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

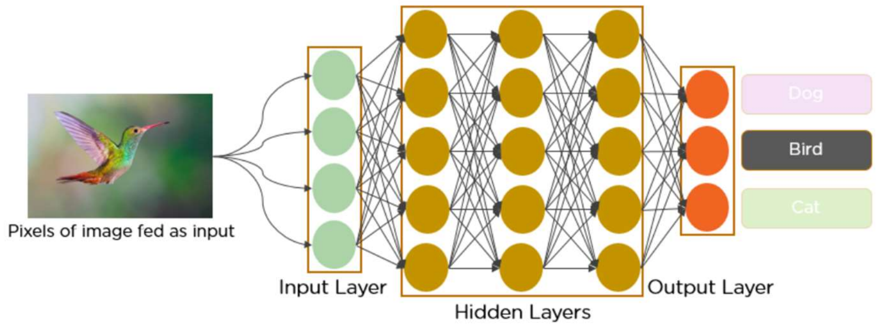
1. INTRODUCTION

1.1 Overview

According to the World Health Organization (WHO), cardiovascular diseases (CVDs) are the number one cause of death today. Over 17.7 million people died from CVDs in the year 2017 all over the world which is about 31% of all deaths, and over 75% of these deaths occur in low and middle-income countries. Arrhythmia is a representative type of CVD that refers to any irregular change from the normal heart rhythms. There are several types of arrhythmia including atrial fibrillation, premature contraction, ventricular fibrillation, and tachycardia. Although a single arrhythmia heartbeat may not have a serious impact on life, continuous arrhythmia beats can result in fatal circumstances. In this project, we build an effective electrocardiogram (ECG) arrhythmia classification method using a convolution al neural network (CNN), in which we classify ECG into seven categories, one being normal and the other six being different types of arrhythmia using deep two-dimensional CNN with grayscale ECG images. We are creating a web application where the user selects the image which is to be classified. The image is fed into the model that is trained and the cited class will be displayed on the webpage.

1.2 Purpose

In the past few decades, Deep Learning has proved to be a compelling tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolution al Neural Networks.

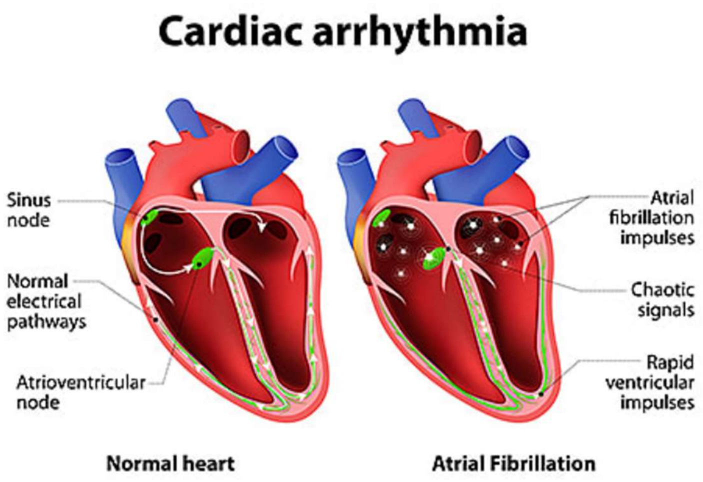


In deep learning, a convolution al neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

2. LITERATURE SURVEY

2.1 Existing problem

Cardiovascular diseases (CVDs) are the number one cause of death today. Over 17.7 million people died from CVDs in the year 2017 all over the world which is about 31% of all deaths, and over 75% of these deaths occur in low and middle-income countries. Arrhythmia is a representative type of CVD that refers to any irregular change from the normal heart rhythms. There are several types of arrhythmia including atrial fibrillation, premature contraction, ventricular fibrillation, and tachycardia.

