CSCE636 – FINAL PROJECT REPORT

NAME: Darpit Dave

UIN – 126004742

Abstract

As part of my course project for CSCE 636 – Deep Learning, the target action I worked on was detecting 'dizziness' through all submissions using a neural network model. This can be an important addition in a smart home. Detecting dizziness among people with medical conditions can be used to send out alerts to the concerned personnel to take necessary actions. In order to overcome the challenge of having limited data available I make use of data augmentation techniques. This helped in improving accuracy over previously trained model and also similar model when used without using augmented data. The model used in this submission has linearly stacked LSTM layers.

Data Description and Processing

The dataset consists of 160 self-made videos of roughly 5-6 seconds each. Out of which, 112 were used for training and 48 for validation, resulting in a 70-30 split. Labels for each video have been generated framewise. For example, the video starts with a person sitting who gets up and starts feeling dizzy. So, the initial frames would be marked as a negative class whereas only the target part would marked as positive labels.

The following steps were taken for processing the videos as input to the neural network model:

Step 1: OpenPose script was used to extract body landmarks from the videos for each frame. These were stored in a json format with x and y coordinates.

Step 2: The JSON files were further combined and stored as a .txt file for reading it as a matrix format.

```
def convert_data(start, end, output_json, output_json_key):
  '''
  start- Starting video number to be added to text file
  end - Last video number to be added
  output_json - Path of json files for videos
  output_json_key -  path of text file where the key-point values are to stored
  '''
  with open(output_json_key,"w") as x:
    for i in range(start, end+1):
          json_files = glob.glob(output_json + '/'+ str(i)+"/*")
          j=0
          for jfile in json_files:
            if j< 140:
              j = j + 1
              with open(jfile) as f:
                  data = json.load(f)
                  if data['people']:
                      kp= data['people'][0]['pose_keypoints_2d']
                      kp = [ str(v) for v in kp]
                      r =','.join(kp)
                      x.write('{}\n'.format(r))
```

Step 3: Once the files are read, separate training and validation data are created in a 70-30 split for training.

```
                                    ###DATA CONVERSION (only required once for generating text files)####
#INITIALIZATION:
#TRAINING:
output_json = '/content/drive/My Drive/CSCE636/JSON_Output'
output_json_key_train = "/content/drive/My Drive/CSCE636/Train_Val_Test/X_train.txt"
start = 1                    #training is from video 1 to video 70 --CHANGE
end = 70
Xtrain = convert_data(start, end, output_json, output_json_key_train)

#VALIDATION:
output_json = '/content/drive/My Drive/CSCE636/JSON_Output'
output_json_key_val = "/content/drive/My Drive/CSCE636/Train_Val_Test/X_val.txt"
start = 71                   #validation is from video 70 to video 100 --CHANGE
end = 100
Xtrain = convert_data(start, end, output_json, output_json_key_val)

                                          ##EXTENSION OF DATASET##
#ADDING MORE TRAINING DATA:
output_json = '/content/drive/My Drive/CSCE636/JSON_Output'
output_json_key_train = "/content/drive/My Drive/CSCE636/Train_Val_Test/X_train_new.txt"
start = 119                    #training is from video 119 to video 160 --CHANGE
end = 160
Xtrain_new = convert_data(start, end, output_json, output_json_key_train)

#ADDING MORE VALIDATION DATA:
output_json = '/content/drive/My Drive/CSCE636/JSON_Output'
output_json_key_val = "/content/drive/My Drive/CSCE636/Train_Val_Test/X_Val_new.txt"
start = 101                  #validation is from video 101 to video 118 --CHANGE
end = 118
Xval_new = convert_data(start, end, output_json, output_json_key_val)
```

Step 4: The read files are reshaped into appropriate tensor dimensions to be used as the final input for the model training.

```
#GENERATING TRAINING AND VALIDATION ARRAYS:

#FOR X-file:
train = "/content/drive/My Drive/CSCE636/Train_Val_Test/X_train.txt"    ##
X_train = read_data(train)

train2 = "/content/drive/My Drive/CSCE636/Train_Val_Test/X_train_new.txt"
X_train_new = read_data(train2)

X_train = np.concatenate((X_train, X_train_new), axis=0)
X_train = X_train.reshape(1568,10,50)

val = "/content/drive/My Drive/CSCE636/Train_Val_Test/X_val.txt"    ##CHAN
X_val = read_data(val)
val2 = "/content/drive/My Drive/CSCE636/Train_Val_Test/X_Val_new.txt"    #
X_val_new = read_data(val2)
X_val = np.concatenate((X_val, X_val_new), axis=0)
X_val = X_val.reshape(672,10,50)

#FOR Y-file:
Y_train,Y_val = generate_label()
Y_train = Y_train.reshape(1568,10,1)
Y_val = Y_val.reshape(672,10,1)

print('X_training = ', X_train.shape)
print('X_validation = ',X_val.shape)
print('Y_training = ', Y_train.shape)
print('Y_validation = ',Y_val.shape)
```

```
X_training =  (1568, 10, 50)
X_validation =  (672, 10, 50)
Y_training =  (1568, 10, 1)
Y_validation =  (672, 10, 1)
```

Step 5: To overcome the limitations of a smaller data, the training dataset was doubled up using data augmentation techniques. This allowed for better training and improved accuracy.

```
  ###DATA AUGMENTATION###
A = np.random.normal(0.0, 10.0, (1568,10,50) )
Xtrain_noise = np.add(Xtrain , A)
Xtrain = np.concatenate((Xtrain,Xtrain_noise), axis=0)
print('X-training shape after adding noise = ' , Xtrain.shape)

Ytrain = np.concatenate((Ytrain,Ytrain), axis=0)
print('Y-training shape after adding noise = ' , Ytrain.shape)
```

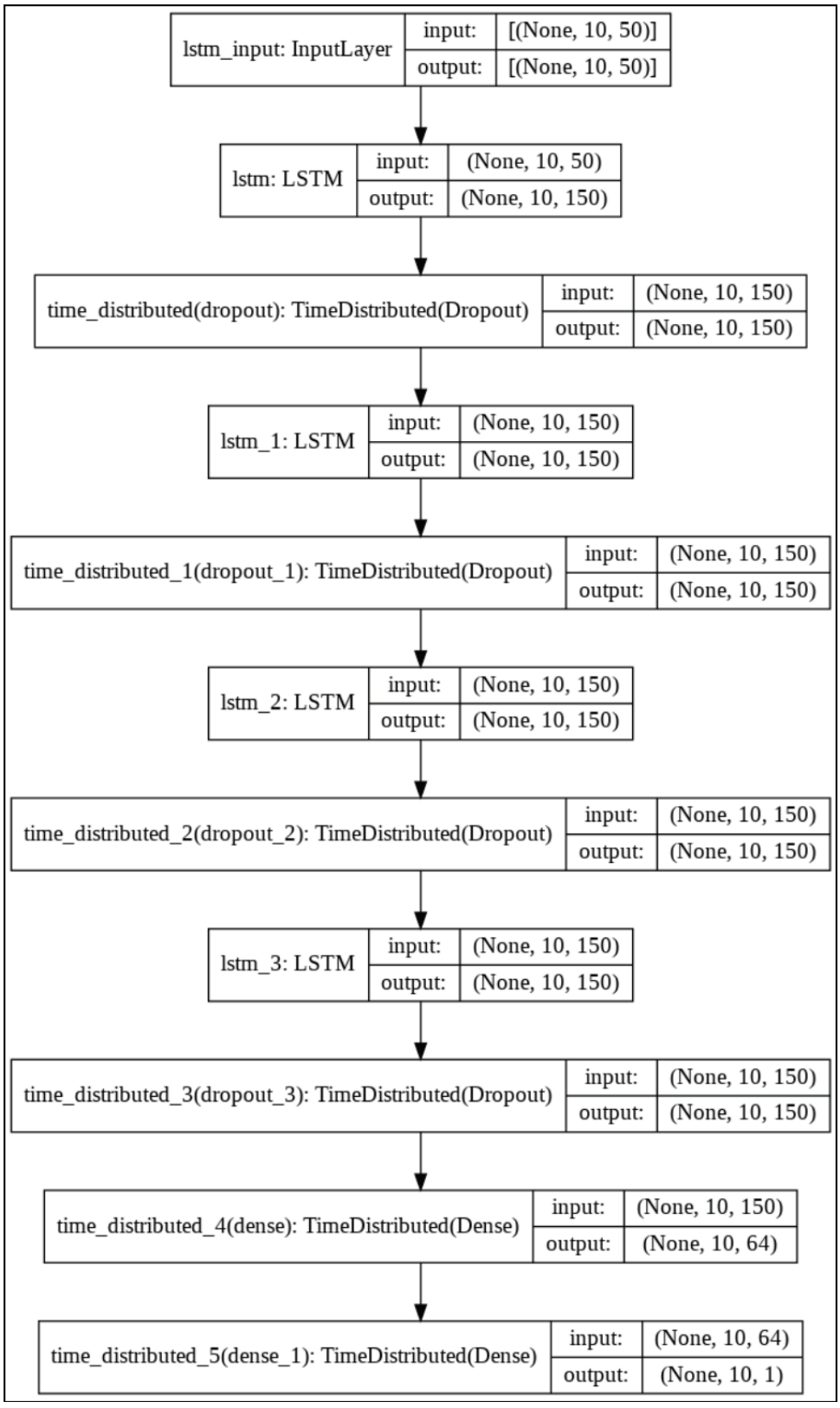
```
X-training shape after adding noise =  (3136, 10, 50)
Y-training shape after adding noise =  (3136, 10, 1)
```

