```
import zipfile

source_zip1 = '/content/drive/MyDrive/epilepsey/f.zip'
source_zip2='/content/drive/MyDrive/epilepsey/n.zip'
source_zip3='/content/drive/MyDrive/epilepsey/o.zip'
source_zip4='/content/drive/MyDrive/epilepsey/s.zip'
source_zip5='/content/drive/MyDrive/epilepsey/z.zip'
dest='/content/drive/MyDrive/epilepsey/dataset'


with zipfile.ZipFile(source_zip1,'r') as zip:
  zip.extractall(dest)

with zipfile.ZipFile(source_zip2,'r') as zip:
  zip.extractall(dest)

with zipfile.ZipFile(source_zip3,'r') as zip:
  zip.extractall(dest)

with zipfile.ZipFile(source_zip4,'r') as zip:
  zip.extractall(dest)

with zipfile.ZipFile(source_zip5,'r') as zip:
  zip.extractall(dest)


DATA_A='/content/drive/MyDrive/epilepsey/dataset/Z/'
DATA_B='/content/drive/MyDrive/epilepsey/dataset/O/'
DATA_C='/content/drive/MyDrive/epilepsey/dataset/N/'
DATA_D='/content/drive/MyDrive/epilepsey/dataset/F/'
DATA_E='/content/drive/MyDrive/epilepsey/dataset/S/'


!pip install tqdm
import os
from tqdm import tqdm

import pandas as pd
import glob
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import csv

LABEL1 = 0
LABEL2 = 1
LABEL3 = 2

def load():
      datafiles = []
      nFiles=0
      for fn in tqdm(os.listdir(DATA_A)):
              i =np.loadtxt(DATA_A +fn)
              datafiles.append([i,np.array(LABEL1)])
              nFiles+=1

      for fn in tqdm(os.listdir(DATA_B)):
              i =np.loadtxt(DATA_B +fn)
              datafiles.append([i,np.array(LABEL1)])
              nFiles+=1

      for fn in tqdm(os.listdir(DATA_C)):
              i =np.loadtxt(DATA_C +fn)
              datafiles.append([i,np.array(LABEL2)])
              nFiles+=1

      for fn in tqdm(os.listdir(DATA_D)):
              i =np.loadtxt(DATA_D +fn)
              datafiles.append([i,np.array(LABEL2)])
              nFiles+=1

      for fn in tqdm(os.listdir(DATA_E)):
              i =np.loadtxt(DATA_E +fn)
              datafiles.append([i,np.array(LABEL3)])
              nFiles+=1
      return datafiles
```

```
data =load()
print(len(data),"Files")
```

```
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (4.65.0)
100%|██████████| 100/100 [00:04<00:00, 20.60it/s]
100%|██████████| 100/100 [00:04<00:00, 23.59it/s]
100%|██████████| 100/100 [00:03<00:00, 33.22it/s]
100%|██████████| 100/100 [00:03<00:00, 32.26it/s]
100%|██████████| 100/100 [00:03<00:00, 27.42it/s]500 Files
```

```
from sklearn.utils import shuffle
from keras.utils import to_categorical

data = shuffle(data)

n_train =round(len(data)*0.8)
train_data = data[0:n_train]
test_data=data[n_train:]

X_train = np.array([d[0] for d in train_data])
Y_train = np.array([d[1] for d in train_data])

X_test = np.array([d[0] for d in test_data])
Y_test = np.array([d[1] for d in test_data])

X_train.shape

X_train = X_train.reshape(X_train.shape[0], 4097, 1)
Y_train = Y_train.reshape(Y_train.shape[0],1)
Y_train = to_categorical(Y_train, num_classes = 3)

X_test = X_test.reshape(X_test.shape[0], 4097, 1)
Y_test = Y_test.reshape(Y_test.shape[0],1)
Y_test = to_categorical(Y_test, num_classes = 3)
```

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.layers import Embedding
from keras.layers import LSTM

batch_size = 6
nb_epoch = 25
hidden_size = 32
use_dropout=True

model = Sequential()
model.add(LSTM(hidden_size, input_shape=(4097,1)))
#model.add(LSTM(hidden_size))
model.add(Dropout(0.2))
model.add(Dense(3))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['mae', 'acc'])

print(model.summary())

history = model.fit(X_train, Y_train, validation_split=0.2, batch_size=batch_size, epochs=nb_epoch)
score = model.evaluate(X_test, Y_test, batch_size=batch_size)
```