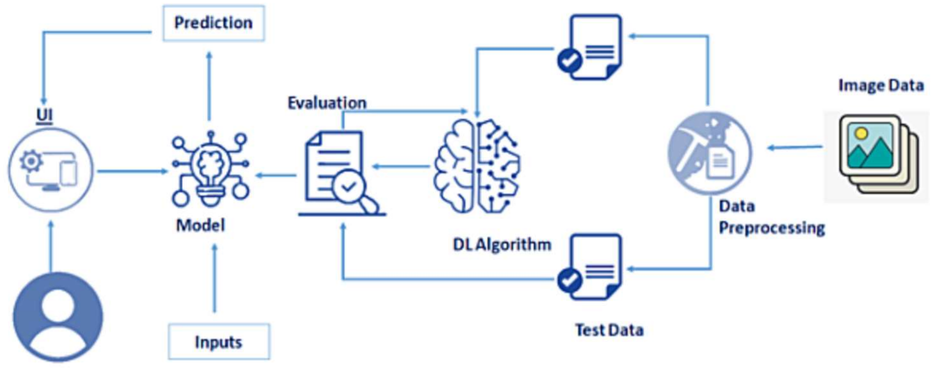


2.2 Proposed solution

An "ambulatory electrocardiogram" or an ECG) about the size of a postcard or digital camera that the patient will be using for 1 to 2 days, or up to 2 weeks. The test measures the movement of electrical signals or waves through the heart. These signals tell the heart to contract (squeeze) and pump blood. The patient will have electrodes taped to your skin. It's painless, although some people have mild skin irritation from the tape used to attach the electrodes to the chest. They can do everything but shower or bathe while wearing the electrodes. After the test period, patient will go back to see your doctor. They will be downloading the information.

3. THEORITICAL ANALYSIS

3.1 Block diagram



We will prepare the project by following the below steps:

- We will be working with Sequential type of modelling.
- We will be working with Keras capabilities.
- We will be working with image processing techniques.
- We will build a web application using the Flask framework.
- Afterwards we will be training our dataset in the IBM cloud and building another model from IBM and we will also test it.

3.2 Hardware / Software designing

Hardware Components used:

Since we are using the IBM cloud as a platform to execute this project we don’t need any hardware components other than our system**.**

Software Components Used: We will be using Anaconda Navigator which is installed in our system and Watson studio from the IBM cloud to complete the project.

4. EXPERIMENTAL INVESTIGATIONS

In this project, we have deployed our training model using CNN on IBM Watson studio and in our local machine. We are deploying 4 types of CNN layers in a sequential manner, starting from:

Convolutional layer 2D:

A 2-D convolutional layer applies sliding convolutional filters to 2-D input. The layer convolves the input by moving the filters along the input vertically and horizontally and computing the dot product of the weights and the input, and then adding a bias term.

Pooling Layer:

Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarises the features present in a region of the feature map generated by a convolution layer.

Fully-Connected layer:

After extracting features from multiple convolution layers and pooling layers, the fully-connected layer is used to expand the connection of all features. Finally, the SoftMax layer makes a logistic regression classification. Fully-connected layer transfers the weighted sum of the output of the previous layer to the activation function.

Dropout Layer:

There is usually a dropout layer before the fully connected layer. The dropout layer will temporarily disconnect some neurons from the network according to the certain probability during the training of the convolution neural network, which reduces the joint adaptability between neuron nodes, reduces overfitting, and enhances the generalization ability of the network.

5. FLOWCHART

5.1 Flow Chart & Results by training model in local machine:

a. Dataset Collection: The dataset contains six classes:

1. Left Bundle Branch Block
2. Normal
3. Premature Atrial Contraction
4. Premature Ventricular Contractions
5. Right Bundle Branch Block
6. Ventricular Fibrillation

b. Image Preprocessing:

Image Pre-processing includes the following main tasks

- **Import ImageDataGenerator Library:**

  Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class

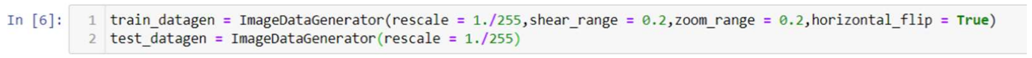
```
In [5]: from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

- **Configure ImageDataGenerator Class:**

  There are five main types of data augmentation techniques for image data; specifically:

  - Image shifts via the width_shift_range and height_shift_range arguments.
  - Image flips via the horizontal_flip and vertical_flip arguments.
  - Image rotates via the rotation_range argument
  - Image brightness via the brightness_range argument.
  - Image zooms via the zoom_range argument.

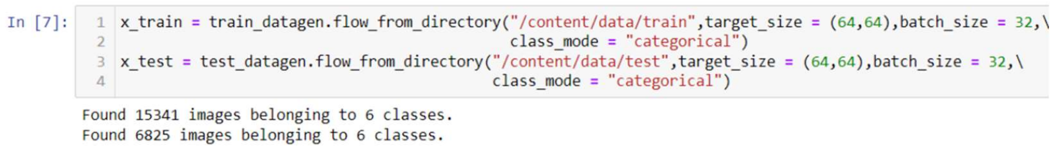
An instance of the ImageDataGenerator class can be constructed for train and test.



```
In [6]:
train_datagen = ImageDataGenerator(rescale = 1./255,shear_range = 0.2,zoom_range = 0.2,horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

● **Applying ImageDataGenerator functionality to the trainset and test set:**

We will apply ImageDataGenerator functionality to Trainset and Testset by using the following code This function will return batches of images from the subdirectories Left Bundle Branch Block, Normal, Premature Atrial Contraction, Premature Ventricular Contractions, Right Bundle Branch Block and Ventricular Fibrillation, together with labels 0 to 5{'Left Bundle Branch Block': 0, 'Normal': 1, 'Premature Atrial Contraction': 2, 'Premature Ventricular Contractions': 3, 'Right Bundle Branch Block': 4, 'Ventricular Fibrillation': 5}
We can see that for training there are 15341 images belonging to 6 classes and for testing there are 6825 images belonging to 6 classes.

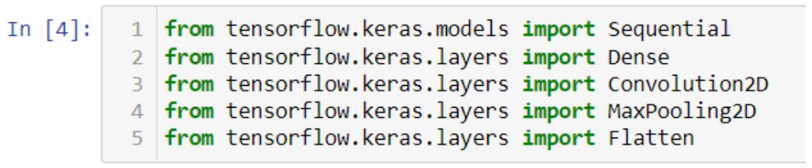


```
In [7]:
x_train = train_datagen.flow_from_directory("/content/data/train",target_size = (64,64),batch_size = 32,\
                                            class_mode = "categorical")
x_test = test_datagen.flow_from_directory("/content/data/test",target_size = (64,64),batch_size = 32,\
                                          class_mode = "categorical")

Found 15341 images belonging to 6 classes.
Found 6825 images belonging to 6 classes.
```

c. 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 model building Libraries:



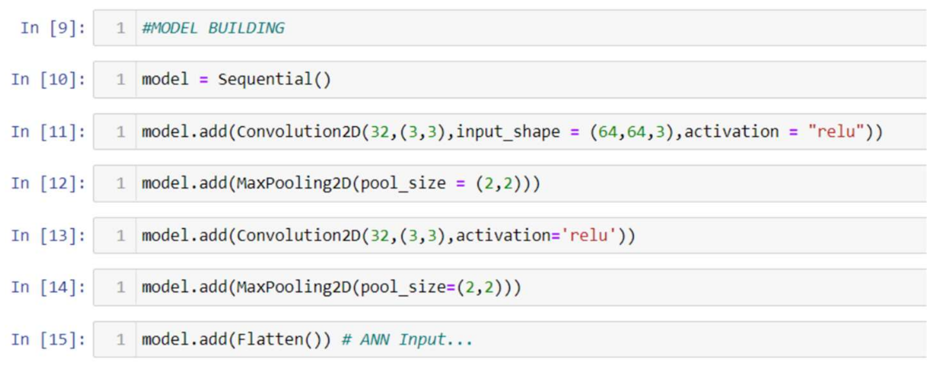
```
In [4]:
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
```

● Initializing the model: Keras has 2 ways to define a neural network:

- Sequential
- Function API

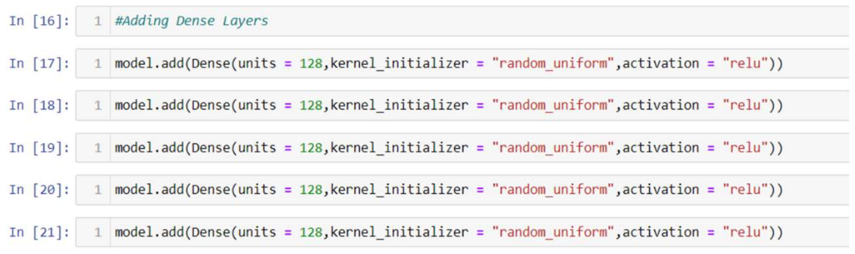
The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add () method. Now, will initialize our model.

