MODEL BUILDING

TEAM ID: PNT2022TMID16902

Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation

Task

1. Model Building

We are ready with the augmented and pre-processed image data,we will begin our build our model by following the below steps:

Import The Libraries:

```
import numpy as np
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense,Flatten
from tensorflow.keras.layers import Conv2D,MaxPooling2D
```

Initializing The Model:

Keras has 2 ways to define a neural network:

- Sequential
- Function API

In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add () method. Now, will initialize our model

The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model.

Adding CNN Layer:

We are adding a convolution layer with an activation function as “relu” and with a small filter size (3,3) and a number of filters as (32) followed by a max-pooling layer.

The Max pool layer is used to downsample the input. The flatten layer flattens the input.

```
model=Sequential()
model.add(Conv2D(32,(3,3),input_shape=(64,64,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
```

Adding Dense & Output Layer:

Dense layer is deeply connected neural network layer. It is most common and frequently used layer.

```
model.add(Dense(32))
model.add(Dense(units=128,kernel_initializer="random_uniform"))
```

Adding Output Layer:

Understanding the model is very important phase to properly use it for training and prediction purposes . keras provides a simple method, summary to get the full information about the model and its layers

```
model.summary()
```

```
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:08:36) [MSC v.1933 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Python/train I test/train.py ==========
Found 15341 images belonging to 6 classes.
Found 6825 images belonging to 6 classes.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

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

 flatten (Flatten)           (None, 30752)             0

 dense (Dense)               (None, 32)                984096

 dense_1 (Dense)             (None, 128)               4224

 dense_2 (Dense)             (None, 6)                 774

=================================================================
Total params: 989,990
Trainable params: 989,990
Non-trainable params: 0
```

Configure The Learning Process:

The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find error or deviation in the learning process. Keras requires loss function during the model compilation proceses

Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer

Metrics is used to evaluate the performance of your model. It is similar to loss function, but not used in the training process.

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

Train The Model:

We will train our model with our image dataset. fit generator functions used to train a deep learning neural network.

```
model.fit(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
```

```
Epoch 1/10
  1/480 [..............................] - ETA: 45:06 - loss: 1.8828 - accuracy: 0.1562
  2/480 [..............................] - ETA: 14:08 - loss: 5.2590 - accuracy: 0.3438
  3/480 [..............................] - ETA: 16:11 - loss: 6.3511 - accuracy: 0.3854
  4/480 [..............................] - ETA: 15:41 - loss: 6.1634 - accuracy: 0.3359
  5/480 [..............................] - ETA: 15:30 - loss: 5.4935 - accuracy: 0.2937
  6/480 [..............................] - ETA: 15:44 - loss: 4.8936 - accuracy: 0.2708
  7/480 [..............................] - ETA: 15:51 - loss: 4.6100 - accuracy: 0.2634
  8/480 [..............................] - ETA: 16:13 - loss: 4.4702 - accurac
y: 0.2539
  9/480 [..............................] - ETA: 16:44 - loss: 4.11
28 - accuracy: 0.2986
 10/480 [..............................] - ETA: 16:38
 - loss: 3.9533 - accuracy: 0.3063
 11/480 [..............................] -
ETA: 16:41 - loss: 3.8012 - accuracy: 0.3182
 12/480 [..............................
.......] - ETA: 16:44 - loss: 3.6622 - accuracy: 0.3229
 13/480 [..........
...................] - ETA: 16:41 - loss: 3.5167 - accuracy: 0.3365
 14/480
```

Save The Model:

The model is saved with .h5 extension as follows.

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

```
model.save('ECG.h5')
```

Test The Model:

Load necessary libraries and load the saved model using load model Taking an image as input and checking the results

The target size should for the image that is should be the same as the target size that you have used for training

```
Efifi• EcFa Format Run Opt@ns Win Help
  rom teosorflow.teras.sedels import load_sedel
from teras.preprocessiog import image
sedel=load sedel(-ECG.R5-)
img=image.load_img(-D:/PyrRoo/traio I test/data/test/Normal/fig_2l14.pog-,target_siee=(64,64))
s=image.img_to_array(img)
s=op.espaod_dims(a,asis=0)
pred=seodel.predicr_classes(
a) pred
iodes=[-Left Buodle BraocR Bloct-,-Normal-,-Premature Atrial Cootracrioo-,-Premature Ventricular Cootracrioos-,-RigRt Buodle BraocR Bloct-,-Ventricular Fibrillatioo-]
result=str(iodea[pred[0]])
result
```