Model Architecture and Design

Submission 2

The model used in this submission is a linear stack of sequential layers. It is a 9-layer unidirectional model. The hidden layer has 4 LSTM layers and 5 time-distributed layers. The output is also a time-distributed layer.

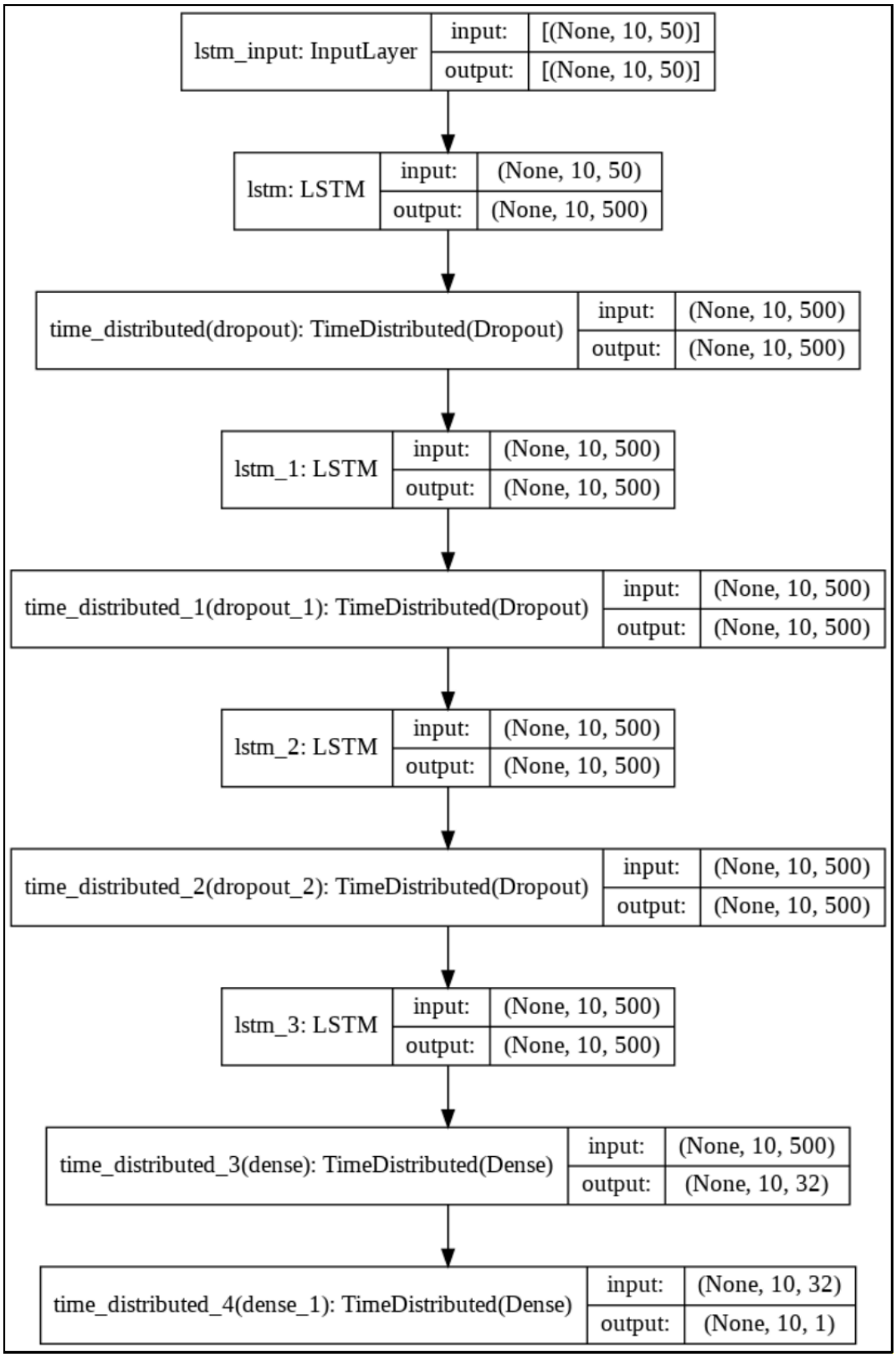
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 10, 500)           1102000
_________________________________________________________________
time_distributed (TimeDistri (None, 10, 500)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 10, 500)           2002000
_________________________________________________________________
time_distributed_1 (TimeDist (None, 10, 500)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 10, 500)           2002000
_________________________________________________________________
time_distributed_2 (TimeDist (None, 10, 500)           0
_________________________________________________________________
lstm_3 (LSTM)                (None, 10, 500)           2002000
_________________________________________________________________
time_distributed_3 (TimeDist (None, 10, 32)            16032
_________________________________________________________________
time_distributed_4 (TimeDist (None, 10, 1)             33
=================================================================
Total params: 7,124,065
Trainable params: 7,124,065
Non-trainable params: 0
_________________________________________________________________
```


lstm_input: InputLayer
input: [(None, 10, 50)]
output: [(None, 10, 50)]
lstm: LSTM
input: (None, 10, 50)
output: (None, 10, 150)
time_distributed(dropout): TimeDistributed(Dropout)
input: (None, 10, 150)
output: (None, 10, 150)
lstm_1: LSTM
input: (None, 10, 150)
output: (None, 10, 150)
time_distributed_1(dropout_1): TimeDistributed(Dropout)
input: (None, 10, 150)
output: (None, 10, 150)
lstm_2: LSTM
input: (None, 10, 150)
output: (None, 10, 150)
time_distributed_2(dropout_2): TimeDistributed(Dropout)
input: (None, 10, 150)
output: (None, 10, 150)
lstm_3: LSTM
input: (None, 10, 150)
output: (None, 10, 150)
time_distributed_3(dropout_3): TimeDistributed(Dropout)
input: (None, 10, 150)
output: (None, 10, 150)
time_distributed_4(dense): TimeDistributed(Dense)
input: (None, 10, 150)
output: (None, 10, 64)
time_distributed_5(dense_1): TimeDistributed(Dense)
input: (None, 10, 64)
output: (None, 10, 1)

Submission 3

The model used in this submission is a linear stack of sequential layers. It is a 7-layer model. The hidden layer has 2 LSTM layers and 4 time-distributed layers. The output is also a time-distributed layer.

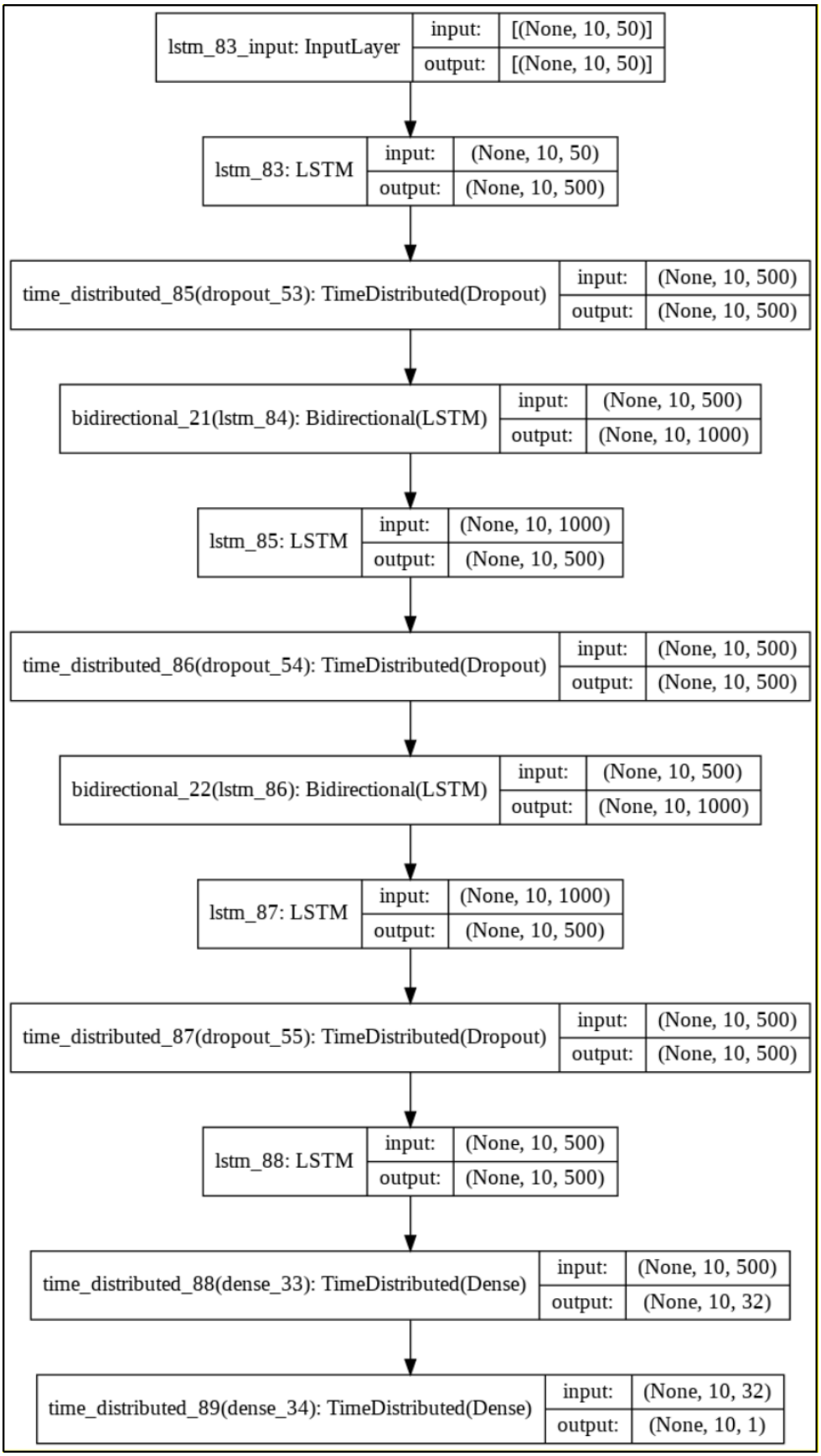
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 10, 500)           1102000
_________________________________________________________________
time_distributed (TimeDistri (None, 10, 500)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 10, 500)           2002000
_________________________________________________________________
time_distributed_1 (TimeDist (None, 10, 500)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 10, 500)           2002000
_________________________________________________________________
time_distributed_2 (TimeDist (None, 10, 500)           0
_________________________________________________________________
lstm_3 (LSTM)                (None, 10, 500)           2002000
_________________________________________________________________
time_distributed_3 (TimeDist (None, 10, 32)            16032
_________________________________________________________________
time_distributed_4 (TimeDist (None, 10, 1)             33
=================================================================
Total params: 7,124,065
Trainable params: 7,124,065
Non-trainable params: 0
_________________________________________________________________
None
```