● Adding CNN Layers: 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 down sample the input. The flatten layer flattens the input.



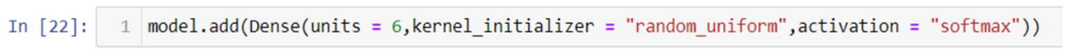
```
In [9]:  #MODEL BUILDING

In [10]: model = Sequential()

In [11]: model.add(Convolution2D(32,(3,3),input_shape = (64,64,3),activation = "relu"))

In [12]: model.add(MaxPooling2D(pool_size = (2,2)))

In [13]: model.add(Convolution2D(32,(3,3),activation='relu'))

In [14]: model.add(MaxPooling2D(pool_size=(2,2)))

In [15]: model.add(Flatten()) # ANN Input...
```

● **Adding Hidden Layers:** Dense layer is deeply connected neural network layer. It is most common and frequently used layer.



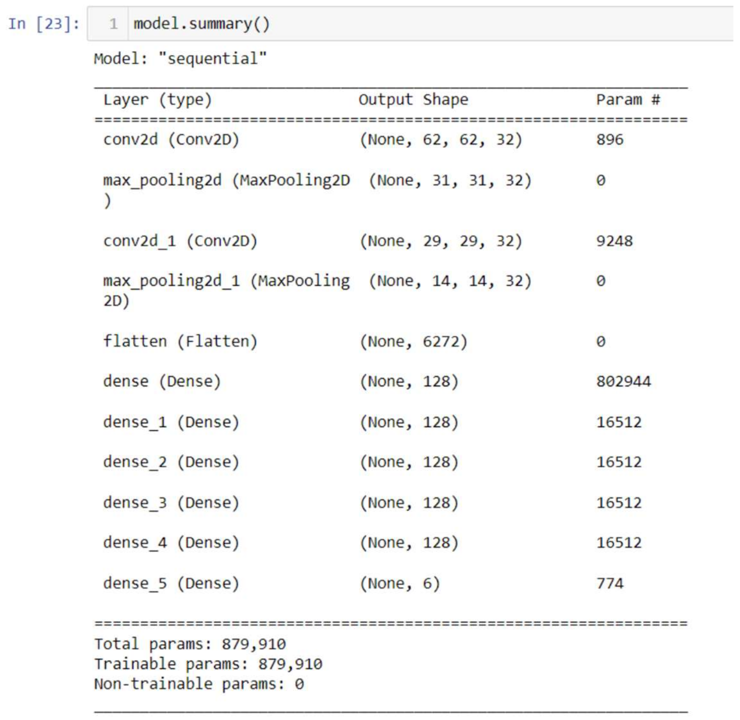
```
In [16]: #Adding Dense Layers

In [17]: model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))

In [18]: model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))

In [19]: model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))

In [20]: model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))

In [21]: model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))
```

● **Adding Output Layer:**



```
In [22]: 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 m odel and its layers



```
In [23]: 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, 128)               802944

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 128)               16512

 dense_3 (Dense)             (None, 128)               16512

 dense_4 (Dense)             (None, 128)               16512

 dense_5 (Dense)             (None, 6)                 774

=================================================================
Total params: 879,910
Trainable params: 879,910
Non-trainable params: 0
_________________________________________________________________
```

● Configure the Learning Process:

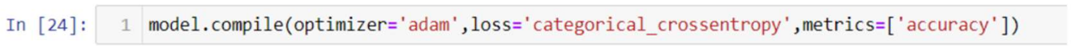
- The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find error or deviation in the learning process. Keras requires loss function during the model compilation process.

- 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.

- **Training the model:**

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

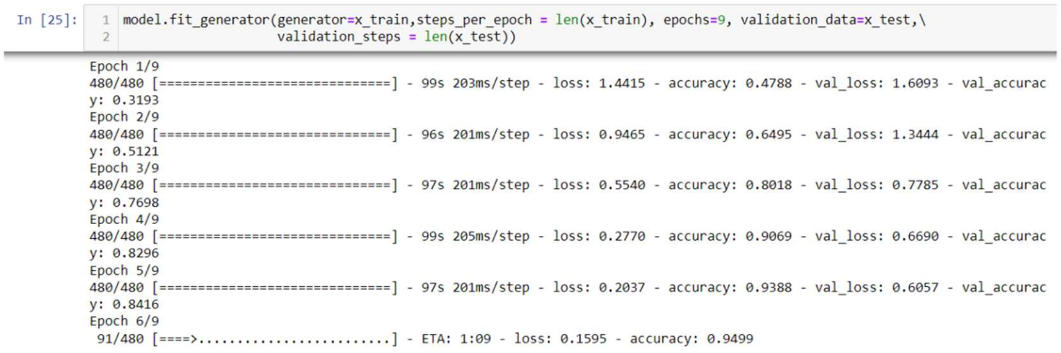
- **Saving the model**: The model is saved with .h5 extension as follows An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.



