```
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 [00:01<00:00, 59.22it/s]
     100%|██████████| 100/100 [00:01<00:00, 74.28it/s]
     100%|██████████| 100/100 [00:02<00:00, 46.91it/s]
     100%|██████████| 100/100 [00:02<00:00, 46.97it/s]
     100%|██████████| 100/100 [00:01<00:00, 65.03it/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.backend import flatten
```

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import StratifiedShuffleSplit
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten, Convolution1D, Dropout, MaxPooling1D,GlobalAveragePooling1D
from keras.optimizers import SGD
from keras.utils import np_utils
import keras
```

```
model = Sequential()
model.add(Convolution1D(64, 10, strides=2, padding='same', activation='relu',  input_shape=(4097, 1)))
model.add(Dropout(0.2))
model.add(MaxPooling1D(3))
model.add(Convolution1D(32, 5, strides=2, padding='same', activation='relu'))
model.add(Dropout(0.2))
model.add(MaxPooling1D(3))
model.add(Convolution1D(16, 4, strides=1, padding='same', activation='relu'))
model.add(Dropout(0.2))
model.add(MaxPooling1D(3))
model.add(GlobalAveragePooling1D())
model.add(Dense(50, activation='relu'))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
```

```
batch_size = 4
n_epoch = 20
hidden_size = 64
use_dropout=True
```

```
use_dropout=True

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=n_epoch)
score = model.evaluate(X_test, Y_test, batch_size=batch_size)
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_18 (Conv1D)          (None, 2049, 64)          704

 dropout_24 (Dropout)        (None, 2049, 64)          0

 max_pooling1d_18 (MaxPoolin  (None, 683, 64)          0
 g1D)

 conv1d_19 (Conv1D)          (None, 342, 32)           10272

 dropout_25 (Dropout)        (None, 342, 32)           0

 max_pooling1d_19 (MaxPoolin  (None, 114, 32)          0
 g1D)

 conv1d_20 (Conv1D)          (None, 114, 16)           2064

 dropout_26 (Dropout)        (None, 114, 16)           0

 max_pooling1d_20 (MaxPoolin  (None, 38, 16)           0
 g1D)

 global_average_pooling1d_6   (None, 16)               0
 (GlobalAveragePooling1D)

