PROJECT DEVELOPMENT PHASE

SPRINT- II

Team ID	PNT2022TMID18693
Project Name	Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation

ECG arrhythmia classification using CNN

Importing Neccessary Libraries

In [1]:
```
import numpy as np#used for numerical analysis
import tensorflow #open source used for both ML and DL for computation
from tensorflow.keras.models import Sequential #it is a plain stack of layers
from tensorflow.keras import layers #A layer consists of a tensor-in tensor-out computation function
#Dense layer is the regular deeply connected neural network layer
from tensorflow.keras.layers import Dense,Flatten
#Faltten-used fot flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D,MaxPooling2D #Convolutional layer
#MaxPooling2D-for downsampling the image
from keras.preprocessing.image import ImageDataGenerator
```

Image Data Agumentation

In [2]:
```
#setting parameter for Image Data agumentation to the traing data
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

In [3]:
```
#Image Data agumentation to the testing data
test_datagen=ImageDataGenerator(rescale=1./255)
```

Loading our data and performing data agumentation

In [4]:
```
#performing data agumentation to train data
x_train=train_datagen.flow_from_directory(directory=r'D:\ECG Image Analysis for Arrhythmia\Dataset\train'
                                          ,target_size=(64,64),batch_size=32,class_mode='categorical')
#performing data agumentation to test data
x_test=test_datagen.flow_from_directory(directory=r'D:\ECG Image Analysis for Arrhythmia\Dataset\test'
                                        ,target_size=(64,64),batch_size=32,class_mode='categorical')
```

```
Found 15341 images belonging to 6 classes.
Found 6825 images belonging to 6 classes.
```

In [5]:
```
print(x_train.class_indices)#checking the number of classes
```

```
{'Left Bundle Branch Block': 0, 'Normal': 1, 'Premature Atrial Contraction': 2, 'Premature Ventricular Contractions': 3, 'Right Bundle Branch Block': 4, 'Ventricular Fibrillation': 5}
```

In [6]:
```
from collections import Counter as c
c(x_train.labels)
```

Out[6]:
```
Counter({0: 504, 1: 7346, 2: 2054, 3: 2759, 4: 2239, 5: 439})
```

Creating the model

In [7]:
```
# create model
model=Sequential()
# adding model layer
model.add(Conv2D(32,(3,3),input_shape=(64,64,3),activation='relu'))#convolutional layer
model.add(MaxPooling2D(pool_size=(2,2))) #MaxPooling2D-for downsampling the input

model.add(Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())#flatten the dimension of the image
model.add(Dense(32))#deeply connected neural network layers.
model.add(Dense(6,activation='softmax'))#output layer with 6 neurons
```

In [8]:
```
model.summary()#summary of our model
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 32)                200736

 dense_1 (Dense)             (None, 6)                 198

=================================================================
Total params: 211,078
Trainable params: 211,078
Non-trainable params: 0
_________________________________________________________________
```

Compiling the model

In [9]:
```
# Compile model
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

Fitting the model

In [10]:
```
# Fit the model
model.fit_generator(generator=x_train,steps_per_epoch = len(x_train),
                    epochs=10, validation_data=x_test,validation_steps = len(x_test))
```

```
C:\Users\akshya\AppData\Local\Temp\ipykernel_26104\1433457599.py:2: UserWarning: `Model.fit_generator` is deprecated and will b
e removed in a future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(generator=x_train,steps_per_epoch = len(x_train),
```

```
Epoch 1/10
480/480 [==============================] - 156s 325ms/step - loss: 0.7763 - accuracy: 0.7349 - val_loss: 0.5478 - val_accuracy:
0.8207
Epoch 2/10
480/480 [==============================] - 39s 81ms/step - loss: 0.3030 - accuracy: 0.9110 - val_loss: 0.5442 - val_accuracy:
0.8488
Epoch 3/10
480/480 [==============================] - 39s 81ms/step - loss: 0.2397 - accuracy: 0.9295 - val_loss: 0.3874 - val_accuracy:
0.8818
Epoch 4/10
480/480 [==============================] - 38s 79ms/step - loss: 0.2102 - accuracy: 0.9357 - val_loss: 0.3702 - val_accuracy:
0.8738
Epoch 5/10
480/480 [==============================] - 38s 80ms/step - loss: 0.1824 - accuracy: 0.9465 - val_loss: 0.3285 - val_accuracy:
0.9127
```

```
Epoch 6/10
480/480 [==============================] - 38s 79ms/step - loss: 0.1705 - accuracy: 0.9475 - val_loss: 0.2943 - val_accuracy:
0.9089
Epoch 7/10
480/480 [==============================] - 39s 80ms/step - loss: 0.1585 - accuracy: 0.9529 - val_loss: 0.2854 - val_accuracy:
0.9111
Epoch 8/10
480/480 [==============================] - 39s 82ms/step - loss: 0.1427 - accuracy: 0.9570 - val_loss: 0.3429 - val_accuracy:
0.9068
Epoch 9/10
480/480 [==============================] - 38s 79ms/step - loss: 0.1345 - accuracy: 0.9597 - val_loss: 0.3325 - val_accuracy:
0.9053
Epoch 10/10
480/480 [==============================] - 38s 79ms/step - loss: 0.1304 - accuracy: 0.9591 - val_loss: 0.3313 - val_accuracy:
0.9103
```

Out[10]:
```
<keras.callbacks.History at 0x258daeb4040>
```

In [11]:
```
#model.fit_generator(x_train,epochs=10,validation_data=x_test)
```

Saving our model

In [12]:
```
# Save the model
model.save('ECG.h5')
```

Predicting our results

In [13]:
```
from tensorflow.keras.utils import load_img,img_to_array
from tensorflow.keras.models import load_model
from keras.preprocessing import image
model = load_model("ECG.h5") #Loading the model for testing
```

In [14]:
```
img = load_img(r"D:\ECG Image Analysis for Arrhythmia\Dataset\test\Left Bundle Branch Block\fig_5934.png",target_size= (64,64))#l
x = img_to_array(img)#image to array
x = np.expand_dims(x,axis = 0)#changing the shape
#pred = model.predict_classes(x)#predicting the classes
#pred
preds=model.predict(x)
pred=np.argmax(preds,axis=1)
preds
```

```
1/1 [==============================] - 0s 98ms/step
```

Out[14]:
```
array([[1., 0., 0., 0., 0., 0.]], dtype=float32)
```

In [15]:
```
index=['Left Bundle Branch Block','Normal','Premature Atrial Contraction',
       'Premature Ventricular Contractions', 'Right Bundle Branch Block','Ventricular Fibrillation']
result=str(index[pred[0]])
result
```

Out[15]:
```
'Left Bundle Branch Block'
```