Sprint-2 Model Buliding

Date	01 Nov 2022
Team ID	PNT2022TMID25331
Project Name	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:

```
from tensorflow.keras.layers import Sequential
from tensorflow.keras.layers import convolution2D
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
```

Initializing The Model:

Keras has 2 ways to define a neural network:

- Sequential
- Function API

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

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

Adding CNN Layer:

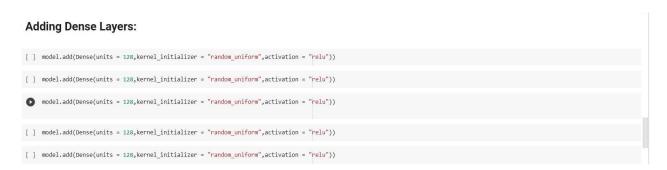
We are adding a convolution layer with an activation function as "relu" and with a small filter size (3,3) and a number of filters as (32) followed by a max-pooling layer.

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



Adding Dense Layer:

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

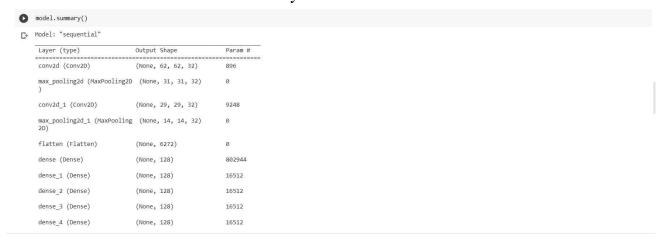


Adding Output Layer:

Adding Output Layer:

```
[ ] model.add(Dense(units = 6,kernel_initializer = "random_uniform",activation = "softmax"))
```

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



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 processes

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.

- Train the model:

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.

Save the model:

```
[ ] #Saving Model.
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

Testing the model:

