

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

MODEL BUILDING

TRAINING THE MODEL

Team ID	PNT2022TMID16055
Project Name	Classification Of Arrhythmia By Using Deep Learning With 2-D ECG Spectral Image Representation

TRAINING THE MODEL:

At this point, we have training data and a fully configured neural network to train. All that is left is to pass the data to the model for the training process to commence, a process that is completed by iterating on the training data. Training begins by calling the fit () method.

The arguments are the batch size as you are using “adam” (bath gradient descent and epochs: no: of times the model should get trained).

steps_per_epoch:

- It specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of steps_per_epoch as the total number of samples in your training folder divided by the batch size.
- Epochs:an integer and number of epochs we want to train our model for.
- Validation_data can be either input and targets list generator inputs, targets, and sample_weights list which can be used to evaluate.

The loss and metrics for any model after any epoch has ended.

- Validation_steps:

Only if the validation_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.

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 ImageDataGenerator
```

IMPORT ImageDataGenerator FROM KERAS:

```
▼ Importing Keras libraries

[1] import keras

▼ Importing ImageDataGenerator from Keras

[13] from matplotlib import pyplot as plt
      from keras.preprocessing.image import ImageDataGenerator

▼ Defining the Parameters

▶ train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,rotation_range=180,zoom_range=0.2,horizontal_flip=True)
  test_datagen=ImageDataGenerator(rescale=1./255)

<keras.preprocessing.image.ImageDataGenerator at 0x7fb7448ac110>
```

APPLYING ImageDataGenerator to train dataset:

`flow_from_directory ()` method for Train folder.

```
▼ Defining the Parameters

[11] train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,rotation_range=180,zoom_range=0.2,horizontal_flip=True)
     test_datagen=ImageDataGenerator(rescale=1./255)

<keras.preprocessing.image.ImageDataGenerator at 0x7fb7448ac110>

▼ Applying ImageDataGenerator functionality to train dataset

[10] from google.colab import drive
     drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

[17] x_train=train_datagen.flow_from_directory('/content/drive/MyDrive/IBM PROJECT/dataset/DATA SET/archive/Dataset/Dataset/train_set',target_size=(128,128),batch_size=32,class_mode='binary')

Found 436 images belonging to 2 classes.
```

APPLYING ImageDataGenerator to test dataset:

Applying the `flow_from_directory ()` method for test folder.

```
▼ Applying ImageDataGenerator functionality to test dataset

▶ x_test=test_datagen.flow_from_directory('/content/drive/MyDrive/IBM PROJECT/dataset/DATA SET/archive/Dataset/Dataset/test_set',target_size=(128,128),batch_size=32,class_mode='binary')

Found 121 images belonging to 2 classes.
```

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_steps=14)
```

```
Epoch 1/10
14/14 [=====] - 322s 19s/step - loss: 1.5998 - accuracy: 0.71
Epoch 2/10
14/14 [=====] - 26s 2s/step - loss: 0.3427 - accuracy: 0.8625
Epoch 3/10
14/14 [=====] - 32s 2s/step - loss: 0.2979 - accuracy: 0.8857
Epoch 4/10
14/14 [=====] - 29s 2s/step - loss: 0.2585 - accuracy: 0.8929
Epoch 5/10
14/14 [=====] - 29s 2s/step - loss: 0.1926 - accuracy: 0.9243
Epoch 6/10
14/14 [=====] - 30s 2s/step - loss: 0.1971 - accuracy: 0.9264
Epoch 7/10
14/14 [=====] - 32s 2s/step - loss: 0.1781 - accuracy: 0.9281
Epoch 8/10
14/14 [=====] - 30s 2s/step - loss: 0.1796 - accuracy: 0.9243
Epoch 9/10
14/14 [=====] - 31s 2s/step - loss: 0.2306 - accuracy: 0.8964
Epoch 10/10
14/14 [=====] - 27s 2s/step - loss: 0.2593 - accuracy: 0.8857
<keras.callbacks.History at 0x7fd537101390>
```