```
In [24]:  model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

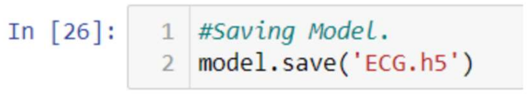
- **Training the model:**

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



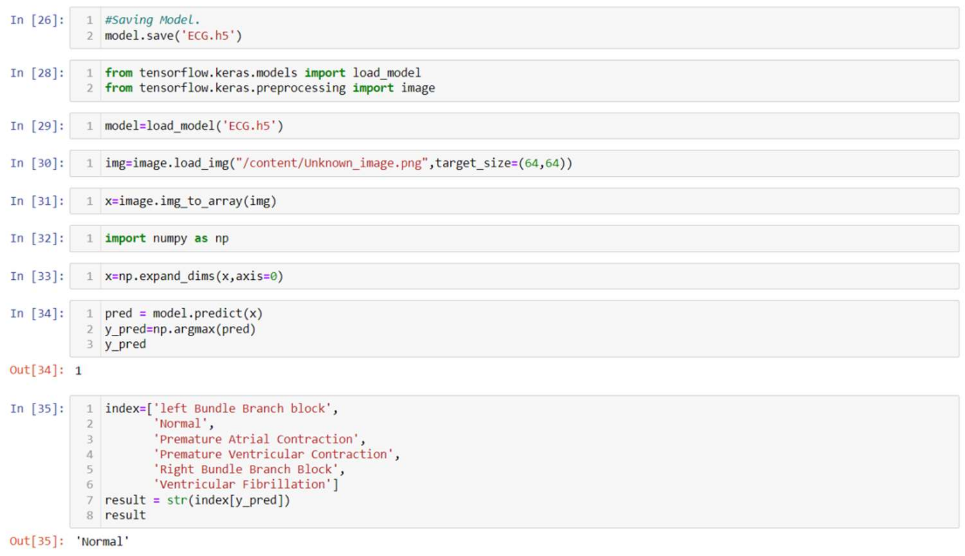
```
In [25]:  model.fit_generator(generator=x_train,steps_per_epoch = len(x_train), epochs=9, validation_data=x_test,\
                              validation_steps = len(x_test))

Epoch 1/9
480/480 [==============================] - 99s 203ms/step - loss: 1.4415 - accuracy: 0.4788 - val_loss: 1.6093 - val_accurac
y: 0.3193
Epoch 2/9
480/480 [==============================] - 96s 201ms/step - loss: 0.9465 - accuracy: 0.6495 - val_loss: 1.3444 - val_accurac
y: 0.5121
Epoch 3/9
480/480 [==============================] - 97s 201ms/step - loss: 0.5540 - accuracy: 0.8018 - val_loss: 0.7785 - val_accurac
y: 0.7698
Epoch 4/9
480/480 [==============================] - 99s 205ms/step - loss: 0.2770 - accuracy: 0.9069 - val_loss: 0.6690 - val_accurac
y: 0.8296
Epoch 5/9
480/480 [==============================] - 97s 201ms/step - loss: 0.2037 - accuracy: 0.9388 - val_loss: 0.6057 - val_accurac
y: 0.8416
Epoch 6/9
 91/480 [====>.........................] - ETA: 1:09 - loss: 0.1595 - accuracy: 0.9499
```

- **Saving 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.