lstm_input: InputLayer
input: [(None, 10, 50)]
output: [(None, 10, 50)]
lstm: LSTM
input: (None, 10, 50)
output: (None, 10, 500)
time_distributed(dropout): TimeDistributed(Dropout)
input: (None, 10, 500)
output: (None, 10, 500)
lstm_1: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
time_distributed_1(dropout_1): TimeDistributed(Dropout)
input: (None, 10, 500)
output: (None, 10, 500)
lstm_2: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
time_distributed_2(dropout_2): TimeDistributed(Dropout)
input: (None, 10, 500)
output: (None, 10, 500)
lstm_3: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
time_distributed_3(dense): TimeDistributed(Dense)
input: (None, 10, 500)
output: (None, 10, 32)
time_distributed_4(dense_1): TimeDistributed(Dense)
input: (None, 10, 32)
output: (None, 10, 1)

Submission 5

The model used in this submission is a linear stack of sequential layers. It is a 11-layer model. The hidden layer has 3 LSTM layers, 5 time-distributed layers and 2 bi-directional layers. The output is also a time-distributed layer.

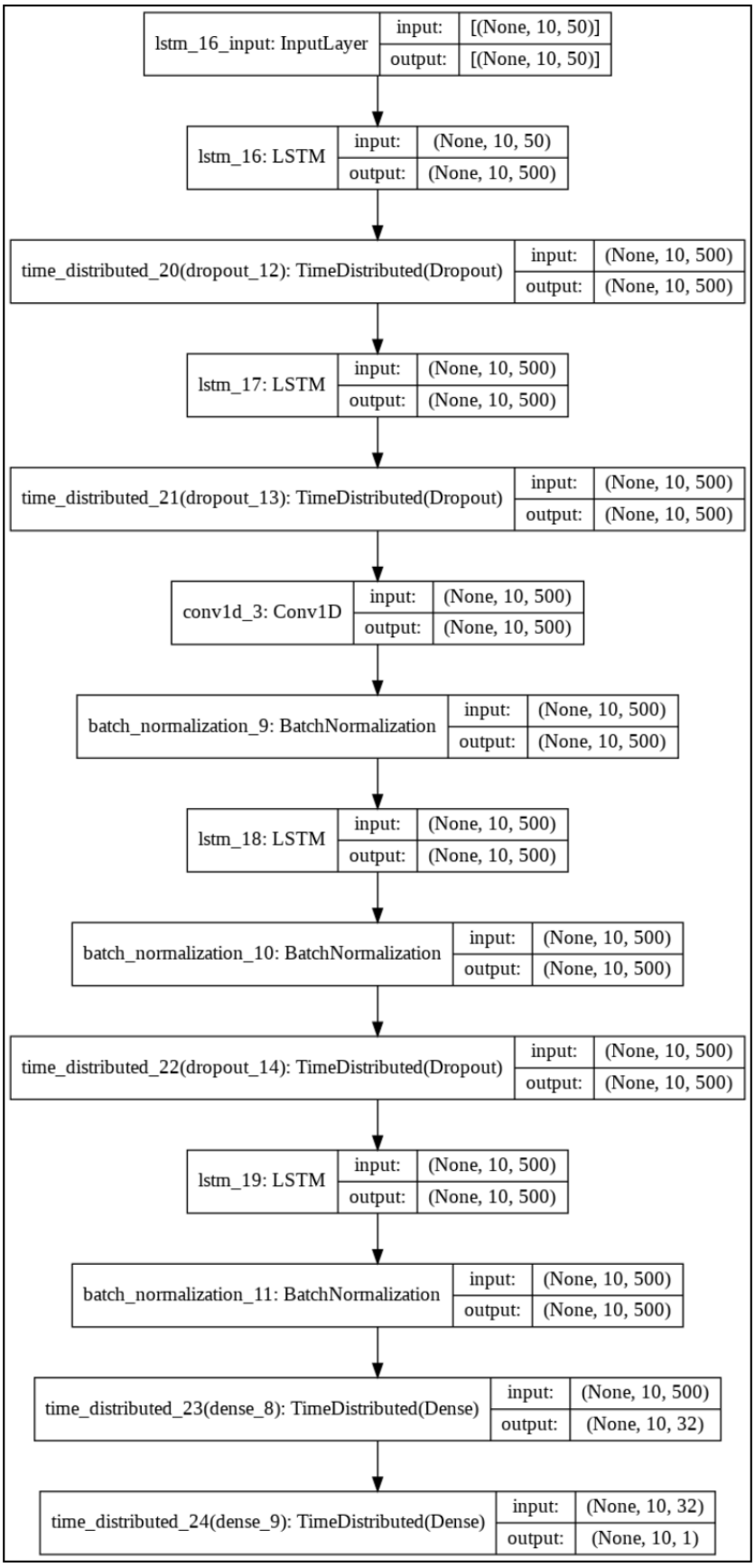
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 10, 500)           1102000
_________________________________________________________________
time_distributed_5 (TimeDist (None, 10, 500)           0
_________________________________________________________________
bidirectional (Bidirectional (None, 10, 1000)          4004000
_________________________________________________________________
lstm_6 (LSTM)                (None, 10, 500)           3002000
_________________________________________________________________
time_distributed_6 (TimeDist (None, 10, 500)           0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 10, 1000)          4004000
_________________________________________________________________
lstm_8 (LSTM)                (None, 10, 500)           3002000
_________________________________________________________________
time_distributed_7 (TimeDist (None, 10, 500)           0
_________________________________________________________________
lstm_9 (LSTM)                (None, 10, 500)           2002000
_________________________________________________________________
time_distributed_8 (TimeDist (None, 10, 32)            16032
_________________________________________________________________
time_distributed_9 (TimeDist (None, 10, 1)             33
=================================================================
Total params: 17,132,065
Trainable params: 17,132,065
Non-trainable params: 0
_________________________________________________________________
None
```


lstm_83_input: InputLayer
input: [(None, 10, 50)]
output: [(None, 10, 50)]
lstm_83: LSTM
input: (None, 10, 50)
output: (None, 10, 500)
time_distributed_85(dropout_53): TimeDistributed(Dropout)
input: (None, 10, 500)
output: (None, 10, 500)
bidirectional_21(lstm_84): Bidirectional(LSTM)
input: (None, 10, 500)
output: (None, 10, 1000)
lstm_85: LSTM
input: (None, 10, 1000)
output: (None, 10, 500)
time_distributed_86(dropout_54): TimeDistributed(Dropout)
input: (None, 10, 500)
output: (None, 10, 500)
bidirectional_22(lstm_86): Bidirectional(LSTM)
input: (None, 10, 500)
output: (None, 10, 1000)
lstm_87: LSTM
input: (None, 10, 1000)
output: (None, 10, 500)
time_distributed_87(dropout_55): TimeDistributed(Dropout)
input: (None, 10, 500)
output: (None, 10, 500)
lstm_88: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
time_distributed_88(dense_33): TimeDistributed(Dense)
