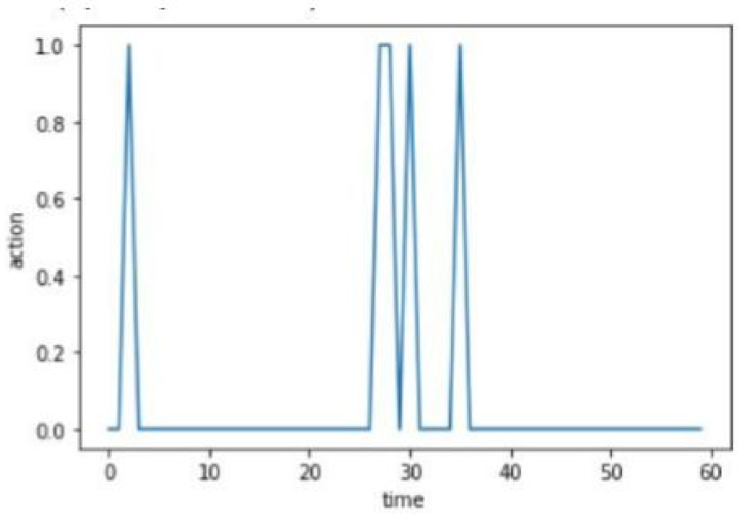
2.3.2 Video2

Aashrith sai Tadkamadla UIN:929004194

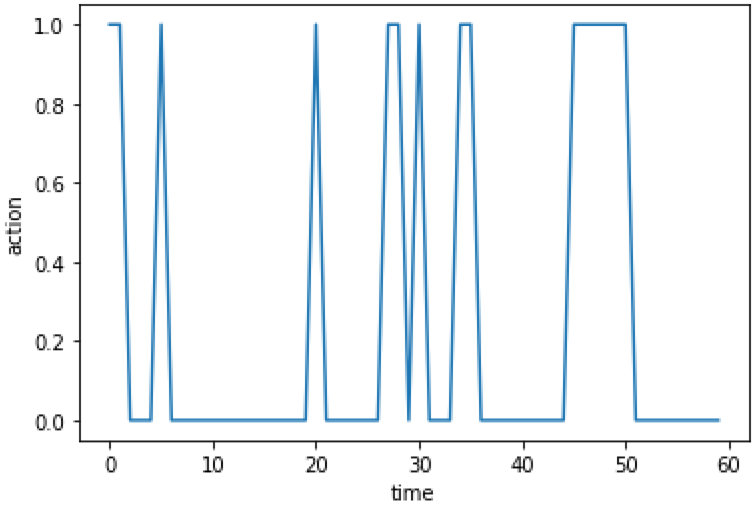
Original



Week 2 (Deep LSTM network)

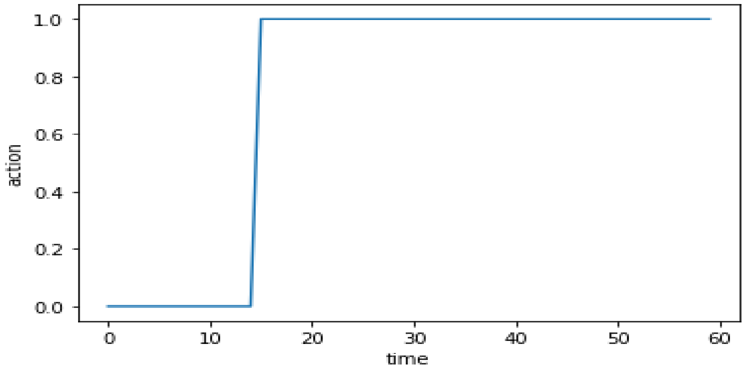


Week 3 (Ensemble Convolutional LSTM + Deep LSTM)

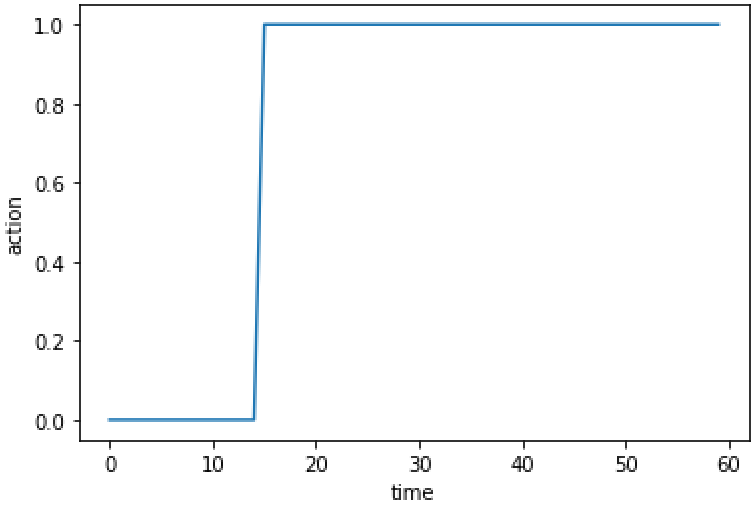


2.3.3 Video3

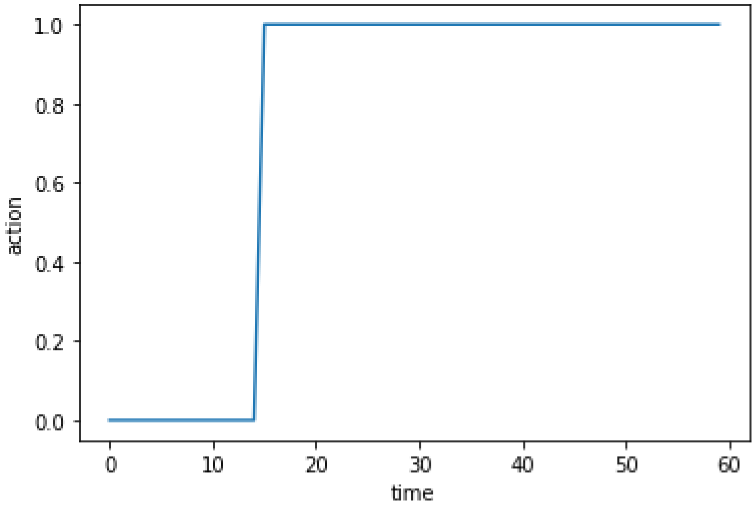
Original Graph



Week 2 (Deep LSTM network)

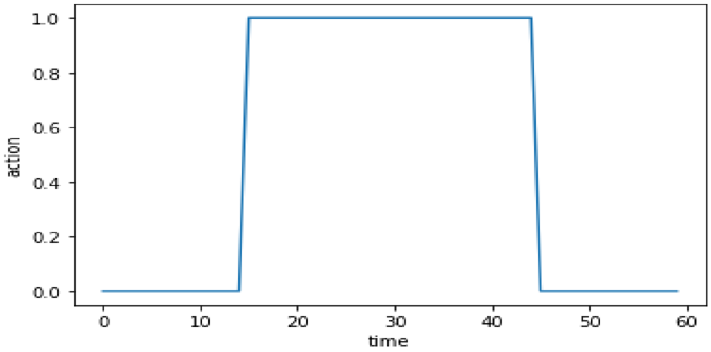


Week 3 (Ensemble Convolutional LSTM + Deep LSTM)

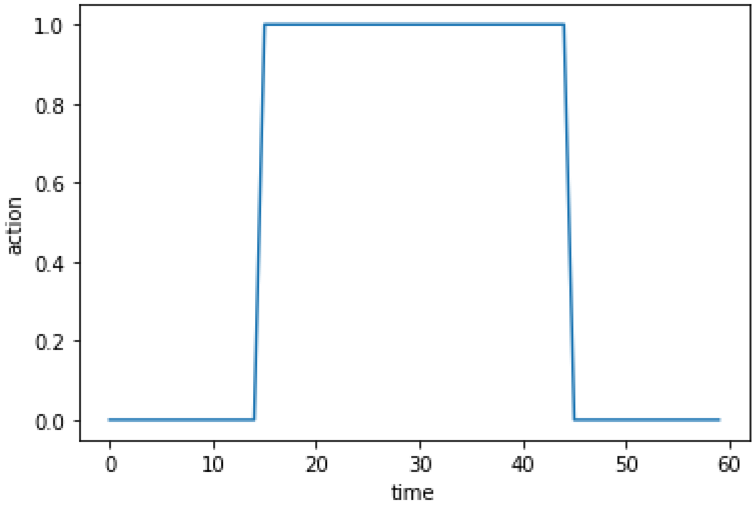


2.3.4 Video4: {15 seconds: 0, 30 seconds:1, 15 seconds:0}

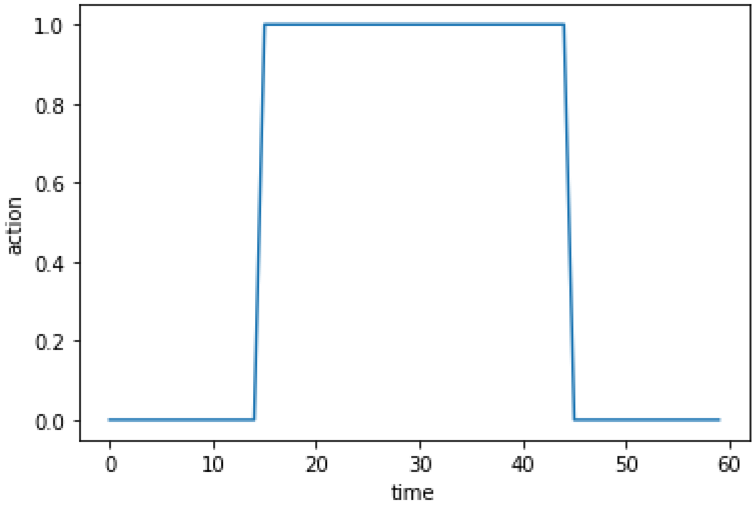
Original Graph



Week 2 (Deep LSTM network)



