Convolutional Neural Network (CNN) for EEG 4-class Motor Classification

Jason Tay
UID: 805599377
jtay20@g.ucla.edu

Henry Wang
UID: 405691801
henrywang3@g.ucla.edu

Lily Zhou
UID: 505750593
lzhou0714@g.ucla.edu

Abstract

This paper reports the findings on various machine learning models performing multiclass classification for electroencephalography (EEG) data collected from a non-invasive electrode with four classes, corresponding to four different imaginary motor tasks of these body parts: left hand, right hand, feet, and tongue. In this project, a Convolutional Neural Network (CNN) was implemented to get a baseline performance followed by testing various combinations implementing self-attention, Long Short-Term Memory (LSTM), and Multi-head Attention (MHA) with the base CNN model. This resulted in 5 models: CNN, CNN + LSTM, CNN + self-attention, CNN + MHA, CNN + LSTM + SA. All models were trained with the entire dataset containing all subjects. The best testing accuracy achieved in this project was 70.4% with CNN+MHA.

1. Introduction

Convolutional Neural Networks (CNNs) have emerged as a popular method for decoding raw Electroencephalography (EEG) signals in motor imagery classification tasks. EEG signals are complex and noisy, making it challenging to extract meaningful Although other methods, like support vector machine (SVM) and multi-layer perceptron (MLP) can also be used, CNNs are better suited for a variety of reasons. EEG signals contain a high degree of noise, so the tolerance of the CNN model to learn from different parts of input data makes it more robust. In addition, CNNs can learn features automatically through filters, whereas with SVM and MLP, some manual feature engineering may be necessary. CNNs can learn local features from the input data through their convolutional layers, capture spatial information using 2D convolutions, and capture temporal information using 1D convolutions or recurrent layers. As a result, they can handle high-dimensional data, learn meaningful features automatically, and take advantage of both spatial and temporal information to produce accurate and efficient classifications in motor imagery classification tasks.

Regardless of the method, pre-processing is necessary in order to overcome various issues like low signal-to-noise ratios. Popular techniques include data trimming, max-pooling, averaging with noise, cropping, and spatial transforms, of which we implemented the first 3 after some experimentation with spatial transforms did not show improvement.

In this project, we implemented 5 CNN-based models using a hybrid of LSTM, self-attention, and MHA with the base CNN model. In particular, we tested the base CNN model, CNN with LSTM, CNN with self-attention, CNN with MHA, and lastly CNN with LSTM and MHA.

1.1 CNN Model

For our study, we implemented a CNN model for motor imagery classification tasks. The model consisted of four convolutional layers, with each convolutional block containing a convolutional layer followed by max pooling, batch normalization, and dropout. This configuration is standard for CNN models, and it provided a strong foundation for our future models when we added more complex features. During our experimentation, we discovered that this model had an exceptional ability to learn complex features from the input data, resulting in high accuracy and robustness. However, we also noted that the model required extensive tuning of hyperparameters to achieve optimal performance.

1.2 Transformer and Self-Attention

A self-attention Transformer maps the EEG signal embeddings using an embedding layer, which is followed by self-attention and feedforward layers in a series. The self-attention layer helps weigh the importance of signal parts and the feedforward layer applies non-linear transformations for learning complex relationships between the input and output. The output of the CNN model is then fed to the self-attention Transformer. The self-attention mechanism captures the complex relationships between different time steps and helps to learn relevant parts of the input signal, which is particularly useful for EEG classification tasks where the signal can be highly complex and non-stationary, and where long-range dependencies and correlations between different time steps are important.

1.3 Multi-head Attention (MHA)

Multi-head attention extends self-attention to enable a model to attend to multiple, separate parts of the input sequence at the same time using several attention mechanisms, or heads. It transforms the input sequence into multiple queries, keys, and values, each of which is independently processed through the attention mechanism to calculate its own attention weights. The final set of weights is produced by combining the attention weights of all the heads, which are used to compute the weighted sum of the values to model complex relationships between different parts of the input sequence. This property is particularly useful for

EEG classification tasks where long-range dependencies and correlations between different time steps are important.

This was built upon the baseline CNN model with an added MultiHeadAttention layer (num_heads = 8) followed by a LayerNormalizationlayer (similar to batch normalization but per example instead of per batch) and finally a GlobalAveragePooling2D layer. We experimented with positional encoding but found that it did not result in better performance. One possible explanation is that the relative position of the input tokens may already be implicitly captured by the model's architecture and the data preprocessing work, so the additional complexity may even be detrimental to performance.

1.4 CNN with LSTM

When a CNN and LSTM are used together, the CNN is used to extract spatial features from the EEG signals, while the LSTM is used to capture the sequential dependencies in the data. This model in theory can effectively learn both spatial and temporal patterns in the data. The output of the last convolutional block is flattened and fed into a fully connected layer with 120 units, which is followed by a reshape operation and an LSTM layer with 10 units. Finally, a dense layer with a softmax activation function is added as the output layer with 4 classes, which are the possible EEG states.

2. Results

We trained on EEG data from the dataset provided in "BCI Competition IV dataset 2a" (http://www.bbci.de/competition/iv/desc_2a.pdf). The dataset consists of EEG data from nine subjects, each performing four different motor imagery tasks (left hand, right hand, foot, and tongue) while wearing an EEG cap. Each of the 9 subjects has 22 channels associated with them, with which data was collected over 1000 timestamps with band-pass filtering between 0.5 Hz and 100 Hz and notch filtering at 50 Hz for noise suppression.

We evaluate our model across all subjects using the training data. As shown in Table 1, we found that the CNN with MHA, CNN with SA and LSTM, and the base CNN had the best testing accuracies of 70.71%, 70.43%, and 70.09%, respectively. The remaining models underperformed as compared to the base CNN model. The CNN with LSTM and the CNN with self-attention had testing accuracies of 67.83% and 68.06%. We also observed a high correlation between the training and validation/testing accuracy, suggesting that our model had a high degree of complexity and avoided overfitting while also maintaining a fast training speed (~2-3 seconds per epoch, batch size of 32). Each model had varying converging speeds, which was often due to the number of layers and parameters. The base CNN model only took 50 epochs, but the other models took significantly longer to train. CNN+LSTM, CNN+MHA, CNN+SA, and CNN+LSTM+SA took 183, 218, 300, and 190 epochs to train, respectively.

We observed that over time, the difference between the training and validation accuracy increases as the model begins to overfit to the training data and the decrease in validation loss becomes less with each successive epoch. For example, this phenomenon can be seen most clearly in Figure 3 after epoch 60; The train and validation accuracy curves begin to diverge with a greater difference between them after each subsequent epoch.

3. Discussion

Although CNN was the simplest model, it was still able to effectively learn spatial patterns in the EEG data and achieve a testing accuracy of 70.09%. CNN+LSTM was able to capture both spatial and temporal patterns and should be more versatile than just CNN alone, it had a lackluster performance. It is possible that the input features were already being captured well in the CNN since EEG data can be interpreted as image data and would thus be suitable for a CNN. With CNN+MHA, the multi-head attention mechanism allowed it to select the most relevant parts of the input data. This model performed better than both CNN and CNN+LSTM, however, it was more complex and computationally expensive as well.

One possibility for these results was that the input data was not balanced well. Although the training set was generally uniformly distributed, there was bias in the test set towards the ‘Right Hand’ class with less distribution for the remaining three classes. It is possible that this imbalance resulted in not only the gap between training and validation accuracy in the 70% range but also a bias away from the CNN+LSTM model. Further analysis could be done via data augmentation and increasing/changing the test set to determine whether this is the case.

3.1 Improving Model Performance

After data preprocessing and input standardization were completed, we worked on improving the performance of each of our models. For example, the CNN + self-attention model was able to reach ~60% test accuracy without many hyperparameter adjustments. Increasing the number of heads and adjusting other factors such as dropout allowed us to reach up to ~68% test accuracy. We believe that dropout was able to significantly improve the models’ performance by

preventing overfitting and encouraging the LSTM models to learn more generalized features. The best validation accuracy we reached was 74%, and was not done by performing PCA or data augmentation, but instead by modifying the activation functions in the convolutional layers, using a combination of GELU and SWIFT instead of ReLU. We believe this may have been due to the following: GELU and SWISH produce smoother gradients, reducing the risk of exploding gradients which we found had occurred with ReLU under certain hyperparameters; they improve the expressiveness of the model, and in particular, GELU and SWISH were found to perform well specifically on image classification, which has many similarities to the EEG data.

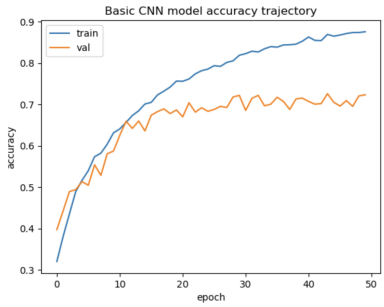


Figure 1. Test accuracy for base CNN model

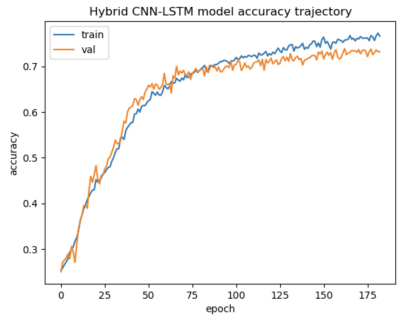


Figure 2. Test accuracy for hybrid CNN-LSTM model

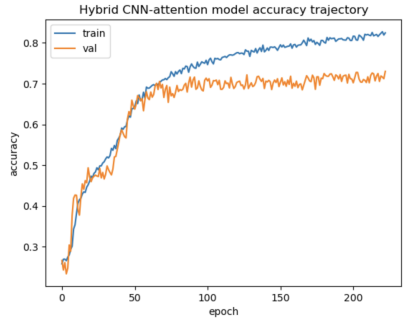


Figure 3. Test accuracy for hybrid CNN + self-attention model

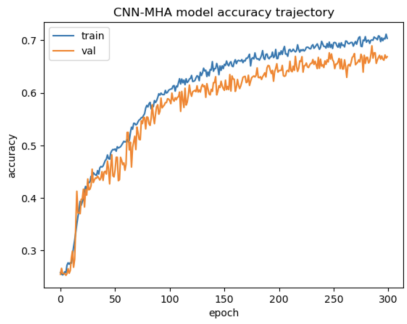


Figure 4. Test accuracy for hybrid CNN-MHA model

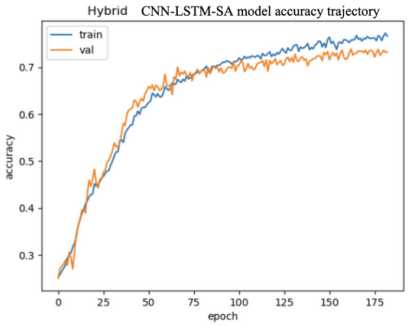
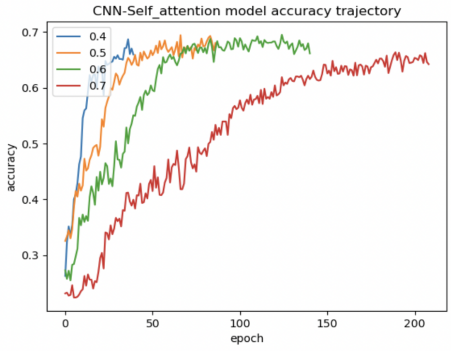


Figure 5. Test accuracy for hybrid CNN-LSTM-SA model



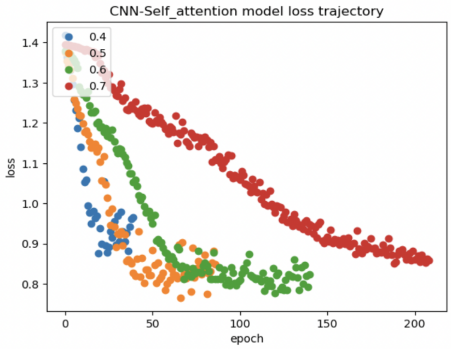


Figure 6. Self-attention model accuracy and model loss trajectory for different dropout values

4. Testing Results

	Validation Accuracy	Test Accuracy
CNN	0.7233	0.7009029
CNN+LSTM	0.6830	0.6783296
CNN+SA	0.7102	0.6805869
CNN+MHA	0.7342	0.7071106
CNN + LSTM + SA	0.7320	0.7042890

Table 1. Validation and test accuracy for CNN, CNN+LSTM, CNN+SA, CNN+MHA, CNN+LSTM+SA

Dropout	**Test Accuracy**
0.4	0.66252821
0.5	0.680033597
0.6	0.68171554804
0.7	0.6551918983

Table 2. Test accuracies for CNN with Self-attention with varying values of dropout

5. Appendix

The below sections describe the details of the architecture for each model and provide further details on the training process.

5.1 Training details

We split 20% of the training dataset to be used as the validation dataset with the remaining 80% as the training data. All models used the Adam optimizer with a learning rate set to 1e-3. Prior to training, all data was prepared in the same process: trimming, max-pooling, averaging, adding noise, and subsampling. We also employed early stopping with a patience of 20.

5.2 CNN

The CNN model consists of 4 convolution blocks with each one performing Conv2D, MaxPooling2D, BatchNormalization, and Dropout. The first filter had a value of 15-30, the second was 40-60, the third was from 80-120, and the final one was 180-220. All layers used ELU as their activation function. Finally, on the output layer, we flattened the input, and output the FC layer with a softmax activation. A dropout value of 0.5 was used for each layer.

5.3 CNN+LSTM

The CNN+LSTM model used a similar structure to the CNN model above, but dropout values of 0.6 were used instead. Furthermore, the first two convolution blocks used Swish as their activation functions and the remaining two convolution blocks used GELU. After the fourth layer, we applied a flattening operation to the output of the CNN block, added an FC layer with 100 units, and then reshaped the output of the FC layer to (120, 1). Finally, this was fed into the LSTM with a dropout value of 10, a recurrent dropout value of 0.1, and a size of 10). The output of the LSTM is then connected to the softmax activation.

5.4 CNN+Self-attention

For the CNN+Self-attention model, we fed the output of the fourth CNN block to the self-attention block. The self-attention block used GELU as its activation function and applied a GlobalAveragePooling1D before connecting to the output layer.

5.5 CNN+MHA

The CNN+MHA is similar to the above model, but CNN+MHA used an MHA instead of the self-attention block. We used 8 heads for the MHA with LayerNormalization (epsilon set to 1e-6) and GlobalAveragePooling2D.

5.6 CNN+LSTM+Self-attention

This model simply used the LSTM block after the fourth CNN block followed by the Multi-head attention block. We used an LSTM with units equal to 124, a dropout rate of 0.6, and a recurrent dropout rate of 0.1. The Multi-head attention block had 8 heads and had LayerNormalization with an epsilon of 1e-6 and GlobalAveragePooling1D.

6. References

Hou, X., Zhang, H., & Jiang, J. (2021). Numerical Simulation of 2D Electron Gas under a Magnetic Field in a Quantum Dot with an Artificial Neural Network. Journal of Physics: Conference Series, 1732(1), 012026. https://doi.org/10.1088/1742-6596/1732/1/012026

Kim, D., Kim, J., & Kang, S. (2020). Deep learning-based EEG analysis for sleep stage classification: A systematic review. arXiv preprint arXiv:2006.16362.

Base CNN Model. In this coding discussion, we will walk you through some basic preprocessing techniques that will help you achieve some performance boost.

This notebook has been created by Tonmoy.

(i) Importing the necessary packages

In [ ]:
```
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten,Dropout
from keras.layers import Conv2D,BatchNormalization,MaxPooling2D,Reshape
from keras.utils import to_categorical
from keras_tuner import RandomSearch
import matplotlib.pyplot as plt
```

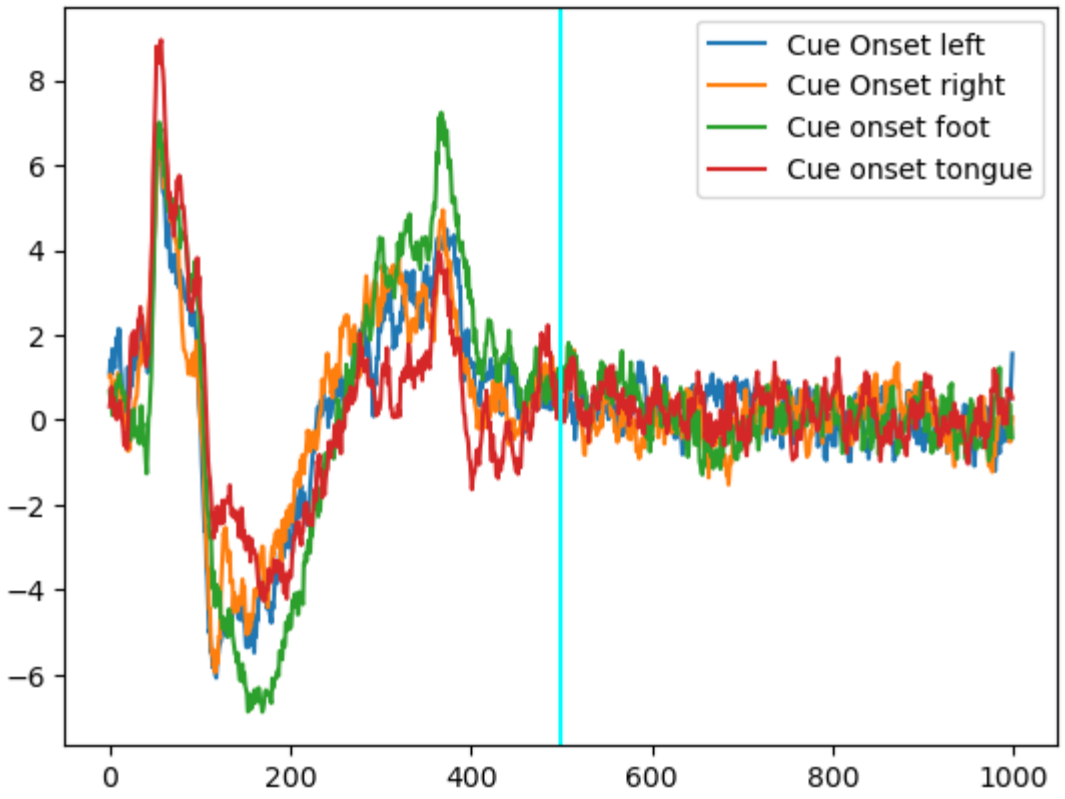
(ii)(a) Loading and visualizing the dataset

In [ ]:

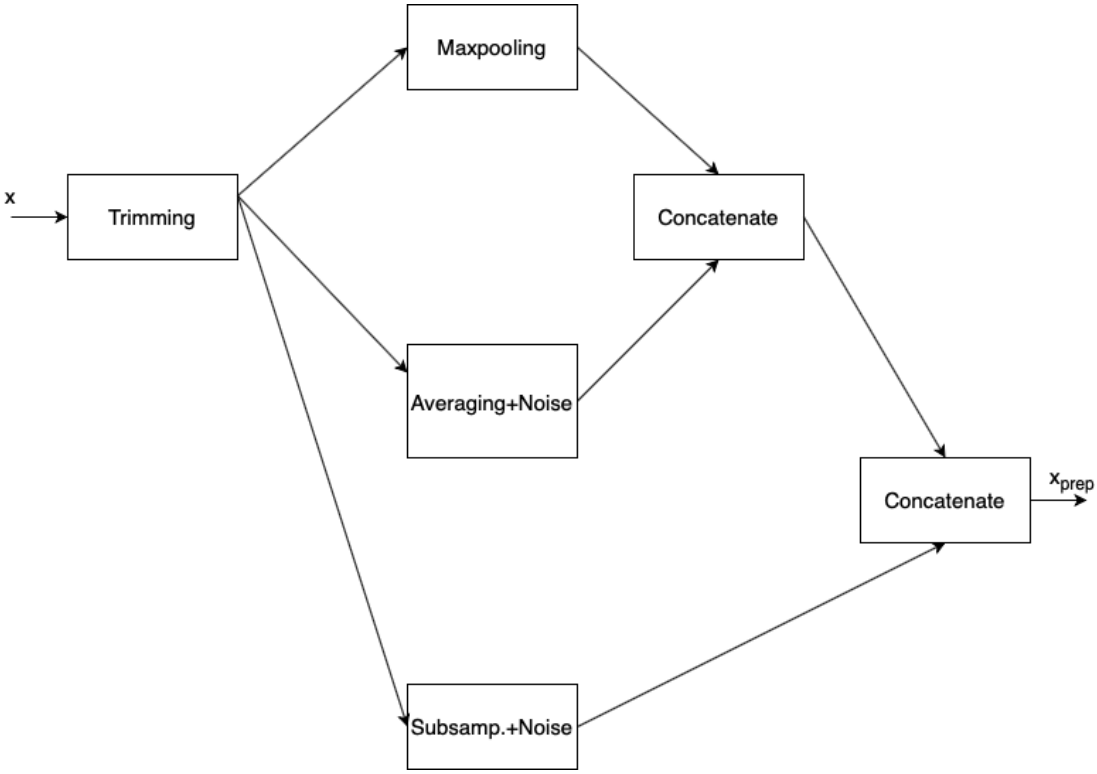
```
## Loading and visualizing the data

## Loading the dataset


X_test = np.load("X_test.npy")
y_test = np.load("y_test.npy")
person_train_valid = np.load("person_train_valid.npy")
X_train_valid = np.load("X_train_valid.npy")
y_train_valid = np.load("y_train_valid.npy")
person_test = np.load("person_test.npy")

## Adjusting the labels so that

# Cue onset left - 0
# Cue onset right - 1
# Cue onset foot - 2
# Cue onset tongue - 3

y_train_valid -= 769
y_test -= 769

## Visualizing the data

ch_data = X_train_valid[:,9,:] # extracts the 9th (out of 22) channel from t


class_0_ind = np.where(y_train_valid == 0) # finds the indices where the lab
ch_data_class_0 = ch_data[class_0_ind] # finds the data where label is 0
avg_ch_data_class_0 = np.mean(ch_data_class_0,axis=0) # finds the average re


class_1_ind = np.where(y_train_valid == 1)
ch_data_class_1 = ch_data[class_1_ind]
avg_ch_data_class_1 = np.mean(ch_data_class_1,axis=0)

class_2_ind = np.where(y_train_valid == 2)
ch_data_class_2 = ch_data[class_2_ind]
avg_ch_data_class_2 = np.mean(ch_data_class_2,axis=0)

class_3_ind = np.where(y_train_valid == 3)
ch_data_class_3 = ch_data[class_3_ind]
avg_ch_data_class_3 = np.mean(ch_data_class_3,axis=0)


plt.plot(np.arange(1000),avg_ch_data_class_0)
plt.plot(np.arange(1000),avg_ch_data_class_1)
plt.plot(np.arange(1000),avg_ch_data_class_2)
plt.plot(np.arange(1000),avg_ch_data_class_3)
plt.axvline(x=500, label='line at t=500',c='cyan')

plt.legend(["Cue Onset left", "Cue Onset right", "Cue onset foot", "Cue onse
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x7feb781e52e0>
```



(ii) (b) Preprocessing the dataset



In [ ]:
```
def data_prep(X,y,sub_sample,average,noise):

    total_X = None
    total_y = None

    # Trimming the data (sample,22,1000) -> (sample,22,500)
    X = X[:,:,0:500]
    print('Shape of X after trimming:',X.shape)

    # Maxpooling the data (sample,22,1000) -> (sample,22,500/sub_sample)
    X_max = np.max(X.reshape(X.shape[0], X.shape[1], -1, sub_sample), axis=3


    total_X = X_max
    total_y = y
    print('Shape of X after maxpooling:',total_X.shape)


    #data augmentation

    # Averaging + noise
    X_average = np.mean(X.reshape(X.shape[0], X.shape[1], -1, average),axis=
    X_average = X_average + np.random.normal(0.0, 0.5, X_average.shape)

    total_X = np.vstack((total_X, X_average))
    total_y = np.hstack((total_y, y))
    print('Shape of X after averaging+noise and concatenating:',total_X.shap

    # Subsampling

    for i in range(sub_sample):

        X_subsample = X[:, :, i::sub_sample] + \
                            (np.random.normal(0.0, 0.5, X[:, :,i::sub_sample

        total_X = np.vstack((total_X, X_subsample))
        total_y = np.hstack((total_y, y))


    print('Shape of X after subsampling and concatenating:',total_X.shape)
    return total_X,total_y


X_train_valid_prep,y_train_valid_prep = data_prep(X_train_valid,y_train_vali
```

```
Shape of X after trimming: (2115, 22, 500)
Shape of X after maxpooling: (2115, 22, 250)
Shape of X after averaging+noise and concatenating: (4230, 22, 250)
Shape of X after subsampling and concatenating: (8460, 22, 250)
```

(ii)(c) Preparing the training, validation, and test datasets

In [ ]:

```
## Random splitting and reshaping the data
# First generating the training and validation indices using random splittir

ind_valid = np.random.choice(2115, 375, replace=False)
ind_train = np.array(list(set(range(2115)).difference(set(ind_valid))))

# Creating the training and validation sets using the generated indices
(X_train, X_valid) = X_train_valid[ind_train], X_train_valid[ind_valid]
(y_train, y_valid) = y_train_valid[ind_train], y_train_valid[ind_valid]


## Preprocessing the dataset
print("training data")
x_train,y_train = data_prep(X_train,y_train,2,2,True)
print("\nvalidation data")
x_valid,y_valid = data_prep(X_valid,y_valid,2,2,True)
print("\ntest data")
X_test_prep,y_test_prep = data_prep(X_test,y_test,2,2,True)


print('Shape of testing set:',X_test_prep.shape)
print('Shape of testing labels:',y_test_prep.shape)

print('Shape of training set:',x_train.shape)
print('Shape of validation set:',x_valid.shape)
print('Shape of training labels:',y_train.shape)
print('Shape of validation labels:',y_valid.shape)


# Converting the labels to categorical variables for multiclass classificati
y_train = to_categorical(y_train, 4)
y_valid = to_categorical(y_valid, 4)
y_test = to_categorical(y_test_prep, 4)
print('Shape of training labels after categorical conversion:',y_train.shape
print('Shape of validation labels after categorical conversion:',y_valid.sha
print('Shape of test labels after categorical conversion:',y_test.shape)

# Adding width of the segment to be 1
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], x_train.shape[
x_valid = x_valid.reshape(x_valid.shape[0], x_valid.shape[1], x_train.shape[
x_test = X_test_prep.reshape(X_test_prep.shape[0], X_test_prep.shape[1], X_t
print('Shape of training set after adding width info:',x_train.shape)
print('Shape of validation set after adding width info:',x_valid.shape)
print('Shape of test set after adding width info:',x_test.shape)


# Reshaping the training and validation dataset
x_train = np.swapaxes(x_train, 1,3)
x_train = np.swapaxes(x_train, 1,2)
x_valid = np.swapaxes(x_valid, 1,3)
x_valid = np.swapaxes(x_valid, 1,2)
x_test = np.swapaxes(x_test, 1,3)
x_test = np.swapaxes(x_test, 1,2)
print('Shape of training set after dimension reshaping:',x_train.shape)
print('Shape of validation set after dimension reshaping:',x_valid.shape)
print('Shape of test set after dimension reshaping:',x_test.shape)
```

```
training data
Shape of X after trimming: (1740, 22, 500)
Shape of X after maxpooling: (1740, 22, 250)
Shape of X after averaging+noise and concatenating: (3480, 22, 250)
Shape of X after subsampling and concatenating: (6960, 22, 250)

validation data
Shape of X after trimming: (375, 22, 500)
Shape of X after maxpooling: (375, 22, 250)
Shape of X after averaging+noise and concatenating: (750, 22, 250)
Shape of X after subsampling and concatenating: (1500, 22, 250)

test data
Shape of X after trimming: (443, 22, 500)
Shape of X after maxpooling: (443, 22, 250)
Shape of X after averaging+noise and concatenating: (886, 22, 250)
Shape of X after subsampling and concatenating: (1772, 22, 250)
Shape of testing set: (1772, 22, 250)
Shape of testing labels: (1772,)
Shape of training set: (6960, 22, 250)
Shape of validation set: (1500, 22, 250)
Shape of training labels: (6960,)
Shape of validation labels: (1500,)
Shape of training labels after categorical conversion: (6960, 4)
Shape of validation labels after categorical conversion: (1500, 4)
Shape of test labels after categorical conversion: (1772, 4)
Shape of training set after adding width info: (6960, 22, 250, 1)
Shape of validation set after adding width info: (1500, 22, 250, 1)
Shape of test set after adding width info: (1772, 22, 250, 1)
Shape of training set after dimension reshaping: (6960, 250, 1, 22)
Shape of validation set after dimension reshaping: (1500, 250, 1, 22)
Shape of test set after dimension reshaping: (1772, 250, 1, 22)
```

(iii)(CNN) Defining the architecture of a basic CNN model

In [ ]:

```
def build_model(params):
    # Building the CNN model using sequential class
    basic_cnn_model = Sequential()

    #tune number of filters
    filter1 = params.Int('filter1', min_value =15, max_value = 30, step = 5)
    filter2 = params.Int('filter2', min_value =40, max_value = 60, step = 5)
    filter3 = params.Int('filter3', min_value =80, max_value = 120, step = 1
    filter4 = params.Int('filter4', min_value =180, max_value = 220, step =



    #tune kernel size
    kernel = params.Choice('kernel_size', values=[3,5,10])
    #tune pool size
    pool = params.Choice('pool_size', values=[2,3,5])
    # tune dropout
    dropout = 0.5
    # 0.5 is optimal for dropout

    # Conv. block 1
    basic_cnn_model.add(Conv2D(filters=filter1, kernel_size=(kernel,1), padd
    basic_cnn_model.add(MaxPooling2D(pool_size=(pool,1), padding='same')) #
    basic_cnn_model.add(BatchNormalization())
    basic_cnn_model.add(Dropout(dropout))

    # Conv. block 2
    basic_cnn_model.add(Conv2D(filters=filter2, kernel_size=(kernel,1), padd
    basic_cnn_model.add(MaxPooling2D(pool_size=(pool,1), padding='same'))
    basic_cnn_model.add(BatchNormalization())
    basic_cnn_model.add(Dropout(dropout))

    # Conv. block 3
    basic_cnn_model.add(Conv2D(filters=filter3, kernel_size=(kernel,1), padd
    basic_cnn_model.add(MaxPooling2D(pool_size=(pool,1), padding='same'))
    basic_cnn_model.add(BatchNormalization())
    basic_cnn_model.add(Dropout(dropout))

    # Conv. block 4
    basic_cnn_model.add(Conv2D(filters=filter4, kernel_size=(kernel,1), padd
    basic_cnn_model.add(MaxPooling2D(pool_size=(pool,1), padding='same'))
    basic_cnn_model.add(BatchNormalization())
    basic_cnn_model.add(Dropout(dropout))

    # Output layer with Softmax activation
    basic_cnn_model.add(Flatten()) # Flattens the input
    basic_cnn_model.add(Dense(4, activation='softmax')) # Output FC layer wi


    # Printing the model summary
    # basic_cnn_model.summary()

    learning_rate = 1e-3
    # epochs = 50
    cnn_optimizer = keras.optimizers.Adam(lr=learning_rate)

    # Compiling the model
    basic_cnn_model.compile(loss='categorical_crossentropy',
                            optimizer=cnn_optimizer
```

```
                    optimizer=cnn_optimizer,
                    metrics=['accuracy'])

    return basic_cnn_model
```

(iv) (CNN) Defining the hyperparameters of the basic CNN model

In [ ]:
```
# Model parameters
# learning_rate = 1e-3
# epochs = 50
# cnn_optimizer = keras.optimizers.Adam(lr=learning_rate)

#early stopping
callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
tuner = RandomSearch(build_model,
                     objective='val_accuracy',
                     max_trials = 20,
                     overwrite=True)
tuner.search(x_train,y_train,epochs=80,validation_data=(x_valid,y_valid), ca
```

```
Trial 20 Complete [00h 02m 23s]
val_accuracy: 0.7133333086967468

Best val_accuracy So Far: 0.7480000257492065
Total elapsed time: 00h 34m 31s
INFO:tensorflow:Oracle triggered exit
```

(v) (CNN) Compiling, training and validating the model

In [ ]:
```
#train with best hyperparameters
best_hps=tuner.get_best_hyperparameters(num_trials=1)[0]
model = tuner.hypermodel.build(best_hps)
history = model.fit(x_train, y_train, epochs=50,batch_size=64, validation_da

val_acc_per_epoch = history.history['val_accuracy']
best_epoch = val_acc_per_epoch.index(max(val_acc_per_epoch)) + 1
print('Best epoch: %d' % (best_epoch,))
```

```
Epoch 1/50
109/109 [==============================] - 4s 34ms/step - loss: 1.9290 - ac
curacy: 0.3203 - val_loss: 1.3925 - val_accuracy: 0.3973
Epoch 2/50
109/109 [==============================] - 4s 33ms/step - loss: 1.5287 - ac
curacy: 0.3807 - val_loss: 1.2395 - val_accuracy: 0.4427
Epoch 3/50
109/109 [==============================] - 4s 32ms/step - loss: 1.3329 - ac
curacy: 0.4353 - val_loss: 1.1689 - val_accuracy: 0.4893
Epoch 4/50
109/109 [==============================] - 4s 32ms/step - loss: 1.2191 - ac
curacy: 0.4892 - val_loss: 1.1516 - val_accuracy: 0.4940
Epoch 5/50
109/109 [==============================] - 4s 33ms/step - loss: 1.1397 - ac
curacy: 0.5165 - val_loss: 1.1252 - val_accuracy: 0.5133
Epoch 6/50
109/109 [==============================] - 4s 33ms/step - loss: 1.0886 - ac
curacy: 0.5398 - val_loss: 1.0996 - val_accuracy: 0.5047
Epoch 7/50
109/109 [==============================] - 4s 33ms/step - loss: 1.0479 - ac
curacy: 0.5737 - val_loss: 1.0523 - val_accuracy: 0.5540
Epoch 8/50
109/109 [==============================] - 4s 34ms/step - loss: 0.9987 - ac
curacy: 0.5822 - val_loss: 1.0802 - val_accuracy: 0.5287
Epoch 9/50
109/109 [==============================] - 4s 32ms/step - loss: 0.9690 - ac
curacy: 0.6037 - val_loss: 0.9931 - val_accuracy: 0.5807
Epoch 10/50
109/109 [==============================] - 4s 32ms/step - loss: 0.9158 - ac
curacy: 0.6313 - val_loss: 0.9705 - val_accuracy: 0.5873
Epoch 11/50
109/109 [==============================] - 4s 32ms/step - loss: 0.8929 - ac
curacy: 0.6408 - val_loss: 0.9327 - val_accuracy: 0.6253
Epoch 12/50
109/109 [==============================] - 4s 32ms/step - loss: 0.8539 - ac
curacy: 0.6566 - val_loss: 0.8794 - val_accuracy: 0.6593
Epoch 13/50
109/109 [==============================] - 4s 33ms/step - loss: 0.8165 - ac
curacy: 0.6733 - val_loss: 0.8591 - val_accuracy: 0.6420
Epoch 14/50
109/109 [==============================] - 4s 33ms/step - loss: 0.7896 - ac
curacy: 0.6846 - val_loss: 0.8255 - val_accuracy: 0.6600
Epoch 15/50
109/109 [==============================] - 4s 34ms/step - loss: 0.7511 - ac
curacy: 0.7010 - val_loss: 0.8646 - val_accuracy: 0.6360
Epoch 16/50
109/109 [==============================] - 4s 35ms/step - loss: 0.7357 - ac
curacy: 0.7050 - val_loss: 0.8328 - val_accuracy: 0.6740
Epoch 17/50
109/109 [==============================] - 4s 33ms/step - loss: 0.7049 - ac
curacy: 0.7231 - val_loss: 0.7896 - val_accuracy: 0.6827
Epoch 18/50
109/109 [==============================] - 4s 32ms/step - loss: 0.6826 - ac
curacy: 0.7320 - val_loss: 0.8439 - val_accuracy: 0.6893
Epoch 19/50
109/109 [==============================] - 4s 33ms/step - loss: 0.6572 - ac
curacy: 0.7420 - val_loss: 0.8132 - val_accuracy: 0.6780
Epoch 20/50
109/109 [==============================] - 4s 33ms/step - loss: 0.6337 - ac
```

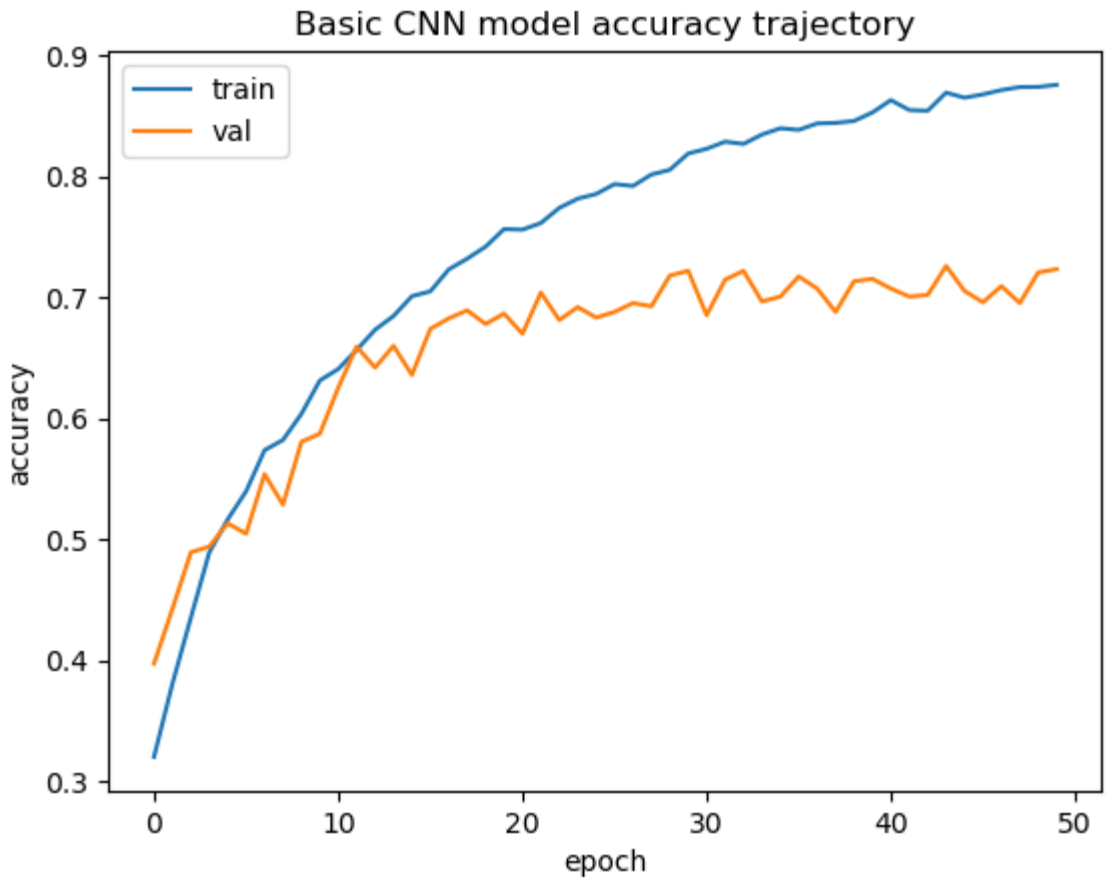
```
curacy: 0.7565 - val_loss: 0.8039 - val_accuracy: 0.6867
Epoch 21/50
109/109 [==============================] - 4s 34ms/step - loss: 0.6274 - ac
curacy: 0.7560 - val_loss: 0.8501 - val_accuracy: 0.6700
Epoch 22/50
109/109 [==============================] - 4s 33ms/step - loss: 0.5989 - ac
curacy: 0.7615 - val_loss: 0.7739 - val_accuracy: 0.7040
Epoch 23/50
109/109 [==============================] - 4s 32ms/step - loss: 0.5868 - ac
curacy: 0.7740 - val_loss: 0.8355 - val_accuracy: 0.6813
Epoch 24/50
109/109 [==============================] - 3s 31ms/step - loss: 0.5583 - ac
curacy: 0.7816 - val_loss: 0.7987 - val_accuracy: 0.6920
Epoch 25/50
109/109 [==============================] - 4s 33ms/step - loss: 0.5590 - ac
curacy: 0.7855 - val_loss: 0.8597 - val_accuracy: 0.6833
Epoch 26/50
109/109 [==============================] - 4s 37ms/step - loss: 0.5476 - ac
curacy: 0.7937 - val_loss: 0.8374 - val_accuracy: 0.6880
Epoch 27/50
109/109 [==============================] - 4s 34ms/step - loss: 0.5341 - ac
curacy: 0.7921 - val_loss: 0.8188 - val_accuracy: 0.6953
Epoch 28/50
109/109 [==============================] - 3s 31ms/step - loss: 0.5117 - ac
curacy: 0.8016 - val_loss: 0.8186 - val_accuracy: 0.6927
Epoch 29/50
109/109 [==============================] - 4s 33ms/step - loss: 0.5068 - ac
curacy: 0.8055 - val_loss: 0.7627 - val_accuracy: 0.7180
Epoch 30/50
109/109 [==============================] - 4s 35ms/step - loss: 0.4664 - ac
curacy: 0.8190 - val_loss: 0.7886 - val_accuracy: 0.7220
Epoch 31/50
109/109 [==============================] - 4s 33ms/step - loss: 0.4611 - ac
curacy: 0.8228 - val_loss: 0.8247 - val_accuracy: 0.6853
Epoch 32/50
109/109 [==============================] - 4s 33ms/step - loss: 0.4540 - ac
curacy: 0.8287 - val_loss: 0.8074 - val_accuracy: 0.7147
Epoch 33/50
109/109 [==============================] - 4s 33ms/step - loss: 0.4533 - ac
curacy: 0.8269 - val_loss: 0.8097 - val_accuracy: 0.7220
Epoch 34/50
109/109 [==============================] - 4s 33ms/step - loss: 0.4337 - ac
curacy: 0.8346 - val_loss: 0.8402 - val_accuracy: 0.6967
Epoch 35/50
109/109 [==============================] - 4s 32ms/step - loss: 0.4223 - ac
curacy: 0.8398 - val_loss: 0.8190 - val_accuracy: 0.7007
Epoch 36/50
109/109 [==============================] - 4s 33ms/step - loss: 0.4064 - ac
curacy: 0.8385 - val_loss: 0.8193 - val_accuracy: 0.7173
Epoch 37/50
109/109 [==============================] - 4s 33ms/step - loss: 0.4194 - ac
curacy: 0.8438 - val_loss: 0.8512 - val_accuracy: 0.7073
Epoch 38/50
109/109 [==============================] - 4s 32ms/step - loss: 0.4064 - ac
curacy: 0.8443 - val_loss: 0.8155 - val_accuracy: 0.6880
Epoch 39/50
109/109 [==============================] - 4s 32ms/step - loss: 0.3954 - ac
curacy: 0.8458 - val_loss: 0.8125 - val_accuracy: 0.7133
Epoch 40/50
```

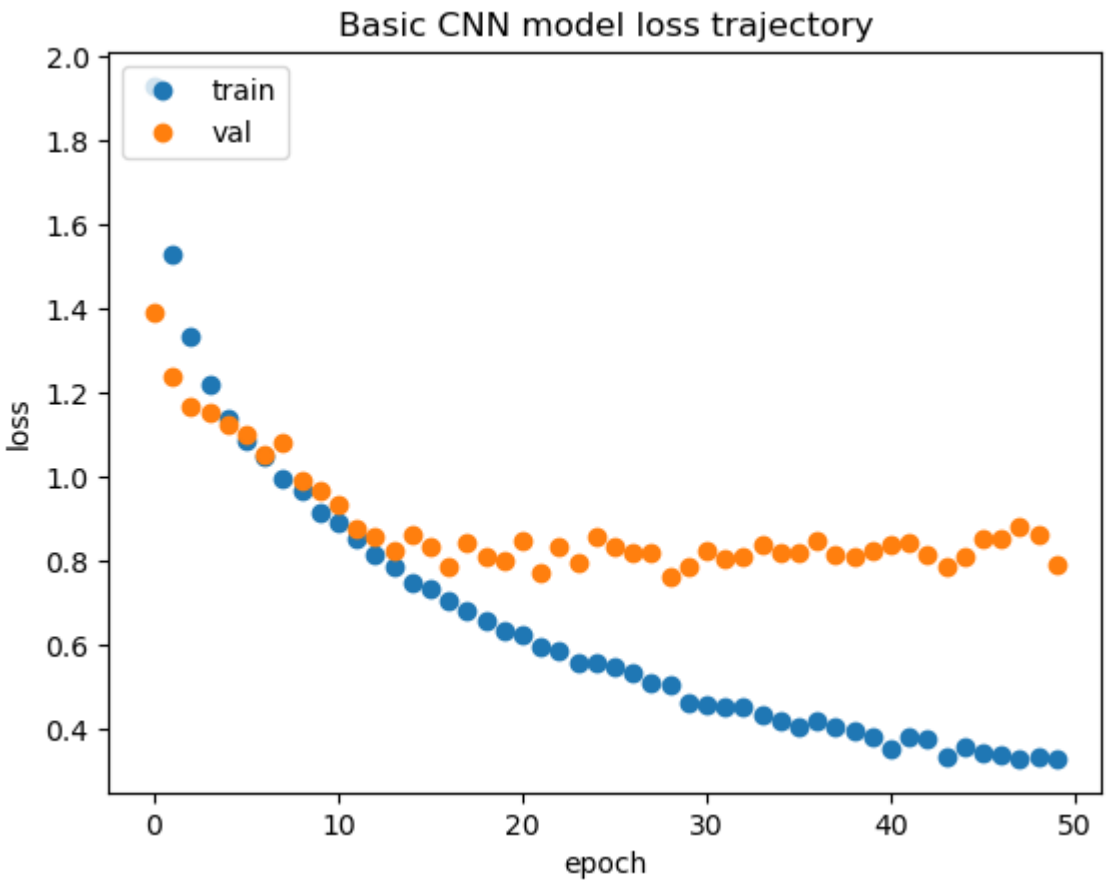
```
109/109 [==============================] - 4s 32ms/step - loss: 0.3815 - ac
curacy: 0.8529 - val_loss: 0.8274 - val_accuracy: 0.7153
Epoch 41/50
109/109 [==============================] - 4s 33ms/step - loss: 0.3567 - ac
curacy: 0.8631 - val_loss: 0.8399 - val_accuracy: 0.7073
Epoch 42/50
109/109 [==============================] - 4s 33ms/step - loss: 0.3832 - ac
curacy: 0.8549 - val_loss: 0.8423 - val_accuracy: 0.7007
Epoch 43/50
109/109 [==============================] - 4s 33ms/step - loss: 0.3788 - ac
curacy: 0.8543 - val_loss: 0.8171 - val_accuracy: 0.7020
Epoch 44/50
109/109 [==============================] - 4s 33ms/step - loss: 0.3375 - ac
curacy: 0.8693 - val_loss: 0.7868 - val_accuracy: 0.7260
Epoch 45/50
109/109 [==============================] - 4s 32ms/step - loss: 0.3601 - ac
curacy: 0.8651 - val_loss: 0.8134 - val_accuracy: 0.7053
Epoch 46/50
109/109 [==============================] - 4s 32ms/step - loss: 0.3461 - ac
curacy: 0.8678 - val_loss: 0.8552 - val_accuracy: 0.6960
Epoch 47/50
109/109 [==============================] - 4s 33ms/step - loss: 0.3419 - ac
curacy: 0.8714 - val_loss: 0.8548 - val_accuracy: 0.7093
Epoch 48/50
109/109 [==============================] - 3s 31ms/step - loss: 0.3327 - ac
curacy: 0.8739 - val_loss: 0.8808 - val_accuracy: 0.6953
Epoch 49/50
109/109 [==============================] - 4s 33ms/step - loss: 0.3358 - ac
curacy: 0.8740 - val_loss: 0.8625 - val_accuracy: 0.7207
Epoch 50/50
109/109 [==============================] - 3s 32ms/step - loss: 0.3314 - ac
curacy: 0.8757 - val_loss: 0.7923 - val_accuracy: 0.7233
Best epoch: 44
```

(vi) (CNN) Visualizing the accuracy and loss trajectory

In [ ]:

```python
import matplotlib.pyplot as plt

# Plotting accuracy trajectory
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Basic CNN model accuracy trajectory')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

# Plotting loss trajectory
plt.plot(history.history['loss'],'o')
plt.plot(history.history['val_loss'],'o')
plt.title('Basic CNN model loss trajectory')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```


Basic CNN model accuracy trajectory
train
val
accuracy
epoch
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
10
20
30
40
50


Basic CNN model loss trajectory
train
val
loss
epoch
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
10
20
30
40
50

(vii)(CNN) Testing the performance of the basic CNN model on the held out test set

In [ ]:
```
## Testing the basic CNN model

cnn_score = model.evaluate(x_test, y_test, verbose=0)
print('Test accuracy of the basic CNN model:',cnn_score[1])
print(best_hps.values)
```

```
Test accuracy of the basic CNN model: 0.7009029388427734
{'filter1': 30, 'filter2': 55, 'filter3': 80, 'filter4': 180, 'kernel_size
': 10, 'pool_size': 3}
```

In this discussion, we will build a basic hybrid CNN-LSTM model for classification on the EEG dataset

This notebook was inspired to Tonmoy with some attempts to tune by us

(i) Importing the necessary packages

In [ ]:
```
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential,Model
from keras.layers import Dense, Activation, Flatten,Dropout
from keras.layers import Conv2D,LSTM,BatchNormalization,MaxPooling2D,Reshape
from keras.utils import to_categorical
import matplotlib.pyplot as plt
```

```
2023-03-18 20:47:02.872775: I tensorflow/core/platform/cpu_feature_guard.c
c:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network
Library (oneDNN) to use the following CPU instructions in performance-criti
cal operations:  SSE4.1 SSE4.2
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

(ii) Preprocessing the dataset and preparing the training, validation, and test datasets

In [ ]:

```
def data_prep(X,y,sub_sample,average,noise):

    total_X = None
    total_y = None

    # Trimming the data (sample,22,1000) -> (sample,22,500)
    X = X[:,:,0:500]
    print('Shape of X after trimming:',X.shape)

    # Maxpooling the data (sample,22,1000) -> (sample,22,500/sub_sample)
    X_max = np.max(X.reshape(X.shape[0], X.shape[1], -1, sub_sample), axis=3


    total_X = X_max
    total_y = y
    print('Shape of X after maxpooling:',total_X.shape)

    # Averaging + noise
    X_average = np.mean(X.reshape(X.shape[0], X.shape[1], -1, average),axis=
    X_average = X_average + np.random.normal(0.0, 0.5, X_average.shape)

    total_X = np.vstack((total_X, X_average))
    total_y = np.hstack((total_y, y))
    print('Shape of X after averaging+noise and concatenating:',total_X.shap

    # Subsampling

    for i in range(sub_sample):

        X_subsample = X[:, :, i::sub_sample] + \
                            (np.random.normal(0.0, 0.5, X[:, :,i::sub_sample

        total_X = np.vstack((total_X, X_subsample))
        total_y = np.hstack((total_y, y))


    print('Shape of X after subsampling and concatenating:',total_X.shape)
    return total_X,total_y
```

In [ ]:

```
## Loading the dataset


X_test = np.load("X_test.npy")
y_test = np.load("y_test.npy")
person_train_valid = np.load("person_train_valid.npy")
X_train_valid = np.load("X_train_valid.npy")
y_train_valid = np.load("y_train_valid.npy")
person_test = np.load("person_test.npy")

## Adjusting the labels so that

# Cue onset left - 0
# Cue onset right - 1
# Cue onset foot - 2
# Cue onset tongue - 3

y_train_valid -= 769
y_test -= 769


## Random splitting and reshaping the data
# First generating the training and validation indices using random splittin

ind_valid = np.random.choice(2115, 375, replace=False)
ind_train = np.array(list(set(range(2115)).difference(set(ind_valid))))

# Creating the training and validation sets using the generated indices
(X_train, X_valid) = X_train_valid[ind_train], X_train_valid[ind_valid]
(y_train, y_valid) = y_train_valid[ind_train], y_train_valid[ind_valid]


## Preprocessing the dataset
x_train,y_train = data_prep(X_train,y_train,2,2,True)
x_valid,y_valid = data_prep(X_valid,y_valid,2,2,True)
X_test_prep,y_test_prep = data_prep(X_test,y_test,2,2,True)


print('Shape of training set:',x_train.shape)
print('Shape of validation set:',x_valid.shape)
print('Shape of training labels:',y_train.shape)
print('Shape of validation labels:',y_valid.shape)
print('Shape of testing set:',X_test_prep.shape)
print('Shape of testing labels:',y_test_prep.shape)


# Converting the labels to categorical variables for multiclass classificati
y_train = to_categorical(y_train, 4)
y_valid = to_categorical(y_valid, 4)
y_test = to_categorical(y_test_prep, 4)
print('Shape of training labels after categorical conversion:',y_train.shape
print('Shape of validation labels after categorical conversion:',y_valid.sha
print('Shape of test labels after categorical conversion:',y_test.shape)

# Adding width of the segment to be 1
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], x_train.shape[
```

```
x_valid = x_valid.reshape(x_valid.shape[0], x_valid.shape[1], x_train.shape[
x_test = X_test_prep.reshape(X_test_prep.shape[0], X_test_prep.shape[1], X_t
print('Shape of training set after adding width info:',x_train.shape)
print('Shape of validation set after adding width info:',x_valid.shape)
print('Shape of test set after adding width info:',x_test.shape)


# Reshaping the training and validation dataset
x_train = np.swapaxes(x_train, 1,3)
x_train = np.swapaxes(x_train, 1,2)
x_valid = np.swapaxes(x_valid, 1,3)
x_valid = np.swapaxes(x_valid, 1,2)
x_test = np.swapaxes(x_test, 1,3)
x_test = np.swapaxes(x_test, 1,2)
print('Shape of training set after dimension reshaping:',x_train.shape)
print('Shape of validation set after dimension reshaping:',x_valid.shape)
print('Shape of test set after dimension reshaping:',x_test.shape)

keras.backend.clear_session()
```

```
Shape of X after trimming: (1740, 22, 500)
Shape of X after maxpooling: (1740, 22, 250)
Shape of X after averaging+noise and concatenating: (3480, 22, 250)
Shape of X after subsampling and concatenating: (6960, 22, 250)
Shape of X after trimming: (375, 22, 500)
Shape of X after maxpooling: (375, 22, 250)
Shape of X after averaging+noise and concatenating: (750, 22, 250)
Shape of X after subsampling and concatenating: (1500, 22, 250)
Shape of X after trimming: (443, 22, 500)
Shape of X after maxpooling: (443, 22, 250)
Shape of X after averaging+noise and concatenating: (886, 22, 250)
Shape of X after subsampling and concatenating: (1772, 22, 250)
Shape of training set: (6960, 22, 250)
Shape of validation set: (1500, 22, 250)
Shape of training labels: (6960,)
Shape of validation labels: (1500,)
Shape of testing set: (1772, 22, 250)
Shape of testing labels: (1772,)
Shape of training labels after categorical conversion: (6960, 4)
Shape of validation labels after categorical conversion: (1500, 4)
Shape of test labels after categorical conversion: (1772, 4)
Shape of training set after adding width info: (6960, 22, 250, 1)
Shape of validation set after adding width info: (1500, 22, 250, 1)
Shape of test set after adding width info: (1772, 22, 250, 1)
Shape of training set after dimension reshaping: (6960, 250, 1, 22)
Shape of validation set after dimension reshaping: (1500, 250, 1, 22)
Shape of test set after dimension reshaping: (1772, 250, 1, 22)
```

(iii)(CNN-LSTM) Defining the architecture of the hybrid CNN-LSTM model

In [ ]:
```
# Building the CNN model using sequential class
def build_model():
# models = []

        # Conv. block 1
    In1 = keras.Input(shape =(250,1,22) )
    c1 = Conv2D(filters=30, kernel_size=(11,1), padding='same', activation='
    p1 =  MaxPooling2D(pool_size=(4,1), padding='same')(c1) # Read the keras
    b1 = BatchNormalization()(p1)
    d1 = Dropout(0.5)(b1)

    # Conv. block 2
    c2 = Conv2D(filters=60, kernel_size=(9,1), padding='same', activation='s
    p2 =  MaxPooling2D(pool_size=(4,1), padding='same')(c2) # Read the keras
    b2 = BatchNormalization()(p2)
    d2 = Dropout(0.6)(b2)

    # Conv. block 3
    c3 = Conv2D(filters=120, kernel_size=(5,1), padding='same', activation='
    p3 =  MaxPooling2D(pool_size=(4,1), padding='same')(c3) # Read the keras
    b3 = BatchNormalization()(p3)
    d3 = Dropout(0.6)(b3)

    # Conv. block 4
    c4 = Conv2D(filters=240, kernel_size=(3,1), padding='same', activation='
    p4 =  MaxPooling2D(pool_size=(4,1), padding='same')(c4) # Read the keras
    b4 = BatchNormalization()(p4)
    d4 = Dropout(0.6)(b4)

    # FC+LSTM layers
    lstm  = Flatten()(d4) # Adding a flattening operation to the output of C
    lstm = Dense((120))(lstm) # FC layer with 100 units
    lstm = Reshape((120,1))(lstm) # Reshape my output of FC layer so that it
    lstm = LSTM(10, dropout=0.6, recurrent_dropout=0.1, input_shape=(120,1),

    # Output layer with Softmax activation
    fc = Dense(4, activation='softmax') (lstm) # Output FC layer with softma

    # Define the final model
    final_model = Model(inputs=In1, outputs=[fc])

    # # Printing the model summary
    final_model.compile(loss='categorical_crossentropy',
            optimizer='adam',
            metrics=['accuracy'])
    final_model.summary()
    return final_model
```

(iv) (CNN-LSTM) Defining the hyperparameters of the hybrid CNN-LSTM model

In [ ]:
```
# Model parameters
learning_rate = 1e-3
epochs = 100
hybrid_cnn_lstm_optimizer = keras.optimizers.Adam(lr=learning_rate)
```

```
/Users/lilyzhou/opt/anaconda3/envs/finalProject147/lib/python3.9/site-packa
ges/keras/optimizers/optimizer_v2/adam.py:114: UserWarning: The `lr` argume
nt is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

(v)(CNN-LSTM) Compiling, training and validating the model

In [ ]:
```
# Compiling the model
keras.backend.clear_session()

# Training and validating the model

batch_sizes = [32,64,128]
import matplotlib.pyplot as plt

hybrid_cnn_lstm_model = build_model()
# Compiling the model
hybrid_cnn_lstm_model.compile(loss='categorical_crossentropy',
              optimizer=hybrid_cnn_lstm_optimizer,
              metrics=['accuracy'])

# Training and validating the model

callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=20) #
hybrid_cnn_lstm_model_results = hybrid_cnn_lstm_model.fit(x_train,
               y_train,
               batch_size=32,
               epochs=epochs,
               validation_data=(x_valid, y_valid),
               verbose=True,
               callbacks = [callback])
```

```
2023-03-18 20:47:10.147668: I tensorflow/core/platform/cpu_feature_guard.c
c:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network
Library (oneDNN) to use the following CPU instructions in performance-criti
cal operations:  SSE4.1 SSE4.2
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 250, 1, 22)]	0
conv2d (Conv2D)	(None, 250, 1, 30)	7290
max_pooling2d (MaxPooling2D)	(None, 63, 1, 30)	0
batch_normalization (BatchNormalization)	(None, 63, 1, 30)	120
dropout (Dropout)	(None, 63, 1, 30)	0
conv2d_1 (Conv2D)	(None, 63, 1, 60)	16260
max_pooling2d_1 (MaxPooling2D)	(None, 16, 1, 60)	0
batch_normalization_1 (BatchNormalization)	(None, 16, 1, 60)	240
dropout_1 (Dropout)	(None, 16, 1, 60)	0
conv2d_2 (Conv2D)	(None, 16, 1, 120)	36120
max_pooling2d_2 (MaxPooling2D)	(None, 4, 1, 120)	0
batch_normalization_2 (BatchNormalization)	(None, 4, 1, 120)	480
dropout_2 (Dropout)	(None, 4, 1, 120)	0
conv2d_3 (Conv2D)	(None, 4, 1, 240)	86640
max_pooling2d_3 (MaxPooling2D)	(None, 1, 1, 240)	0
batch_normalization_3 (BatchNormalization)	(None, 1, 1, 240)	960
dropout_3 (Dropout)	(None, 1, 1, 240)	0
flatten (Flatten)	(None, 240)	0
dense (Dense)	(None, 120)	28920
reshape (Reshape)	(None, 120, 1)	0
lstm (LSTM)	(None, 10)	480
dense_1 (Dense)	(None, 4)	44

```
=================================================================
Total params: 177,554
Trainable params: 176,654
Non-trainable params: 900
_________________________________________________________________
Epoch 1/100
218/218 [==============================] - 30s 126ms/step - loss: 1.3888 -
accuracy: 0.2530 - val_loss: 1.3825 - val_accuracy: 0.2587
Epoch 2/100
218/218 [==============================] - 26s 122ms/step - loss: 1.3784 -
accuracy: 0.2855 - val_loss: 1.3781 - val_accuracy: 0.2933
Epoch 3/100
218/218 [==============================] - 26s 121ms/step - loss: 1.3601 -
accuracy: 0.3174 - val_loss: 1.3142 - val_accuracy: 0.3740
Epoch 4/100
218/218 [==============================] - 27s 122ms/step - loss: 1.3147 -
accuracy: 0.3671 - val_loss: 1.3202 - val_accuracy: 0.3513
Epoch 5/100
218/218 [==============================] - 27s 122ms/step - loss: 1.2898 -
accuracy: 0.3856 - val_loss: 1.2761 - val_accuracy: 0.3987
Epoch 6/100
218/218 [==============================] - 28s 128ms/step - loss: 1.2602 -
accuracy: 0.4088 - val_loss: 1.2645 - val_accuracy: 0.4027
Epoch 7/100
218/218 [==============================] - 28s 127ms/step - loss: 1.2473 -
accuracy: 0.4164 - val_loss: 1.2164 - val_accuracy: 0.4293
Epoch 8/100
218/218 [==============================] - 28s 129ms/step - loss: 1.2289 -
accuracy: 0.4332 - val_loss: 1.1946 - val_accuracy: 0.4140
Epoch 9/100
218/218 [==============================] - 27s 126ms/step - loss: 1.2133 -
accuracy: 0.4376 - val_loss: 1.1867 - val_accuracy: 0.4393
Epoch 10/100
218/218 [==============================] - 28s 129ms/step - loss: 1.1955 -
accuracy: 0.4527 - val_loss: 1.1772 - val_accuracy: 0.4187
Epoch 11/100
218/218 [==============================] - 25s 115ms/step - loss: 1.1801 -
accuracy: 0.4611 - val_loss: 1.1564 - val_accuracy: 0.4753
Epoch 12/100
218/218 [==============================] - 24s 112ms/step - loss: 1.1599 -
accuracy: 0.4744 - val_loss: 1.1605 - val_accuracy: 0.4460
Epoch 13/100
218/218 [==============================] - 24s 111ms/step - loss: 1.1481 -
accuracy: 0.4839 - val_loss: 1.1220 - val_accuracy: 0.4713
Epoch 14/100
218/218 [==============================] - 28s 128ms/step - loss: 1.1344 -
accuracy: 0.4907 - val_loss: 1.1273 - val_accuracy: 0.4793
Epoch 15/100
218/218 [==============================] - 26s 117ms/step - loss: 1.1247 -
accuracy: 0.4894 - val_loss: 1.1509 - val_accuracy: 0.4580
Epoch 16/100
218/218 [==============================] - 27s 123ms/step - loss: 1.1074 -
accuracy: 0.4981 - val_loss: 1.1036 - val_accuracy: 0.4813
Epoch 17/100
218/218 [==============================] - 24s 112ms/step - loss: 1.0924 -
accuracy: 0.5076 - val_loss: 1.1046 - val_accuracy: 0.4793
```

```
Epoch 18/100
218/218 [==============================] - 26s 118ms/step - loss: 1.0796 -
accuracy: 0.5194 - val_loss: 1.1236 - val_accuracy: 0.4827
Epoch 19/100
218/218 [==============================] - 26s 118ms/step - loss: 1.0717 -
accuracy: 0.5162 - val_loss: 1.0904 - val_accuracy: 0.4873
Epoch 20/100
218/218 [==============================] - 26s 118ms/step - loss: 1.0579 -
accuracy: 0.5283 - val_loss: 1.1677 - val_accuracy: 0.4513
Epoch 21/100
218/218 [==============================] - 26s 119ms/step - loss: 1.0630 -
accuracy: 0.5341 - val_loss: 1.0984 - val_accuracy: 0.4867
Epoch 22/100
218/218 [==============================] - 24s 112ms/step - loss: 1.0345 -
accuracy: 0.5434 - val_loss: 1.1001 - val_accuracy: 0.4653
Epoch 23/100
218/218 [==============================] - 25s 113ms/step - loss: 1.0343 -
accuracy: 0.5447 - val_loss: 1.0842 - val_accuracy: 0.5007
Epoch 24/100
218/218 [==============================] - 25s 116ms/step - loss: 1.0207 -
accuracy: 0.5468 - val_loss: 1.1349 - val_accuracy: 0.4713
Epoch 25/100
218/218 [==============================] - 25s 116ms/step - loss: 1.0206 -
accuracy: 0.5563 - val_loss: 1.1191 - val_accuracy: 0.4733
Epoch 26/100
218/218 [==============================] - 27s 124ms/step - loss: 1.0016 -
accuracy: 0.5619 - val_loss: 1.0863 - val_accuracy: 0.4967
Epoch 27/100
218/218 [==============================] - 25s 117ms/step - loss: 1.0061 -
accuracy: 0.5585 - val_loss: 1.0849 - val_accuracy: 0.4847
Epoch 28/100
218/218 [==============================] - 25s 117ms/step - loss: 0.9898 -
accuracy: 0.5644 - val_loss: 1.0872 - val_accuracy: 0.4927
Epoch 29/100
218/218 [==============================] - 25s 116ms/step - loss: 0.9812 -
accuracy: 0.5615 - val_loss: 1.1203 - val_accuracy: 0.4833
Epoch 30/100
218/218 [==============================] - 26s 119ms/step - loss: 0.9754 -
accuracy: 0.5721 - val_loss: 1.0724 - val_accuracy: 0.5260
Epoch 31/100
218/218 [==============================] - 26s 119ms/step - loss: 0.9756 -
accuracy: 0.5720 - val_loss: 1.1420 - val_accuracy: 0.4860
Epoch 32/100
218/218 [==============================] - 26s 118ms/step - loss: 0.9658 -
accuracy: 0.5843 - val_loss: 1.0819 - val_accuracy: 0.4993
Epoch 33/100
218/218 [==============================] - 29s 135ms/step - loss: 0.9469 -
accuracy: 0.5874 - val_loss: 1.0912 - val_accuracy: 0.5160
Epoch 34/100
218/218 [==============================] - 39s 178ms/step - loss: 0.9495 -
accuracy: 0.5861 - val_loss: 1.0730 - val_accuracy: 0.5173
Epoch 35/100
218/218 [==============================] - 28s 126ms/step - loss: 0.9444 -
accuracy: 0.5948 - val_loss: 1.1131 - val_accuracy: 0.5193
Epoch 36/100
218/218 [==============================] - 27s 122ms/step - loss: 0.9452 -
```

```
accuracy: 0.5922 - val_loss: 1.0920 - val_accuracy: 0.5173
Epoch 37/100
218/218 [==============================] - 25s 114ms/step - loss: 0.9203 -
accuracy: 0.6010 - val_loss: 1.0875 - val_accuracy: 0.5147
Epoch 38/100
218/218 [==============================] - 28s 130ms/step - loss: 0.9207 -
accuracy: 0.6080 - val_loss: 1.1145 - val_accuracy: 0.5273
Epoch 39/100
218/218 [==============================] - 33s 151ms/step - loss: 0.9098 -
accuracy: 0.6092 - val_loss: 1.1204 - val_accuracy: 0.5133
Epoch 40/100
218/218 [==============================] - 37s 172ms/step - loss: 0.9177 -
accuracy: 0.6043 - val_loss: 1.1075 - val_accuracy: 0.5213
Epoch 41/100
218/218 [==============================] - 39s 178ms/step - loss: 0.9066 -
accuracy: 0.6154 - val_loss: 1.0882 - val_accuracy: 0.5380
Epoch 42/100
218/218 [==============================] - 39s 180ms/step - loss: 0.8933 -
accuracy: 0.6157 - val_loss: 1.0898 - val_accuracy: 0.5267
Epoch 43/100
218/218 [==============================] - 43s 195ms/step - loss: 0.8898 -
accuracy: 0.6261 - val_loss: 1.1282 - val_accuracy: 0.5333
Epoch 44/100
218/218 [==============================] - 36s 167ms/step - loss: 0.8853 -
accuracy: 0.6177 - val_loss: 1.1274 - val_accuracy: 0.5453
Epoch 45/100
218/218 [==============================] - 26s 121ms/step - loss: 0.8816 -
accuracy: 0.6224 - val_loss: 1.1006 - val_accuracy: 0.5440
Epoch 46/100
218/218 [==============================] - 28s 128ms/step - loss: 0.8881 -
accuracy: 0.6332 - val_loss: 1.0269 - val_accuracy: 0.6000
Epoch 47/100
218/218 [==============================] - 34s 157ms/step - loss: 0.8632 -
accuracy: 0.6425 - val_loss: 1.1025 - val_accuracy: 0.5507
Epoch 48/100
218/218 [==============================] - 46s 213ms/step - loss: 0.8664 -
accuracy: 0.6392 - val_loss: 1.0858 - val_accuracy: 0.5747
Epoch 49/100
218/218 [==============================] - 44s 200ms/step - loss: 0.8622 -
accuracy: 0.6404 - val_loss: 1.0234 - val_accuracy: 0.5833
Epoch 50/100
218/218 [==============================] - 42s 194ms/step - loss: 0.8523 -
accuracy: 0.6526 - val_loss: 1.0947 - val_accuracy: 0.5593
Epoch 51/100
218/218 [==============================] - 40s 183ms/step - loss: 0.8420 -
accuracy: 0.6557 - val_loss: 1.0424 - val_accuracy: 0.5860
Epoch 52/100
218/218 [==============================] - 43s 196ms/step - loss: 0.8377 -
accuracy: 0.6582 - val_loss: 1.0468 - val_accuracy: 0.5927
Epoch 53/100
218/218 [==============================] - 43s 196ms/step - loss: 0.8355 -
accuracy: 0.6658 - val_loss: 1.1237 - val_accuracy: 0.5513
Epoch 54/100
218/218 [==============================] - 41s 189ms/step - loss: 0.8258 -
accuracy: 0.6690 - val_loss: 1.0386 - val_accuracy: 0.6073
Epoch 55/100
```

```
218/218 [==============================] - 41s 188ms/step - loss: 0.8243 -
accuracy: 0.6658 - val_loss: 1.0317 - val_accuracy: 0.6053
Epoch 56/100
218/218 [==============================] - 38s 173ms/step - loss: 0.8162 -
accuracy: 0.6700 - val_loss: 1.0584 - val_accuracy: 0.5920
Epoch 57/100
218/218 [==============================] - 38s 172ms/step - loss: 0.8020 -
accuracy: 0.6823 - val_loss: 1.0689 - val_accuracy: 0.6193
Epoch 58/100
218/218 [==============================] - 38s 173ms/step - loss: 0.8004 -
accuracy: 0.6826 - val_loss: 1.0095 - val_accuracy: 0.6267
Epoch 59/100
218/218 [==============================] - 36s 166ms/step - loss: 0.7959 -
accuracy: 0.6931 - val_loss: 1.0293 - val_accuracy: 0.6320
Epoch 60/100
218/218 [==============================] - 45s 204ms/step - loss: 0.8029 -
accuracy: 0.6886 - val_loss: 0.9983 - val_accuracy: 0.6167
Epoch 61/100
218/218 [==============================] - 41s 190ms/step - loss: 0.7871 -
accuracy: 0.6935 - val_loss: 0.9741 - val_accuracy: 0.6433
Epoch 62/100
218/218 [==============================] - 40s 183ms/step - loss: 0.7792 -
accuracy: 0.6934 - val_loss: 1.0127 - val_accuracy: 0.6187
Epoch 63/100
218/218 [==============================] - 41s 190ms/step - loss: 0.7799 -
accuracy: 0.6950 - val_loss: 0.9592 - val_accuracy: 0.6427
Epoch 64/100
218/218 [==============================] - 41s 189ms/step - loss: 0.7686 -
accuracy: 0.6986 - val_loss: 0.9687 - val_accuracy: 0.6373
Epoch 65/100
218/218 [==============================] - 41s 189ms/step - loss: 0.7457 -
accuracy: 0.7108 - val_loss: 0.9956 - val_accuracy: 0.6433
Epoch 66/100
218/218 [==============================] - 42s 194ms/step - loss: 0.7464 -
accuracy: 0.7151 - val_loss: 1.0392 - val_accuracy: 0.6293
Epoch 67/100
218/218 [==============================] - 42s 193ms/step - loss: 0.7421 -
accuracy: 0.7152 - val_loss: 1.0770 - val_accuracy: 0.6227
Epoch 68/100
218/218 [==============================] - 43s 197ms/step - loss: 0.7391 -
accuracy: 0.7131 - val_loss: 1.0092 - val_accuracy: 0.6207
Epoch 69/100
218/218 [==============================] - 40s 185ms/step - loss: 0.7318 -
accuracy: 0.7220 - val_loss: 1.0386 - val_accuracy: 0.6273
Epoch 70/100
218/218 [==============================] - 37s 172ms/step - loss: 0.7245 -
accuracy: 0.7266 - val_loss: 0.9895 - val_accuracy: 0.6320
Epoch 71/100
218/218 [==============================] - 24s 111ms/step - loss: 0.7143 -
accuracy: 0.7325 - val_loss: 0.9589 - val_accuracy: 0.6407
Epoch 72/100
218/218 [==============================] - 25s 117ms/step - loss: 0.7122 -
accuracy: 0.7282 - val_loss: 0.9545 - val_accuracy: 0.6407
Epoch 73/100
218/218 [==============================] - 25s 117ms/step - loss: 0.7008 -
accuracy: 0.7346 - val_loss: 0.9777 - val_accuracy: 0.6527
```

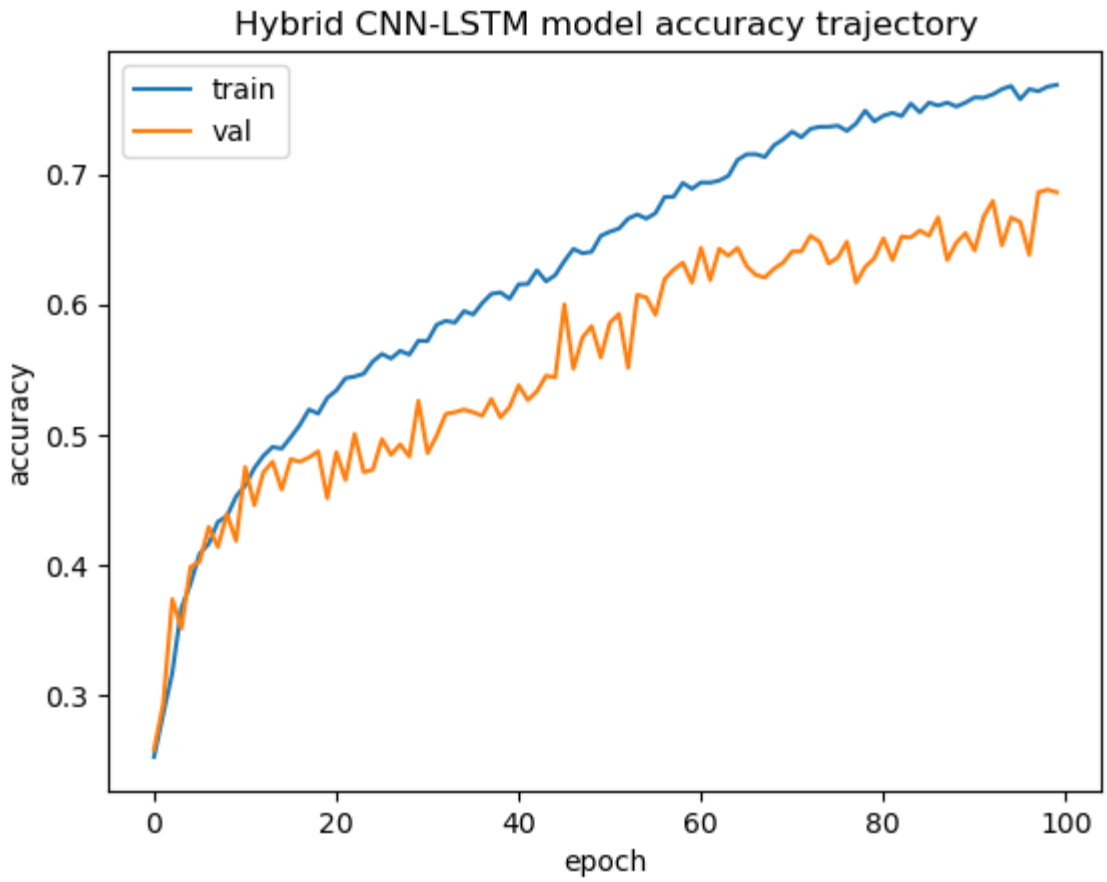
```
Epoch 74/100
218/218 [==============================] - 25s 112ms/step - loss: 0.6979 -
accuracy: 0.7362 - val_loss: 0.9727 - val_accuracy: 0.6480
Epoch 75/100
218/218 [==============================] - 24s 111ms/step - loss: 0.6917 -
accuracy: 0.7362 - val_loss: 0.9871 - val_accuracy: 0.6313
Epoch 76/100
218/218 [==============================] - 22s 99ms/step - loss: 0.6941 - a
ccuracy: 0.7371 - val_loss: 0.9648 - val_accuracy: 0.6360
Epoch 77/100
218/218 [==============================] - 19s 85ms/step - loss: 0.7047 - a
ccuracy: 0.7330 - val_loss: 0.9653 - val_accuracy: 0.6480
Epoch 78/100
218/218 [==============================] - 18s 84ms/step - loss: 0.6903 - a
ccuracy: 0.7386 - val_loss: 1.0132 - val_accuracy: 0.6167
Epoch 79/100
218/218 [==============================] - 18s 84ms/step - loss: 0.6685 - a
ccuracy: 0.7486 - val_loss: 1.0329 - val_accuracy: 0.6287
Epoch 80/100
218/218 [==============================] - 18s 85ms/step - loss: 0.6888 - a
ccuracy: 0.7404 - val_loss: 0.9615 - val_accuracy: 0.6353
Epoch 81/100
218/218 [==============================] - 19s 85ms/step - loss: 0.6706 - a
ccuracy: 0.7447 - val_loss: 0.9342 - val_accuracy: 0.6507
Epoch 82/100
218/218 [==============================] - 18s 84ms/step - loss: 0.6798 - a
ccuracy: 0.7468 - val_loss: 0.9883 - val_accuracy: 0.6340
Epoch 83/100
218/218 [==============================] - 18s 83ms/step - loss: 0.6730 - a
ccuracy: 0.7444 - val_loss: 0.9681 - val_accuracy: 0.6520
Epoch 84/100
218/218 [==============================] - 19s 85ms/step - loss: 0.6567 - a
ccuracy: 0.7540 - val_loss: 0.9574 - val_accuracy: 0.6513
Epoch 85/100
218/218 [==============================] - 18s 84ms/step - loss: 0.6687 - a
ccuracy: 0.7473 - val_loss: 0.9385 - val_accuracy: 0.6567
Epoch 86/100
218/218 [==============================] - 18s 84ms/step - loss: 0.6514 - a
ccuracy: 0.7547 - val_loss: 0.9621 - val_accuracy: 0.6527
Epoch 87/100
218/218 [==============================] - 16s 71ms/step - loss: 0.6498 - a
ccuracy: 0.7524 - val_loss: 0.9330 - val_accuracy: 0.6667
Epoch 88/100
218/218 [==============================] - 14s 66ms/step - loss: 0.6520 - a
ccuracy: 0.7547 - val_loss: 0.9660 - val_accuracy: 0.6340
Epoch 89/100
218/218 [==============================] - 15s 67ms/step - loss: 0.6533 - a
ccuracy: 0.7517 - val_loss: 0.9687 - val_accuracy: 0.6473
Epoch 90/100
218/218 [==============================] - 15s 66ms/step - loss: 0.6489 - a
ccuracy: 0.7547 - val_loss: 0.9672 - val_accuracy: 0.6547
Epoch 91/100
218/218 [==============================] - 15s 67ms/step - loss: 0.6363 - a
ccuracy: 0.7588 - val_loss: 0.9459 - val_accuracy: 0.6413
Epoch 92/100
218/218 [==============================] - 15s 67ms/step - loss: 0.6472 - a
```

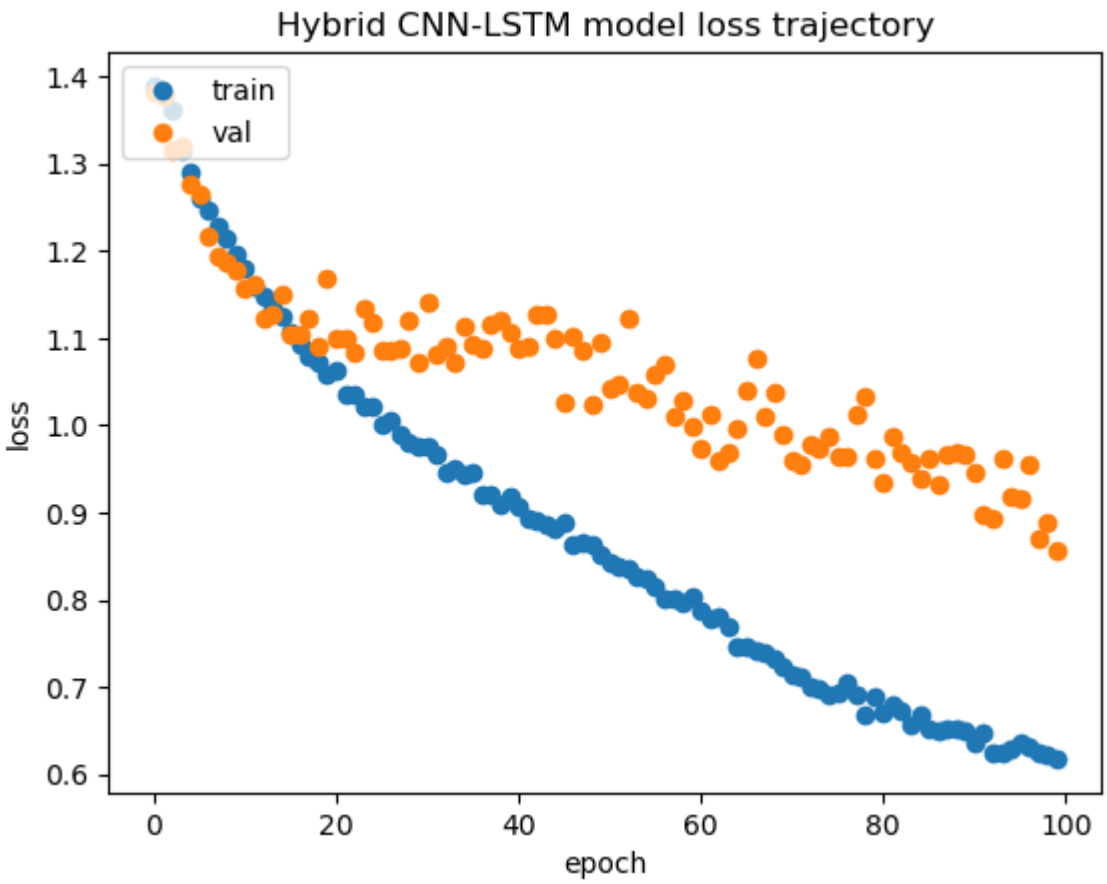
```
ccuracy: 0.7585 - val_loss: 0.8979 - val_accuracy: 0.6673
Epoch 93/100
218/218 [==============================] - 15s 69ms/step - loss: 0.6253 - a
ccuracy: 0.7611 - val_loss: 0.8922 - val_accuracy: 0.6793
Epoch 94/100
218/218 [==============================] - 14s 66ms/step - loss: 0.6239 - a
ccuracy: 0.7651 - val_loss: 0.9626 - val_accuracy: 0.6453
Epoch 95/100
218/218 [==============================] - 13s 62ms/step - loss: 0.6283 - a
ccuracy: 0.7675 - val_loss: 0.9193 - val_accuracy: 0.6667
Epoch 96/100
218/218 [==============================] - 13s 61ms/step - loss: 0.6360 - a
ccuracy: 0.7573 - val_loss: 0.9163 - val_accuracy: 0.6633
Epoch 97/100
218/218 [==============================] - 13s 61ms/step - loss: 0.6319 - a
ccuracy: 0.7651 - val_loss: 0.9545 - val_accuracy: 0.6380
Epoch 98/100
218/218 [==============================] - 13s 61ms/step - loss: 0.6256 - a
ccuracy: 0.7635 - val_loss: 0.8699 - val_accuracy: 0.6860
Epoch 99/100
218/218 [==============================] - 13s 62ms/step - loss: 0.6233 - a
ccuracy: 0.7670 - val_loss: 0.8874 - val_accuracy: 0.6880
Epoch 100/100
218/218 [==============================] - 14s 64ms/step - loss: 0.6180 - a
ccuracy: 0.7684 - val_loss: 0.8565 - val_accuracy: 0.6860
```

(vi)(CNN-LSTM) Visualizing the accuracy and loss trajectory

In [ ]:

```python
import matplotlib.pyplot as plt

# Plotting accuracy trajectory
plt.plot(hybrid_cnn_lstm_model_results.history['accuracy'])
plt.plot(hybrid_cnn_lstm_model_results.history['val_accuracy'])
plt.title('Hybrid CNN-LSTM model accuracy trajectory')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

# Plotting loss trajectory
plt.plot(hybrid_cnn_lstm_model_results.history['loss'],'o')
plt.plot(hybrid_cnn_lstm_model_results.history['val_loss'],'o')
plt.title('Hybrid CNN-LSTM model loss trajectory')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```


Hybrid CNN-LSTM model accuracy trajectory
train
val
accuracy
epoch
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100


Hybrid CNN-LSTM model loss trajectory
train
val
loss
epoch
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
1.4
0
20
40
60
80
100

(vii)(CNN-LSTM) Testing the performance of the hybrid CNN-LSTM model on the held out test set

In [ ]:
```
## Testing the hybrid CNN-LSTM model

hybrid_cnn_lstm_score = hybrid_cnn_lstm_model.evaluate(x_test, y_test, verbo
print('Test accuracy of the hybrid CNN-LSTM model:',hybrid_cnn_lstm_score[1]
```

```
Test accuracy of the hybrid CNN-LSTM model: 0.6834085583686829
```

In [ ]:

CNN-SeqSelfAttention

This notebook was inspired to Tonmoy with some attempts to tune by us

(i) Importing the necessary packages

In [ ]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
import keras
from keras_self_attention import SeqSelfAttention

from keras import layers
from keras.models import Sequential,Model
from keras.layers import Dense, Activation, Flatten,Dropout, MultiHeadAttent
from keras.layers import Conv2D,LSTM,BatchNormalization,MaxPooling2D,Reshape
from keras.utils import to_categorical
import matplotlib.pyplot as plt

tf.compat.v1.random.set_random_seed(0)
```

(ii) Preprocessing the dataset and preparing the training, validation, and test datasets

In [ ]:

```
def data_prep(X,y,sub_sample,average,noise):

    total_X = None
    total_y = None

    # Trimming the data (sample,22,1000) -> (sample,22,500)
    X = X[:,:,0:500]
    print('Shape of X after trimming:',X.shape)

    # Maxpooling the data (sample,22,1000) -> (sample,22,500/sub_sample)
    X_max = np.max(X.reshape(X.shape[0], X.shape[1], -1, sub_sample), axis=3


    total_X = X_max
    total_y = y
    print('Shape of X after maxpooling:',total_X.shape)

    # Averaging + noise
    X_average = np.mean(X.reshape(X.shape[0], X.shape[1], -1, average),axis=
    X_average = X_average + np.random.normal(0.0, 0.5, X_average.shape)

    total_X = np.vstack((total_X, X_average))
    total_y = np.hstack((total_y, y))
    print('Shape of X after averaging+noise and concatenating:',total_X.shap

    # Subsampling

    for i in range(sub_sample):

        X_subsample = X[:, :, i::sub_sample] + \
                            (np.random.normal(0.0, 0.5, X[:, :,i::sub_sample

        total_X = np.vstack((total_X, X_subsample))
        total_y = np.hstack((total_y, y))


    print('Shape of X after subsampling and concatenating:',total_X.shape)
    return total_X,total_y
```

In [ ]:

```
## Loading the dataset


X_test = np.load("X_test.npy")
y_test = np.load("y_test.npy")
person_train_valid = np.load("person_train_valid.npy")
X_train_valid = np.load("X_train_valid.npy")
y_train_valid = np.load("y_train_valid.npy")
person_test = np.load("person_test.npy")

## Adjusting the labels so that

# Cue onset left - 0
# Cue onset right - 1
# Cue onset foot - 2
# Cue onset tongue - 3

y_train_valid -= 769
y_test -= 769


## Random splitting and reshaping the data
# First generating the training and validation indices using random splittin

ind_valid = np.random.choice(2115, 375, replace=False)
ind_train = np.array(list(set(range(2115)).difference(set(ind_valid))))

# Creating the training and validation sets using the generated indices
(X_train, X_valid) = X_train_valid[ind_train], X_train_valid[ind_valid]
(y_train, y_valid) = y_train_valid[ind_train], y_train_valid[ind_valid]


## Preprocessing the dataset
x_train,y_train = data_prep(X_train,y_train,2,2,True)
x_valid,y_valid = data_prep(X_valid,y_valid,2,2,True)
X_test_prep,y_test_prep = data_prep(X_test,y_test,2,2,True)


print('Shape of training set:',x_train.shape)
print('Shape of validation set:',x_valid.shape)
print('Shape of training labels:',y_train.shape)
print('Shape of validation labels:',y_valid.shape)
print('Shape of testing set:',X_test_prep.shape)
print('Shape of testing labels:',y_test_prep.shape)


# Converting the labels to categorical variables for multiclass classificati
y_train = to_categorical(y_train, 4)
y_valid = to_categorical(y_valid, 4)
y_test = to_categorical(y_test_prep, 4)
print('Shape of training labels after categorical conversion:',y_train.shape
print('Shape of validation labels after categorical conversion:',y_valid.sha
print('Shape of test labels after categorical conversion:',y_test.shape)

# Adding width of the segment to be 1
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], x_train.shape[
```

```
x_valid = x_valid.reshape(x_valid.shape[0], x_valid.shape[1], x_train.shape[
x_test = X_test_prep.reshape(X_test_prep.shape[0], X_test_prep.shape[1], X_t
print('Shape of training set after adding width info:',x_train.shape)
print('Shape of validation set after adding width info:',x_valid.shape)
print('Shape of test set after adding width info:',x_test.shape)


# Reshaping the training and validation dataset
x_train = np.swapaxes(x_train, 1,3)
x_train = np.swapaxes(x_train, 1,2)
x_valid = np.swapaxes(x_valid, 1,3)
x_valid = np.swapaxes(x_valid, 1,2)
x_test = np.swapaxes(x_test, 1,3)
x_test = np.swapaxes(x_test, 1,2)
print('Shape of training set after dimension reshaping:',x_train.shape)
print('Shape of validation set after dimension reshaping:',x_valid.shape)
print('Shape of test set after dimension reshaping:',x_test.shape)

keras.backend.clear_session()
```

```
Shape of X after trimming: (1740, 22, 500)
Shape of X after maxpooling: (1740, 22, 250)
Shape of X after averaging+noise and concatenating: (3480, 22, 250)
Shape of X after subsampling and concatenating: (6960, 22, 250)
Shape of X after trimming: (375, 22, 500)
Shape of X after maxpooling: (375, 22, 250)
Shape of X after averaging+noise and concatenating: (750, 22, 250)
Shape of X after subsampling and concatenating: (1500, 22, 250)
Shape of X after trimming: (443, 22, 500)
Shape of X after maxpooling: (443, 22, 250)
Shape of X after averaging+noise and concatenating: (886, 22, 250)
Shape of X after subsampling and concatenating: (1772, 22, 250)
Shape of training set: (6960, 22, 250)
Shape of validation set: (1500, 22, 250)
Shape of training labels: (6960,)
Shape of validation labels: (1500,)
Shape of testing set: (1772, 22, 250)
Shape of testing labels: (1772,)
Shape of training labels after categorical conversion: (6960, 4)
Shape of validation labels after categorical conversion: (1500, 4)
Shape of test labels after categorical conversion: (1772, 4)
Shape of training set after adding width info: (6960, 22, 250, 1)
Shape of validation set after adding width info: (1500, 22, 250, 1)
Shape of test set after adding width info: (1772, 22, 250, 1)
Shape of training set after dimension reshaping: (6960, 250, 1, 22)
Shape of validation set after dimension reshaping: (1500, 250, 1, 22)
Shape of test set after dimension reshaping: (1772, 250, 1, 22)
```

(iii)(CNN-Self-Attention) Defining the architecture of the hybrid CNN-LSTM model

In [ ]:
```
# Building the CNN model using functional class
def build_model():
# models = []

    n_frames = 250
    n_channels = 22
    # Conv. block 1
    In1 = keras.Input(shape =(250,1,22) )
    c1 = Conv2D(filters=30, kernel_size=(11,1), padding='same', activation='
    p1 =  MaxPooling2D(pool_size=(4,1), padding='same')(c1) # Read the keras
    b1 = BatchNormalization()(p1)
    d1 = Dropout(0.8)(b1)

    # Conv. block 2
    c2 = Conv2D(filters=60, kernel_size=(9,1), padding='same', activation='s
    p2 =  MaxPooling2D(pool_size=(4,1), padding='same')(c2) # Read the keras
    b2 = BatchNormalization()(p2)
    d2 = Dropout(0.7)(b2)

    # Conv. block 3
    c3 = Conv2D(filters=120, kernel_size=(5,1), padding='same', activation='
    p3 =  MaxPooling2D(pool_size=(4,1), padding='same')(c3) # Read the keras
    b3 = BatchNormalization()(p3)
    d3 = Dropout(0.6)(b3)

    c4 = Conv2D(filters=240, kernel_size=(3,1), padding='same', activation='
    p4 =  MaxPooling2D(pool_size=(4,1), padding='same')(c4) # Read the keras
    b4 = BatchNormalization()(p4)
    d4 = Dropout(0.5)(b4)

    # self attention block
    selfatt = tf.squeeze(d4, axis=2)
    selfatt = SeqSelfAttention(attention_activation='gelu')(selfatt)
    selfatt = GlobalAveragePooling1D()(selfatt)

    # Add fully connected layers
    fc1 = Dense(64, activation='relu')(selfatt)
    fc1_dropout = Dropout(rate=0.5)(fc1)
    fc2 = Dense(4, activation='softmax')(fc1_dropout)

    # Define the final model
    final_model = Model(inputs=In1, outputs=[fc2])

    # # Printing the model summary
    final_model.compile(loss='categorical_crossentropy',
            optimizer='adam',
            metrics=['accuracy'])
    final_model.summary()
    return final_model
```

(iv) (CNN-LSTM) Defining the hyperparameters of the hybrid CNN-LSTM model

In [ ]:
```
# Model parameters
epochs = 300
initial_learning_rate = 1e-3
decay_steps = 1000
decay_rate = 0.99

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate,
    decay_steps=decay_steps,
    decay_rate=decay_rate
)

optimizer = keras.optimizers.Adam(learning_rate=lr_schedule)
```

(v)Attention at the end

In [ ]:
```
# Compiling the model
keras.backend.clear_session()

# Training and validating the model

batch_sizes = [32,64,128]
import matplotlib.pyplot as plt

hybrid_cnn_lstm_model = build_model()
# Compiling the model
hybrid_cnn_lstm_model.compile(loss='categorical_crossentropy',
               optimizer=optimizer,
               metrics=['accuracy'])

# Training and validating the model
acc = 0.73
class myCallback(tf.keras.callbacks.Callback):
        def on_epoch_end(self, epoch, logs={}):
                if((logs.get('val_accuracy') > acc)):
                        print("\nval accuracy high enough and difference bet
                        self.model.stop_training = True

# Instantiate a callback object
callback = myCallback()
# callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)
hybrid_cnn_lstm_model_results = hybrid_cnn_lstm_model.fit(x_train,
                y_train,
                batch_size=32,
                epochs=epochs,
                validation_data=(x_valid, y_valid),
                verbose=True,
                callbacks = [callback]
                )
```

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 250, 1, 22)]	0
conv2d (Conv2D)	(None, 250, 1, 30)	7290
max_pooling2d (MaxPooling2D)	(None, 63, 1, 30)	0
batch_normalization (BatchNormalization)	(None, 63, 1, 30)	120
dropout (Dropout)	(None, 63, 1, 30)	0
conv2d_1 (Conv2D)	(None, 63, 1, 60)	16260
max_pooling2d_1 (MaxPooling2D)	(None, 16, 1, 60)	0
batch_normalization_1 (BatchNormalization)	(None, 16, 1, 60)	240
dropout_1 (Dropout)	(None, 16, 1, 60)	0
conv2d_2 (Conv2D)	(None, 16, 1, 120)	36120
max_pooling2d_2 (MaxPooling2D)	(None, 4, 1, 120)	0
batch_normalization_2 (BatchNormalization)	(None, 4, 1, 120)	480
dropout_2 (Dropout)	(None, 4, 1, 120)	0
conv2d_3 (Conv2D)	(None, 4, 1, 240)	86640
max_pooling2d_3 (MaxPooling2D)	(None, 1, 1, 240)	0
batch_normalization_3 (BatchNormalization)	(None, 1, 1, 240)	960
dropout_3 (Dropout)	(None, 1, 1, 240)	0
tf.compat.v1.squeeze (TFOpLambda)	(None, 1, 240)	0
seq_self_attention (SeqSelfAttention)	(None, 1, 240)	15425
global_average_pooling1d (GlobalAveragePooling1D)	(None, 240)	0
dense (Dense)	(None, 64)	15424

```
 dropout_4 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 4)                 260

=================================================================
Total params: 179,219
Trainable params: 178,319
Non-trainable params: 900
_________________________________________________________________
Epoch 1/300
218/218 [==============================] - 4s 14ms/step - loss: 2.0168 - ac
curacy: 0.2511 - val_loss: 1.4256 - val_accuracy: 0.2720
Epoch 2/300
218/218 [==============================] - 3s 15ms/step - loss: 1.5723 - ac
curacy: 0.2645 - val_loss: 1.3932 - val_accuracy: 0.2720
Epoch 3/300
218/218 [==============================] - 3s 14ms/step - loss: 1.4544 - ac
curacy: 0.2670 - val_loss: 1.3795 - val_accuracy: 0.2667
Epoch 4/300
218/218 [==============================] - 4s 18ms/step - loss: 1.4107 - ac
curacy: 0.2707 - val_loss: 1.3814 - val_accuracy: 0.2680
Epoch 5/300
218/218 [==============================] - 3s 15ms/step - loss: 1.4007 - ac
curacy: 0.2740 - val_loss: 1.3816 - val_accuracy: 0.2740
Epoch 6/300
218/218 [==============================] - 3s 15ms/step - loss: 1.3872 - ac
curacy: 0.2763 - val_loss: 1.3782 - val_accuracy: 0.2807
Epoch 7/300
218/218 [==============================] - 4s 17ms/step - loss: 1.3792 - ac
curacy: 0.2865 - val_loss: 1.3789 - val_accuracy: 0.2767
Epoch 8/300
218/218 [==============================] - 3s 15ms/step - loss: 1.3788 - ac
curacy: 0.2772 - val_loss: 1.3762 - val_accuracy: 0.2867
Epoch 9/300
218/218 [==============================] - 3s 14ms/step - loss: 1.3747 - ac
curacy: 0.2912 - val_loss: 1.3781 - val_accuracy: 0.2773
Epoch 10/300
218/218 [==============================] - 3s 14ms/step - loss: 1.3723 - ac
curacy: 0.2920 - val_loss: 1.3802 - val_accuracy: 0.2993
Epoch 11/300
218/218 [==============================] - 3s 14ms/step - loss: 1.3601 - ac
curacy: 0.3195 - val_loss: 1.3610 - val_accuracy: 0.2960
Epoch 12/300
218/218 [==============================] - 3s 14ms/step - loss: 1.3547 - ac
curacy: 0.3236 - val_loss: 1.3573 - val_accuracy: 0.3060
Epoch 13/300
218/218 [==============================] - 3s 14ms/step - loss: 1.3361 - ac
curacy: 0.3437 - val_loss: 1.3372 - val_accuracy: 0.3127
Epoch 14/300
218/218 [==============================] - 3s 14ms/step - loss: 1.3325 - ac
curacy: 0.3458 - val_loss: 1.3455 - val_accuracy: 0.3107
Epoch 15/300
218/218 [==============================] - 3s 14ms/step - loss: 1.3198 - ac
curacy: 0.3612 - val_loss: 1.3095 - val_accuracy: 0.3840
Epoch 16/300
```

```
218/218 [==============================] - 3s 14ms/step - loss: 1.3052 - ac
curacy: 0.3761 - val_loss: 1.3052 - val_accuracy: 0.3700
Epoch 17/300
218/218 [==============================] - 3s 14ms/step - loss: 1.2919 - ac
curacy: 0.3787 - val_loss: 1.3137 - val_accuracy: 0.3400
Epoch 18/300
218/218 [==============================] - 3s 14ms/step - loss: 1.2799 - ac
curacy: 0.4009 - val_loss: 1.2754 - val_accuracy: 0.3900
Epoch 19/300
218/218 [==============================] - 3s 14ms/step - loss: 1.2698 - ac
curacy: 0.4046 - val_loss: 1.3078 - val_accuracy: 0.3700
Epoch 20/300
218/218 [==============================] - 3s 14ms/step - loss: 1.2562 - ac
curacy: 0.4109 - val_loss: 1.2546 - val_accuracy: 0.3913
Epoch 21/300
218/218 [==============================] - 3s 14ms/step - loss: 1.2535 - ac
curacy: 0.4190 - val_loss: 1.2611 - val_accuracy: 0.3773
Epoch 22/300
218/218 [==============================] - 3s 14ms/step - loss: 1.2408 - ac
curacy: 0.4249 - val_loss: 1.2488 - val_accuracy: 0.4073
Epoch 23/300
218/218 [==============================] - 3s 14ms/step - loss: 1.2230 - ac
curacy: 0.4425 - val_loss: 1.2628 - val_accuracy: 0.3727
Epoch 24/300
218/218 [==============================] - 3s 14ms/step - loss: 1.2225 - ac
curacy: 0.4476 - val_loss: 1.2547 - val_accuracy: 0.3733
Epoch 25/300
218/218 [==============================] - 3s 15ms/step - loss: 1.2171 - ac
curacy: 0.4441 - val_loss: 1.2371 - val_accuracy: 0.4087
Epoch 26/300
218/218 [==============================] - 3s 14ms/step - loss: 1.2101 - ac
curacy: 0.4474 - val_loss: 1.2268 - val_accuracy: 0.4227
Epoch 27/300
218/218 [==============================] - 3s 15ms/step - loss: 1.2093 - ac
curacy: 0.4460 - val_loss: 1.2175 - val_accuracy: 0.4333
Epoch 28/300
218/218 [==============================] - 3s 16ms/step - loss: 1.2044 - ac
curacy: 0.4556 - val_loss: 1.2295 - val_accuracy: 0.4493
Epoch 29/300
218/218 [==============================] - 3s 14ms/step - loss: 1.1899 - ac
curacy: 0.4615 - val_loss: 1.2126 - val_accuracy: 0.4360
Epoch 30/300
218/218 [==============================] - 3s 15ms/step - loss: 1.2003 - ac
curacy: 0.4537 - val_loss: 1.2128 - val_accuracy: 0.4593
Epoch 31/300
218/218 [==============================] - 3s 15ms/step - loss: 1.1867 - ac
curacy: 0.4664 - val_loss: 1.2029 - val_accuracy: 0.4687
Epoch 32/300
218/218 [==============================] - 3s 15ms/step - loss: 1.1817 - ac
curacy: 0.4733 - val_loss: 1.2153 - val_accuracy: 0.4193
Epoch 33/300
218/218 [==============================] - 3s 14ms/step - loss: 1.1787 - ac
curacy: 0.4726 - val_loss: 1.1927 - val_accuracy: 0.4700
Epoch 34/300
218/218 [==============================] - 3s 15ms/step - loss: 1.1686 - ac
curacy: 0.4772 - val_loss: 1.2066 - val_accuracy: 0.4473
```

```
Epoch 35/300
218/218 [==============================] - 3s 15ms/step - loss: 1.1747 - ac
curacy: 0.4825 - val_loss: 1.1915 - val_accuracy: 0.4747
Epoch 36/300
218/218 [==============================] - 3s 15ms/step - loss: 1.1660 - ac
curacy: 0.4819 - val_loss: 1.1909 - val_accuracy: 0.4513
Epoch 37/300
218/218 [==============================] - 3s 14ms/step - loss: 1.1577 - ac
curacy: 0.4825 - val_loss: 1.2013 - val_accuracy: 0.4567
Epoch 38/300
218/218 [==============================] - 3s 14ms/step - loss: 1.1555 - ac
curacy: 0.4898 - val_loss: 1.1875 - val_accuracy: 0.4800
Epoch 39/300
218/218 [==============================] - 3s 14ms/step - loss: 1.1356 - ac
curacy: 0.5032 - val_loss: 1.1509 - val_accuracy: 0.4987
Epoch 40/300
218/218 [==============================] - 3s 14ms/step - loss: 1.1382 - ac
curacy: 0.5109 - val_loss: 1.1679 - val_accuracy: 0.4813
Epoch 41/300
218/218 [==============================] - 3s 14ms/step - loss: 1.1493 - ac
curacy: 0.5033 - val_loss: 1.1466 - val_accuracy: 0.5233
Epoch 42/300
218/218 [==============================] - 3s 14ms/step - loss: 1.1235 - ac
curacy: 0.5155 - val_loss: 1.1349 - val_accuracy: 0.5247
Epoch 43/300
218/218 [==============================] - 4s 16ms/step - loss: 1.1241 - ac
curacy: 0.5168 - val_loss: 1.1264 - val_accuracy: 0.5153
Epoch 44/300
218/218 [==============================] - 3s 14ms/step - loss: 1.1172 - ac
curacy: 0.5240 - val_loss: 1.1199 - val_accuracy: 0.5347
Epoch 45/300
218/218 [==============================] - 3s 15ms/step - loss: 1.1101 - ac
curacy: 0.5306 - val_loss: 1.1233 - val_accuracy: 0.5227
Epoch 46/300
218/218 [==============================] - 3s 15ms/step - loss: 1.1086 - ac
curacy: 0.5280 - val_loss: 1.1228 - val_accuracy: 0.5260
Epoch 47/300
218/218 [==============================] - 3s 15ms/step - loss: 1.1130 - ac
curacy: 0.5323 - val_loss: 1.1104 - val_accuracy: 0.5173
Epoch 48/300
218/218 [==============================] - 3s 15ms/step - loss: 1.0981 - ac
curacy: 0.5421 - val_loss: 1.1077 - val_accuracy: 0.5333
Epoch 49/300
218/218 [==============================] - 3s 15ms/step - loss: 1.1007 - ac
curacy: 0.5297 - val_loss: 1.1121 - val_accuracy: 0.5320
Epoch 50/300
218/218 [==============================] - 3s 15ms/step - loss: 1.0872 - ac
curacy: 0.5394 - val_loss: 1.0993 - val_accuracy: 0.5600
Epoch 51/300
218/218 [==============================] - 3s 15ms/step - loss: 1.0763 - ac
curacy: 0.5457 - val_loss: 1.0881 - val_accuracy: 0.5567
Epoch 52/300
218/218 [==============================] - 3s 14ms/step - loss: 1.0754 - ac
curacy: 0.5543 - val_loss: 1.0767 - val_accuracy: 0.5593
Epoch 53/300
218/218 [==============================] - 3s 14ms/step - loss: 1.0785 - ac
```

```
curacy: 0.5507 - val_loss: 1.0654 - val_accuracy: 0.5687
Epoch 54/300
218/218 [==============================] - 3s 14ms/step - loss: 1.0780 - ac
curacy: 0.5507 - val_loss: 1.0614 - val_accuracy: 0.5533
Epoch 55/300
218/218 [==============================] - 3s 15ms/step - loss: 1.0595 - ac
curacy: 0.5585 - val_loss: 1.0747 - val_accuracy: 0.5653
Epoch 56/300
218/218 [==============================] - 3s 15ms/step - loss: 1.0625 - ac
curacy: 0.5583 - val_loss: 1.0577 - val_accuracy: 0.5447
Epoch 57/300
218/218 [==============================] - 3s 15ms/step - loss: 1.0499 - ac
curacy: 0.5601 - val_loss: 1.0412 - val_accuracy: 0.5653
Epoch 58/300
218/218 [==============================] - 3s 15ms/step - loss: 1.0552 - ac
curacy: 0.5649 - val_loss: 1.0378 - val_accuracy: 0.5700
Epoch 59/300
218/218 [==============================] - 3s 15ms/step - loss: 1.0462 - ac
curacy: 0.5661 - val_loss: 1.0221 - val_accuracy: 0.5807
Epoch 60/300
218/218 [==============================] - 3s 15ms/step - loss: 1.0331 - ac
curacy: 0.5716 - val_loss: 1.0317 - val_accuracy: 0.5680
Epoch 61/300
218/218 [==============================] - 3s 15ms/step - loss: 1.0399 - ac
curacy: 0.5727 - val_loss: 1.0149 - val_accuracy: 0.5887
Epoch 62/300
218/218 [==============================] - 3s 15ms/step - loss: 1.0330 - ac
curacy: 0.5753 - val_loss: 1.0180 - val_accuracy: 0.5827
Epoch 63/300
218/218 [==============================] - 3s 15ms/step - loss: 1.0382 - ac
curacy: 0.5823 - val_loss: 1.0146 - val_accuracy: 0.5827
Epoch 64/300
218/218 [==============================] - 3s 16ms/step - loss: 1.0276 - ac
curacy: 0.5770 - val_loss: 0.9921 - val_accuracy: 0.5913
Epoch 65/300
218/218 [==============================] - 3s 14ms/step - loss: 1.0179 - ac
curacy: 0.5879 - val_loss: 1.0140 - val_accuracy: 0.5673
Epoch 66/300
218/218 [==============================] - 3s 14ms/step - loss: 1.0159 - ac
curacy: 0.5812 - val_loss: 0.9827 - val_accuracy: 0.5973
Epoch 67/300
218/218 [==============================] - 3s 14ms/step - loss: 1.0095 - ac
curacy: 0.5891 - val_loss: 0.9887 - val_accuracy: 0.6107
Epoch 68/300
218/218 [==============================] - 3s 15ms/step - loss: 1.0017 - ac
curacy: 0.5898 - val_loss: 0.9859 - val_accuracy: 0.5967
Epoch 69/300
218/218 [==============================] - 3s 15ms/step - loss: 0.9987 - ac
curacy: 0.5981 - val_loss: 0.9772 - val_accuracy: 0.6160
Epoch 70/300
218/218 [==============================] - 3s 14ms/step - loss: 1.0044 - ac
curacy: 0.5953 - val_loss: 0.9594 - val_accuracy: 0.6187
Epoch 71/300
218/218 [==============================] - 3s 16ms/step - loss: 0.9993 - ac
curacy: 0.5909 - val_loss: 0.9747 - val_accuracy: 0.6113
Epoch 72/300
```

```
218/218 [==============================] - 3s 15ms/step - loss: 1.0034 - ac
curacy: 0.5914 - val_loss: 0.9672 - val_accuracy: 0.6300
Epoch 73/300
218/218 [==============================] - 3s 16ms/step - loss: 0.9914 - ac
curacy: 0.5953 - val_loss: 0.9646 - val_accuracy: 0.6280
Epoch 74/300
218/218 [==============================] - 3s 15ms/step - loss: 0.9867 - ac
curacy: 0.6026 - val_loss: 0.9425 - val_accuracy: 0.6287
Epoch 75/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9857 - ac
curacy: 0.5980 - val_loss: 0.9492 - val_accuracy: 0.6247
Epoch 76/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9766 - ac
curacy: 0.6019 - val_loss: 0.9416 - val_accuracy: 0.6233
Epoch 77/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9677 - ac
curacy: 0.6069 - val_loss: 0.9531 - val_accuracy: 0.6147
Epoch 78/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9521 - ac
curacy: 0.6114 - val_loss: 0.9284 - val_accuracy: 0.6200
Epoch 79/300
218/218 [==============================] - 3s 15ms/step - loss: 0.9755 - ac
curacy: 0.6098 - val_loss: 0.9257 - val_accuracy: 0.6340
Epoch 80/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9587 - ac
curacy: 0.6096 - val_loss: 0.9299 - val_accuracy: 0.6180
Epoch 81/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9618 - ac
curacy: 0.6122 - val_loss: 0.9382 - val_accuracy: 0.6400
Epoch 82/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9433 - ac
curacy: 0.6191 - val_loss: 0.9188 - val_accuracy: 0.6320
Epoch 83/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9577 - ac
curacy: 0.6111 - val_loss: 0.9246 - val_accuracy: 0.6320
Epoch 84/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9548 - ac
curacy: 0.6128 - val_loss: 0.8958 - val_accuracy: 0.6480
Epoch 85/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9483 - ac
curacy: 0.6263 - val_loss: 0.8905 - val_accuracy: 0.6413
Epoch 86/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9617 - ac
curacy: 0.6108 - val_loss: 0.8969 - val_accuracy: 0.6527
Epoch 87/300
218/218 [==============================] - 3s 15ms/step - loss: 0.9498 - ac
curacy: 0.6267 - val_loss: 0.8967 - val_accuracy: 0.6480
Epoch 88/300
218/218 [==============================] - 3s 15ms/step - loss: 0.9456 - ac
curacy: 0.6134 - val_loss: 0.8981 - val_accuracy: 0.6447
Epoch 89/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9494 - ac
curacy: 0.6132 - val_loss: 0.8953 - val_accuracy: 0.6360
Epoch 90/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9390 - ac
curacy: 0.6233 - val_loss: 0.8784 - val_accuracy: 0.6447
```

```
Epoch 91/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9453 - ac
curacy: 0.6167 - val_loss: 0.8902 - val_accuracy: 0.6420
Epoch 92/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9368 - ac
curacy: 0.6204 - val_loss: 0.8958 - val_accuracy: 0.6353
Epoch 93/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9331 - ac
curacy: 0.6310 - val_loss: 0.8938 - val_accuracy: 0.6447
Epoch 94/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9345 - ac
curacy: 0.6264 - val_loss: 0.8774 - val_accuracy: 0.6480
Epoch 95/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9241 - ac
curacy: 0.6303 - val_loss: 0.8753 - val_accuracy: 0.6587
Epoch 96/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9313 - ac
curacy: 0.6302 - val_loss: 0.8929 - val_accuracy: 0.6447
Epoch 97/300
218/218 [==============================] - 3s 15ms/step - loss: 0.9311 - ac
curacy: 0.6307 - val_loss: 0.8948 - val_accuracy: 0.6427
Epoch 98/300
218/218 [==============================] - 3s 16ms/step - loss: 0.9347 - ac
curacy: 0.6287 - val_loss: 0.8800 - val_accuracy: 0.6373
Epoch 99/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9240 - ac
curacy: 0.6351 - val_loss: 0.8899 - val_accuracy: 0.6407
Epoch 100/300
218/218 [==============================] - 3s 15ms/step - loss: 0.9180 - ac
curacy: 0.6299 - val_loss: 0.8936 - val_accuracy: 0.6380
Epoch 101/300
218/218 [==============================] - 3s 15ms/step - loss: 0.9224 - ac
curacy: 0.6356 - val_loss: 0.8694 - val_accuracy: 0.6573
Epoch 102/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9200 - ac
curacy: 0.6361 - val_loss: 0.8829 - val_accuracy: 0.6427
Epoch 103/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9252 - ac
curacy: 0.6338 - val_loss: 0.8786 - val_accuracy: 0.6393
Epoch 104/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9285 - ac
curacy: 0.6290 - val_loss: 0.8868 - val_accuracy: 0.6493
Epoch 105/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9032 - ac
curacy: 0.6366 - val_loss: 0.8827 - val_accuracy: 0.6333
Epoch 106/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9035 - ac
curacy: 0.6352 - val_loss: 0.8669 - val_accuracy: 0.6700
Epoch 107/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9063 - ac
curacy: 0.6385 - val_loss: 0.8633 - val_accuracy: 0.6533
Epoch 108/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9074 - ac
curacy: 0.6372 - val_loss: 0.8580 - val_accuracy: 0.6527
Epoch 109/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9171 - ac
```

```
curacy: 0.6299 - val_loss: 0.8571 - val_accuracy: 0.6753
Epoch 110/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9001 - ac
curacy: 0.6467 - val_loss: 0.8469 - val_accuracy: 0.6647
Epoch 111/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9083 - ac
curacy: 0.6326 - val_loss: 0.8562 - val_accuracy: 0.6553
Epoch 112/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9064 - ac
curacy: 0.6358 - val_loss: 0.8592 - val_accuracy: 0.6580
Epoch 113/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8957 - ac
curacy: 0.6411 - val_loss: 0.8361 - val_accuracy: 0.6653
Epoch 114/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8912 - ac
curacy: 0.6481 - val_loss: 0.8464 - val_accuracy: 0.6673
Epoch 115/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9073 - ac
curacy: 0.6424 - val_loss: 0.8586 - val_accuracy: 0.6560
Epoch 116/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9029 - ac
curacy: 0.6421 - val_loss: 0.8502 - val_accuracy: 0.6607
Epoch 117/300
218/218 [==============================] - 3s 14ms/step - loss: 0.9008 - ac
curacy: 0.6478 - val_loss: 0.8452 - val_accuracy: 0.6593
Epoch 118/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8920 - ac
curacy: 0.6497 - val_loss: 0.8317 - val_accuracy: 0.6740
Epoch 119/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8969 - ac
curacy: 0.6448 - val_loss: 0.8531 - val_accuracy: 0.6580
Epoch 120/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8928 - ac
curacy: 0.6501 - val_loss: 0.8450 - val_accuracy: 0.6633
Epoch 121/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8928 - ac
curacy: 0.6481 - val_loss: 0.8441 - val_accuracy: 0.6513
Epoch 122/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8966 - ac
curacy: 0.6468 - val_loss: 0.8444 - val_accuracy: 0.6620
Epoch 123/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8912 - ac
curacy: 0.6478 - val_loss: 0.8398 - val_accuracy: 0.6473
Epoch 124/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8901 - ac
curacy: 0.6401 - val_loss: 0.8177 - val_accuracy: 0.6647
Epoch 125/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8904 - ac
curacy: 0.6451 - val_loss: 0.8372 - val_accuracy: 0.6753
Epoch 126/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8898 - ac
curacy: 0.6409 - val_loss: 0.8351 - val_accuracy: 0.6587
Epoch 127/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8872 - ac
curacy: 0.6503 - val_loss: 0.8351 - val_accuracy: 0.6680
Epoch 128/300
```

```
218/218 [==============================] - 3s 14ms/step - loss: 0.8766 - ac
curacy: 0.6579 - val_loss: 0.8316 - val_accuracy: 0.6760
Epoch 129/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8911 - ac
curacy: 0.6480 - val_loss: 0.8288 - val_accuracy: 0.6680
Epoch 130/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8792 - ac
curacy: 0.6438 - val_loss: 0.8332 - val_accuracy: 0.6707
Epoch 131/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8684 - ac
curacy: 0.6559 - val_loss: 0.8165 - val_accuracy: 0.6720
Epoch 132/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8778 - ac
curacy: 0.6536 - val_loss: 0.8223 - val_accuracy: 0.6787
Epoch 133/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8645 - ac
curacy: 0.6628 - val_loss: 0.8365 - val_accuracy: 0.6673
Epoch 134/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8770 - ac
curacy: 0.6537 - val_loss: 0.8285 - val_accuracy: 0.6707
Epoch 135/300
218/218 [==============================] - 4s 16ms/step - loss: 0.8835 - ac
curacy: 0.6532 - val_loss: 0.8322 - val_accuracy: 0.6613
Epoch 136/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8697 - ac
curacy: 0.6570 - val_loss: 0.8322 - val_accuracy: 0.6680
Epoch 137/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8748 - ac
curacy: 0.6537 - val_loss: 0.8191 - val_accuracy: 0.6760
Epoch 138/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8652 - ac
curacy: 0.6569 - val_loss: 0.8331 - val_accuracy: 0.6673
Epoch 139/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8734 - ac
curacy: 0.6566 - val_loss: 0.8306 - val_accuracy: 0.6733
Epoch 140/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8675 - ac
curacy: 0.6539 - val_loss: 0.8267 - val_accuracy: 0.6747
Epoch 141/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8735 - ac
curacy: 0.6595 - val_loss: 0.8348 - val_accuracy: 0.6787
Epoch 142/300
218/218 [==============================] - 3s 16ms/step - loss: 0.8632 - ac
curacy: 0.6612 - val_loss: 0.8191 - val_accuracy: 0.6720
Epoch 143/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8559 - ac
curacy: 0.6605 - val_loss: 0.8340 - val_accuracy: 0.6753
Epoch 144/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8550 - ac
curacy: 0.6609 - val_loss: 0.8101 - val_accuracy: 0.6780
Epoch 145/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8545 - ac
curacy: 0.6638 - val_loss: 0.8196 - val_accuracy: 0.6727
Epoch 146/300
218/218 [==============================] - 3s 16ms/step - loss: 0.8616 - ac
curacy: 0.6639 - val_loss: 0.8012 - val_accuracy: 0.6847
```

```
Epoch 147/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8641 - ac
curacy: 0.6652 - val_loss: 0.8185 - val_accuracy: 0.6847
Epoch 148/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8506 - ac
curacy: 0.6655 - val_loss: 0.8199 - val_accuracy: 0.6747
Epoch 149/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8440 - ac
curacy: 0.6701 - val_loss: 0.8176 - val_accuracy: 0.6807
Epoch 150/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8453 - ac
curacy: 0.6688 - val_loss: 0.8117 - val_accuracy: 0.6873
Epoch 151/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8474 - ac
curacy: 0.6710 - val_loss: 0.8127 - val_accuracy: 0.6727
Epoch 152/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8617 - ac
curacy: 0.6672 - val_loss: 0.8074 - val_accuracy: 0.6873
Epoch 153/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8620 - ac
curacy: 0.6631 - val_loss: 0.8065 - val_accuracy: 0.6840
Epoch 154/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8484 - ac
curacy: 0.6672 - val_loss: 0.8223 - val_accuracy: 0.6720
Epoch 155/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8476 - ac
curacy: 0.6649 - val_loss: 0.8126 - val_accuracy: 0.6733
Epoch 156/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8544 - ac
curacy: 0.6602 - val_loss: 0.8142 - val_accuracy: 0.6833
Epoch 157/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8564 - ac
curacy: 0.6649 - val_loss: 0.8077 - val_accuracy: 0.6787
Epoch 158/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8532 - ac
curacy: 0.6707 - val_loss: 0.8185 - val_accuracy: 0.6720
Epoch 159/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8497 - ac
curacy: 0.6677 - val_loss: 0.8079 - val_accuracy: 0.6827
Epoch 160/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8532 - ac
curacy: 0.6714 - val_loss: 0.8115 - val_accuracy: 0.6793
Epoch 161/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8457 - ac
curacy: 0.6691 - val_loss: 0.8274 - val_accuracy: 0.6800
Epoch 162/300
218/218 [==============================] - 3s 16ms/step - loss: 0.8462 - ac
curacy: 0.6661 - val_loss: 0.8115 - val_accuracy: 0.6907
Epoch 163/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8336 - ac
curacy: 0.6693 - val_loss: 0.8123 - val_accuracy: 0.6787
Epoch 164/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8325 - ac
curacy: 0.6698 - val_loss: 0.7994 - val_accuracy: 0.6847
Epoch 165/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8440 - ac
```

```
curacy: 0.6662 - val_loss: 0.8121 - val_accuracy: 0.6860
Epoch 166/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8322 - ac
curacy: 0.6773 - val_loss: 0.7937 - val_accuracy: 0.6960
Epoch 167/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8472 - ac
curacy: 0.6694 - val_loss: 0.7945 - val_accuracy: 0.6867
Epoch 168/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8404 - ac
curacy: 0.6736 - val_loss: 0.8104 - val_accuracy: 0.6893
Epoch 169/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8144 - ac
curacy: 0.6764 - val_loss: 0.8099 - val_accuracy: 0.6873
Epoch 170/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8482 - ac
curacy: 0.6624 - val_loss: 0.8267 - val_accuracy: 0.6807
Epoch 171/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8246 - ac
curacy: 0.6766 - val_loss: 0.8021 - val_accuracy: 0.6980
Epoch 172/300
218/218 [==============================] - 3s 16ms/step - loss: 0.8475 - ac
curacy: 0.6677 - val_loss: 0.7989 - val_accuracy: 0.6907
Epoch 173/300
218/218 [==============================] - 3s 16ms/step - loss: 0.8215 - ac
curacy: 0.6744 - val_loss: 0.7995 - val_accuracy: 0.6880
Epoch 174/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8460 - ac
curacy: 0.6658 - val_loss: 0.7989 - val_accuracy: 0.6980
Epoch 175/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8281 - ac
curacy: 0.6753 - val_loss: 0.7841 - val_accuracy: 0.7000
Epoch 176/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8322 - ac
curacy: 0.6793 - val_loss: 0.8024 - val_accuracy: 0.6793
Epoch 177/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8373 - ac
curacy: 0.6753 - val_loss: 0.7986 - val_accuracy: 0.6880
Epoch 178/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8278 - ac
curacy: 0.6793 - val_loss: 0.7926 - val_accuracy: 0.6887
Epoch 179/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8267 - ac
curacy: 0.6724 - val_loss: 0.7980 - val_accuracy: 0.6887
Epoch 180/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8219 - ac
curacy: 0.6861 - val_loss: 0.8141 - val_accuracy: 0.6827
Epoch 181/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8173 - ac
curacy: 0.6862 - val_loss: 0.7946 - val_accuracy: 0.6887
Epoch 182/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8296 - ac
curacy: 0.6763 - val_loss: 0.7824 - val_accuracy: 0.7000
Epoch 183/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8358 - ac
curacy: 0.6717 - val_loss: 0.7904 - val_accuracy: 0.6887
Epoch 184/300
```

```
218/218 [==============================] - 3s 14ms/step - loss: 0.8003 - ac
curacy: 0.6912 - val_loss: 0.7849 - val_accuracy: 0.6853
Epoch 185/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8127 - ac
curacy: 0.6802 - val_loss: 0.7966 - val_accuracy: 0.6920
Epoch 186/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8278 - ac
curacy: 0.6772 - val_loss: 0.7958 - val_accuracy: 0.6987
Epoch 187/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8022 - ac
curacy: 0.6869 - val_loss: 0.7979 - val_accuracy: 0.6913
Epoch 188/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8121 - ac
curacy: 0.6819 - val_loss: 0.7994 - val_accuracy: 0.6853
Epoch 189/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8315 - ac
curacy: 0.6708 - val_loss: 0.7895 - val_accuracy: 0.6907
Epoch 190/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8261 - ac
curacy: 0.6843 - val_loss: 0.7963 - val_accuracy: 0.6927
Epoch 191/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8102 - ac
curacy: 0.6875 - val_loss: 0.7918 - val_accuracy: 0.6800
Epoch 192/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8142 - ac
curacy: 0.6825 - val_loss: 0.7829 - val_accuracy: 0.6840
Epoch 193/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8187 - ac
curacy: 0.6812 - val_loss: 0.7874 - val_accuracy: 0.6927
Epoch 194/300
218/218 [==============================] - 3s 14ms/step - loss: 0.8091 - ac
curacy: 0.6895 - val_loss: 0.7932 - val_accuracy: 0.6980
Epoch 195/300
218/218 [==============================] - 4s 17ms/step - loss: 0.7962 - ac
curacy: 0.6889 - val_loss: 0.7708 - val_accuracy: 0.7073
Epoch 196/300
218/218 [==============================] - 4s 19ms/step - loss: 0.8132 - ac
curacy: 0.6751 - val_loss: 0.7964 - val_accuracy: 0.6987
Epoch 197/300
218/218 [==============================] - 3s 16ms/step - loss: 0.8008 - ac
curacy: 0.6921 - val_loss: 0.7975 - val_accuracy: 0.7007
Epoch 198/300
218/218 [==============================] - 4s 16ms/step - loss: 0.7931 - ac
curacy: 0.6858 - val_loss: 0.7893 - val_accuracy: 0.6953
Epoch 199/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8154 - ac
curacy: 0.6816 - val_loss: 0.7822 - val_accuracy: 0.6967
Epoch 200/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8058 - ac
curacy: 0.6874 - val_loss: 0.7844 - val_accuracy: 0.6953
Epoch 201/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8170 - ac
curacy: 0.6793 - val_loss: 0.7749 - val_accuracy: 0.6940
Epoch 202/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8239 - ac
curacy: 0.6769 - val_loss: 0.7814 - val_accuracy: 0.6967
```

```
Epoch 203/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7947 - ac
curacy: 0.6974 - val_loss: 0.7917 - val_accuracy: 0.6833
Epoch 204/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8142 - ac
curacy: 0.6826 - val_loss: 0.7879 - val_accuracy: 0.6993
Epoch 205/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8207 - ac
curacy: 0.6773 - val_loss: 0.7804 - val_accuracy: 0.7047
Epoch 206/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8062 - ac
curacy: 0.6846 - val_loss: 0.7917 - val_accuracy: 0.6947
Epoch 207/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8185 - ac
curacy: 0.6868 - val_loss: 0.7834 - val_accuracy: 0.6973
Epoch 208/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8031 - ac
curacy: 0.6898 - val_loss: 0.7828 - val_accuracy: 0.7140
Epoch 209/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8019 - ac
curacy: 0.6816 - val_loss: 0.7952 - val_accuracy: 0.6980
Epoch 210/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8113 - ac
curacy: 0.6832 - val_loss: 0.7946 - val_accuracy: 0.7013
Epoch 211/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8034 - ac
curacy: 0.6874 - val_loss: 0.7900 - val_accuracy: 0.6933
Epoch 212/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7868 - ac
curacy: 0.6912 - val_loss: 0.7852 - val_accuracy: 0.6940
Epoch 213/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8103 - ac
curacy: 0.6862 - val_loss: 0.7932 - val_accuracy: 0.6913
Epoch 214/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8057 - ac
curacy: 0.6841 - val_loss: 0.7882 - val_accuracy: 0.6827
Epoch 215/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8061 - ac
curacy: 0.6886 - val_loss: 0.7858 - val_accuracy: 0.6960
Epoch 216/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8000 - ac
curacy: 0.6912 - val_loss: 0.7990 - val_accuracy: 0.6933
Epoch 217/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7949 - ac
curacy: 0.6908 - val_loss: 0.7805 - val_accuracy: 0.7033
Epoch 218/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7924 - ac
curacy: 0.7013 - val_loss: 0.7933 - val_accuracy: 0.6900
Epoch 219/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7944 - ac
curacy: 0.6977 - val_loss: 0.7838 - val_accuracy: 0.7020
Epoch 220/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7961 - ac
curacy: 0.6905 - val_loss: 0.7958 - val_accuracy: 0.6960
Epoch 221/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8037 - ac
```

```
curacy: 0.6937 - val_loss: 0.7798 - val_accuracy: 0.6967
Epoch 222/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7902 - ac
curacy: 0.6938 - val_loss: 0.7805 - val_accuracy: 0.7100
Epoch 223/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7910 - ac
curacy: 0.6996 - val_loss: 0.8013 - val_accuracy: 0.6860
Epoch 224/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7959 - ac
curacy: 0.6862 - val_loss: 0.7726 - val_accuracy: 0.7033
Epoch 225/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7895 - ac
curacy: 0.6974 - val_loss: 0.7804 - val_accuracy: 0.7020
Epoch 226/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7857 - ac
curacy: 0.6958 - val_loss: 0.7907 - val_accuracy: 0.6947
Epoch 227/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7949 - ac
curacy: 0.6930 - val_loss: 0.7842 - val_accuracy: 0.6947
Epoch 228/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7760 - ac
curacy: 0.7034 - val_loss: 0.7833 - val_accuracy: 0.6893
Epoch 229/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7888 - ac
curacy: 0.6947 - val_loss: 0.7745 - val_accuracy: 0.6987
Epoch 230/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7857 - ac
curacy: 0.6898 - val_loss: 0.7810 - val_accuracy: 0.7033
Epoch 231/300
218/218 [==============================] - 4s 17ms/step - loss: 0.7878 - ac
curacy: 0.6974 - val_loss: 0.7680 - val_accuracy: 0.7080
Epoch 232/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7885 - ac
curacy: 0.6960 - val_loss: 0.7849 - val_accuracy: 0.7007
Epoch 233/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7808 - ac
curacy: 0.6958 - val_loss: 0.7803 - val_accuracy: 0.7027
Epoch 234/300
218/218 [==============================] - 3s 15ms/step - loss: 0.8025 - ac
curacy: 0.6864 - val_loss: 0.7728 - val_accuracy: 0.7160
Epoch 235/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7765 - ac
curacy: 0.7042 - val_loss: 0.7765 - val_accuracy: 0.7093
Epoch 236/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7921 - ac
curacy: 0.6931 - val_loss: 0.7571 - val_accuracy: 0.7127
Epoch 237/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7819 - ac
curacy: 0.6955 - val_loss: 0.7729 - val_accuracy: 0.7153
Epoch 238/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7865 - ac
curacy: 0.6958 - val_loss: 0.7722 - val_accuracy: 0.7120
Epoch 239/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7827 - ac
curacy: 0.6924 - val_loss: 0.7738 - val_accuracy: 0.7120
Epoch 240/300
```

```
218/218 [==============================] - 3s 14ms/step - loss: 0.7883 - ac
curacy: 0.6955 - val_loss: 0.7809 - val_accuracy: 0.7060
Epoch 241/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7885 - ac
curacy: 0.6954 - val_loss: 0.7805 - val_accuracy: 0.7073
Epoch 242/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7938 - ac
curacy: 0.6922 - val_loss: 0.7896 - val_accuracy: 0.7100
Epoch 243/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7833 - ac
curacy: 0.6940 - val_loss: 0.7702 - val_accuracy: 0.7100
Epoch 244/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7860 - ac
curacy: 0.7013 - val_loss: 0.7783 - val_accuracy: 0.7067
Epoch 245/300
218/218 [==============================] - 4s 17ms/step - loss: 0.7848 - ac
curacy: 0.6934 - val_loss: 0.7831 - val_accuracy: 0.7107
Epoch 246/300
218/218 [==============================] - 4s 18ms/step - loss: 0.7934 - ac
curacy: 0.6905 - val_loss: 0.7670 - val_accuracy: 0.7227
Epoch 247/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7823 - ac
curacy: 0.6941 - val_loss: 0.7670 - val_accuracy: 0.7073
Epoch 248/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7749 - ac
curacy: 0.7056 - val_loss: 0.7864 - val_accuracy: 0.6960
Epoch 249/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7709 - ac
curacy: 0.7070 - val_loss: 0.7611 - val_accuracy: 0.7133
Epoch 250/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7763 - ac
curacy: 0.7011 - val_loss: 0.7634 - val_accuracy: 0.7167
Epoch 251/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7714 - ac
curacy: 0.6957 - val_loss: 0.7571 - val_accuracy: 0.7080
Epoch 252/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7743 - ac
curacy: 0.6968 - val_loss: 0.7765 - val_accuracy: 0.7080
Epoch 253/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7687 - ac
curacy: 0.7032 - val_loss: 0.7812 - val_accuracy: 0.7093
Epoch 254/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7860 - ac
curacy: 0.6970 - val_loss: 0.7713 - val_accuracy: 0.7213
Epoch 255/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7687 - ac
curacy: 0.7057 - val_loss: 0.7505 - val_accuracy: 0.7153
Epoch 256/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7743 - ac
curacy: 0.6955 - val_loss: 0.7613 - val_accuracy: 0.7113
Epoch 257/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7733 - ac
curacy: 0.7089 - val_loss: 0.7638 - val_accuracy: 0.7093
Epoch 258/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7662 - ac
curacy: 0.7004 - val_loss: 0.7738 - val_accuracy: 0.7113
```

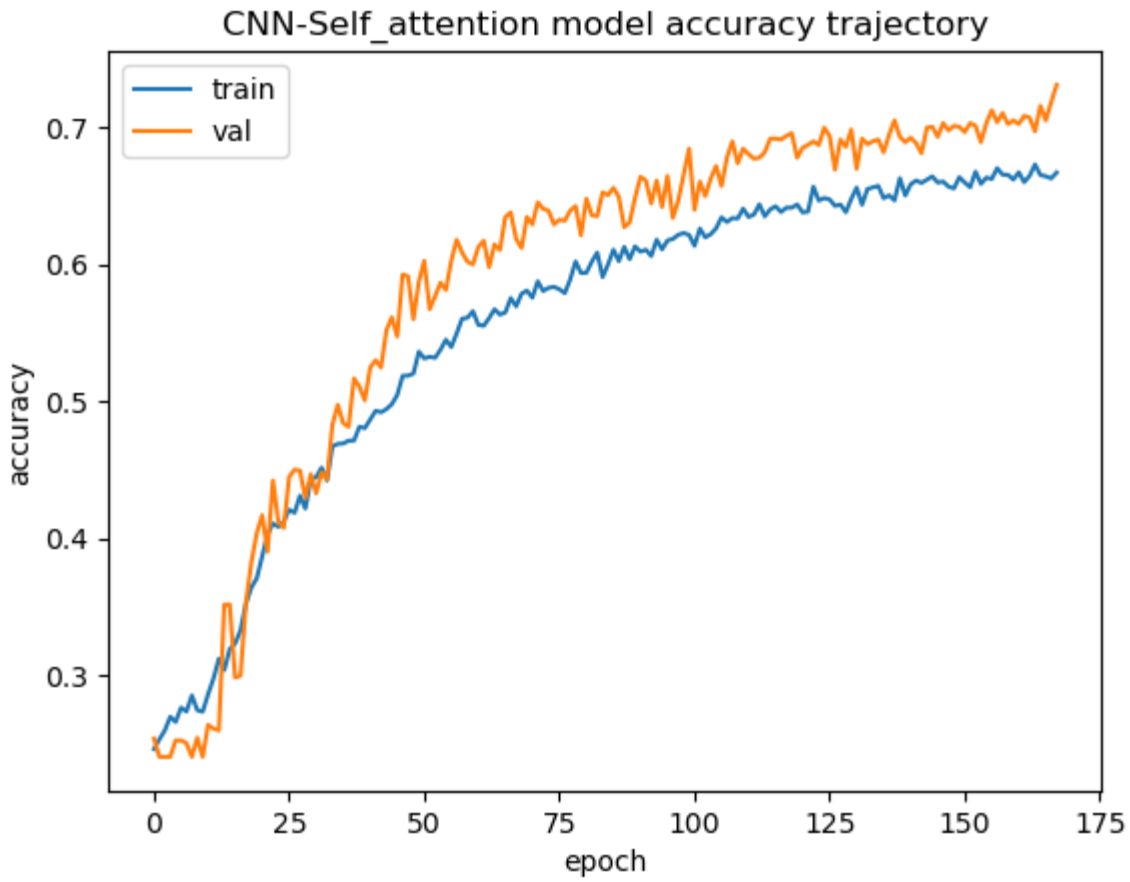
```
Epoch 259/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7748 - ac
curacy: 0.7060 - val_loss: 0.7614 - val_accuracy: 0.7067
Epoch 260/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7774 - ac
curacy: 0.7003 - val_loss: 0.7770 - val_accuracy: 0.7027
Epoch 261/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7577 - ac
curacy: 0.7088 - val_loss: 0.7674 - val_accuracy: 0.7173
Epoch 262/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7704 - ac
curacy: 0.6980 - val_loss: 0.7681 - val_accuracy: 0.7100
Epoch 263/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7767 - ac
curacy: 0.6983 - val_loss: 0.7647 - val_accuracy: 0.7160
Epoch 264/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7677 - ac
curacy: 0.7019 - val_loss: 0.7717 - val_accuracy: 0.7033
Epoch 265/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7934 - ac
curacy: 0.6993 - val_loss: 0.7482 - val_accuracy: 0.7193
Epoch 266/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7698 - ac
curacy: 0.7004 - val_loss: 0.7708 - val_accuracy: 0.7147
Epoch 267/300
218/218 [==============================] - 4s 17ms/step - loss: 0.7769 - ac
curacy: 0.7055 - val_loss: 0.7570 - val_accuracy: 0.7040
Epoch 268/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7549 - ac
curacy: 0.7036 - val_loss: 0.7565 - val_accuracy: 0.7087
Epoch 269/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7721 - ac
curacy: 0.6986 - val_loss: 0.7735 - val_accuracy: 0.7020
Epoch 270/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7726 - ac
curacy: 0.6981 - val_loss: 0.7740 - val_accuracy: 0.7060
Epoch 271/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7788 - ac
curacy: 0.6983 - val_loss: 0.7652 - val_accuracy: 0.7120
Epoch 272/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7774 - ac
curacy: 0.7014 - val_loss: 0.7718 - val_accuracy: 0.7167
Epoch 273/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7675 - ac
curacy: 0.7046 - val_loss: 0.7724 - val_accuracy: 0.7147
Epoch 274/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7695 - ac
curacy: 0.7026 - val_loss: 0.7682 - val_accuracy: 0.7100
Epoch 275/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7650 - ac
curacy: 0.7029 - val_loss: 0.7778 - val_accuracy: 0.7060
Epoch 276/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7732 - ac
curacy: 0.7029 - val_loss: 0.7809 - val_accuracy: 0.7033
Epoch 277/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7732 - ac
```

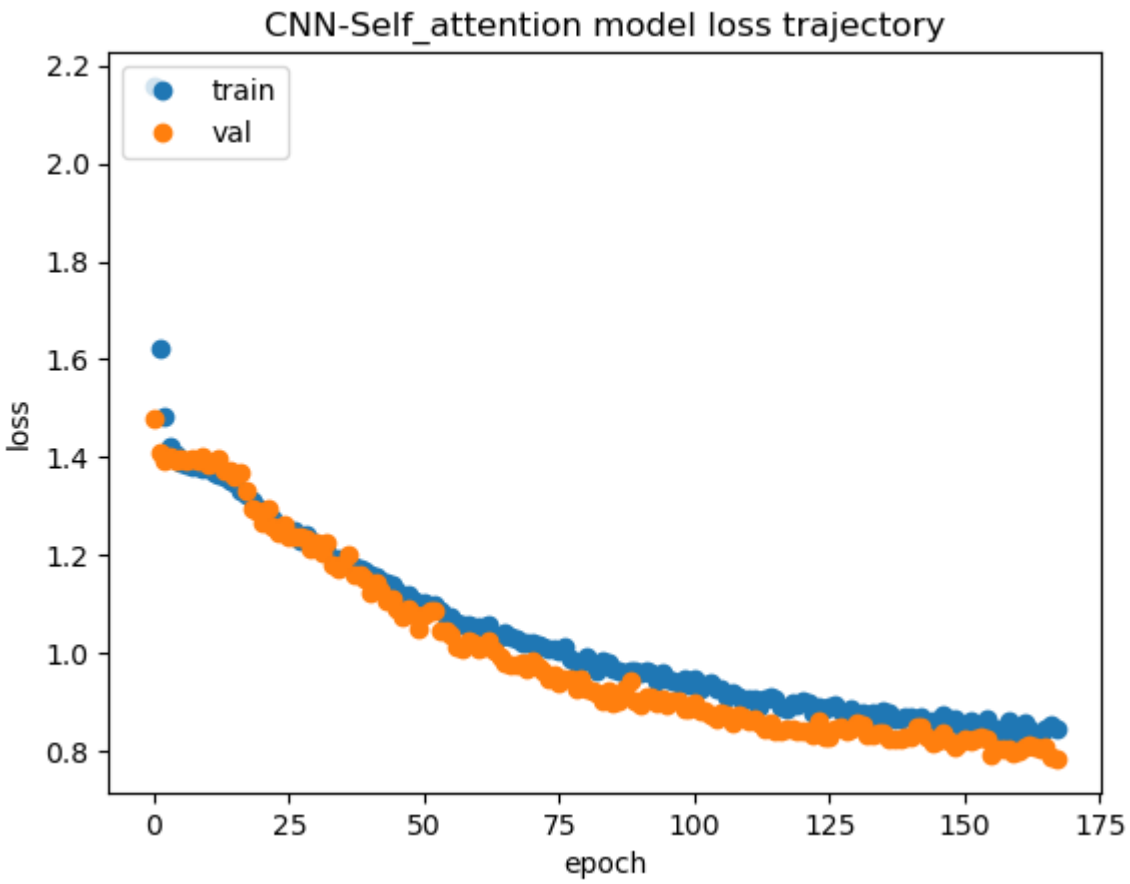
```
curacy: 0.7027 - val_loss: 0.7703 - val_accuracy: 0.7007
Epoch 278/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7691 - ac
curacy: 0.7046 - val_loss: 0.7699 - val_accuracy: 0.7167
Epoch 279/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7570 - ac
curacy: 0.7029 - val_loss: 0.7591 - val_accuracy: 0.7113
Epoch 280/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7739 - ac
curacy: 0.6993 - val_loss: 0.7622 - val_accuracy: 0.7140
Epoch 281/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7543 - ac
curacy: 0.7034 - val_loss: 0.7614 - val_accuracy: 0.7173
Epoch 282/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7723 - ac
curacy: 0.7055 - val_loss: 0.7574 - val_accuracy: 0.7060
Epoch 283/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7607 - ac
curacy: 0.7129 - val_loss: 0.7503 - val_accuracy: 0.7193
Epoch 284/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7732 - ac
curacy: 0.6944 - val_loss: 0.7693 - val_accuracy: 0.7027
Epoch 285/300
218/218 [==============================] - 3s 14ms/step - loss: 0.7660 - ac
curacy: 0.6968 - val_loss: 0.7675 - val_accuracy: 0.7133
Epoch 286/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7517 - ac
curacy: 0.7151 - val_loss: 0.7482 - val_accuracy: 0.7213
Epoch 287/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7707 - ac
curacy: 0.7039 - val_loss: 0.7610 - val_accuracy: 0.7113
Epoch 288/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7680 - ac
curacy: 0.7017 - val_loss: 0.7626 - val_accuracy: 0.7167
Epoch 289/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7527 - ac
curacy: 0.7086 - val_loss: 0.7659 - val_accuracy: 0.7187
Epoch 290/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7696 - ac
curacy: 0.7001 - val_loss: 0.7622 - val_accuracy: 0.7153
Epoch 291/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7628 - ac
curacy: 0.7065 - val_loss: 0.7657 - val_accuracy: 0.7067
Epoch 292/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7536 - ac
curacy: 0.7065 - val_loss: 0.7678 - val_accuracy: 0.7053
Epoch 293/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7564 - ac
curacy: 0.7091 - val_loss: 0.7646 - val_accuracy: 0.7107
Epoch 294/300
218/218 [==============================] - 3s 16ms/step - loss: 0.7580 - ac
curacy: 0.7091 - val_loss: 0.7542 - val_accuracy: 0.7140
Epoch 295/300
218/218 [==============================] - 3s 15ms/step - loss: 0.7521 - ac
curacy: 0.7109 - val_loss: 0.7635 - val_accuracy: 0.7160
Epoch 296/300
```

```
218/218 [==============================] - 3s 15ms/step - loss: 0.7527 - ac
curacy: 0.7060 - val_loss: 0.7545 - val_accuracy: 0.7220
Epoch 297/300
169/218 [======================>.......] - ETA: 0s - loss: 0.7493 - accurac
y: 0.7110
```

(vi)(CNN-Attention) Visualizing the accuracy and loss trajectory

In [ ]:
```python
import matplotlib.pyplot as plt

# Plotting accuracy trajectory
plt.plot(hybrid_cnn_lstm_model_results.history['accuracy'])
plt.plot(hybrid_cnn_lstm_model_results.history['val_accuracy'])
plt.title('CNN-Self_attention model accuracy trajectory')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

# Plotting loss trajectory
plt.plot(hybrid_cnn_lstm_model_results.history['loss'],'o')
plt.plot(hybrid_cnn_lstm_model_results.history['val_loss'],'o')
plt.title('CNN-Self_attention model loss trajectory')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```


CNN-Self_attention model accuracy trajectory
train
val
accuracy
epoch
0.3
0.4
0.5
0.6
0.7
0
25
50
75
100
125
150
175


CNN-Self_attention model loss trajectory
train
val
loss
epoch
0.8
1.0
1.2
1.4
1.6
1.8
2.0
2.2
0
25
50
75
100
125
150
175

(vii)(CNN-Self-Attention) Testing the performance of the hybrid CNN-Self-Attention model on the held out test set

In [ ]:
```
## Testing the hybrid CNN-LSTM model

hybrid_cnn_lstm_score = hybrid_cnn_lstm_model.evaluate(x_test, y_test, verbo
print('Test accuracy of the hybrid CNN-attention model:',hybrid_cnn_lstm_sco
```

```
Test accuracy of the hybrid CNN-attention model: 0.6805869340896606
```

In [ ]:

In this discussion, we will build a basic hybrid CNN-MHA model for classification on the EEG dataset

This notebook was inspired to Tonmoy with some attempts to tune by us

(i) Importing the necessary packages

In [ ]:
```
import numpy as np
import pandas as pd
import keras
from keras import layers
from keras.models import Sequential, Model
from keras.layers import Dense, Activation, Flatten,Dropout,Bidirectional, C
from keras.layers import Conv2D, LayerNormalization, LSTM,BatchNormalizatior
from keras.utils import to_categorical
import matplotlib.pyplot as plt
```

(ii) Preprocessing the dataset and preparing the training, validation, and test datasets

In [ ]:

```
def data_prep(X,y,sub_sample,average,noise):

    total_X = None
    total_y = None

    # Trimming the data (sample,22,1000) -> (sample,22,500)
    X = X[:,:,0:500]
    print('Shape of X after trimming:',X.shape)

    # Maxpooling the data (sample,22,1000) -> (sample,22,500/sub_sample)
    X_max = np.max(X.reshape(X.shape[0], X.shape[1], -1, sub_sample), axis=3


    total_X = X_max
    total_y = y
    print('Shape of X after maxpooling:',total_X.shape)

    # Averaging + noise
    X_average = np.mean(X.reshape(X.shape[0], X.shape[1], -1, average),axis=
    X_average = X_average + np.random.normal(0.0, 0.5, X_average.shape)

    total_X = np.vstack((total_X, X_average))
    total_y = np.hstack((total_y, y))
    print('Shape of X after averaging+noise and concatenating:',total_X.shap

    # Subsampling

    for i in range(sub_sample):

        X_subsample = X[:, :, i::sub_sample] + \
                            (np.random.normal(0.0, 0.5, X[:, :,i::sub_sample

        total_X = np.vstack((total_X, X_subsample))
        total_y = np.hstack((total_y, y))


    print('Shape of X after subsampling and concatenating:',total_X.shape)
    return total_X,total_y
```

In [ ]:

```
## Loading the dataset


X_test = np.load("X_test.npy")
y_test = np.load("y_test.npy")
person_train_valid = np.load("person_train_valid.npy")
X_train_valid = np.load("X_train_valid.npy")
y_train_valid = np.load("y_train_valid.npy")
person_test = np.load("person_test.npy")

## Adjusting the labels so that

# Cue onset left - 0
# Cue onset right - 1
# Cue onset foot - 2
# Cue onset tongue - 3

y_train_valid -= 769
y_test -= 769


## Random splitting and reshaping the data
# First generating the training and validation indices using random splittir

ind_valid = np.random.choice(2115, 375, replace=False)
ind_train = np.array(list(set(range(2115)).difference(set(ind_valid))))

# Creating the training and validation sets using the generated indices
(X_train, X_valid) = X_train_valid[ind_train], X_train_valid[ind_valid]
(y_train, y_valid) = y_train_valid[ind_train], y_train_valid[ind_valid]


## Preprocessing the dataset
x_train,y_train = data_prep(X_train,y_train,2,2,True)
x_valid,y_valid = data_prep(X_valid,y_valid,2,2,True)
X_test_prep,y_test_prep = data_prep(X_test,y_test,2,2,True)


print('Shape of training set:',x_train.shape)
print('Shape of validation set:',x_valid.shape)
print('Shape of training labels:',y_train.shape)
print('Shape of validation labels:',y_valid.shape)
print('Shape of testing set:',X_test_prep.shape)
print('Shape of testing labels:',y_test_prep.shape)


# Converting the labels to categorical variables for multiclass classificati
y_train = to_categorical(y_train, 4)
y_valid = to_categorical(y_valid, 4)
y_test = to_categorical(y_test_prep, 4)
print('Shape of training labels after categorical conversion:',y_train.shape
print('Shape of validation labels after categorical conversion:',y_valid.sha
print('Shape of test labels after categorical conversion:',y_test.shape)

# Adding width of the segment to be 1
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], x_train.shape[
x_valid = x_valid.reshape(x_valid.shape[0], x_valid.shape[1], x_train.shape[
x_test = X_test_prep.reshape(X_test_prep.shape[0], X_test_prep.shape[1], X_t
print('Shape of training set after adding width info:',x_train.shape)
```

```
print('Shape of training set after adding width info:',x_train.shape)
print('Shape of validation set after adding width info:',x_valid.shape)
print('Shape of test set after adding width info:',x_test.shape)


# Reshaping the training and validation dataset
x_train = np.swapaxes(x_train, 1,3)
x_train = np.swapaxes(x_train, 1,2)
x_valid = np.swapaxes(x_valid, 1,3)
x_valid = np.swapaxes(x_valid, 1,2)
x_test = np.swapaxes(x_test, 1,3)
x_test = np.swapaxes(x_test, 1,2)
print('Shape of training set after dimension reshaping:',x_train.shape)
print('Shape of validation set after dimension reshaping:',x_valid.shape)
print('Shape of test set after dimension reshaping:',x_test.shape)

keras.backend.clear_session()
```

```
Shape of X after trimming: (1740, 22, 500)
Shape of X after maxpooling: (1740, 22, 250)
Shape of X after averaging+noise and concatenating: (3480, 22, 250)
Shape of X after subsampling and concatenating: (6960, 22, 250)
Shape of X after trimming: (375, 22, 500)
Shape of X after maxpooling: (375, 22, 250)
Shape of X after averaging+noise and concatenating: (750, 22, 250)
Shape of X after subsampling and concatenating: (1500, 22, 250)
Shape of X after trimming: (443, 22, 500)
Shape of X after maxpooling: (443, 22, 250)
Shape of X after averaging+noise and concatenating: (886, 22, 250)
Shape of X after subsampling and concatenating: (1772, 22, 250)
Shape of training set: (6960, 22, 250)
Shape of validation set: (1500, 22, 250)
Shape of training labels: (6960,)
Shape of validation labels: (1500,)
Shape of testing set: (1772, 22, 250)
Shape of testing labels: (1772,)
Shape of training labels after categorical conversion: (6960, 4)
Shape of validation labels after categorical conversion: (1500, 4)
Shape of test labels after categorical conversion: (1772, 4)
Shape of training set after adding width info: (6960, 22, 250, 1)
Shape of validation set after adding width info: (1500, 22, 250, 1)
Shape of test set after adding width info: (1772, 22, 250, 1)
Shape of training set after dimension reshaping: (6960, 250, 1, 22)
Shape of validation set after dimension reshaping: (1500, 250, 1, 22)
Shape of test set after dimension reshaping: (1772, 250, 1, 22)
```

(iii)(CNN-MHA) Defining the architecture of the hybrid CNN-MHA model

In [ ]:

```
# Building the CNN model using functional class
def build_model():
# models = []

    n_frames = 250
    n_channels = 22
    # Conv. block 1
    In1 = keras.Input(shape =(250,1,22) )
    c1 = Conv2D(filters=30, kernel_size=(11,1), padding='same', activation='
    p1 =  MaxPooling2D(pool_size=(4,1), padding='same')(c1) # Read the keras
    b1 = BatchNormalization()(p1)
    d1 = Dropout(0.5)(b1)

    # Conv. block 2
    c2 = Conv2D(filters=60, kernel_size=(9,1), padding='same', activation='s
    p2 =  MaxPooling2D(pool_size=(4,1), padding='same')(c2) # Read the keras
    b2 = BatchNormalization()(p2)
    d2 = Dropout(0.6)(b2)

    # Conv. block 3
    c3 = Conv2D(filters=120, kernel_size=(5,1), padding='same', activation='
    p3 =  MaxPooling2D(pool_size=(4,1), padding='same')(c3) # Read the keras
    b3 = BatchNormalization()(p3)
    d3 = Dropout(0.6)(b3)

    # Conv. block 4
    c4 = Conv2D(filters=240, kernel_size=(3,1), padding='same', activation='
    p4 =  MaxPooling2D(pool_size=(4,1), padding='same')(c4) # Read the keras
    b4 = BatchNormalization()(p4)
    d4 = Dropout(0.6)(b4)


    #attention
    query_inputs = d4
    key_inputs = d4
    value_inputs = d4

    multi_head_attention = MultiHeadAttention(num_heads=8, key_dim=n_channel
    multi_head_attention = LayerNormalization(epsilon=1e-6)(multi_head_atter
    multi_head_attention = GlobalAveragePooling2D()(multi_head_attention)

    # Add fully connected layers
    fc1 = Dense(64, activation='relu')(multi_head_attention)
    fc1_dropout = Dropout(rate=0.5)(fc1)
    fc2 = Dense(4, activation='softmax')(fc1_dropout)

    # Define the final model
    final_model = Model(inputs=In1, outputs=[fc2])

    # Compile the final model
    # # # Apply spatial attention
    # # hybrid_cnn_lstm_model.add(Lambda(spatial_attention))



    # # FC
    # hybrid_cnn_lstm_model.add(Flatten()) # Adding a flattening operation t
    # hybrid_cnn_lstm_model.add(Dense(100, activation='Elu')) # FC layer wit
    # hybrid_cnn_lstm_model.add(Reshape((100,1))) # Reshape my output of FC
```

```
    # hybrid_cnn_lstm_model.add(Reshape((100,1))) # Reshape my output of FC
    # # hybrid_cnn_lstm_model.add(LSTM(10, dropout=0.6, recurrent_dropout=0.


    # # Output layer with Softmax activation
    # hybrid_cnn_lstm_model.add(Dense(4, activation='softmax')) # Output FC

    # # models.append(hybrid_cnn_lstm_model)
    # # Printing the model summary
    final_model.compile(loss='categorical_crossentropy',
            optimizer='adam',
            metrics=['accuracy'])
    final_model.summary()
    return final_model
```

(iv) (CNN-MHA) Defining the hyperparameters of the hybrid CNN-MHA model

In [ ]:

```
import tensorflow as tf
# Model parameters
epochs = 300
initial_learning_rate = 1e-3
decay_steps = 1000
decay_rate = 0.99

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate,
    decay_steps=decay_steps,
    decay_rate=decay_rate
)

optimizer = keras.optimizers.Adam(learning_rate=lr_schedule)
```

(v) Attention at the end

In [ ]:

```
# Compiling the model
keras.backend.clear_session()

# Training and validating the model

batch_sizes = [32,64,128]
import matplotlib.pyplot as plt

hybrid_cnn_lstm_model = build_model()
# Compiling the model
hybrid_cnn_lstm_model.compile(loss='categorical_crossentropy',
               optimizer=optimizer,
               metrics=['accuracy'])

# Training and validating the model
ACCURACY_THRESHOLD = 0.725
class myCallback(tf.keras.callbacks.Callback):
        def on_epoch_end(self, epoch, logs={}):
                if(logs.get('val_accuracy') > ACCURACY_THRESHOLD):
                        print("\nReached %2.2f%% accuracy, so stopping trair
                        self.model.stop_training = True

# Instantiate a callback object
callback = myCallback()
# callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)
hybrid_cnn_lstm_model_results = hybrid_cnn_lstm_model.fit(x_train,
                y_train,
                batch_size=32,
                epochs=epochs,
                validation_data=(x_valid, y_valid),
                verbose=True,
                callbacks = [callback]
                )
```

```
Model: "model"
__________________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected
to
==========================================================================================
 input_1 (InputLayer)        [(None, 250, 1, 22)]      0           []

 conv2d (Conv2D)             (None, 250, 1, 30)        7290        ['input_1
[0][0]']

 max_pooling2d (MaxPooling2D)   (None, 63, 1, 30)      0           ['conv2d
[0][0]']
__________________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected
to
==========================================================================================
 input_1 (InputLayer)        [(None, 250, 1, 22)]      0           []

 conv2d (Conv2D)             (None, 250, 1, 30)        7290        ['input_1
[0][0]']

 max_pooling2d (MaxPooling2D)   (None, 63, 1, 30)      0           ['conv2d
[0][0]']

 batch_normalization (BatchNorm  (None, 63, 1, 30)     120         ['max_pool
ing2d[0][0]']
 alization)

 dropout (Dropout)           (None, 63, 1, 30)         0           ['batch_no
rmalization[0][0]']

 conv2d_1 (Conv2D)           (None, 63, 1, 60)         16260       ['dropout
[0][0]']

 max_pooling2d_1 (MaxPooling2D)  (None, 16, 1, 60)     0           ['conv2d_1
[0][0]']

 batch_normalization_1 (BatchNo  (None, 16, 1, 60)     240         ['max_pool
ing2d_1[0][0]']
 rmalization)

 dropout_1 (Dropout)         (None, 16, 1, 60)         0           ['batch_no
rmalization_1[0][0]']

 conv2d_2 (Conv2D)           (None, 16, 1, 120)        36120       ['dropout_
1[0][0]']

 max_pooling2d_2 (MaxPooling2D)  (None, 4, 1, 120)     0           ['conv2d_2
[0][0]']

 batch_normalization_2 (BatchNo  (None, 4, 1, 120)     480         ['max_pool
ing2d_2[0][0]']
 rmalization)
```

```
 dropout_2 (Dropout)            (None, 4, 1, 120)    0           ['batch_no
rmalization_2[0][0]']

 conv2d_3 (Conv2D)              (None, 4, 1, 240)    86640       ['dropout_
2[0][0]']

 max_pooling2d_3 (MaxPooling2D)  (None, 1, 1, 240)   0           ['conv2d_3
[0][0]']

 batch_normalization_3 (BatchNo  (None, 1, 1, 240)   960         ['max_pool
ing2d_3[0][0]']
 rmalization)

 dropout_3 (Dropout)            (None, 1, 1, 240)    0           ['batch_no
rmalization_3[0][0]']

 multi_head_attention (MultiHea  (None, 1, 1, 240)   15648       ['dropout_
3[0][0]',
 dAttention)                                                     'dropout_
3[0][0]',
                                                                 'dropout_
3[0][0]']

 layer_normalization (LayerNorm  (None, 1, 1, 240)   480         ['multi_he
ad_attention[0][0]']
 alization)

 global_average_pooling2d (Glob  (None, 240)         0           ['layer_no
rmalization[0][0]']
 alAveragePooling2D)

 dense (Dense)                  (None, 64)           15424       ['global_a
verage_pooling2d[0][0]'
                                                                ]

 dropout_4 (Dropout)            (None, 64)           0           ['dense
[0][0]']

 dense_1 (Dense)                (None, 4)            260         ['dropout_
4[0][0]']

==================================================================================
======================
Total params: 179,922
Trainable params: 179,022
Non-trainable params: 900
__________________________________________________________________________________
_____________________
Epoch 1/300
218/218 [==============================] - 5s 18ms/step - loss: 1.5216 - ac
curacy: 0.2579 - val_loss: 1.3850 - val_accuracy: 0.2660
Epoch 2/300
218/218 [==============================] - 3s 16ms/step - loss: 1.3928 - ac
curacy: 0.2691 - val_loss: 1.3945 - val_accuracy: 0.2427
Epoch 3/300
218/218 [==============================] - 3s 16ms/step - loss: 1.3872 - ac
curacy: 0.2694 - val_loss: 1.3978 - val_accuracy: 0.2607
Epoch 4/300
```

```
218/218 [==============================] - 4s 17ms/step - loss: 1.3859 - ac
curacy: 0.2655 - val_loss: 1.3903 - val_accuracy: 0.2333
Epoch 5/300
218/218 [==============================] - 3s 16ms/step - loss: 1.3844 - ac
curacy: 0.2744 - val_loss: 1.3935 - val_accuracy: 0.2473
Epoch 6/300
218/218 [==============================] - 4s 17ms/step - loss: 1.3799 - ac
curacy: 0.2796 - val_loss: 1.3728 - val_accuracy: 0.3040
Epoch 7/300
218/218 [==============================] - 4s 17ms/step - loss: 1.3711 - ac
curacy: 0.2932 - val_loss: 1.3667 - val_accuracy: 0.2887
Epoch 8/300
218/218 [==============================] - 3s 16ms/step - loss: 1.3619 - ac
curacy: 0.3010 - val_loss: 1.3272 - val_accuracy: 0.3767
Epoch 9/300
218/218 [==============================] - 4s 16ms/step - loss: 1.3381 - ac
curacy: 0.3431 - val_loss: 1.2786 - val_accuracy: 0.4187
Epoch 10/300
218/218 [==============================] - 3s 16ms/step - loss: 1.3203 - ac
curacy: 0.3539 - val_loss: 1.2826 - val_accuracy: 0.4267
Epoch 11/300
218/218 [==============================] - 4s 17ms/step - loss: 1.3013 - ac
curacy: 0.3816 - val_loss: 1.2479 - val_accuracy: 0.4260
Epoch 12/300
218/218 [==============================] - 3s 16ms/step - loss: 1.2741 - ac
curacy: 0.4066 - val_loss: 1.2875 - val_accuracy: 0.3907
Epoch 13/300
218/218 [==============================] - 4s 16ms/step - loss: 1.2720 - ac
curacy: 0.4152 - val_loss: 1.2768 - val_accuracy: 0.3773
Epoch 14/300
218/218 [==============================] - 3s 16ms/step - loss: 1.2564 - ac
curacy: 0.4187 - val_loss: 1.2149 - val_accuracy: 0.4220
Epoch 15/300
218/218 [==============================] - 3s 15ms/step - loss: 1.2449 - ac
curacy: 0.4292 - val_loss: 1.1978 - val_accuracy: 0.4540
Epoch 16/300
218/218 [==============================] - 3s 13ms/step - loss: 1.2384 - ac
curacy: 0.4343 - val_loss: 1.2065 - val_accuracy: 0.4407
Epoch 17/300
218/218 [==============================] - 3s 12ms/step - loss: 1.2326 - ac
curacy: 0.4330 - val_loss: 1.1704 - val_accuracy: 0.4620
Epoch 18/300
218/218 [==============================] - 2s 11ms/step - loss: 1.2186 - ac
curacy: 0.4458 - val_loss: 1.1929 - val_accuracy: 0.4567
Epoch 19/300
218/218 [==============================] - 2s 11ms/step - loss: 1.2013 - ac
curacy: 0.4519 - val_loss: 1.1434 - val_accuracy: 0.4933
Epoch 20/300
218/218 [==============================] - 2s 11ms/step - loss: 1.1920 - ac
curacy: 0.4606 - val_loss: 1.1529 - val_accuracy: 0.4713
Epoch 21/300
218/218 [==============================] - 3s 12ms/step - loss: 1.1816 - ac
curacy: 0.4730 - val_loss: 1.2039 - val_accuracy: 0.4593
Epoch 22/300
218/218 [==============================] - 2s 11ms/step - loss: 1.1720 - ac
curacy: 0.4694 - val_loss: 1.1412 - val_accuracy: 0.4687
Epoch 23/300
218/218 [==============================] - 2s 11ms/step - loss: 1.1581 - ac
curacy: 0.4796 - val_loss: 1.1460 - val_accuracy: 0.4727
```

```
Epoch 24/300
218/218 [==============================] - 2s 11ms/step - loss: 1.1471 - ac
curacy: 0.4832 - val_loss: 1.1541 - val_accuracy: 0.4753
Epoch 25/300
218/218 [==============================] - 2s 11ms/step - loss: 1.1398 - ac
curacy: 0.4935 - val_loss: 1.1531 - val_accuracy: 0.4733
Epoch 26/300
218/218 [==============================] - 2s 11ms/step - loss: 1.1291 - ac
curacy: 0.4886 - val_loss: 1.1213 - val_accuracy: 0.4720
Epoch 27/300
218/218 [==============================] - 3s 12ms/step - loss: 1.1164 - ac
curacy: 0.4981 - val_loss: 1.1176 - val_accuracy: 0.4920
Epoch 28/300
218/218 [==============================] - 3s 12ms/step - loss: 1.1230 - ac
curacy: 0.4981 - val_loss: 1.1643 - val_accuracy: 0.4687
Epoch 29/300
218/218 [==============================] - 3s 12ms/step - loss: 1.1037 - ac
curacy: 0.5052 - val_loss: 1.1061 - val_accuracy: 0.4820
Epoch 30/300
218/218 [==============================] - 3s 12ms/step - loss: 1.0976 - ac
curacy: 0.5082 - val_loss: 1.1314 - val_accuracy: 0.4660
Epoch 31/300
218/218 [==============================] - 2s 11ms/step - loss: 1.0831 - ac
curacy: 0.5152 - val_loss: 1.1478 - val_accuracy: 0.4773
Epoch 32/300
218/218 [==============================] - 3s 12ms/step - loss: 1.0739 - ac
curacy: 0.5201 - val_loss: 1.0792 - val_accuracy: 0.4980
Epoch 33/300
218/218 [==============================] - 3s 13ms/step - loss: 1.0650 - ac
curacy: 0.5177 - val_loss: 1.1413 - val_accuracy: 0.4887
Epoch 34/300
218/218 [==============================] - 3s 12ms/step - loss: 1.0627 - ac
curacy: 0.5233 - val_loss: 1.1294 - val_accuracy: 0.4820
Epoch 35/300
218/218 [==============================] - 2s 11ms/step - loss: 1.0517 - ac
curacy: 0.5411 - val_loss: 1.1005 - val_accuracy: 0.4760
Epoch 36/300
218/218 [==============================] - 3s 11ms/step - loss: 1.0405 - ac
curacy: 0.5365 - val_loss: 1.1255 - val_accuracy: 0.4907
Epoch 37/300
218/218 [==============================] - 2s 11ms/step - loss: 1.0364 - ac
curacy: 0.5483 - val_loss: 1.0937 - val_accuracy: 0.5200
Epoch 38/300
218/218 [==============================] - 3s 12ms/step - loss: 1.0266 - ac
curacy: 0.5404 - val_loss: 1.1014 - val_accuracy: 0.5213
Epoch 39/300
218/218 [==============================] - 3s 12ms/step - loss: 1.0200 - ac
curacy: 0.5605 - val_loss: 1.0836 - val_accuracy: 0.5387
Epoch 40/300
218/218 [==============================] - 3s 12ms/step - loss: 1.0102 - ac
curacy: 0.5662 - val_loss: 1.0574 - val_accuracy: 0.5560
Epoch 41/300
218/218 [==============================] - 2s 11ms/step - loss: 1.0037 - ac
curacy: 0.5734 - val_loss: 1.0650 - val_accuracy: 0.5753
Epoch 42/300
218/218 [==============================] - 3s 12ms/step - loss: 0.9884 - ac
curacy: 0.5909 - val_loss: 1.0392 - val_accuracy: 0.5867
Epoch 43/300
218/218 [==============================] - 3s 12ms/step - loss: 0.9985 - ac
```

```
curacy: 0.5878 - val_loss: 1.0511 - val_accuracy: 0.5780
Epoch 44/300
218/218 [==============================] - 3s 12ms/step - loss: 0.9883 - ac
curacy: 0.5940 - val_loss: 1.0550 - val_accuracy: 0.5700
Epoch 45/300
218/218 [==============================] - 3s 12ms/step - loss: 0.9822 - ac
curacy: 0.5950 - val_loss: 1.0311 - val_accuracy: 0.5667
Epoch 46/300
218/218 [==============================] - 2s 11ms/step - loss: 0.9571 - ac
curacy: 0.6111 - val_loss: 0.9798 - val_accuracy: 0.6133
Epoch 47/300
218/218 [==============================] - 2s 11ms/step - loss: 0.9473 - ac
curacy: 0.6205 - val_loss: 0.9626 - val_accuracy: 0.6313
Epoch 48/300
218/218 [==============================] - 2s 11ms/step - loss: 0.9514 - ac
curacy: 0.6177 - val_loss: 0.9532 - val_accuracy: 0.6227
Epoch 49/300
218/218 [==============================] - 2s 11ms/step - loss: 0.9154 - ac
curacy: 0.6382 - val_loss: 0.9526 - val_accuracy: 0.6593
Epoch 50/300
218/218 [==============================] - 2s 11ms/step - loss: 0.9214 - ac
curacy: 0.6391 - val_loss: 0.9451 - val_accuracy: 0.6453
Epoch 51/300
218/218 [==============================] - 2s 11ms/step - loss: 0.9032 - ac
curacy: 0.6430 - val_loss: 0.9546 - val_accuracy: 0.6373
Epoch 52/300
218/218 [==============================] - 2s 11ms/step - loss: 0.8931 - ac
curacy: 0.6573 - val_loss: 0.9344 - val_accuracy: 0.6500
Epoch 53/300
218/218 [==============================] - 2s 11ms/step - loss: 0.8815 - ac
curacy: 0.6510 - val_loss: 0.8903 - val_accuracy: 0.6707
Epoch 54/300
218/218 [==============================] - 3s 11ms/step - loss: 0.8661 - ac
curacy: 0.6721 - val_loss: 0.9081 - val_accuracy: 0.6653
Epoch 55/300
218/218 [==============================] - 2s 11ms/step - loss: 0.8824 - ac
curacy: 0.6583 - val_loss: 0.8910 - val_accuracy: 0.6580
Epoch 56/300
218/218 [==============================] - 2s 11ms/step - loss: 0.8682 - ac
curacy: 0.6603 - val_loss: 0.8873 - val_accuracy: 0.6633
Epoch 57/300
218/218 [==============================] - 2s 11ms/step - loss: 0.8507 - ac
curacy: 0.6789 - val_loss: 0.9000 - val_accuracy: 0.6333
Epoch 58/300
218/218 [==============================] - 2s 11ms/step - loss: 0.8582 - ac
curacy: 0.6641 - val_loss: 0.8813 - val_accuracy: 0.6613
Epoch 59/300
218/218 [==============================] - 2s 11ms/step - loss: 0.8400 - ac
curacy: 0.6915 - val_loss: 0.8540 - val_accuracy: 0.6800
Epoch 60/300
218/218 [==============================] - 2s 11ms/step - loss: 0.8285 - ac
curacy: 0.6886 - val_loss: 0.8620 - val_accuracy: 0.6647
Epoch 61/300
218/218 [==============================] - 2s 11ms/step - loss: 0.8198 - ac
curacy: 0.6889 - val_loss: 0.8879 - val_accuracy: 0.6607
Epoch 62/300
218/218 [==============================] - 2s 11ms/step - loss: 0.8199 - ac
curacy: 0.6922 - val_loss: 0.8401 - val_accuracy: 0.6793
Epoch 63/300
```

```
218/218 [==============================] - 2s 11ms/step - loss: 0.8049 - ac
curacy: 0.6957 - val_loss: 0.8632 - val_accuracy: 0.6667
Epoch 64/300
218/218 [==============================] - 2s 11ms/step - loss: 0.8107 - ac
curacy: 0.6974 - val_loss: 0.8707 - val_accuracy: 0.6733
Epoch 65/300
218/218 [==============================] - 2s 11ms/step - loss: 0.8033 - ac
curacy: 0.6945 - val_loss: 0.8397 - val_accuracy: 0.6947
Epoch 66/300
218/218 [==============================] - 2s 11ms/step - loss: 0.8052 - ac
curacy: 0.7001 - val_loss: 0.8171 - val_accuracy: 0.7007
Epoch 67/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7980 - ac
curacy: 0.6960 - val_loss: 0.8706 - val_accuracy: 0.6853
Epoch 68/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7969 - ac
curacy: 0.7029 - val_loss: 0.8181 - val_accuracy: 0.7053
Epoch 69/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7896 - ac
curacy: 0.7057 - val_loss: 0.8351 - val_accuracy: 0.6893
Epoch 70/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7785 - ac
curacy: 0.7089 - val_loss: 0.8155 - val_accuracy: 0.6980
Epoch 71/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7698 - ac
curacy: 0.7095 - val_loss: 0.8953 - val_accuracy: 0.6660
Epoch 72/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7696 - ac
curacy: 0.7125 - val_loss: 0.8294 - val_accuracy: 0.6860
Epoch 73/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7623 - ac
curacy: 0.7096 - val_loss: 0.8084 - val_accuracy: 0.6953
Epoch 74/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7629 - ac
curacy: 0.7144 - val_loss: 0.8431 - val_accuracy: 0.6540
Epoch 75/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7452 - ac
curacy: 0.7170 - val_loss: 0.7928 - val_accuracy: 0.6913
Epoch 76/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7444 - ac
curacy: 0.7292 - val_loss: 0.8405 - val_accuracy: 0.6700
Epoch 77/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7497 - ac
curacy: 0.7241 - val_loss: 0.8513 - val_accuracy: 0.6753
Epoch 78/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7463 - ac
curacy: 0.7178 - val_loss: 0.8445 - val_accuracy: 0.6660
Epoch 79/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7392 - ac
curacy: 0.7293 - val_loss: 0.8467 - val_accuracy: 0.6787
Epoch 80/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7449 - ac
curacy: 0.7236 - val_loss: 0.8807 - val_accuracy: 0.6953
Epoch 81/300
218/218 [==============================] - 3s 12ms/step - loss: 0.7203 - ac
curacy: 0.7299 - val_loss: 0.8642 - val_accuracy: 0.6807
Epoch 82/300
218/218 [==============================] - 3s 12ms/step - loss: 0.7240 - ac
curacy: 0.7333 - val_loss: 0.8387 - val_accuracy: 0.6853
```

```
Epoch 83/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7350 - ac
curacy: 0.7244 - val_loss: 0.8107 - val_accuracy: 0.6833
Epoch 84/300
218/218 [==============================] - 3s 12ms/step - loss: 0.7112 - ac
curacy: 0.7375 - val_loss: 0.8666 - val_accuracy: 0.6860
Epoch 85/300
218/218 [==============================] - 3s 13ms/step - loss: 0.7151 - ac
curacy: 0.7359 - val_loss: 0.8273 - val_accuracy: 0.6960
Epoch 86/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7091 - ac
curacy: 0.7430 - val_loss: 0.7873 - val_accuracy: 0.7007
Epoch 87/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7073 - ac
curacy: 0.7418 - val_loss: 0.8552 - val_accuracy: 0.6793
Epoch 88/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7252 - ac
curacy: 0.7279 - val_loss: 0.8027 - val_accuracy: 0.7013
Epoch 89/300
218/218 [==============================] - 2s 11ms/step - loss: 0.7015 - ac
curacy: 0.7411 - val_loss: 0.8444 - val_accuracy: 0.7160
Epoch 90/300
218/218 [==============================] - 2s 11ms/step - loss: 0.6871 - ac
curacy: 0.7483 - val_loss: 0.8116 - val_accuracy: 0.6913
Epoch 91/300
218/218 [==============================] - 3s 12ms/step - loss: 0.7028 - ac
curacy: 0.7421 - val_loss: 0.7976 - val_accuracy: 0.7087
Epoch 92/300
218/218 [==============================] - 3s 12ms/step - loss: 0.7022 - ac
curacy: 0.7384 - val_loss: 0.7912 - val_accuracy: 0.7160
Epoch 93/300
218/218 [==============================] - 3s 12ms/step - loss: 0.7055 - ac
curacy: 0.7376 - val_loss: 0.8224 - val_accuracy: 0.6860
Epoch 94/300
218/218 [==============================] - 3s 12ms/step - loss: 0.6770 - ac
curacy: 0.7527 - val_loss: 0.8235 - val_accuracy: 0.6893
Epoch 95/300
218/218 [==============================] - 3s 12ms/step - loss: 0.6906 - ac
curacy: 0.7420 - val_loss: 0.7930 - val_accuracy: 0.7067
Epoch 96/300
218/218 [==============================] - 2s 11ms/step - loss: 0.6849 - ac
curacy: 0.7474 - val_loss: 0.7904 - val_accuracy: 0.6933
Epoch 97/300
218/218 [==============================] - 2s 11ms/step - loss: 0.6669 - ac
curacy: 0.7534 - val_loss: 0.8639 - val_accuracy: 0.6847
Epoch 98/300
218/218 [==============================] - 2s 11ms/step - loss: 0.6780 - ac
curacy: 0.7513 - val_loss: 0.8125 - val_accuracy: 0.6820
Epoch 99/300
218/218 [==============================] - 3s 12ms/step - loss: 0.6855 - ac
curacy: 0.7499 - val_loss: 0.7995 - val_accuracy: 0.7080
Epoch 100/300
218/218 [==============================] - 3s 12ms/step - loss: 0.6595 - ac
curacy: 0.7572 - val_loss: 0.8387 - val_accuracy: 0.7133
Epoch 101/300
218/218 [==============================] - 3s 12ms/step - loss: 0.6813 - ac
curacy: 0.7466 - val_loss: 0.8080 - val_accuracy: 0.7067
Epoch 102/300
218/218 [==============================] - 3s 12ms/step - loss: 0.6633 - ac
```

```
curacy: 0.7580 - val_loss: 0.7842 - val_accuracy: 0.7120
Epoch 103/300
218/218 [==============================] - 3s 12ms/step - loss: 0.6560 - ac
curacy: 0.7622 - val_loss: 0.7990 - val_accuracy: 0.6907
Epoch 104/300
218/218 [==============================] - 3s 13ms/step - loss: 0.6758 - ac
curacy: 0.7536 - val_loss: 0.8567 - val_accuracy: 0.6960
Epoch 105/300
218/218 [==============================] - 4s 17ms/step - loss: 0.6693 - ac
curacy: 0.7483 - val_loss: 0.8337 - val_accuracy: 0.7073
Epoch 106/300
218/218 [==============================] - 3s 14ms/step - loss: 0.6604 - ac
curacy: 0.7621 - val_loss: 0.8186 - val_accuracy: 0.7067
Epoch 107/300
218/218 [==============================] - 3s 16ms/step - loss: 0.6480 - ac
curacy: 0.7605 - val_loss: 0.7851 - val_accuracy: 0.7087
Epoch 108/300
218/218 [==============================] - 3s 15ms/step - loss: 0.6491 - ac
curacy: 0.7616 - val_loss: 0.8051 - val_accuracy: 0.6973
Epoch 109/300
218/218 [==============================] - 3s 16ms/step - loss: 0.6629 - ac
curacy: 0.7589 - val_loss: 0.7961 - val_accuracy: 0.7080
Epoch 110/300
218/218 [==============================] - 3s 15ms/step - loss: 0.6476 - ac
curacy: 0.7583 - val_loss: 0.8239 - val_accuracy: 0.6860
Epoch 111/300
218/218 [==============================] - 4s 16ms/step - loss: 0.6393 - ac
curacy: 0.7642 - val_loss: 0.8211 - val_accuracy: 0.6860
Epoch 112/300
218/218 [==============================] - 5s 24ms/step - loss: 0.6437 - ac
curacy: 0.7652 - val_loss: 0.8326 - val_accuracy: 0.6907
Epoch 113/300
218/218 [==============================] - 4s 17ms/step - loss: 0.6440 - ac
curacy: 0.7667 - val_loss: 0.7880 - val_accuracy: 0.7067
Epoch 114/300
218/218 [==============================] - 4s 17ms/step - loss: 0.6446 - ac
curacy: 0.7639 - val_loss: 0.7916 - val_accuracy: 0.7000
Epoch 115/300
218/218 [==============================] - 3s 15ms/step - loss: 0.6356 - ac
curacy: 0.7677 - val_loss: 0.8291 - val_accuracy: 0.7013
Epoch 116/300
218/218 [==============================] - 3s 16ms/step - loss: 0.6399 - ac
curacy: 0.7704 - val_loss: 0.8235 - val_accuracy: 0.6893
Epoch 117/300
218/218 [==============================] - 3s 16ms/step - loss: 0.6323 - ac
curacy: 0.7705 - val_loss: 0.7894 - val_accuracy: 0.7153
Epoch 118/300
218/218 [==============================] - 3s 16ms/step - loss: 0.6352 - ac
curacy: 0.7695 - val_loss: 0.7969 - val_accuracy: 0.6993
Epoch 119/300
218/218 [==============================] - 3s 16ms/step - loss: 0.6287 - ac
curacy: 0.7707 - val_loss: 0.8088 - val_accuracy: 0.6880
Epoch 120/300
218/218 [==============================] - 3s 15ms/step - loss: 0.6270 - ac
curacy: 0.7733 - val_loss: 0.8398 - val_accuracy: 0.6927
Epoch 121/300
218/218 [==============================] - 3s 16ms/step - loss: 0.6174 - ac
curacy: 0.7756 - val_loss: 0.7783 - val_accuracy: 0.7027
Epoch 122/300
```

```
218/218 [==============================] - 3s 15ms/step - loss: 0.6261 - ac
curacy: 0.7760 - val_loss: 0.7977 - val_accuracy: 0.7047
Epoch 123/300
218/218 [==============================] - 3s 16ms/step - loss: 0.6321 - ac
curacy: 0.7733 - val_loss: 0.7787 - val_accuracy: 0.7060
Epoch 124/300
218/218 [==============================] - 3s 15ms/step - loss: 0.6285 - ac
curacy: 0.7749 - val_loss: 0.8206 - val_accuracy: 0.6953
Epoch 125/300
218/218 [==============================] - 3s 15ms/step - loss: 0.6201 - ac
curacy: 0.7744 - val_loss: 0.7899 - val_accuracy: 0.6967
Epoch 126/300
218/218 [==============================] - 3s 16ms/step - loss: 0.6200 - ac
curacy: 0.7713 - val_loss: 0.8000 - val_accuracy: 0.6867
Epoch 127/300
218/218 [==============================] - 3s 15ms/step - loss: 0.6078 - ac
curacy: 0.7828 - val_loss: 0.7818 - val_accuracy: 0.6967
Epoch 128/300
218/218 [==============================] - 3s 15ms/step - loss: 0.6033 - ac
curacy: 0.7795 - val_loss: 0.7662 - val_accuracy: 0.7213
Epoch 129/300
218/218 [==============================] - 3s 15ms/step - loss: 0.6042 - ac
curacy: 0.7795 - val_loss: 0.8351 - val_accuracy: 0.6880
Epoch 130/300
218/218 [==============================] - 3s 15ms/step - loss: 0.6312 - ac
curacy: 0.7667 - val_loss: 0.8173 - val_accuracy: 0.6953
Epoch 131/300
218/218 [==============================] - 3s 15ms/step - loss: 0.6114 - ac
curacy: 0.7815 - val_loss: 0.7965 - val_accuracy: 0.7020
Epoch 132/300
218/218 [==============================] - 3s 15ms/step - loss: 0.6050 - ac
curacy: 0.7792 - val_loss: 0.7894 - val_accuracy: 0.6927
Epoch 133/300
218/218 [==============================] - 3s 13ms/step - loss: 0.6054 - ac
curacy: 0.7774 - val_loss: 0.8047 - val_accuracy: 0.6953
Epoch 134/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5974 - ac
curacy: 0.7859 - val_loss: 0.8263 - val_accuracy: 0.7027
Epoch 135/300
218/218 [==============================] - 3s 12ms/step - loss: 0.6100 - ac
curacy: 0.7750 - val_loss: 0.7890 - val_accuracy: 0.7167
Epoch 136/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5921 - ac
curacy: 0.7830 - val_loss: 0.7721 - val_accuracy: 0.7180
Epoch 137/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5916 - ac
curacy: 0.7879 - val_loss: 0.7850 - val_accuracy: 0.7107
Epoch 138/300
218/218 [==============================] - 2s 11ms/step - loss: 0.6084 - ac
curacy: 0.7787 - val_loss: 0.7939 - val_accuracy: 0.7053
Epoch 139/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5866 - ac
curacy: 0.7944 - val_loss: 0.7638 - val_accuracy: 0.7127
Epoch 140/300
218/218 [==============================] - 2s 11ms/step - loss: 0.6029 - ac
curacy: 0.7779 - val_loss: 0.8294 - val_accuracy: 0.6880
Epoch 141/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5926 - ac
curacy: 0.7815 - val_loss: 0.8059 - val_accuracy: 0.7060
```

```
Epoch 142/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5916 - ac
curacy: 0.7874 - val_loss: 0.7802 - val_accuracy: 0.7040
Epoch 143/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5994 - ac
curacy: 0.7813 - val_loss: 0.7834 - val_accuracy: 0.7080
Epoch 144/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5721 - ac
curacy: 0.7895 - val_loss: 0.7932 - val_accuracy: 0.7047
Epoch 145/300
218/218 [==============================] - 3s 11ms/step - loss: 0.5865 - ac
curacy: 0.7895 - val_loss: 0.8439 - val_accuracy: 0.6853
Epoch 146/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5857 - ac
curacy: 0.7885 - val_loss: 0.8054 - val_accuracy: 0.7013
Epoch 147/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5807 - ac
curacy: 0.7902 - val_loss: 0.7850 - val_accuracy: 0.7087
Epoch 148/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5854 - ac
curacy: 0.7866 - val_loss: 0.7767 - val_accuracy: 0.7027
Epoch 149/300
218/218 [==============================] - 3s 13ms/step - loss: 0.5997 - ac
curacy: 0.7816 - val_loss: 0.7935 - val_accuracy: 0.7120
Epoch 150/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5708 - ac
curacy: 0.7886 - val_loss: 0.8585 - val_accuracy: 0.6927
Epoch 151/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5794 - ac
curacy: 0.7951 - val_loss: 0.7866 - val_accuracy: 0.7033
Epoch 152/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5829 - ac
curacy: 0.7898 - val_loss: 0.7841 - val_accuracy: 0.7140
Epoch 153/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5800 - ac
curacy: 0.7944 - val_loss: 0.7745 - val_accuracy: 0.6967
Epoch 154/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5783 - ac
curacy: 0.7889 - val_loss: 0.7824 - val_accuracy: 0.7047
Epoch 155/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5819 - ac
curacy: 0.7901 - val_loss: 0.8203 - val_accuracy: 0.6933
Epoch 156/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5665 - ac
curacy: 0.7915 - val_loss: 0.7891 - val_accuracy: 0.7087
Epoch 157/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5760 - ac
curacy: 0.7918 - val_loss: 0.7837 - val_accuracy: 0.6980
Epoch 158/300
218/218 [==============================] - 3s 13ms/step - loss: 0.5771 - ac
curacy: 0.7974 - val_loss: 0.7694 - val_accuracy: 0.7180
Epoch 159/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5785 - ac
curacy: 0.7868 - val_loss: 0.8073 - val_accuracy: 0.6887
Epoch 160/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5607 - ac
curacy: 0.7950 - val_loss: 0.8216 - val_accuracy: 0.7007
Epoch 161/300
218/218 [==============================] - 3s 15ms/step - loss: 0.5783 - ac
```

```
curacy: 0.7961 - val_loss: 0.8311 - val_accuracy: 0.6933
Epoch 162/300
218/218 [==============================] - 3s 13ms/step - loss: 0.5637 - ac
curacy: 0.7895 - val_loss: 0.7625 - val_accuracy: 0.7060
Epoch 163/300
218/218 [==============================] - 3s 16ms/step - loss: 0.5718 - ac
curacy: 0.7940 - val_loss: 0.8056 - val_accuracy: 0.6900
Epoch 164/300
218/218 [==============================] - 3s 15ms/step - loss: 0.5590 - ac
curacy: 0.7947 - val_loss: 0.7888 - val_accuracy: 0.6980
Epoch 165/300
218/218 [==============================] - 3s 16ms/step - loss: 0.5578 - ac
curacy: 0.7950 - val_loss: 0.7757 - val_accuracy: 0.6993
Epoch 166/300
218/218 [==============================] - 3s 16ms/step - loss: 0.5511 - ac
curacy: 0.7999 - val_loss: 0.8030 - val_accuracy: 0.6847
Epoch 167/300
218/218 [==============================] - 3s 16ms/step - loss: 0.5512 - ac
curacy: 0.8065 - val_loss: 0.8061 - val_accuracy: 0.6960
Epoch 168/300
218/218 [==============================] - 3s 16ms/step - loss: 0.5341 - ac
curacy: 0.8003 - val_loss: 0.7774 - val_accuracy: 0.7080
Epoch 169/300
218/218 [==============================] - 4s 17ms/step - loss: 0.5617 - ac
curacy: 0.7932 - val_loss: 0.7573 - val_accuracy: 0.7193
Epoch 170/300
218/218 [==============================] - 3s 15ms/step - loss: 0.5379 - ac
curacy: 0.8119 - val_loss: 0.7899 - val_accuracy: 0.7140
Epoch 171/300
218/218 [==============================] - 3s 16ms/step - loss: 0.5575 - ac
curacy: 0.8045 - val_loss: 0.7925 - val_accuracy: 0.7067
Epoch 172/300
218/218 [==============================] - 3s 15ms/step - loss: 0.5377 - ac
curacy: 0.8089 - val_loss: 0.8021 - val_accuracy: 0.7047
Epoch 173/300
218/218 [==============================] - 3s 16ms/step - loss: 0.5500 - ac
curacy: 0.8022 - val_loss: 0.7653 - val_accuracy: 0.7207
Epoch 174/300
218/218 [==============================] - 3s 15ms/step - loss: 0.5504 - ac
curacy: 0.8055 - val_loss: 0.8066 - val_accuracy: 0.7140
Epoch 175/300
218/218 [==============================] - 3s 15ms/step - loss: 0.5645 - ac
curacy: 0.7961 - val_loss: 0.8357 - val_accuracy: 0.6860
Epoch 176/300
218/218 [==============================] - 4s 16ms/step - loss: 0.5459 - ac
curacy: 0.8013 - val_loss: 0.7636 - val_accuracy: 0.7167
Epoch 177/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5536 - ac
curacy: 0.7983 - val_loss: 0.7931 - val_accuracy: 0.7167
Epoch 178/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5616 - ac
curacy: 0.7938 - val_loss: 0.7987 - val_accuracy: 0.6960
Epoch 179/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5290 - ac
curacy: 0.8059 - val_loss: 0.8029 - val_accuracy: 0.7133
Epoch 180/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5385 - ac
curacy: 0.8059 - val_loss: 0.7840 - val_accuracy: 0.7120
Epoch 181/300
```