 dense_12 (Dense)            (None, 50)                850

 dropout_27 (Dropout)        (None, 50)                0

 flatten_6 (Flatten)         (None, 50)                0

 dense_13 (Dense)            (None, 3)                 153

=================================================================
Total params: 14,043
Trainable params: 14,043
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/20
80/80 [==============================] - 3s 9ms/step - loss: 2.8653 - mae: 0.3689 - acc: 0.4906 - val_loss: 1.1977 - val_mae: 0.3979
Epoch 2/20
80/80 [==============================] - 0s 6ms/step - loss: 1.0148 - mae: 0.3400 - acc: 0.5375 - val_loss: 0.9474 - val_mae: 0.3609
Epoch 3/20
80/80 [==============================] - 0s 6ms/step - loss: 0.7363 - mae: 0.2601 - acc: 0.7344 - val_loss: 0.6986 - val_mae: 0.2428
Epoch 4/20
80/80 [==============================] - 0s 5ms/step - loss: 0.7071 - mae: 0.2546 - acc: 0.7437 - val_loss: 0.5445 - val_mae: 0.2340
Epoch 5/20
80/80 [==============================] - 0s 6ms/step - loss: 0.6074 - mae: 0.1937 - acc: 0.8406 - val_loss: 0.4057 - val_mae: 0.1840
Epoch 6/20
80/80 [==============================] - 0s 5ms/step - loss: 0.6308 - mae: 0.1918 - acc: 0.8219 - val_loss: 0.5004 - val_mae: 0.2388
Epoch 7/20
80/80 [==============================] - 0s 6ms/step - loss: 0.5155 - mae: 0.1695 - acc: 0.8813 - val_loss: 0.4180 - val_mae: 0.1517
Epoch 8/20
```

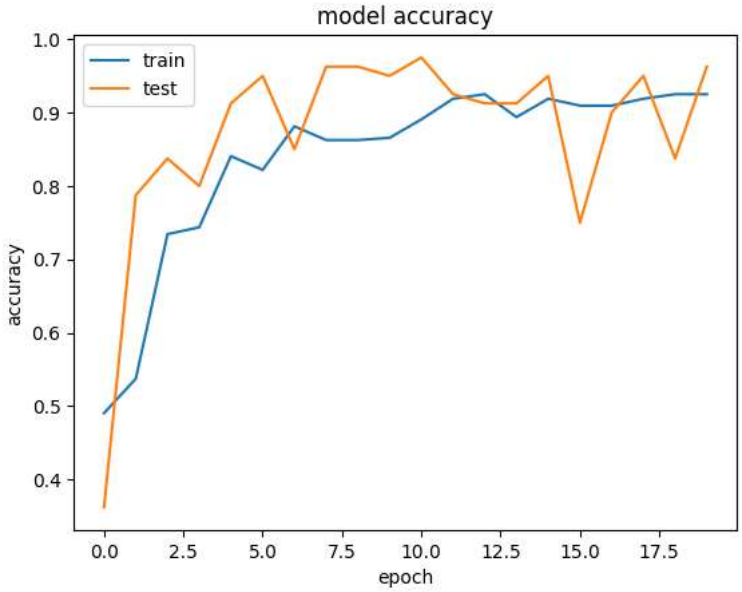
```
from google.colab import drive
drive.mount('/content/drive')
```

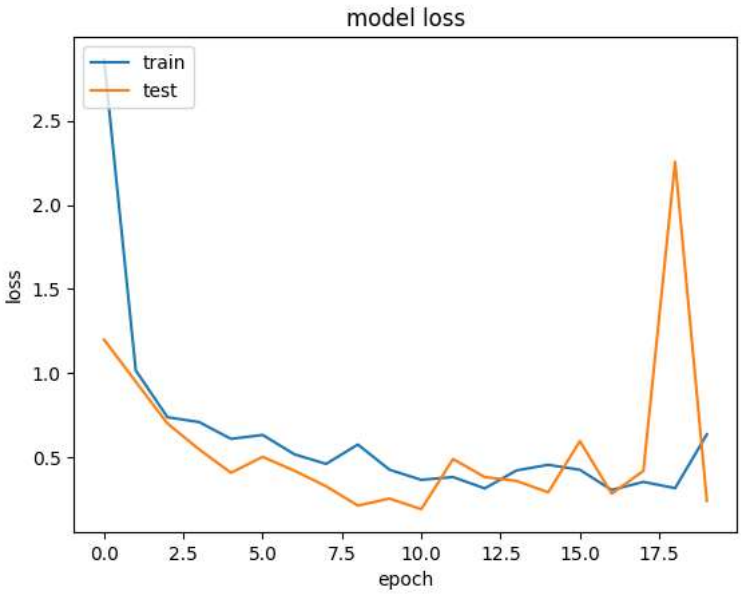
```
Mounted at /content/drive
```

```
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()
```

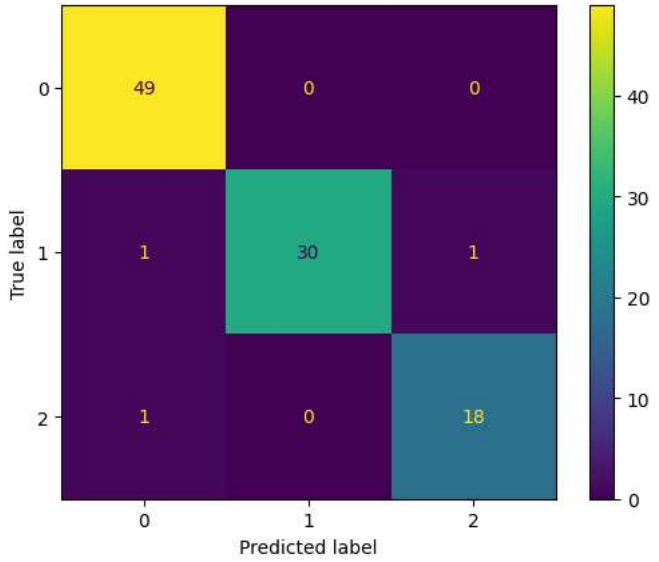
```
dict_keys(['loss', 'mae', 'acc', 'val_loss', 'val_mae', 'val_acc'])
```




```
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 [==============================] - 0s 5ms/step
[[49  0  0]
 [ 1 30  1]
 [ 1  0 18]]
              precision    recall  f1-score   support

           0       0.96      0.98      0.97        49
           1       1.00      0.94      0.97        32
           2       0.95      0.95      0.95        19

   micro avg       0.97      0.96      0.96       100
   macro avg       0.97      0.95      0.96       100
weighted avg       0.97      0.96      0.96       100
 samples avg       0.96      0.96      0.96       100

96.0
0.965
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
```




✓ 0s completed at 10:21 AM