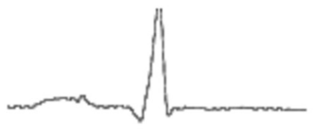
```
In [26]:  #Saving Model.
          model.save('ECG.h5')
```

- **Testing the model:** Load necessary libraries and load the saved model using load_model Taking an image as input and checking the results Note: The target size should for the image that is should be the same as the target size that you have used for training.



```
In [26]:  #Saving Model.
          model.save('ECG.h5')

In [28]:  from tensorflow.keras.models import load_model
          from tensorflow.keras.preprocessing import image

In [29]:  model=load_model('ECG.h5')

In [30]:  img=image.load_img("/content/Unknown_image.png",target_size=(64,64))

In [31]:  x=image.img_to_array(img)

In [32]:  import numpy as np

In [33]:  x=np.expand_dims(x,axis=0)

In [34]:  pred = model.predict(x)
          y_pred=np.argmax(pred)
          y_pred

Out[34]:  1

In [35]:  index=['left Bundle Branch block',
                 'Normal',
                 'Premature Atrial Contraction',
                 'Premature Ventricular Contraction',
                 'Right Bundle Branch Block',
                 'Ventricular Fibrillation']
          result = str(index[y_pred])
          result

Out[35]:  'Normal'
```

The unknown image uploaded is:

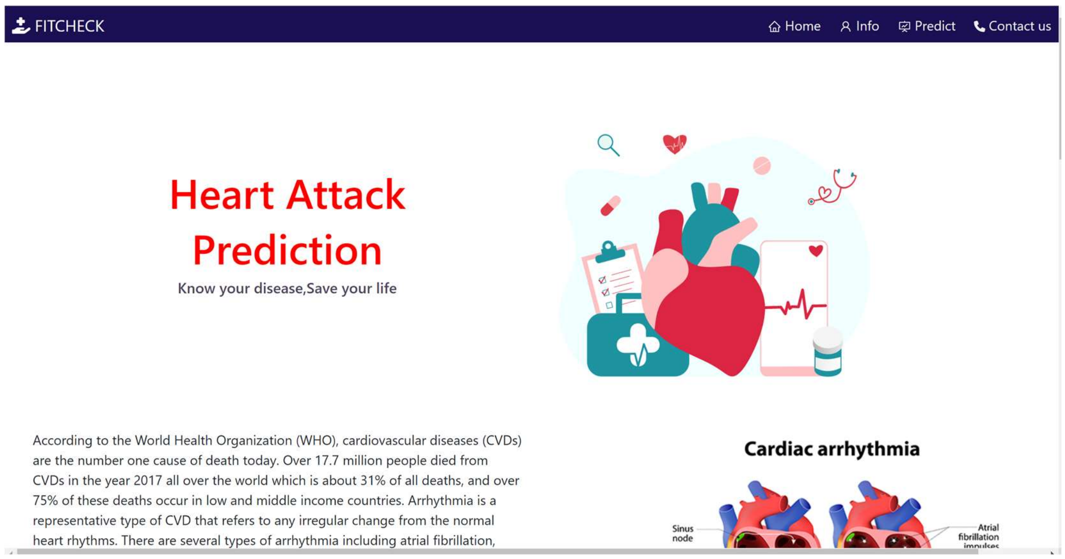


Here the output for the uploaded result is normal.

d. Application Building: In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has uploaded an image. The uploaded image is given to the saved model and prediction is showcased on the UI. This section has the following tasks

● Building HTML Pages:

- We use HTML to create the front end part of the web page.
- Here, we created 4 html pages- home.html, predict_base.html, predict.html, information.html.
- home.html displays the home page.