input: (None, 10, 500)
output: (None, 10, 32)
time_distributed_89(dense_34): TimeDistributed(Dense)
input: (None, 10, 32)
output: (None, 10, 1)

Submisison 6

The model used in this submission is a linear stack of sequential layers. It is a 12-layer model. The hidden layer has 3 LSTM layers, 3 time-distributed layers and 1 1D convolutional layer. The model utilizes 3 batchnormalization layers. The output is also a time-distributed layer.

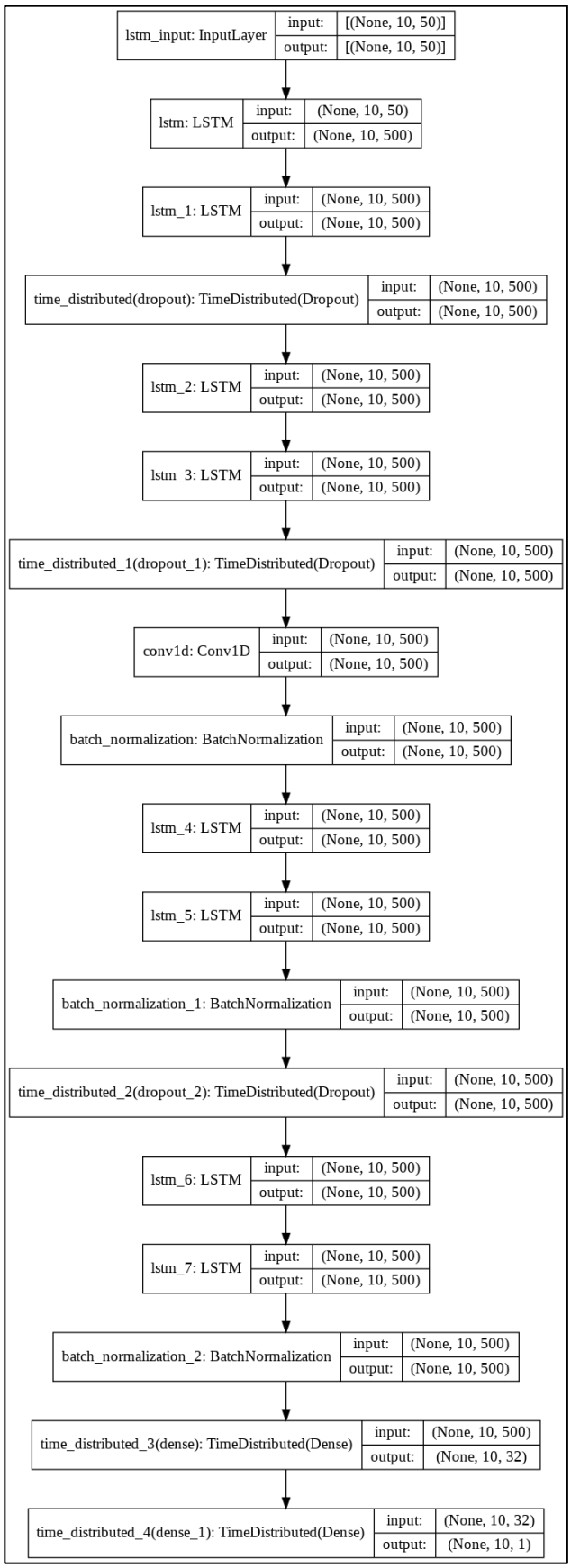
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_10 (LSTM)               (None, 10, 500)           1102000
_________________________________________________________________
time_distributed_10 (TimeDis (None, 10, 500)           0
_________________________________________________________________
lstm_11 (LSTM)               (None, 10, 500)           2002000
_________________________________________________________________
time_distributed_11 (TimeDis (None, 10, 500)           0
_________________________________________________________________
conv1d (Conv1D)              (None, 10, 500)           500500
_________________________________________________________________
batch_normalization (BatchNo (None, 10, 500)           2000
_________________________________________________________________
lstm_12 (LSTM)               (None, 10, 500)           2002000
_________________________________________________________________
batch_normalization_1 (Batch (None, 10, 500)           2000
_________________________________________________________________
time_distributed_12 (TimeDis (None, 10, 500)           0
_________________________________________________________________
lstm_13 (LSTM)               (None, 10, 500)           2002000
_________________________________________________________________
batch_normalization_2 (Batch (None, 10, 500)           2000
_________________________________________________________________
time_distributed_13 (TimeDis (None, 10, 32)            16032
_________________________________________________________________
time_distributed_14 (TimeDis (None, 10, 1)             33
=================================================================
Total params: 7,630,565
Trainable params: 7,627,565
Non-trainable params: 3,000
_________________________________________________________________
```


lstm_16_input: InputLayer
input: [(None, 10, 50)]
output: [(None, 10, 50)]
lstm_16: LSTM
input: (None, 10, 50)
output: (None, 10, 500)
time_distributed_20(dropout_12): TimeDistributed(Dropout)
input: (None, 10, 500)
output: (None, 10, 500)
lstm_17: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
time_distributed_21(dropout_13): TimeDistributed(Dropout)
input: (None, 10, 500)
output: (None, 10, 500)
conv1d_3: Conv1D
input: (None, 10, 500)
output: (None, 10, 500)
batch_normalization_9: BatchNormalization
input: (None, 10, 500)
output: (None, 10, 500)
lstm_18: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
batch_normalization_10: BatchNormalization
input: (None, 10, 500)
output: (None, 10, 500)
time_distributed_22(dropout_14): TimeDistributed(Dropout)
input: (None, 10, 500)
output: (None, 10, 500)
lstm_19: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
batch_normalization_11: BatchNormalization
input: (None, 10, 500)
output: (None, 10, 500)
time_distributed_23(dense_8): TimeDistributed(Dense)
input: (None, 10, 500)
output: (None, 10, 32)
time_distributed_24(dense_9): TimeDistributed(Dense)
input: (None, 10, 32)
output: (None, 10, 1)

