Sprint-2
Model Building

Tasks:

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.models import Sequential
from tensorflow.keras.layers import Dense
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

```
model=Sequential()
```

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

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

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

model.add(Flatten())

model.add(Dense(32))
model.add(Dense(6, 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

```
[ ] model.summary()
```

```
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
_________________________________________________________________
```

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'])
```

1. 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=x_train,  epochs=5, validation_data=x_test)

     Epoch 1/5
     480/480 [==============================] - 11057s 23s/step - loss: 0.6662 - accuracy: 0.7746 - val_loss: 0.4629 - val_accuracy: 0.8517
     Epoch 2/5
     480/480 [==============================] - 117s 244ms/step - loss: 0.2755 - accuracy: 0.9192 - val_loss: 0.4083 - val_accuracy: 0.8614
     Epoch 3/5
     480/480 [==============================] - 114s 237ms/step - loss: 0.2328 - accuracy: 0.9316 - val_loss: 0.3736 - val_accuracy: 0.8816
     Epoch 4/5
     480/480 [==============================] - 118s 245ms/step - loss: 0.2038 - accuracy: 0.9402 - val_loss: 0.3344 - val_accuracy: 0.8900
     Epoch 5/5
     480/480 [==============================] - 119s 247ms/step - loss: 0.1790 - accuracy: 0.9460 - val_loss: 0.3636 - val_accuracy: 0.8882
     <keras.callbacks.History at 0x7f8dd28a5c50>
```

2. 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('/content/drive/MyDrive/IBM project/ECG.h5')
```

3. 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.

- Importing necessary libraries for testing

```
from tensorflow.keras.models import load_model
from keras.utils import load_img
from keras.utils import img_to_array
import numpy as np
```

- Loading the saved model:

```
[ ]  model=load_model('/content/drive/MyDrive/IBM project/ECG.h5')
```

- Predicting the class:

```
img=load_img('/content/drive/MyDrive/IBM project/dataset/test/Left Bundle Branch Block/fig_5906.png', target_size=(64,64))
x=img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=model.predict(x)
pred_classes=['Left bundle branch block', 'Normal', 'Premature atrial contraction', 'Premature ventricular contraction','Right bundle branch block','Ventricular fibrillation']
classes_x=np.argmax(pred)
print(pred)
print(pred_classes[classes_x])
```

```
1/1 [==============================] - 0s 310ms/step
[[1. 0. 0. 0. 0. 0.]]
Left bundle branch block
```