```
=================================================================
Total params: 4,451
Trainable params: 4,451
```

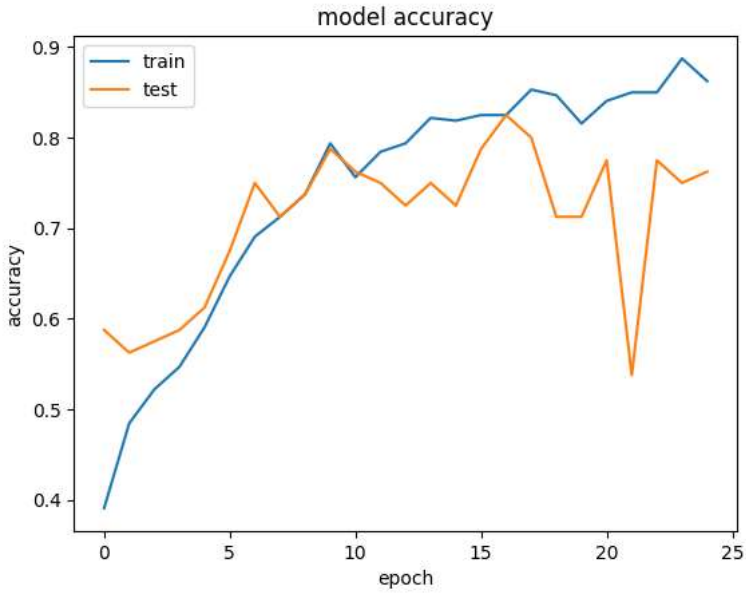
```
Epoch 6/25
54/54 [==============================] - 7s 126ms/step - loss: 0.7905 - mae: 0.3441 - acc: 0.6469 - val_loss: 0.7485 - val_mae: 0.336
Epoch 7/25
54/54 [==============================] - 6s 110ms/step - loss: 0.7506 - mae: 0.3276 - acc: 0.6906 - val_loss: 0.6959 - val_mae: 0.317
Epoch 8/25
54/54 [==============================] - 5s 92ms/step - loss: 0.6957 - mae: 0.3080 - acc: 0.7125 - val_loss: 0.6681 - val_mae: 0.300
Epoch 9/25
54/54 [==============================] - 6s 110ms/step - loss: 0.6633 - mae: 0.2909 - acc: 0.7375 - val_loss: 0.6252 - val_mae: 0.279
Epoch 10/25
54/54 [==============================] - 5s 89ms/step - loss: 0.6031 - mae: 0.2671 - acc: 0.7937 - val_loss: 0.5986 - val_mae: 0.264
Epoch 11/25
54/54 [==============================] - 5s 90ms/step - loss: 0.5846 - mae: 0.2572 - acc: 0.7563 - val_loss: 0.6069 - val_mae: 0.258
Epoch 12/25
54/54 [==============================] - 6s 110ms/step - loss: 0.5452 - mae: 0.2406 - acc: 0.7844 - val_loss: 0.5700 - val_mae: 0.236
Epoch 13/25
54/54 [==============================] - 5s 91ms/step - loss: 0.5252 - mae: 0.2220 - acc: 0.7937 - val_loss: 0.6740 - val_mae: 0.2504
Epoch 14/25
54/54 [==============================] - 8s 144ms/step - loss: 0.4809 - mae: 0.2054 - acc: 0.8219 - val_loss: 0.6379 - val_mae: 0.238
Epoch 15/25
54/54 [==============================] - 5s 91ms/step - loss: 0.4938 - mae: 0.2026 - acc: 0.8188 - val_loss: 0.6773 - val_mae: 0.2376
Epoch 16/25
54/54 [==============================] - 5s 99ms/step - loss: 0.4682 - mae: 0.1883 - acc: 0.8250 - val_loss: 0.5439 - val_mae: 0.2160
Epoch 17/25
54/54 [==============================] - 5s 100ms/step - loss: 0.4700 - mae: 0.1925 - acc: 0.8250 - val_loss: 0.4757 - val_mae: 0.19
Epoch 18/25
54/54 [==============================] - 5s 89ms/step - loss: 0.4285 - mae: 0.1762 - acc: 0.8531 - val_loss: 0.5084 - val_mae: 0.1907
Epoch 19/25
54/54 [==============================] - 7s 133ms/step - loss: 0.4250 - mae: 0.1729 - acc: 0.8469 - val_loss: 0.7413 - val_mae: 0.23
Epoch 20/25
54/54 [==============================] - 5s 92ms/step - loss: 0.5023 - mae: 0.1826 - acc: 0.8156 - val_loss: 0.7281 - val_mae: 0.2288
Epoch 21/25
54/54 [==============================] - 6s 104ms/step - loss: 0.4532 - mae: 0.1771 - acc: 0.8406 - val_loss: 0.5955 - val_mae: 0.20
Epoch 22/25
54/54 [==============================] - 5s 99ms/step - loss: 0.4065 - mae: 0.1599 - acc: 0.8500 - val_loss: 1.1161 - val_mae: 0.317
Epoch 23/25
54/54 [==============================] - 5s 91ms/step - loss: 0.4491 - mae: 0.1697 - acc: 0.8500 - val_loss: 0.5046 - val_mae: 0.1836
Epoch 24/25
54/54 [==============================] - 6s 113ms/step - loss: 0.3512 - mae: 0.1458 - acc: 0.8875 - val_loss: 0.4922 - val_mae: 0.184
Epoch 25/25
54/54 [==============================] - 5s 90ms/step - loss: 0.3695 - mae: 0.1533 - acc: 0.8625 - val_loss: 0.5781 - val_mae: 0.1856
17/17 [==============================] - 1s 57ms/step - loss: 0.4831 - mae: 0.1393 - acc: 0.8600
```

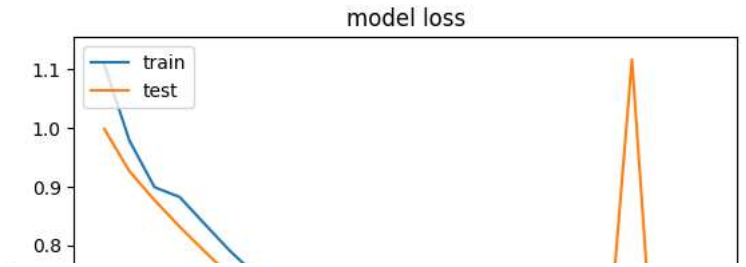
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
print(history.history.keys())

plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend ([ 'train', 'test'], loc= 'upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend ([ 'train', 'test'], loc= 'upper left')
plt.show()
```

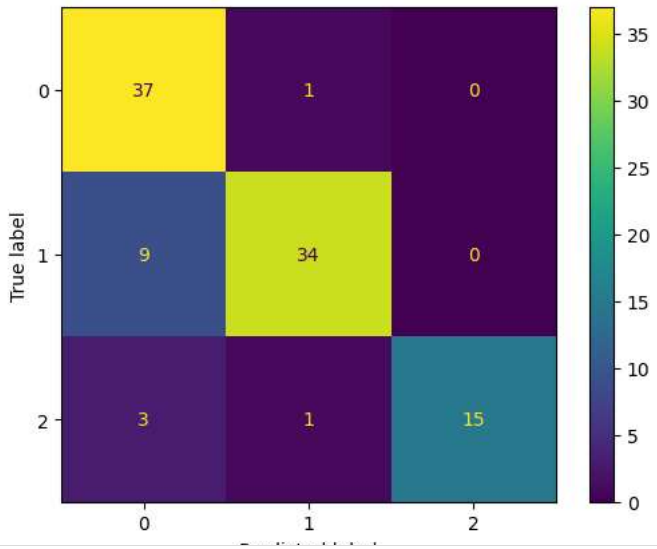
```
dict_keys(['loss', 'mae', 'acc', 'val_loss', 'val_mae', 'val_acc'])
```






```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report, accuracy_score, f1_score

Y_pred=model.predict(X_test)
Y_pred= np.round(Y_pred)

#print(Y_pred)

cm = confusion_matrix(Y_test.argmax(axis=1),Y_pred.argmax(axis=1))
print(cm)
print(classification_report (Y_test, Y_pred))
print(round((accuracy_score (Y_test, Y_pred)*100),2))
print(round (f1_score (Y_test, Y_pred, average='weighted'), 3))
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

```
4/4 [==============================] - 1s 57ms/step
[[37  1  0]
 [ 9 34  0]
 [ 3  1 15]]
              precision    recall  f1-score   support

           0       0.76      0.97      0.85        38
           1       0.94      0.79      0.86        43
           2       1.00      0.79      0.88        19

   micro avg       0.86      0.86      0.86       100
   macro avg       0.90      0.85      0.86       100
weighted avg       0.88      0.86      0.86       100
 samples avg       0.86      0.86      0.86       100

86.0
0.861
```



ancel contracts here

✓ 0s completed at 11:33 AM