Submission 8

The model used in this submission is a linear stack of sequential layers. It is a 14-layer model. The hidden layer has 8 LSTM layers, 5 time-distributed layers and 1 1D convolutional layer. The model utilizes 3 batchnormalization layers. The output is also a time-distributed layer.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_14 (LSTM)               (None, 10, 500)           1102000
_________________________________________________________________
lstm_15 (LSTM)               (None, 10, 500)           2002000
_________________________________________________________________
time_distributed_15 (TimeDis (None, 10, 500)           0
_________________________________________________________________
lstm_16 (LSTM)               (None, 10, 500)           2002000
_________________________________________________________________
lstm_17 (LSTM)               (None, 10, 500)           2002000
_________________________________________________________________
time_distributed_16 (TimeDis (None, 10, 500)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 10, 500)           500500
_________________________________________________________________
batch_normalization_3 (Batch (None, 10, 500)           2000
_________________________________________________________________
lstm_18 (LSTM)               (None, 10, 500)           2002000
_________________________________________________________________
lstm_19 (LSTM)               (None, 10, 500)           2002000
_________________________________________________________________
batch_normalization_4 (Batch (None, 10, 500)           2000
_________________________________________________________________
time_distributed_17 (TimeDis (None, 10, 500)           0
_________________________________________________________________
lstm_20 (LSTM)               (None, 10, 500)           2002000
_________________________________________________________________
lstm_21 (LSTM)               (None, 10, 500)           2002000
_________________________________________________________________
batch_normalization_5 (Batch (None, 10, 500)           2000
_________________________________________________________________
time_distributed_18 (TimeDis (None, 10, 32)            16032
_________________________________________________________________
time_distributed_19 (TimeDis (None, 10, 1)             33
=================================================================
Total params: 15,638,565
```


lstm_input: InputLayer
input: [(None, 10, 50)]
output: [(None, 10, 50)]
lstm: LSTM
input: (None, 10, 50)
output: (None, 10, 500)
lstm_1: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
time_distributed(dropout): TimeDistributed(Dropout)
input: (None, 10, 500)
output: (None, 10, 500)
lstm_2: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
lstm_3: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
time_distributed_1(dropout_1): TimeDistributed(Dropout)
input: (None, 10, 500)
output: (None, 10, 500)
