Classification Of Arrhythmia By Using Deep Learning With 2-D ECG Spectral Image Representation

MODEL BUILDING

PREDICTIONS

Team ID	PNT2022TMID31222
Project Name	Classification Of Arrhythmia By Using Deep Learning
	With 2-D ECG Spectral Image Representation

PREDICTIONS:

The last and final step is to save the model to do predictions. For that we have a class in a keras called Load_model is used to load our saved model h5 file (alert.h5).

IMPORT LIBRARIES:

11/7/22, 12:35 AM

Untitled8.ipynb - Colaboratory

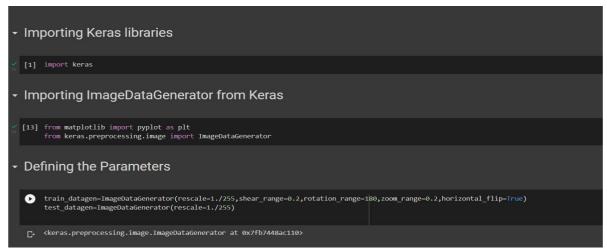
Importing Keras libraries

import keras

Importing ImageDataGenerator from Keras

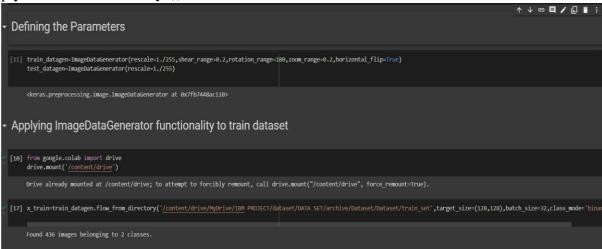
 $from \ keras.preprocessing.image \ import \ Image Data Generator$

IMPORT ImageDataGenerator FROM KERAS:



APPLYING ImageDataGenerator to train dataset:

plyflow_from_directory ()methodfor Train folder.



APPLYING ImageDataGenerator to test dataset:

Applying the **flow_from_directory** () methodfortest folder.



IMPORTING MODEL BUILDING LIBRARIES:

11/8/22, 1:16 AM

Main code - Colaboratory

Importing Model Building Libraries

```
#to define the linear Initialisation import sequential
from keras.models import Sequential
#to add layers import Dense
from keras.layers import Dense
#to create Convolutional kernel import convolution2D
from keras.layers import Convolution2D
#import Maxpooling layer
from keras.layers import MaxPooling2D
#import flatten layer
from keras.layers import Flatten
import warnings
warnings.filterwarnings('ignore')
```

INITIALIZING THE MODEL:

Initializing the model

```
model=Sequential()
```

ADDING CNN LAYERS:

Adding CNN Layers

```
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
#add maxpooling layers
model.add(MaxPooling2D(pool_size=(2,2)))
#add faltten layer
model.add(Flatten())
```

ADDING DENSE LAYERS:

Add Dense layers

```
#add hidden layers
model.add(Dense(150,activation='relu'))
#add output layer
model.add(Dense(1,activation='sigmoid'))
```

CONFIGURING THE LEARNING PROCESS:

configuring the learning process

```
model.compile(loss='binary_crossentropy',optimizer="adam",metrics=["accuracy"])
```

TRAINING THE MODEL:

Training the model

```
model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_data=x_test,validation
  Epoch 1/10
  Epoch 2/10
  Epoch 3/10
  Epoch 4/10
  Epoch 5/10
  Epoch 6/10
  14/14 [============= ] - 30s 2s/step - loss: 0.1971 - accuracy: 0.926
  Epoch 7/10
  Epoch 8/10
  Epoch 9/10
  14/14 [============= ] - 31s 2s/step - loss: 0.2306 - accuracy: 0.896
  Epoch 10/10
  14/14 [==============] - 27s 2s/step - loss: 0.2593 - accuracy: 0.889
  <keras.callbacks.History at 0x7fd537101390>
```