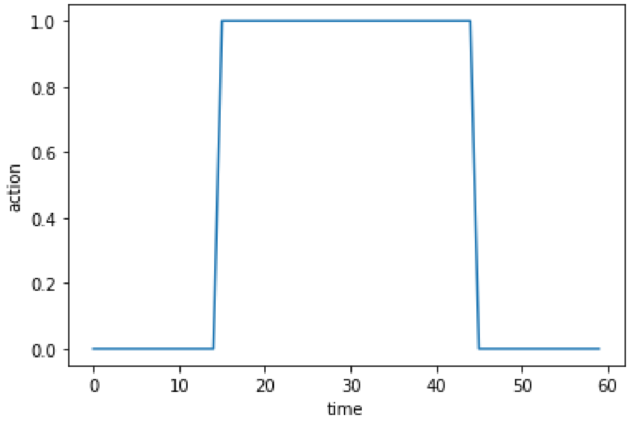
Week 3 (Ensemble Convolutional LSTM + Deep LSTM)



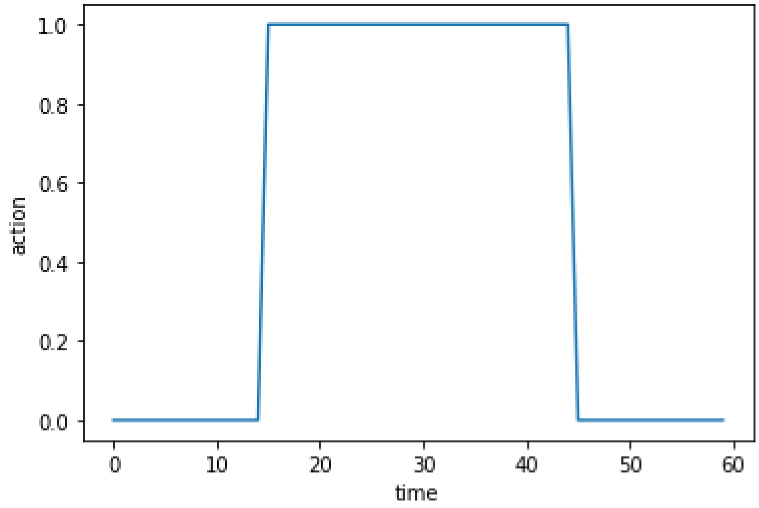
2.3.5 Video5

Aashrith sai Tadkamadla UIN:929004194

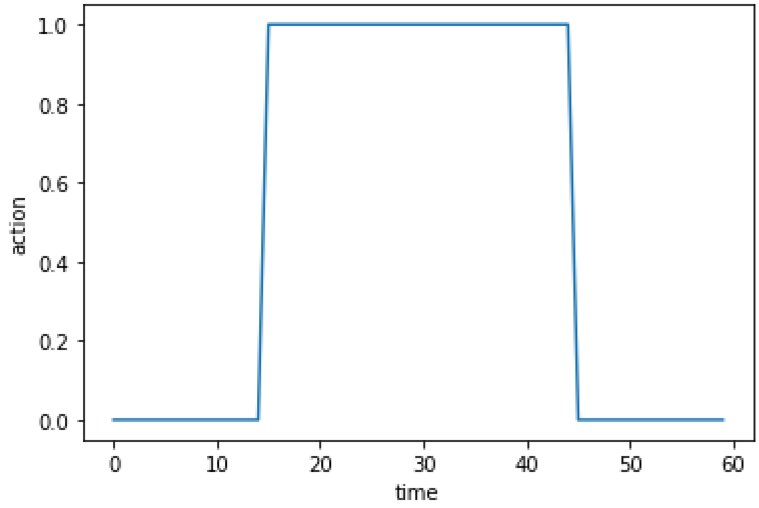
Original Graph{15 seconds: 0, 30 seconds:1, 15 seconds:0}



Week 2 (Deep LSTM network)



Week 3 (Ensemble Convolutional LSTM + Deep LSTM)



2.4 Future Improvements

2.4.1 I will be looking into more number of edge cases where the network does not perform well and will add that data to make it more useful for real time applications.

2.4.2 As I have mentioned before I have checked several networks to give better ensembled results, I will be adding and giving different datasets to the networks and try to get better results.

2.4.3 I have observed that there are several false positives that are being formed by the network I will try to tune more number of hyper parameters to decrease this number.