conv1d: Conv1D
input: (None, 10, 500)
output: (None, 10, 500)
batch_normalization: BatchNormalization
input: (None, 10, 500)
output: (None, 10, 500)
lstm_4: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
lstm_5: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
batch_normalization_1: BatchNormalization
input: (None, 10, 500)
output: (None, 10, 500)
time_distributed_2(dropout_2): TimeDistributed(Dropout)
input: (None, 10, 500)
output: (None, 10, 500)
lstm_6: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
lstm_7: LSTM
input: (None, 10, 500)
output: (None, 10, 500)
batch_normalization_2: BatchNormalization
input: (None, 10, 500)
output: (None, 10, 500)
time_distributed_3(dense): TimeDistributed(Dense)
input: (None, 10, 500)
output: (None, 10, 32)
time_distributed_4(dense_1): TimeDistributed(Dense)
input: (None, 10, 32)
output: (None, 10, 1)

Submission 10

For the final submission, I worked on using an ensemble model from the following:

(1) Submission 3 (Model A)
(2) Submission 5 (Model B)
(3) Submission 6 (Model C)
(4) Submission 8 (Model D)

The final prediction would be done by giving equal weights (0.25) to predictions from each model and adding it up.

HYPERPARAMETERS

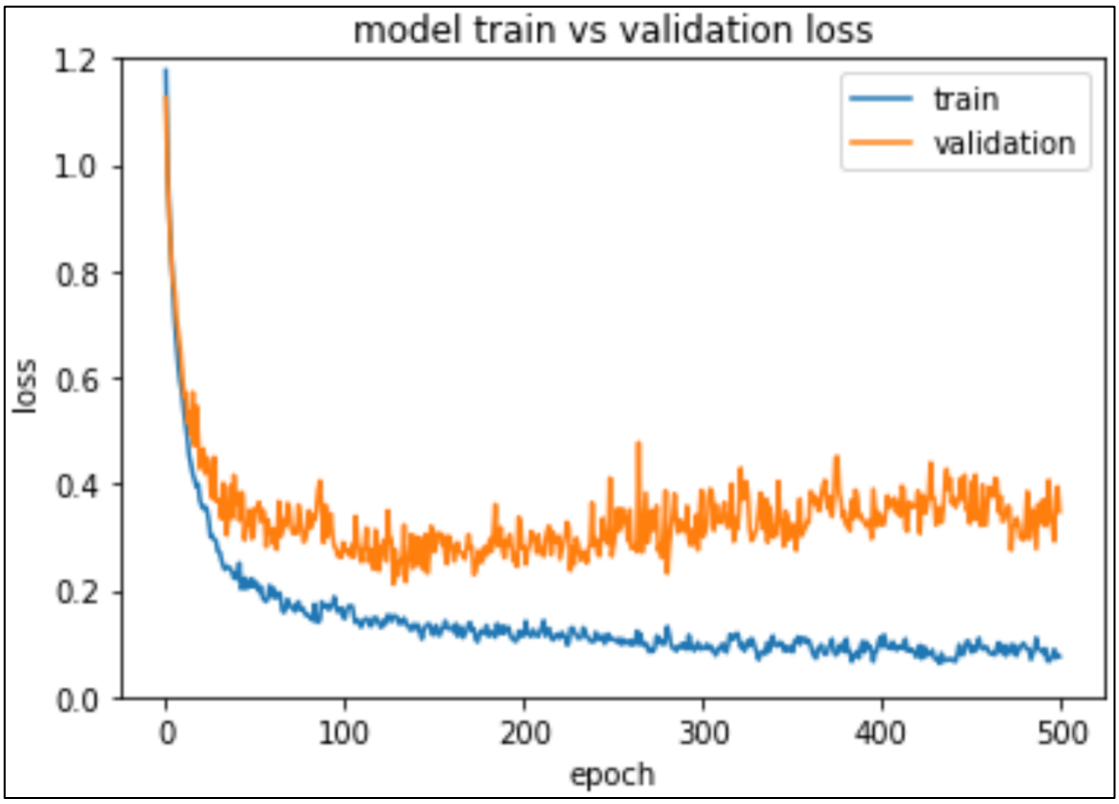
The following hyperparameters were tuned: units, batch_size, epochs, learning rate. The below table summarizes the different values used and the optimal values found.

Hyperparamter	Values Tried	Optimal
Batch Size	50, 100, 200, 300, 400, 500	100
Units	100, 150, 200, 250, 333, 350, 400, 500, 750, 1000	500
Learning rate	0.001, 0.0001, 0.01, 0.00001	0.0001

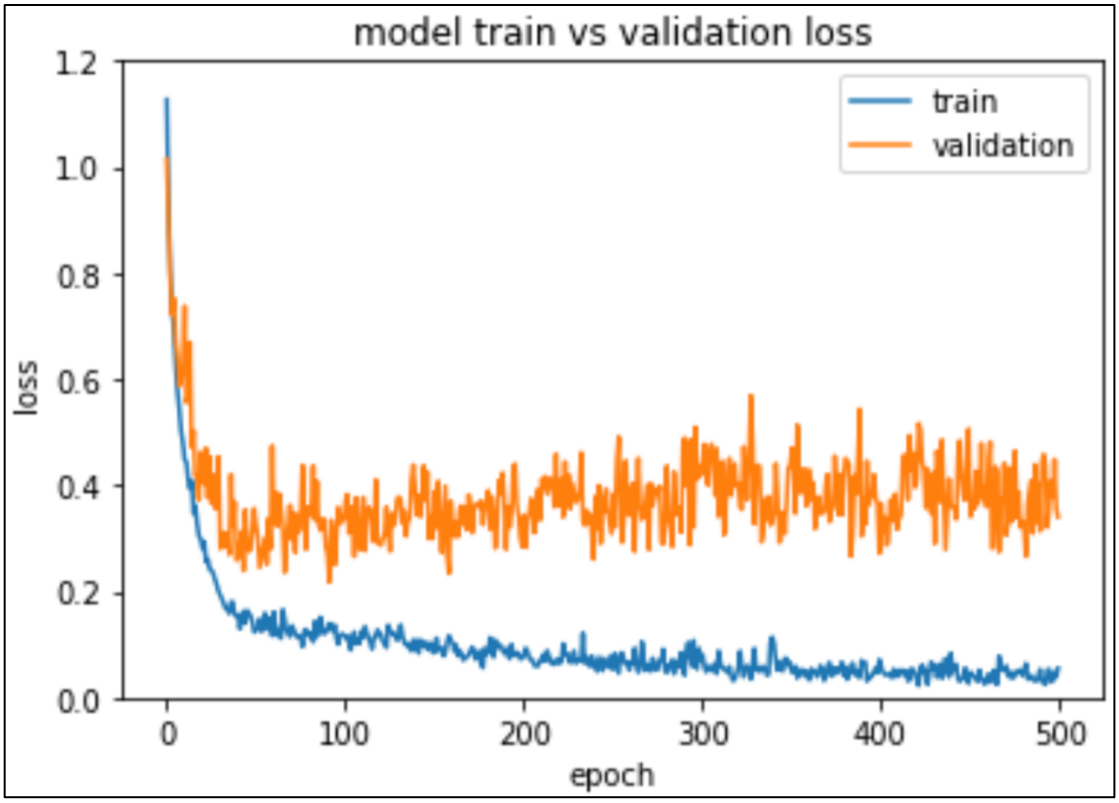
The above 2 hyperparameters were manually tried for various combinations to select the best output. For choosing the optimal epochs, keras features like early stopping and model checkpoint were utilized with 'validation_loss' to be the monitoring metric.

Training and Validation Performance

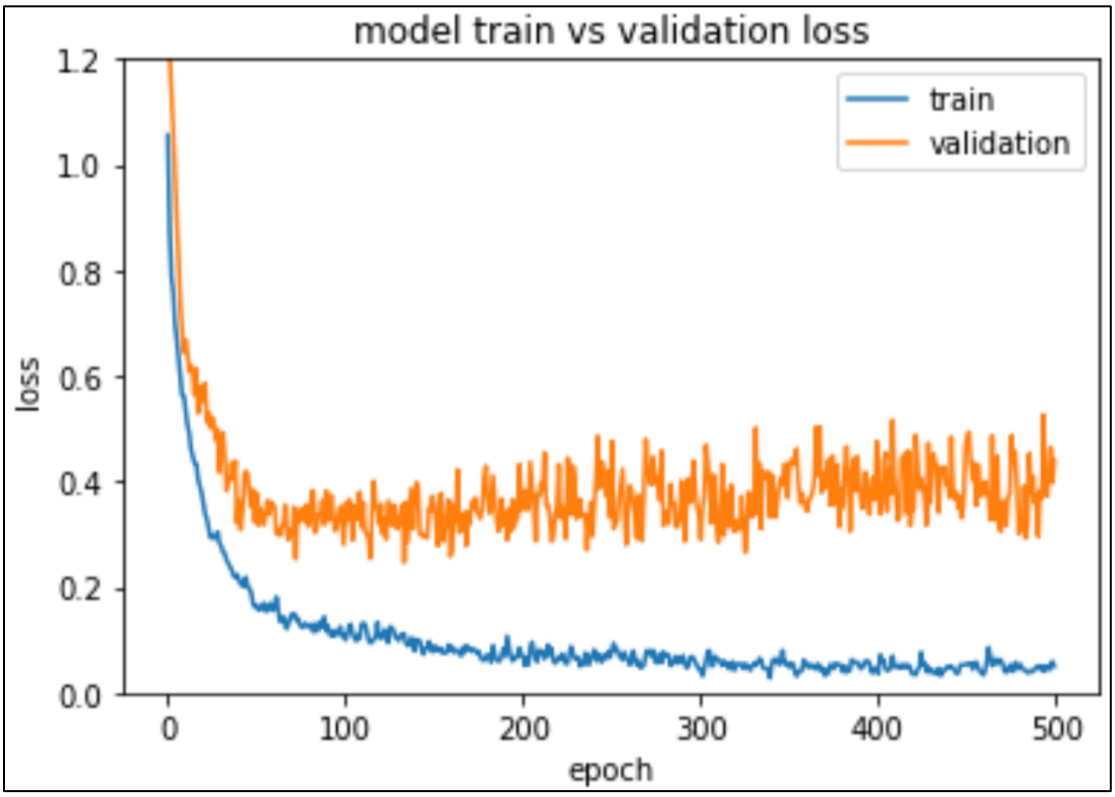
1. Validation Loss: Model A



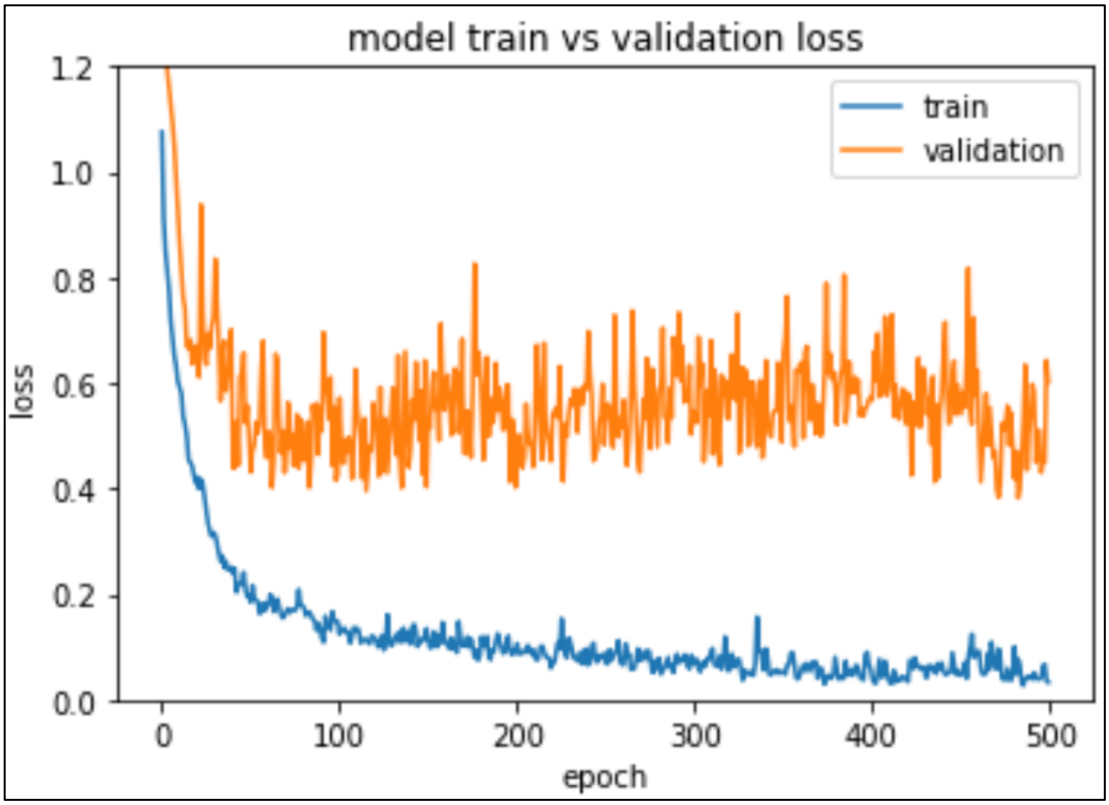
2. Validation Loss: Model B



3. Validation Loss: Model C



4. Validation Loss: Model D



Prediction Results

The ensemble outperformed all individual models on the validation dataset. However, when checking on the external test videos, Model for submission 6 was the best performing model. I am submitting it for testing on the submitted metadata as my final model.

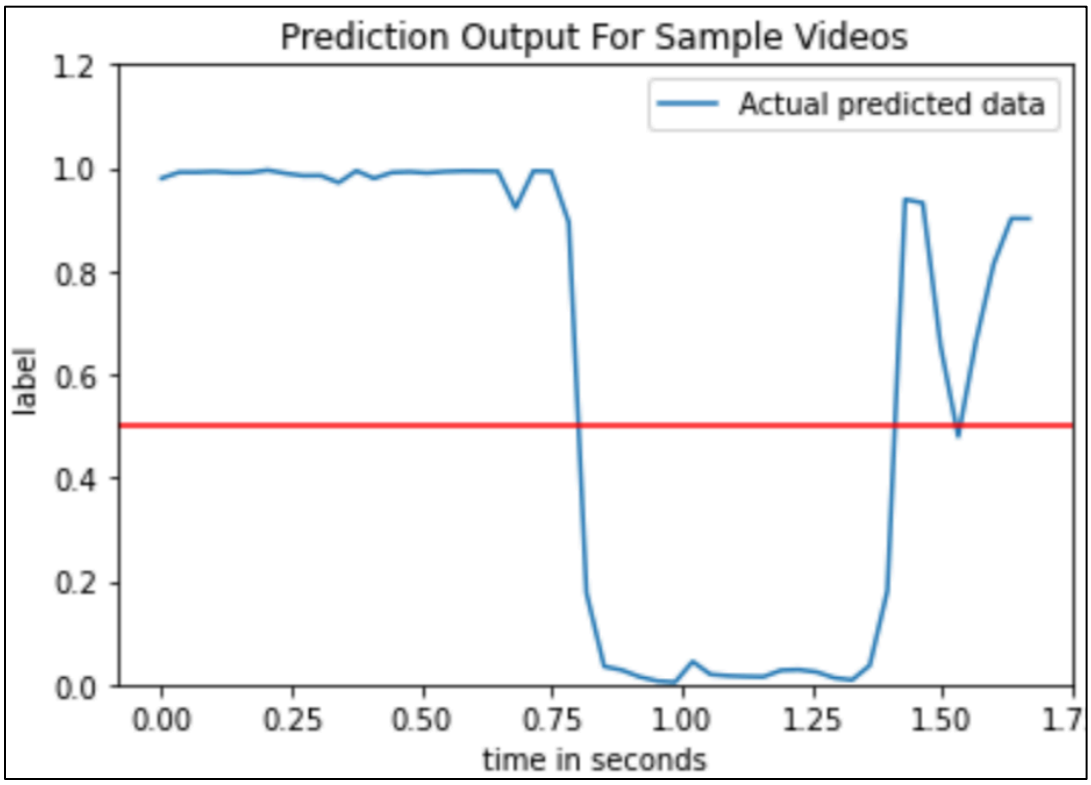
```
import sklearn
from sklearn.metrics import accuracy_score
print("Model 1 prediction accuracy:", sklearn.metrics.accuracy_score(predicted_class1, true))
print("Model 2 prediction accuracy:", sklearn.metrics.accuracy_score(predicted_class2, true))
print("Model 3 prediction accuracy:", sklearn.metrics.accuracy_score(predicted_class3, true))
print("Model 4 prediction accuracy:", sklearn.metrics.accuracy_score(predicted_class4, true))
print("Ensemble prediction accuracy:", sklearn.metrics.accuracy_score(total_predicted, true))
```