```
218/218 [==============================] - 2s 11ms/step - loss: 0.5422 - ac
curacy: 0.8014 - val_loss: 0.7359 - val_accuracy: 0.7247
Epoch 182/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5549 - ac
curacy: 0.7993 - val_loss: 0.7671 - val_accuracy: 0.7200
Epoch 183/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5260 - ac
curacy: 0.8132 - val_loss: 0.7879 - val_accuracy: 0.7180
Epoch 184/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5362 - ac
curacy: 0.8065 - val_loss: 0.7922 - val_accuracy: 0.7180
Epoch 185/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5263 - ac
curacy: 0.8125 - val_loss: 0.7671 - val_accuracy: 0.7233
Epoch 186/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5230 - ac
curacy: 0.8101 - val_loss: 0.7806 - val_accuracy: 0.7240
Epoch 187/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5272 - ac
curacy: 0.8109 - val_loss: 0.8070 - val_accuracy: 0.7167
Epoch 188/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5262 - ac
curacy: 0.8115 - val_loss: 0.7815 - val_accuracy: 0.7080
Epoch 189/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5338 - ac
curacy: 0.8036 - val_loss: 0.7698 - val_accuracy: 0.7213
Epoch 190/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5446 - ac
curacy: 0.8066 - val_loss: 0.7576 - val_accuracy: 0.7187
Epoch 191/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5311 - ac
curacy: 0.8096 - val_loss: 0.7833 - val_accuracy: 0.7113
Epoch 192/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5362 - ac
curacy: 0.8098 - val_loss: 0.7433 - val_accuracy: 0.7220
Epoch 193/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5265 - ac
curacy: 0.8106 - val_loss: 0.7677 - val_accuracy: 0.7053
Epoch 194/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5347 - ac
curacy: 0.8075 - val_loss: 0.7814 - val_accuracy: 0.7180
Epoch 195/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5383 - ac
curacy: 0.8069 - val_loss: 0.7632 - val_accuracy: 0.7200
Epoch 196/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5311 - ac
curacy: 0.8066 - val_loss: 0.7473 - val_accuracy: 0.7207
Epoch 197/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5278 - ac
curacy: 0.8125 - val_loss: 0.8008 - val_accuracy: 0.7013
Epoch 198/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5082 - ac
curacy: 0.8116 - val_loss: 0.7281 - val_accuracy: 0.7227
Epoch 199/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5356 - ac
curacy: 0.8057 - val_loss: 0.7429 - val_accuracy: 0.7120
Epoch 200/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5289 - ac
curacy: 0.8057 - val_loss: 0.7806 - val_accuracy: 0.7113
```

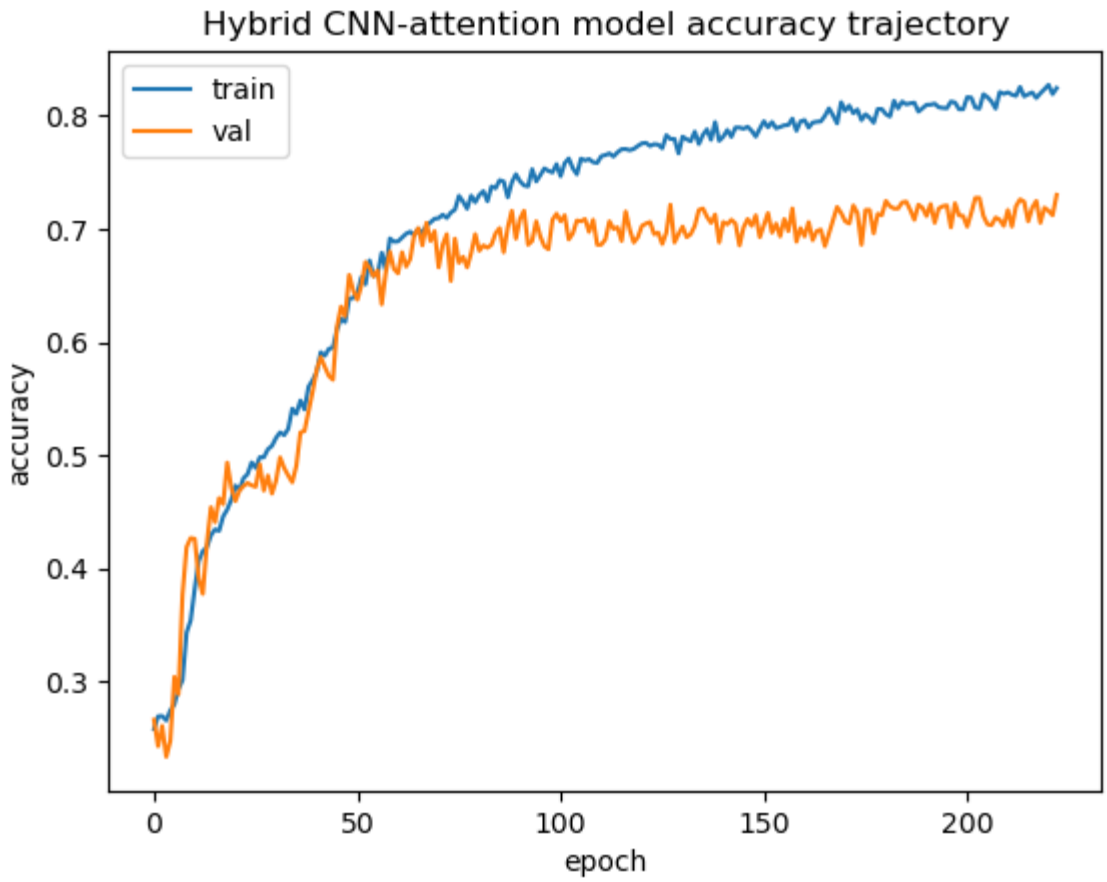
```
Epoch 201/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5092 - ac
curacy: 0.8164 - val_loss: 0.7941 - val_accuracy: 0.7020
Epoch 202/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5143 - ac
curacy: 0.8159 - val_loss: 0.7493 - val_accuracy: 0.7140
Epoch 203/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5207 - ac
curacy: 0.8072 - val_loss: 0.7328 - val_accuracy: 0.7273
Epoch 204/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5295 - ac
curacy: 0.8063 - val_loss: 0.7353 - val_accuracy: 0.7273
Epoch 205/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5053 - ac
curacy: 0.8190 - val_loss: 0.7600 - val_accuracy: 0.7113
Epoch 206/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5125 - ac
curacy: 0.8164 - val_loss: 0.8007 - val_accuracy: 0.7040
Epoch 207/300
218/218 [==============================] - 3s 11ms/step - loss: 0.5160 - ac
curacy: 0.8125 - val_loss: 0.7882 - val_accuracy: 0.7033
Epoch 208/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5303 - ac
curacy: 0.8063 - val_loss: 0.7639 - val_accuracy: 0.7100
Epoch 209/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5073 - ac
curacy: 0.8210 - val_loss: 0.7992 - val_accuracy: 0.7080
Epoch 210/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5115 - ac
curacy: 0.8193 - val_loss: 0.7941 - val_accuracy: 0.7027
Epoch 211/300
218/218 [==============================] - 3s 12ms/step - loss: 0.4991 - ac
curacy: 0.8207 - val_loss: 0.7627 - val_accuracy: 0.7167
Epoch 212/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5105 - ac
curacy: 0.8187 - val_loss: 0.7915 - val_accuracy: 0.7020
Epoch 213/300
218/218 [==============================] - 2s 11ms/step - loss: 0.4960 - ac
curacy: 0.8175 - val_loss: 0.7536 - val_accuracy: 0.7200
Epoch 214/300
218/218 [==============================] - 2s 11ms/step - loss: 0.4911 - ac
curacy: 0.8254 - val_loss: 0.7320 - val_accuracy: 0.7260
Epoch 215/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5054 - ac
curacy: 0.8171 - val_loss: 0.7580 - val_accuracy: 0.7240
Epoch 216/300
218/218 [==============================] - 2s 11ms/step - loss: 0.5013 - ac
curacy: 0.8190 - val_loss: 0.7749 - val_accuracy: 0.7067
Epoch 217/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5023 - ac
curacy: 0.8205 - val_loss: 0.7923 - val_accuracy: 0.7187
Epoch 218/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5119 - ac
curacy: 0.8157 - val_loss: 0.7317 - val_accuracy: 0.7247
Epoch 219/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5017 - ac
curacy: 0.8198 - val_loss: 0.7757 - val_accuracy: 0.7053
Epoch 220/300
218/218 [==============================] - 2s 11ms/step - loss: 0.4942 - ac
```

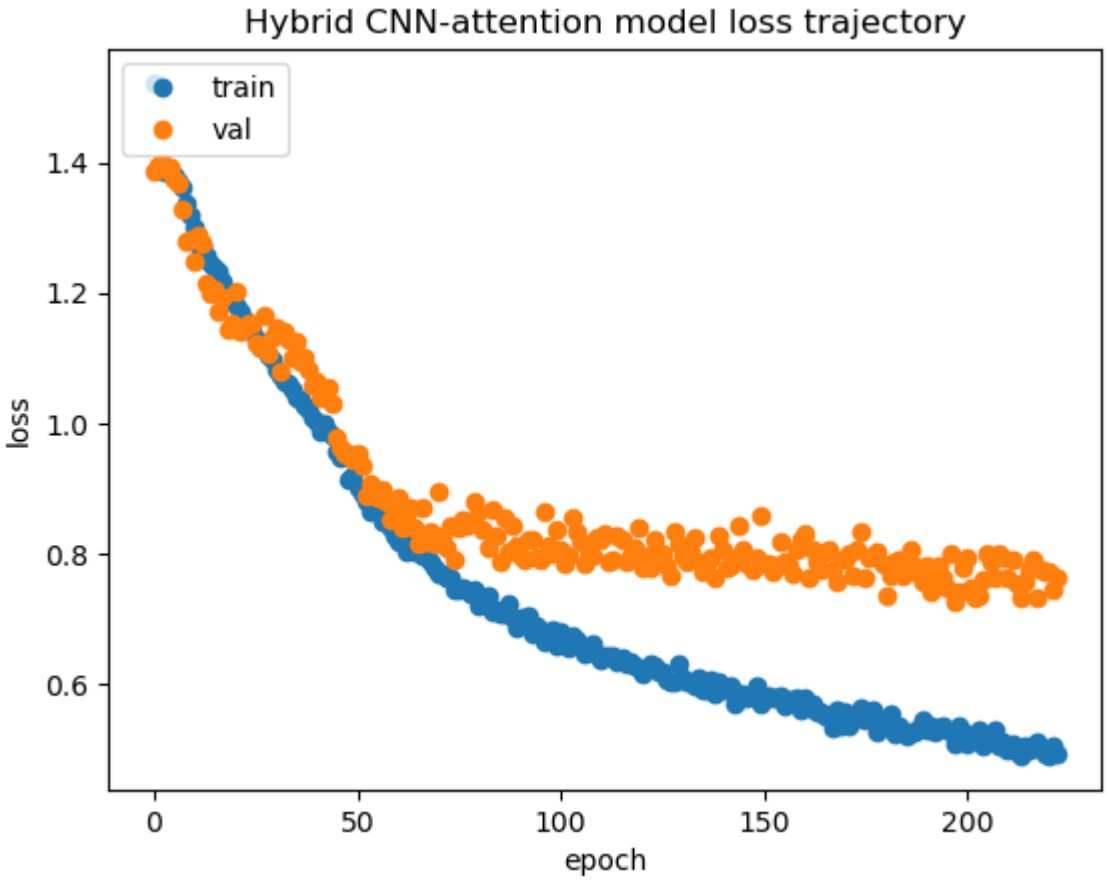
```
curacy: 0.8234 - val_loss: 0.7727 - val_accuracy: 0.7187
Epoch 221/300
218/218 [==============================] - 2s 11ms/step - loss: 0.4915 - ac
curacy: 0.8273 - val_loss: 0.7725 - val_accuracy: 0.7160
Epoch 222/300
218/218 [==============================] - 3s 12ms/step - loss: 0.5070 - ac
curacy: 0.8191 - val_loss: 0.7453 - val_accuracy: 0.7120
Epoch 223/300
214/218 [============================>.] - ETA: 0s - loss: 0.4946 - accurac
y: 0.8245
Reached 73.00% accuracy, so stopping training!!
218/218 [==============================] - 3s 12ms/step - loss: 0.4953 - ac
curacy: 0.8244 - val_loss: 0.7651 - val_accuracy: 0.7300
```

(vi)(CNN-MHA) Visualizing the accuracy and loss trajectory

In [ ]:

```python
import matplotlib.pyplot as plt

# Plotting accuracy trajectory
plt.plot(hybrid_cnn_lstm_model_results.history['accuracy'])
plt.plot(hybrid_cnn_lstm_model_results.history['val_accuracy'])
plt.title('Hybrid CNN-attention model accuracy trajectory')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

# Plotting loss trajectory
plt.plot(hybrid_cnn_lstm_model_results.history['loss'],'o')
plt.plot(hybrid_cnn_lstm_model_results.history['val_loss'],'o')
plt.title('Hybrid CNN-attention model loss trajectory')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```


Hybrid CNN-attention model accuracy trajectory
train
val
accuracy
epoch
0.3
0.4
0.5
0.6
0.7
0.8
0
50
100
150
200


Hybrid CNN-attention model loss trajectory
train
val
loss
epoch
0.6
0.8
1.0
1.2
1.4
0
50
100
150
200

(vii)(CNN-MHA) Testing the performance of the hybrid CNN-MHA model on the held out test set

In [ ]:
```
## Testing the hybrid CNN-MHA model

hybrid_cnn_mha_score = hybrid_cnn_mha_model.evaluate(x_test, y_test, verbose
print('Test accuracy of the hybrid CNN-MHA model:',hybrid_cnn_mha_score[1])
```

```
Test accuracy of the hybrid CNN-attention model: 0.707110583782196
```

In [ ]:

CNN-LSTM-Self Attention

(i) Importing the necessary packages

In [ ]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
import keras
from keras_self_attention import SeqSelfAttention

from keras import layers
from keras.models import Sequential,Model
from keras.layers import Dense, Activation, Flatten,Dropout, MultiHeadAttent
from keras.layers import Conv2D,LSTM,BatchNormalization,MaxPooling2D,Reshape
from keras.utils import to_categorical
import matplotlib.pyplot as plt
```

(ii) Preprocessing the dataset and preparing the training, validation, and test datasets

In [ ]:

```
def data_prep(X,y,sub_sample,average,noise):

    total_X = None
    total_y = None

    # Trimming the data (sample,22,1000) -> (sample,22,500)
    X = X[:,:,0:500]
    print('Shape of X after trimming:',X.shape)

    # Maxpooling the data (sample,22,1000) -> (sample,22,500/sub_sample)
    X_max = np.max(X.reshape(X.shape[0], X.shape[1], -1, sub_sample), axis=3


    total_X = X_max
    total_y = y
    print('Shape of X after maxpooling:',total_X.shape)

    # Averaging + noise
    X_average = np.mean(X.reshape(X.shape[0], X.shape[1], -1, average),axis=
    X_average = X_average + np.random.normal(0.0, 0.5, X_average.shape)

    total_X = np.vstack((total_X, X_average))
    total_y = np.hstack((total_y, y))
    print('Shape of X after averaging+noise and concatenating:',total_X.shap

    # Subsampling

    for i in range(sub_sample):

        X_subsample = X[:, :, i::sub_sample] + \
                            (np.random.normal(0.0, 0.5, X[:, :,i::sub_sample

        total_X = np.vstack((total_X, X_subsample))
        total_y = np.hstack((total_y, y))


    print('Shape of X after subsampling and concatenating:',total_X.shape)
    return total_X,total_y
```

In [ ]:

```
## Loading the dataset


X_test = np.load("X_test.npy")
y_test = np.load("y_test.npy")
person_train_valid = np.load("person_train_valid.npy")
X_train_valid = np.load("X_train_valid.npy")
y_train_valid = np.load("y_train_valid.npy")
person_test = np.load("person_test.npy")

## Adjusting the labels so that

# Cue onset left - 0
# Cue onset right - 1
# Cue onset foot - 2
# Cue onset tongue - 3

y_train_valid -= 769
y_test -= 769


## Random splitting and reshaping the data
# First generating the training and validation indices using random splittir

ind_valid = np.random.choice(2115, 375, replace=False)
ind_train = np.array(list(set(range(2115)).difference(set(ind_valid))))

# Creating the training and validation sets using the generated indices
(X_train, X_valid) = X_train_valid[ind_train], X_train_valid[ind_valid]
(y_train, y_valid) = y_train_valid[ind_train], y_train_valid[ind_valid]


## Preprocessing the dataset
x_train,y_train = data_prep(X_train,y_train,2,2,True)
x_valid,y_valid = data_prep(X_valid,y_valid,2,2,True)
X_test_prep,y_test_prep = data_prep(X_test,y_test,2,2,True)


print('Shape of training set:',x_train.shape)
print('Shape of validation set:',x_valid.shape)
print('Shape of training labels:',y_train.shape)
print('Shape of validation labels:',y_valid.shape)
print('Shape of testing set:',X_test_prep.shape)
print('Shape of testing labels:',y_test_prep.shape)


# Converting the labels to categorical variables for multiclass classificati
y_train = to_categorical(y_train, 4)
y_valid = to_categorical(y_valid, 4)
y_test = to_categorical(y_test_prep, 4)
print('Shape of training labels after categorical conversion:',y_train.shape
print('Shape of validation labels after categorical conversion:',y_valid.sha
print('Shape of test labels after categorical conversion:',y_test.shape)

# Adding width of the segment to be 1
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], x_train.shape[
x_valid = x_valid.reshape(x_valid.shape[0], x_valid.shape[1], x_train.shape[
x_test = X_test_prep.reshape(X_test_prep.shape[0], X_test_prep.shape[1], X_t
print('Shape of training set after adding width info:',x_train.shape)
```

```
print('Shape of training set after adding width info:',x_train.shape)
print('Shape of validation set after adding width info:',x_valid.shape)
print('Shape of test set after adding width info:',x_test.shape)


# Reshaping the training and validation dataset
x_train = np.swapaxes(x_train, 1,3)
x_train = np.swapaxes(x_train, 1,2)
x_valid = np.swapaxes(x_valid, 1,3)
x_valid = np.swapaxes(x_valid, 1,2)
x_test = np.swapaxes(x_test, 1,3)
x_test = np.swapaxes(x_test, 1,2)
print('Shape of training set after dimension reshaping:',x_train.shape)
print('Shape of validation set after dimension reshaping:',x_valid.shape)
print('Shape of test set after dimension reshaping:',x_test.shape)

keras.backend.clear_session()
```

```
Shape of X after trimming: (1740, 22, 500)
Shape of X after maxpooling: (1740, 22, 250)
Shape of X after averaging+noise and concatenating: (3480, 22, 250)
Shape of X after subsampling and concatenating: (6960, 22, 250)
Shape of X after trimming: (375, 22, 500)
Shape of X after maxpooling: (375, 22, 250)
Shape of X after averaging+noise and concatenating: (750, 22, 250)
Shape of X after subsampling and concatenating: (1500, 22, 250)
Shape of X after trimming: (443, 22, 500)
Shape of X after maxpooling: (443, 22, 250)
Shape of X after averaging+noise and concatenating: (886, 22, 250)
Shape of X after subsampling and concatenating: (1772, 22, 250)
Shape of training set: (6960, 22, 250)
Shape of validation set: (1500, 22, 250)
Shape of training labels: (6960,)
Shape of validation labels: (1500,)
Shape of testing set: (1772, 22, 250)
Shape of testing labels: (1772,)
Shape of training labels after categorical conversion: (6960, 4)
Shape of validation labels after categorical conversion: (1500, 4)
Shape of test labels after categorical conversion: (1772, 4)
Shape of training set after adding width info: (6960, 22, 250, 1)
Shape of validation set after adding width info: (1500, 22, 250, 1)
Shape of test set after adding width info: (1772, 22, 250, 1)
Shape of training set after dimension reshaping: (6960, 250, 1, 22)
Shape of validation set after dimension reshaping: (1500, 250, 1, 22)
Shape of test set after dimension reshaping: (1772, 250, 1, 22)
```

(iii)(CNN-LSTM) Defining the architecture of the hybrid CNN-LSTM model

In [ ]:

```
def build_model(num_heads=2, input_shape=(250, 1, 22), num_classes=4):
    n_frames = 250
    n_channels = 22
    # Conv. block 1
    In1 = keras.Input(shape =(250,1,22) )
    c1 = Conv2D(filters=30, kernel_size=(11,1), padding='same', activation='
    p1 =  MaxPooling2D(pool_size=(4,1), padding='same')(c1) # Read the keras
    b1 = BatchNormalization()(p1)
    d1 = Dropout(0.6)(b1)

    # Conv. block 2
    c2 = Conv2D(filters=60, kernel_size=(9,1), padding='same', activation='s
    p2 =  MaxPooling2D(pool_size=(4,1), padding='same')(c2) # Read the keras
    b2 = BatchNormalization()(p2)
    d2 = Dropout(0.6)(b2)

    # Conv. block 3
    c3 = Conv2D(filters=120, kernel_size=(5,1), padding='same', activation='
    p3 =  MaxPooling2D(pool_size=(4,1), padding='same')(c3) # Read the keras
    b3 = BatchNormalization()(p3)
    d3 = Dropout(0.6)(b3)

    # Conv. block 4
    c4 = Conv2D(filters=240, kernel_size=(3,1), padding='same', activation='
    p4 =  MaxPooling2D(pool_size=(4,1), padding='same')(c4) # Read the keras
    b4 = BatchNormalization()(p4)
    d4 = Dropout(0.6)(b4)

    # LSTM block
    lstm1 = Reshape((1, -1))(d4)
    lstm1 = Bidirectional(LSTM(units=124, dropout=0.6, recurrent_dropout=0.1
    lstm1 = LayerNormalization()(lstm1)

    # self attention block
    selfatt = SeqSelfAttention(attention_activation='gelu')(lstm1)
    selfatt = GlobalAveragePooling1D()(selfatt)



    # # # Add fully connected layers
    fc2 = Dense(4, activation='softmax')(selfatt)

    # Define the final model
    final_model = Model(inputs=In1, outputs=[fc2])
    final_model.summary()
    return final_model
```

(iv) (CNN-LSTM) Defining the hyperparameters of the hybrid CNN-LSTM model

In [ ]:
```
# Model parameters

epochs = 200
initial_learning_rate = 1e-3
decay_steps = 1000
decay_rate = 0.99

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate,
    decay_steps=decay_steps,
    decay_rate=decay_rate
)

optimizer = keras.optimizers.Adam(learning_rate=lr_schedule)
```

(v) (CNN-LSTM) Compiling, training and validating the model

In [ ]:
```
# Compiling the model
hybrid_cnn_lstm_model = build_model()
# Compiling the model
hybrid_cnn_lstm_model.compile(loss='categorical_crossentropy',
              optimizer=optimizer,
              metrics=['accuracy'])

# Training and validating the model
callback = keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=20
hybrid_cnn_lstm_model_results = hybrid_cnn_lstm_model.fit(x_train,
               y_train,
               batch_size=32,
               epochs=epochs,
               validation_data=(x_valid, y_valid),
               verbose=True,
               callbacks = [callback])
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 250, 1, 22)]      0

 conv2d (Conv2D)             (None, 250, 1, 30)        7290

 max_pooling2d (MaxPooling2D  (None, 63, 1, 30)        0
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 250, 1, 22)]      0

 conv2d (Conv2D)             (None, 250, 1, 30)        7290

 max_pooling2d (MaxPooling2D  (None, 63, 1, 30)        0
 )

 batch_normalization (BatchN  (None, 63, 1, 30)        120
 ormalization)

 dropout (Dropout)           (None, 63, 1, 30)         0

 conv2d_1 (Conv2D)           (None, 63, 1, 60)         16260

 max_pooling2d_1 (MaxPooling  (None, 16, 1, 60)        0
 2D)

 batch_normalization_1 (Batc  (None, 16, 1, 60)        240
 hNormalization)

 dropout_1 (Dropout)         (None, 16, 1, 60)         0

 conv2d_2 (Conv2D)           (None, 16, 1, 120)        36120

 max_pooling2d_2 (MaxPooling  (None, 4, 1, 120)        0
 2D)

 batch_normalization_2 (Batc  (None, 4, 1, 120)        480
 hNormalization)

 dropout_2 (Dropout)         (None, 4, 1, 120)         0

 conv2d_3 (Conv2D)           (None, 4, 1, 240)         86640

 max_pooling2d_3 (MaxPooling  (None, 1, 1, 240)        0
 2D)

 batch_normalization_3 (Batc  (None, 1, 1, 240)        960
 hNormalization)

 dropout_3 (Dropout)         (None, 1, 1, 240)         0

 reshape (Reshape)           (None, 1, 240)            0

 bidirectional (Bidirectiona  (None, 1, 248)           362080
 l)
```

```
 layer_normalization (LayerN  (None, 1, 248)          496
 ormalization)

 seq_self_attention (SeqSelf  (None, 1, 248)          15937
 Attention)

 global_average_pooling1d (G  (None, 248)             0
 lobalAveragePooling1D)

 dense (Dense)               (None, 4)                996

=================================================================
Total params: 527,619
Trainable params: 526,719
Non-trainable params: 900
_________________________________________________________________
Epoch 1/200
218/218 [==============================] - 9s 22ms/step - loss: 1.7706 - ac
curacy: 0.2533 - val_loss: 1.6084 - val_accuracy: 0.2507
Epoch 2/200
218/218 [==============================] - 4s 17ms/step - loss: 1.5303 - ac
curacy: 0.2592 - val_loss: 1.4583 - val_accuracy: 0.2720
Epoch 3/200
218/218 [==============================] - 4s 19ms/step - loss: 1.4488 - ac
curacy: 0.2662 - val_loss: 1.4491 - val_accuracy: 0.2760
Epoch 4/200
218/218 [==============================] - 4s 18ms/step - loss: 1.4170 - ac
curacy: 0.2721 - val_loss: 1.4806 - val_accuracy: 0.2813
Epoch 5/200
218/218 [==============================] - 4s 18ms/step - loss: 1.3968 - ac
curacy: 0.2792 - val_loss: 1.4810 - val_accuracy: 0.2893
Epoch 6/200
218/218 [==============================] - 4s 18ms/step - loss: 1.3848 - ac
curacy: 0.2932 - val_loss: 1.4235 - val_accuracy: 0.2787
Epoch 7/200
218/218 [==============================] - 4s 18ms/step - loss: 1.3796 - ac
curacy: 0.2996 - val_loss: 1.4273 - val_accuracy: 0.3060
Epoch 8/200
218/218 [==============================] - 4s 18ms/step - loss: 1.3748 - ac
curacy: 0.3050 - val_loss: 1.4436 - val_accuracy: 0.2933
Epoch 9/200
218/218 [==============================] - 4s 18ms/step - loss: 1.3648 - ac
curacy: 0.3172 - val_loss: 1.4524 - val_accuracy: 0.2707
Epoch 10/200
218/218 [==============================] - 4s 18ms/step - loss: 1.3563 - ac
curacy: 0.3251 - val_loss: 1.4445 - val_accuracy: 0.3100
Epoch 11/200
218/218 [==============================] - 4s 18ms/step - loss: 1.3384 - ac
curacy: 0.3402 - val_loss: 1.4544 - val_accuracy: 0.3447
Epoch 12/200
218/218 [==============================] - 4s 20ms/step - loss: 1.3167 - ac
curacy: 0.3626 - val_loss: 1.3783 - val_accuracy: 0.3607
Epoch 13/200
218/218 [==============================] - 4s 18ms/step - loss: 1.3033 - ac
curacy: 0.3750 - val_loss: 1.3561 - val_accuracy: 0.3747
Epoch 14/200
218/218 [==============================] - 4s 18ms/step - loss: 1.2848 - ac
curacy: 0.3884 - val_loss: 1.3375 - val_accuracy: 0.3953
Epoch 15/200
```

```
218/218 [==============================] - 4s 18ms/step - loss: 1.2729 - ac
curacy: 0.3964 - val_loss: 1.3498 - val_accuracy: 0.3960
Epoch 16/200
218/218 [==============================] - 4s 18ms/step - loss: 1.2651 - ac
curacy: 0.4082 - val_loss: 1.3246 - val_accuracy: 0.3893
Epoch 17/200
218/218 [==============================] - 4s 17ms/step - loss: 1.2564 - ac
curacy: 0.4152 - val_loss: 1.2874 - val_accuracy: 0.4287
Epoch 18/200
218/218 [==============================] - 4s 18ms/step - loss: 1.2459 - ac
curacy: 0.4234 - val_loss: 1.2568 - val_accuracy: 0.4593
Epoch 19/200
218/218 [==============================] - 4s 18ms/step - loss: 1.2374 - ac
curacy: 0.4290 - val_loss: 1.2368 - val_accuracy: 0.4460
Epoch 20/200
218/218 [==============================] - 4s 18ms/step - loss: 1.2355 - ac
curacy: 0.4297 - val_loss: 1.2585 - val_accuracy: 0.4627
Epoch 21/200
218/218 [==============================] - 4s 18ms/step - loss: 1.2108 - ac
curacy: 0.4519 - val_loss: 1.2341 - val_accuracy: 0.4827
Epoch 22/200
218/218 [==============================] - 5s 22ms/step - loss: 1.2057 - ac
curacy: 0.4470 - val_loss: 1.2907 - val_accuracy: 0.4533
Epoch 23/200
218/218 [==============================] - 4s 18ms/step - loss: 1.2003 - ac
curacy: 0.4507 - val_loss: 1.2393 - val_accuracy: 0.4427
Epoch 24/200
218/218 [==============================] - 4s 19ms/step - loss: 1.1898 - ac
curacy: 0.4559 - val_loss: 1.2273 - val_accuracy: 0.4613
Epoch 25/200
218/218 [==============================] - 4s 18ms/step - loss: 1.1877 - ac
curacy: 0.4647 - val_loss: 1.2187 - val_accuracy: 0.4620
Epoch 26/200
218/218 [==============================] - 4s 18ms/step - loss: 1.1817 - ac
curacy: 0.4677 - val_loss: 1.1949 - val_accuracy: 0.4760
Epoch 27/200
218/218 [==============================] - 4s 18ms/step - loss: 1.1743 - ac
curacy: 0.4737 - val_loss: 1.1762 - val_accuracy: 0.4813
Epoch 28/200
218/218 [==============================] - 4s 16ms/step - loss: 1.1713 - ac
curacy: 0.4787 - val_loss: 1.1786 - val_accuracy: 0.4980
Epoch 29/200
218/218 [==============================] - 3s 15ms/step - loss: 1.1564 - ac
curacy: 0.4802 - val_loss: 1.2041 - val_accuracy: 0.5013
Epoch 30/200
218/218 [==============================] - 3s 13ms/step - loss: 1.1465 - ac
curacy: 0.4921 - val_loss: 1.1558 - val_accuracy: 0.5120
Epoch 31/200
218/218 [==============================] - 3s 14ms/step - loss: 1.1426 - ac
curacy: 0.5000 - val_loss: 1.1332 - val_accuracy: 0.5220
Epoch 32/200
218/218 [==============================] - 3s 16ms/step - loss: 1.1304 - ac
curacy: 0.5116 - val_loss: 1.1288 - val_accuracy: 0.5387
Epoch 33/200
218/218 [==============================] - 3s 14ms/step - loss: 1.1169 - ac
curacy: 0.5194 - val_loss: 1.1211 - val_accuracy: 0.5307
Epoch 34/200
218/218 [==============================] - 3s 13ms/step - loss: 1.1120 - ac
curacy: 0.5198 - val_loss: 1.1439 - val_accuracy: 0.5313
```

```
Epoch 35/200
218/218 [==============================] - 3s 14ms/step - loss: 1.0922 - ac
curacy: 0.5435 - val_loss: 1.1107 - val_accuracy: 0.5373
Epoch 36/200
218/218 [==============================] - 3s 14ms/step - loss: 1.0777 - ac
curacy: 0.5448 - val_loss: 1.0957 - val_accuracy: 0.5593
Epoch 37/200
218/218 [==============================] - 3s 14ms/step - loss: 1.0892 - ac
curacy: 0.5398 - val_loss: 1.0651 - val_accuracy: 0.5800
Epoch 38/200
218/218 [==============================] - 3s 14ms/step - loss: 1.0583 - ac
curacy: 0.5591 - val_loss: 1.0853 - val_accuracy: 0.5760
Epoch 39/200
218/218 [==============================] - 3s 14ms/step - loss: 1.0536 - ac
curacy: 0.5664 - val_loss: 1.0426 - val_accuracy: 0.6007
Epoch 40/200
218/218 [==============================] - 3s 14ms/step - loss: 1.0397 - ac
curacy: 0.5717 - val_loss: 1.0266 - val_accuracy: 0.6087
Epoch 41/200
218/218 [==============================] - 3s 15ms/step - loss: 1.0346 - ac
curacy: 0.5767 - val_loss: 1.0194 - val_accuracy: 0.6107
Epoch 42/200
218/218 [==============================] - 3s 14ms/step - loss: 1.0192 - ac
curacy: 0.5774 - val_loss: 1.0060 - val_accuracy: 0.6140
Epoch 43/200
218/218 [==============================] - 3s 14ms/step - loss: 1.0003 - ac
curacy: 0.5955 - val_loss: 0.9844 - val_accuracy: 0.6293
Epoch 44/200
218/218 [==============================] - 3s 15ms/step - loss: 0.9918 - ac
curacy: 0.5963 - val_loss: 0.9557 - val_accuracy: 0.6273
Epoch 45/200
218/218 [==============================] - 3s 15ms/step - loss: 0.9783 - ac
curacy: 0.6065 - val_loss: 0.9676 - val_accuracy: 0.6147
Epoch 46/200
218/218 [==============================] - 3s 14ms/step - loss: 0.9884 - ac
curacy: 0.5996 - val_loss: 0.9613 - val_accuracy: 0.6267
Epoch 47/200
218/218 [==============================] - 3s 14ms/step - loss: 0.9546 - ac
curacy: 0.6121 - val_loss: 0.9217 - val_accuracy: 0.6333
Epoch 48/200
218/218 [==============================] - 3s 13ms/step - loss: 0.9514 - ac
curacy: 0.6144 - val_loss: 0.9535 - val_accuracy: 0.6273
Epoch 49/200
218/218 [==============================] - 3s 14ms/step - loss: 0.9567 - ac
curacy: 0.6141 - val_loss: 0.9620 - val_accuracy: 0.6433
Epoch 50/200
218/218 [==============================] - 3s 14ms/step - loss: 0.9487 - ac
curacy: 0.6201 - val_loss: 0.9379 - val_accuracy: 0.6500
Epoch 51/200
218/218 [==============================] - 3s 15ms/step - loss: 0.9380 - ac
curacy: 0.6250 - val_loss: 0.9080 - val_accuracy: 0.6587
Epoch 52/200
218/218 [==============================] - 3s 14ms/step - loss: 0.9305 - ac
curacy: 0.6276 - val_loss: 0.9093 - val_accuracy: 0.6547
Epoch 53/200
218/218 [==============================] - 3s 14ms/step - loss: 0.9114 - ac
curacy: 0.6445 - val_loss: 0.8866 - val_accuracy: 0.6620
Epoch 54/200
218/218 [==============================] - 3s 13ms/step - loss: 0.9074 - ac
```

```
curacy: 0.6409 - val_loss: 0.8938 - val_accuracy: 0.6500
Epoch 55/200
218/218 [==============================] - 3s 14ms/step - loss: 0.9141 - ac
curacy: 0.6368 - val_loss: 0.8879 - val_accuracy: 0.6613
Epoch 56/200
218/218 [==============================] - 3s 13ms/step - loss: 0.8951 - ac
curacy: 0.6438 - val_loss: 0.9048 - val_accuracy: 0.6587
Epoch 57/200
218/218 [==============================] - 3s 14ms/step - loss: 0.8966 - ac
curacy: 0.6372 - val_loss: 0.9343 - val_accuracy: 0.6500
Epoch 58/200
218/218 [==============================] - 3s 14ms/step - loss: 0.8995 - ac
curacy: 0.6366 - val_loss: 0.8736 - val_accuracy: 0.6533
Epoch 59/200
218/218 [==============================] - 3s 14ms/step - loss: 0.8881 - ac
curacy: 0.6441 - val_loss: 0.9022 - val_accuracy: 0.6600
Epoch 60/200
218/218 [==============================] - 3s 14ms/step - loss: 0.8665 - ac
curacy: 0.6579 - val_loss: 0.8582 - val_accuracy: 0.6847
Epoch 61/200
218/218 [==============================] - 3s 14ms/step - loss: 0.8731 - ac
curacy: 0.6540 - val_loss: 0.8959 - val_accuracy: 0.6600
Epoch 62/200
218/218 [==============================] - 3s 13ms/step - loss: 0.8737 - ac
curacy: 0.6506 - val_loss: 0.8673 - val_accuracy: 0.6600
Epoch 63/200
218/218 [==============================] - 3s 14ms/step - loss: 0.8636 - ac
curacy: 0.6562 - val_loss: 0.8622 - val_accuracy: 0.6620
Epoch 64/200
218/218 [==============================] - 3s 13ms/step - loss: 0.8537 - ac
curacy: 0.6667 - val_loss: 0.8862 - val_accuracy: 0.6413
Epoch 65/200
218/218 [==============================] - 3s 13ms/step - loss: 0.8565 - ac
curacy: 0.6631 - val_loss: 0.8682 - val_accuracy: 0.6793
Epoch 66/200
218/218 [==============================] - 3s 14ms/step - loss: 0.8451 - ac
curacy: 0.6639 - val_loss: 0.8812 - val_accuracy: 0.6740
Epoch 67/200
218/218 [==============================] - 3s 14ms/step - loss: 0.8554 - ac
curacy: 0.6736 - val_loss: 0.8109 - val_accuracy: 0.7000
Epoch 68/200
218/218 [==============================] - 3s 14ms/step - loss: 0.8365 - ac
curacy: 0.6698 - val_loss: 0.8635 - val_accuracy: 0.6813
Epoch 69/200
218/218 [==============================] - 3s 13ms/step - loss: 0.8481 - ac
curacy: 0.6674 - val_loss: 0.8491 - val_accuracy: 0.6893
Epoch 70/200
218/218 [==============================] - 3s 13ms/step - loss: 0.8340 - ac
curacy: 0.6744 - val_loss: 0.8419 - val_accuracy: 0.6853
Epoch 71/200
218/218 [==============================] - 3s 14ms/step - loss: 0.8234 - ac
curacy: 0.6705 - val_loss: 0.8427 - val_accuracy: 0.6913
Epoch 72/200
218/218 [==============================] - 3s 13ms/step - loss: 0.8252 - ac
curacy: 0.6815 - val_loss: 0.8506 - val_accuracy: 0.6807
Epoch 73/200
218/218 [==============================] - 3s 13ms/step - loss: 0.8208 - ac
curacy: 0.6750 - val_loss: 0.8613 - val_accuracy: 0.6807
Epoch 74/200
```

```
218/218 [==============================] - 3s 15ms/step - loss: 0.8243 - ac
curacy: 0.6777 - val_loss: 0.8547 - val_accuracy: 0.6880
Epoch 75/200
218/218 [==============================] - 3s 13ms/step - loss: 0.8109 - ac
curacy: 0.6829 - val_loss: 0.8944 - val_accuracy: 0.6713
Epoch 76/200
218/218 [==============================] - 3s 13ms/step - loss: 0.8064 - ac
curacy: 0.6846 - val_loss: 0.8365 - val_accuracy: 0.6853
Epoch 77/200
218/218 [==============================] - 3s 13ms/step - loss: 0.8044 - ac
curacy: 0.6861 - val_loss: 0.8286 - val_accuracy: 0.6887
Epoch 78/200
218/218 [==============================] - 3s 14ms/step - loss: 0.7959 - ac
curacy: 0.6924 - val_loss: 0.8424 - val_accuracy: 0.6953
Epoch 79/200
218/218 [==============================] - 3s 14ms/step - loss: 0.7945 - ac
curacy: 0.6865 - val_loss: 0.8245 - val_accuracy: 0.6880
Epoch 80/200
218/218 [==============================] - 4s 16ms/step - loss: 0.7956 - ac
curacy: 0.6908 - val_loss: 0.8079 - val_accuracy: 0.6900
Epoch 81/200
218/218 [==============================] - 4s 19ms/step - loss: 0.7771 - ac
curacy: 0.6945 - val_loss: 0.8684 - val_accuracy: 0.6933
Epoch 82/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7828 - ac
curacy: 0.6977 - val_loss: 0.8150 - val_accuracy: 0.6933
Epoch 83/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7708 - ac
curacy: 0.7020 - val_loss: 0.8339 - val_accuracy: 0.6787
Epoch 84/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7928 - ac
curacy: 0.6898 - val_loss: 0.7983 - val_accuracy: 0.6987
Epoch 85/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7898 - ac
curacy: 0.6898 - val_loss: 0.8378 - val_accuracy: 0.6860
Epoch 86/200
218/218 [==============================] - 4s 20ms/step - loss: 0.7696 - ac
curacy: 0.7019 - val_loss: 0.8059 - val_accuracy: 0.6960
Epoch 87/200
218/218 [==============================] - 5s 24ms/step - loss: 0.7814 - ac
curacy: 0.6999 - val_loss: 0.8118 - val_accuracy: 0.7027
Epoch 88/200
218/218 [==============================] - 5s 23ms/step - loss: 0.7791 - ac
curacy: 0.7009 - val_loss: 0.8154 - val_accuracy: 0.6973
Epoch 89/200
218/218 [==============================] - 5s 23ms/step - loss: 0.7644 - ac
curacy: 0.7039 - val_loss: 0.8034 - val_accuracy: 0.6993
Epoch 90/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7572 - ac
curacy: 0.7039 - val_loss: 0.8146 - val_accuracy: 0.6900
Epoch 91/200
218/218 [==============================] - 5s 21ms/step - loss: 0.7623 - ac
curacy: 0.7068 - val_loss: 0.8115 - val_accuracy: 0.6987
Epoch 92/200
218/218 [==============================] - 5s 23ms/step - loss: 0.7484 - ac
curacy: 0.7102 - val_loss: 0.8236 - val_accuracy: 0.6880
Epoch 93/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7496 - ac
curacy: 0.7103 - val_loss: 0.8296 - val_accuracy: 0.6887
```

```
Epoch 94/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7564 - ac
curacy: 0.7132 - val_loss: 0.8102 - val_accuracy: 0.6940
Epoch 95/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7544 - ac
curacy: 0.7121 - val_loss: 0.7939 - val_accuracy: 0.7000
Epoch 96/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7502 - ac
curacy: 0.7106 - val_loss: 0.8095 - val_accuracy: 0.7000
Epoch 97/200
218/218 [==============================] - 5s 21ms/step - loss: 0.7566 - ac
curacy: 0.7063 - val_loss: 0.7860 - val_accuracy: 0.6973
Epoch 98/200
218/218 [==============================] - 5s 23ms/step - loss: 0.7453 - ac
curacy: 0.7111 - val_loss: 0.7874 - val_accuracy: 0.7127
Epoch 99/200
218/218 [==============================] - 4s 21ms/step - loss: 0.7433 - ac
curacy: 0.7116 - val_loss: 0.8281 - val_accuracy: 0.6913
Epoch 100/200
218/218 [==============================] - 4s 21ms/step - loss: 0.7355 - ac
curacy: 0.7132 - val_loss: 0.8037 - val_accuracy: 0.7027
Epoch 101/200
218/218 [==============================] - 5s 23ms/step - loss: 0.7364 - ac
curacy: 0.7195 - val_loss: 0.7993 - val_accuracy: 0.7040
Epoch 102/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7388 - ac
curacy: 0.7154 - val_loss: 0.8082 - val_accuracy: 0.7127
Epoch 103/200
218/218 [==============================] - 5s 21ms/step - loss: 0.7334 - ac
curacy: 0.7165 - val_loss: 0.8104 - val_accuracy: 0.7100
Epoch 104/200
218/218 [==============================] - 5s 23ms/step - loss: 0.7168 - ac
curacy: 0.7228 - val_loss: 0.8311 - val_accuracy: 0.6907
Epoch 105/200
218/218 [==============================] - 5s 21ms/step - loss: 0.7298 - ac
curacy: 0.7185 - val_loss: 0.8098 - val_accuracy: 0.6973
Epoch 106/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7224 - ac
curacy: 0.7216 - val_loss: 0.8183 - val_accuracy: 0.7073
Epoch 107/200
218/218 [==============================] - 5s 23ms/step - loss: 0.7218 - ac
curacy: 0.7198 - val_loss: 0.8175 - val_accuracy: 0.6980
Epoch 108/200
218/218 [==============================] - 5s 21ms/step - loss: 0.7197 - ac
curacy: 0.7240 - val_loss: 0.8019 - val_accuracy: 0.7007
Epoch 109/200
218/218 [==============================] - 5s 21ms/step - loss: 0.7263 - ac
curacy: 0.7227 - val_loss: 0.8067 - val_accuracy: 0.6940
Epoch 110/200
218/218 [==============================] - 5s 21ms/step - loss: 0.7129 - ac
curacy: 0.7213 - val_loss: 0.8316 - val_accuracy: 0.6980
Epoch 111/200
218/218 [==============================] - 5s 21ms/step - loss: 0.7169 - ac
curacy: 0.7244 - val_loss: 0.8222 - val_accuracy: 0.7067
Epoch 112/200
218/218 [==============================] - 5s 23ms/step - loss: 0.7152 - ac
curacy: 0.7236 - val_loss: 0.7948 - val_accuracy: 0.7093
Epoch 113/200
218/218 [==============================] - 4s 21ms/step - loss: 0.7264 - ac
```

```
curacy: 0.7175 - val_loss: 0.7847 - val_accuracy: 0.7100
Epoch 114/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7024 - ac
curacy: 0.7292 - val_loss: 0.7891 - val_accuracy: 0.7140
Epoch 115/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7252 - ac
curacy: 0.7249 - val_loss: 0.8450 - val_accuracy: 0.7013
Epoch 116/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7023 - ac
curacy: 0.7251 - val_loss: 0.8033 - val_accuracy: 0.7153
Epoch 117/200
218/218 [==============================] - 5s 23ms/step - loss: 0.7115 - ac
curacy: 0.7287 - val_loss: 0.7836 - val_accuracy: 0.6913
Epoch 118/200
218/218 [==============================] - 5s 23ms/step - loss: 0.7033 - ac
curacy: 0.7326 - val_loss: 0.7737 - val_accuracy: 0.7147
Epoch 119/200
218/218 [==============================] - 5s 22ms/step - loss: 0.7119 - ac
curacy: 0.7226 - val_loss: 0.7818 - val_accuracy: 0.7073
Epoch 120/200
218/218 [==============================] - 5s 23ms/step - loss: 0.7106 - ac
curacy: 0.7276 - val_loss: 0.7806 - val_accuracy: 0.7113
Epoch 121/200
218/218 [==============================] - 5s 23ms/step - loss: 0.7131 - ac
curacy: 0.7249 - val_loss: 0.7736 - val_accuracy: 0.7180
Epoch 122/200
218/218 [==============================] - 6s 25ms/step - loss: 0.6994 - ac
curacy: 0.7310 - val_loss: 0.8013 - val_accuracy: 0.7053
Epoch 123/200
218/218 [==============================] - 5s 24ms/step - loss: 0.7053 - ac
curacy: 0.7277 - val_loss: 0.7816 - val_accuracy: 0.7100
Epoch 124/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6994 - ac
curacy: 0.7338 - val_loss: 0.7855 - val_accuracy: 0.7140
Epoch 125/200
218/218 [==============================] - 5s 25ms/step - loss: 0.6769 - ac
curacy: 0.7398 - val_loss: 0.8210 - val_accuracy: 0.7040
Epoch 126/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6864 - ac
curacy: 0.7336 - val_loss: 0.8061 - val_accuracy: 0.7067
Epoch 127/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6982 - ac
curacy: 0.7300 - val_loss: 0.7828 - val_accuracy: 0.7173
Epoch 128/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6793 - ac
curacy: 0.7409 - val_loss: 0.7991 - val_accuracy: 0.7213
Epoch 129/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6927 - ac
curacy: 0.7361 - val_loss: 0.8099 - val_accuracy: 0.7120
Epoch 130/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6833 - ac
curacy: 0.7356 - val_loss: 0.7737 - val_accuracy: 0.7200
Epoch 131/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6770 - ac
curacy: 0.7445 - val_loss: 0.8112 - val_accuracy: 0.7107
Epoch 132/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6687 - ac
curacy: 0.7470 - val_loss: 0.7771 - val_accuracy: 0.7280
Epoch 133/200
```

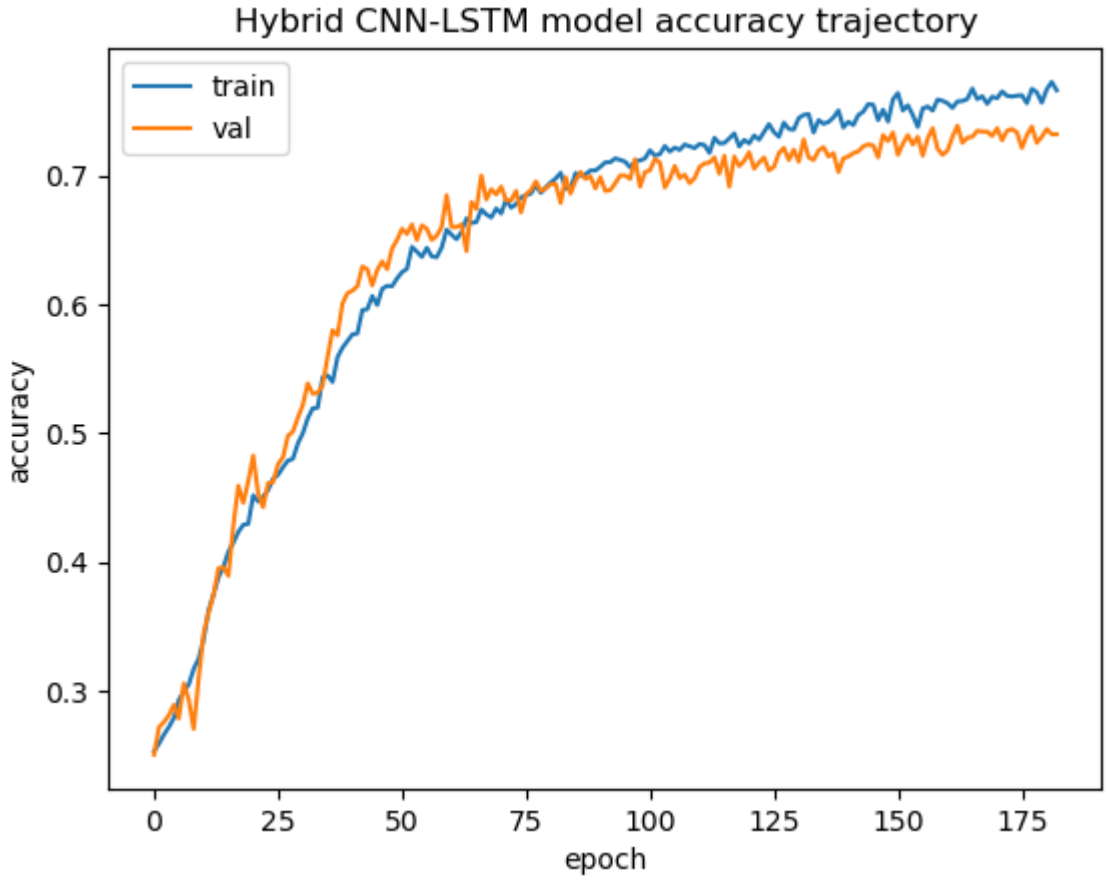
```
218/218 [==============================] - 5s 24ms/step - loss: 0.6677 - ac
curacy: 0.7476 - val_loss: 0.7952 - val_accuracy: 0.7140
Epoch 134/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6915 - ac
curacy: 0.7333 - val_loss: 0.7854 - val_accuracy: 0.7120
Epoch 135/200
218/218 [==============================] - 6s 27ms/step - loss: 0.6659 - ac
curacy: 0.7430 - val_loss: 0.7829 - val_accuracy: 0.7193
Epoch 136/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6687 - ac
curacy: 0.7401 - val_loss: 0.7602 - val_accuracy: 0.7220
Epoch 137/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6726 - ac
curacy: 0.7415 - val_loss: 0.7693 - val_accuracy: 0.7147
Epoch 138/200
218/218 [==============================] - 5s 24ms/step - loss: 0.6691 - ac
curacy: 0.7450 - val_loss: 0.7824 - val_accuracy: 0.7173
Epoch 139/200
218/218 [==============================] - 5s 21ms/step - loss: 0.6573 - ac
curacy: 0.7504 - val_loss: 0.7811 - val_accuracy: 0.7027
Epoch 140/200
218/218 [==============================] - 5s 21ms/step - loss: 0.6666 - ac
curacy: 0.7372 - val_loss: 0.7932 - val_accuracy: 0.7133
Epoch 141/200
218/218 [==============================] - 5s 21ms/step - loss: 0.6715 - ac
curacy: 0.7417 - val_loss: 0.7731 - val_accuracy: 0.7147
Epoch 142/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6819 - ac
curacy: 0.7391 - val_loss: 0.7669 - val_accuracy: 0.7173
Epoch 143/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6508 - ac
curacy: 0.7468 - val_loss: 0.8015 - val_accuracy: 0.7187
Epoch 144/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6586 - ac
curacy: 0.7494 - val_loss: 0.7884 - val_accuracy: 0.7227
Epoch 145/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6355 - ac
curacy: 0.7553 - val_loss: 0.7763 - val_accuracy: 0.7240
Epoch 146/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6454 - ac
curacy: 0.7549 - val_loss: 0.7590 - val_accuracy: 0.7233
Epoch 147/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6600 - ac
curacy: 0.7430 - val_loss: 0.8053 - val_accuracy: 0.7140
Epoch 148/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6545 - ac
curacy: 0.7510 - val_loss: 0.7602 - val_accuracy: 0.7313
Epoch 149/200
218/218 [==============================] - 5s 24ms/step - loss: 0.6615 - ac
curacy: 0.7418 - val_loss: 0.7660 - val_accuracy: 0.7267
Epoch 150/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6388 - ac
curacy: 0.7591 - val_loss: 0.7630 - val_accuracy: 0.7333
Epoch 151/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6239 - ac
curacy: 0.7639 - val_loss: 0.8129 - val_accuracy: 0.7160
Epoch 152/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6503 - ac
curacy: 0.7504 - val_loss: 0.7669 - val_accuracy: 0.7247
```

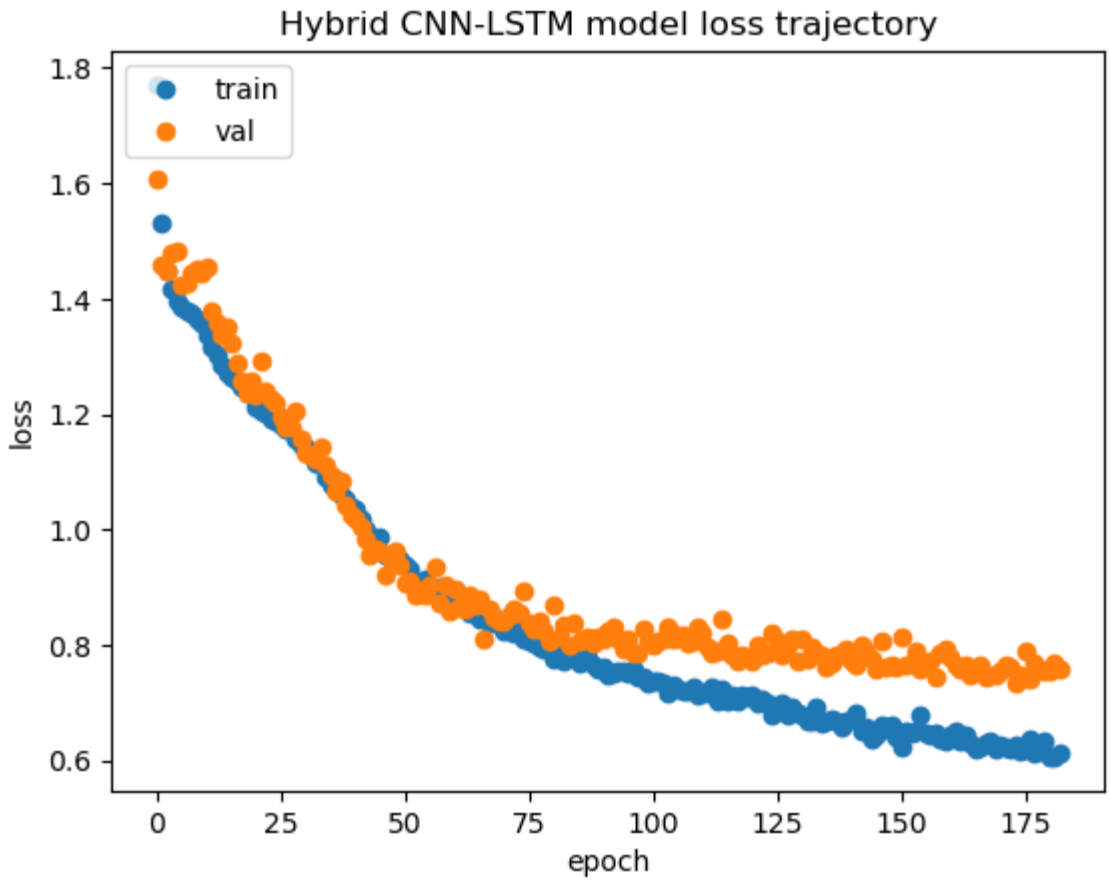
```
Epoch 153/200
218/218 [==============================] - 5s 24ms/step - loss: 0.6462 - ac
curacy: 0.7545 - val_loss: 0.7681 - val_accuracy: 0.7313
Epoch 154/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6525 - ac
curacy: 0.7468 - val_loss: 0.7879 - val_accuracy: 0.7240
Epoch 155/200
218/218 [==============================] - 5s 25ms/step - loss: 0.6795 - ac
curacy: 0.7376 - val_loss: 0.7587 - val_accuracy: 0.7300
Epoch 156/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6477 - ac
curacy: 0.7520 - val_loss: 0.7637 - val_accuracy: 0.7153
Epoch 157/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6443 - ac
curacy: 0.7534 - val_loss: 0.7714 - val_accuracy: 0.7287
Epoch 158/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6465 - ac
curacy: 0.7503 - val_loss: 0.7434 - val_accuracy: 0.7367
Epoch 159/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6361 - ac
curacy: 0.7589 - val_loss: 0.7864 - val_accuracy: 0.7207
Epoch 160/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6345 - ac
curacy: 0.7579 - val_loss: 0.7932 - val_accuracy: 0.7160
Epoch 161/200
218/218 [==============================] - 5s 24ms/step - loss: 0.6427 - ac
curacy: 0.7560 - val_loss: 0.7789 - val_accuracy: 0.7193
Epoch 162/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6506 - ac
curacy: 0.7524 - val_loss: 0.7639 - val_accuracy: 0.7300
Epoch 163/200
218/218 [==============================] - 5s 24ms/step - loss: 0.6319 - ac
curacy: 0.7575 - val_loss: 0.7576 - val_accuracy: 0.7387
Epoch 164/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6426 - ac
curacy: 0.7578 - val_loss: 0.7635 - val_accuracy: 0.7253
Epoch 165/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6299 - ac
curacy: 0.7593 - val_loss: 0.7492 - val_accuracy: 0.7293
Epoch 166/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6185 - ac
curacy: 0.7674 - val_loss: 0.7571 - val_accuracy: 0.7300
Epoch 167/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6233 - ac
curacy: 0.7592 - val_loss: 0.7638 - val_accuracy: 0.7347
Epoch 168/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6295 - ac
curacy: 0.7615 - val_loss: 0.7428 - val_accuracy: 0.7340
Epoch 169/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6346 - ac
curacy: 0.7556 - val_loss: 0.7496 - val_accuracy: 0.7340
Epoch 170/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6191 - ac
curacy: 0.7611 - val_loss: 0.7466 - val_accuracy: 0.7307
Epoch 171/200
218/218 [==============================] - 6s 27ms/step - loss: 0.6276 - ac
curacy: 0.7599 - val_loss: 0.7556 - val_accuracy: 0.7367
Epoch 172/200
218/218 [==============================] - 4s 20ms/step - loss: 0.6225 - ac
```

```
curacy: 0.7652 - val_loss: 0.7655 - val_accuracy: 0.7273
Epoch 173/200
218/218 [==============================] - 5s 21ms/step - loss: 0.6208 - ac
curacy: 0.7618 - val_loss: 0.7604 - val_accuracy: 0.7347
Epoch 174/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6252 - ac
curacy: 0.7612 - val_loss: 0.7347 - val_accuracy: 0.7360
Epoch 175/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6155 - ac
curacy: 0.7619 - val_loss: 0.7463 - val_accuracy: 0.7333
Epoch 176/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6190 - ac
curacy: 0.7622 - val_loss: 0.7883 - val_accuracy: 0.7213
Epoch 177/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6355 - ac
curacy: 0.7560 - val_loss: 0.7409 - val_accuracy: 0.7313
Epoch 178/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6140 - ac
curacy: 0.7672 - val_loss: 0.7677 - val_accuracy: 0.7380
Epoch 179/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6157 - ac
curacy: 0.7645 - val_loss: 0.7584 - val_accuracy: 0.7253
Epoch 180/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6325 - ac
curacy: 0.7565 - val_loss: 0.7561 - val_accuracy: 0.7300
Epoch 181/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6063 - ac
curacy: 0.7668 - val_loss: 0.7545 - val_accuracy: 0.7360
Epoch 182/200
218/218 [==============================] - 5s 23ms/step - loss: 0.6061 - ac
curacy: 0.7726 - val_loss: 0.7674 - val_accuracy: 0.7320
Epoch 183/200
218/218 [==============================] - 5s 22ms/step - loss: 0.6124 - ac
curacy: 0.7661 - val_loss: 0.7575 - val_accuracy: 0.7320
```

(vi)(CNN-LSTM) Visualizing the accuracy and loss trajectory

In [ ]:
```
import matplotlib.pyplot as plt

# Plotting accuracy trajectory
plt.plot(hybrid_cnn_lstm_model_results.history['accuracy'])
plt.plot(hybrid_cnn_lstm_model_results.history['val_accuracy'])
plt.title('Hybrid CNN-LSTM model accuracy trajectory')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

# Plotting loss trajectory
plt.plot(hybrid_cnn_lstm_model_results.history['loss'],'o')
plt.plot(hybrid_cnn_lstm_model_results.history['val_loss'],'o')
plt.title('Hybrid CNN-LSTM model loss trajectory')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```





(vii)(CNN-LSTM) Testing the performance of the hybrid CNN-LSTM model on the held out test set

In [ ]:
```
## Testing the hybrid CNN-LSTM model

hybrid_cnn_lstm_score = hybrid_cnn_lstm_model.evaluate(x_test, y_test, verbo
print('Test accuracy of the hybrid CNN-LSTM-self_attention model:',hybrid_cr
```

```
Test accuracy of the hybrid CNN-LSTM-self_attention model: 0.70428895950317
38
```