```
Model 1 prediction accuracy: 0.9151785714285714
Model 2 prediction accuracy: 0.9105654761904762
Model 3 prediction accuracy: 0.8907738095238096
Model 4 prediction accuracy: 0.8967261904761905
Ensemble prediction accuracy: 0.9311011904761904
```

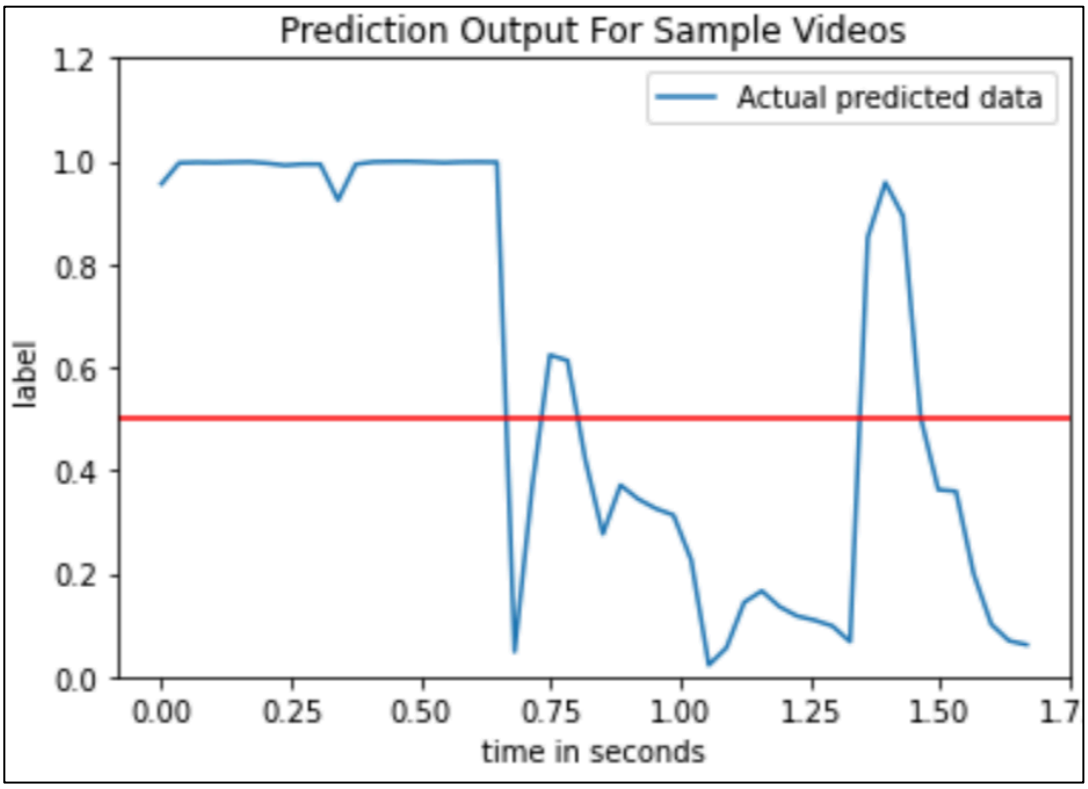
The following are a few sample results for positive (contains target action) and negative classes (does not have target action).

Positive Class

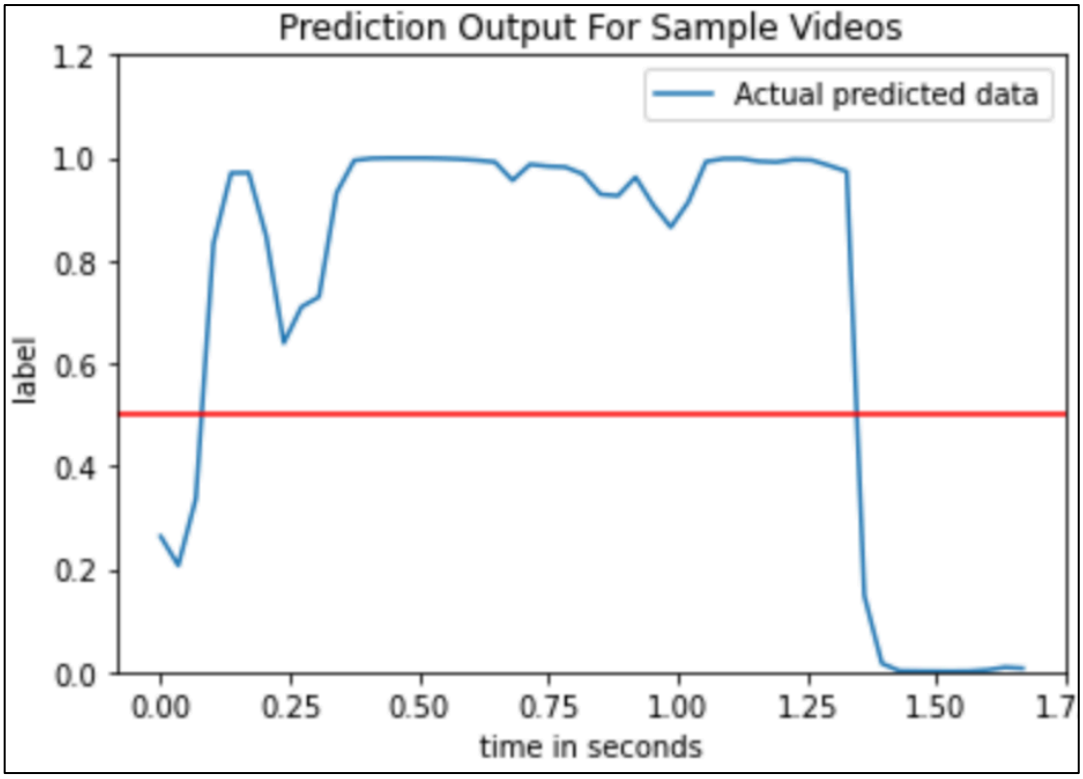
1. **Positive Sample – 1 (Video #: 1)**



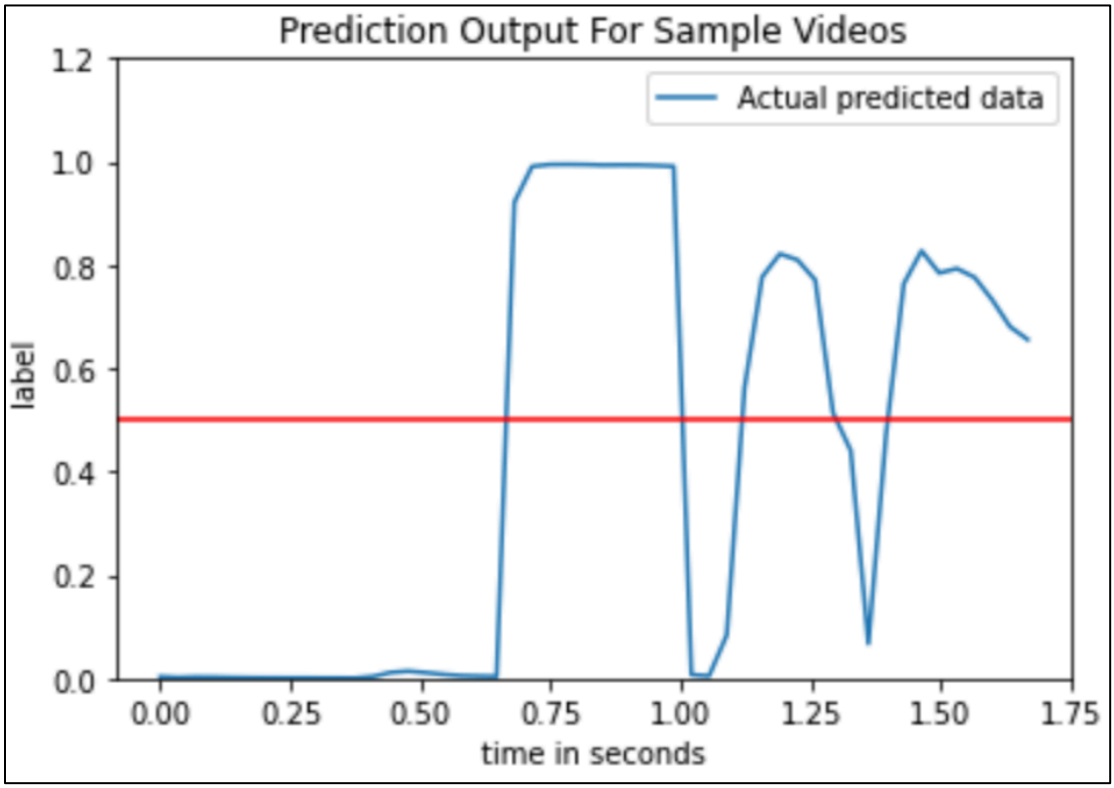
2. Positive Sample – 2 (Video #: 2)



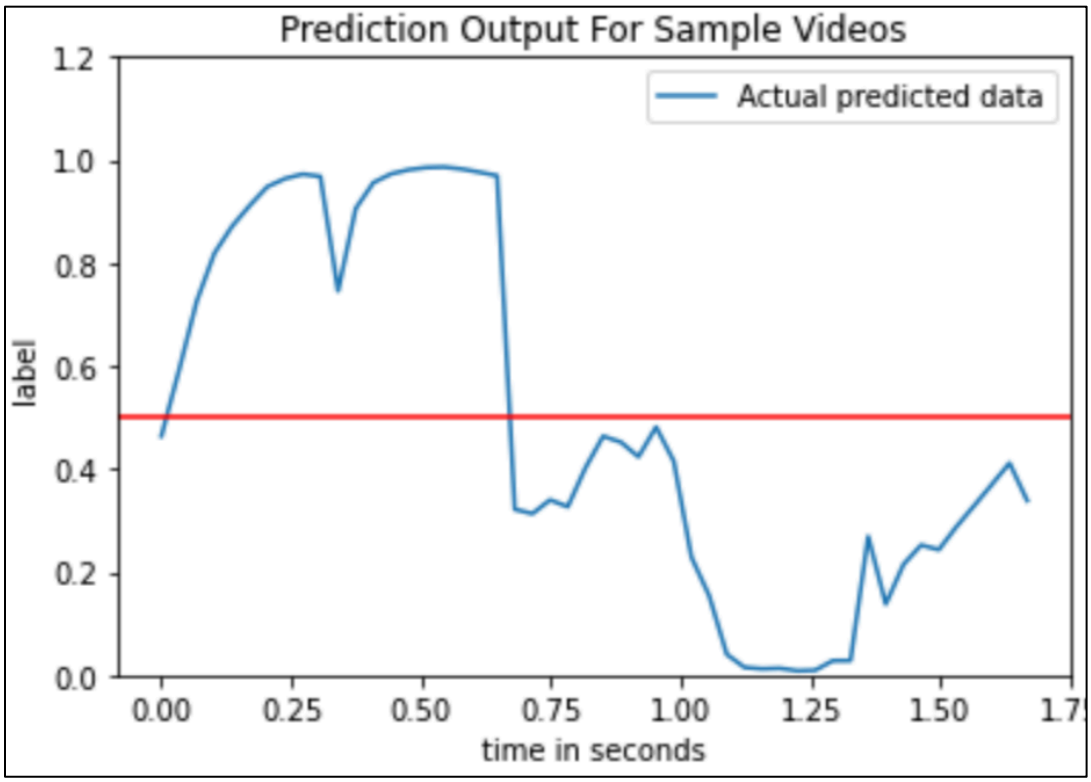
3. Positive Sample - 3 (Video #: 3)



4. **Positive Sample - 4 (Video #:4)**



5. **Positive Sample - 5 (Video #: 5)**



IMPORTANT NOTE:

As part of final submission, I worked on transfer learning (VGG, ResNET) as well as utilizing multi-task learning via the SNORKEL approach. I read through the links provided on the course website and read a few papers on it since I also am trying to implement the same in my research. However, for the purpose of the project, I was unsuccessful in completing a working script and needed to change at the end moment since I also wanted to complete my submission before the deadline.

INSTRUCTION ON HOW TO RUN THE PREDICTION MODEL:

1. Download CSCE636_Project_Submission10_Prediction_final.ipynb from shared github repository: https://github.com/darpitdavetamu/CSCE636-DeepLearning/tree/main/Submission10
2. Upload the file to google colab.
3. JSON files and text file required to run the code can also be accessed at https://drive.google.com/drive/folders/1Ua297X--JqXMMfB33V4TeSG0rxCbzRZP?usp=sharing
4. The path written in the code is same as the google drive link shared.

LINK TO DEMO VIDEO:

https://www.youtube.com/watch?v=cK1aSOpTAn4