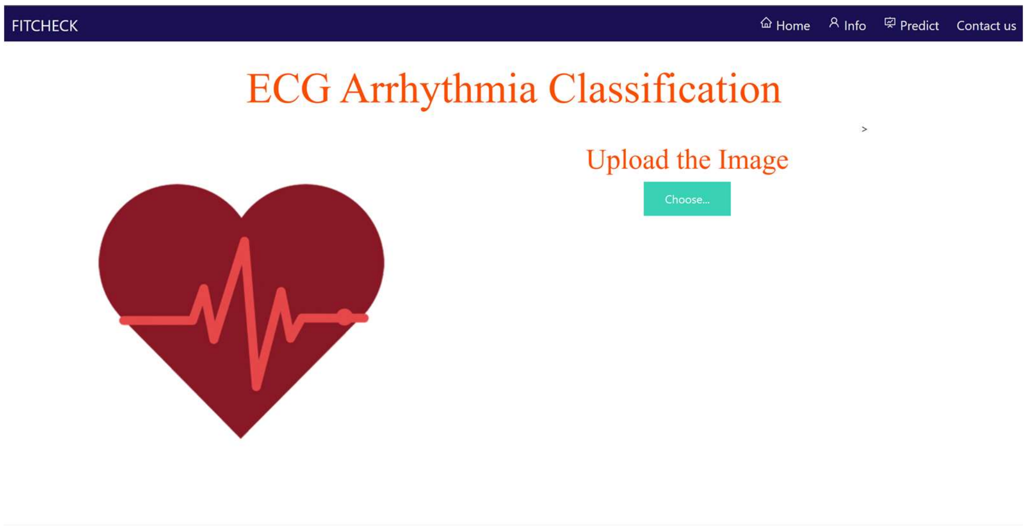


Heart Attack Prediction

Know your disease,Save your life

According to the World Health Organization (WHO), cardiovascular diseases (CVDs) are the number one cause of death today. Over 17.7 million people died from CVDs in the year 2017 all over the world which is about 31% of all deaths, and over 75% of these deaths occur in low and middle income countries. Arrhythmia is a representative type of CVD that refers to any irregular change from the normal heart rhythms. There are several types of arrhythmia including atrial fibrillation,

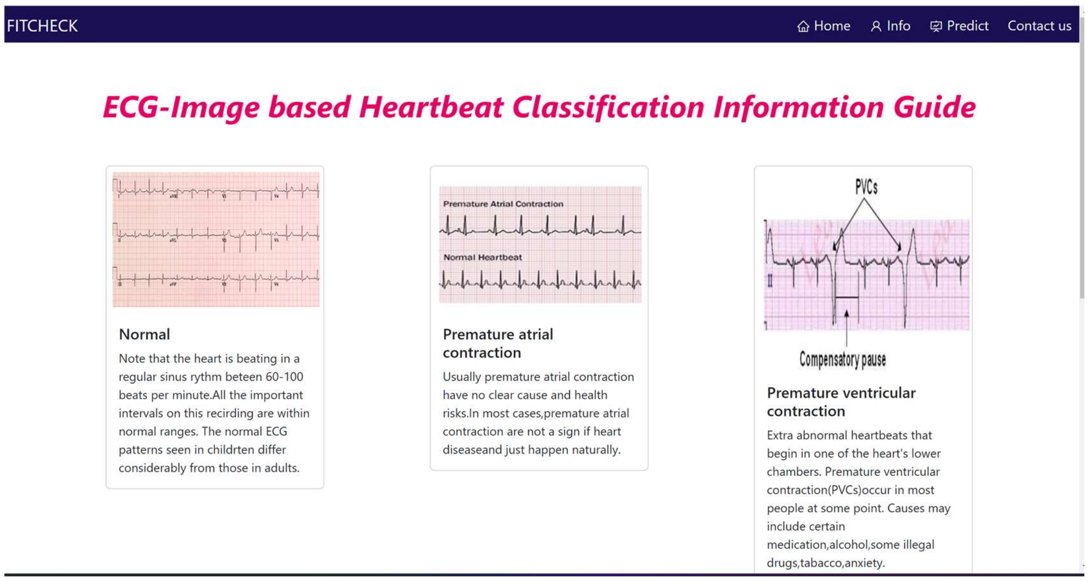
Cardiac arrhythmia



ECG Arrhythmia Classification

Upload the Image

- information.html displays all important details to be known about ECG.



ECG-Image based Heartbeat Classification Information Guide

Normal

Note that the heart is beating in a regular sinus rythm beteen 60-100 beats per minute.All the important intervals on this recirding are within normal ranges. The normal ECG patterns seen in childrten differ considerably from those in adults.

Premature atrial contraction

Usually premature atrial contraction have no clear cause and health risks.In most cases,premature atrial contraction are not a sign if heart diseaseand just happen naturally.

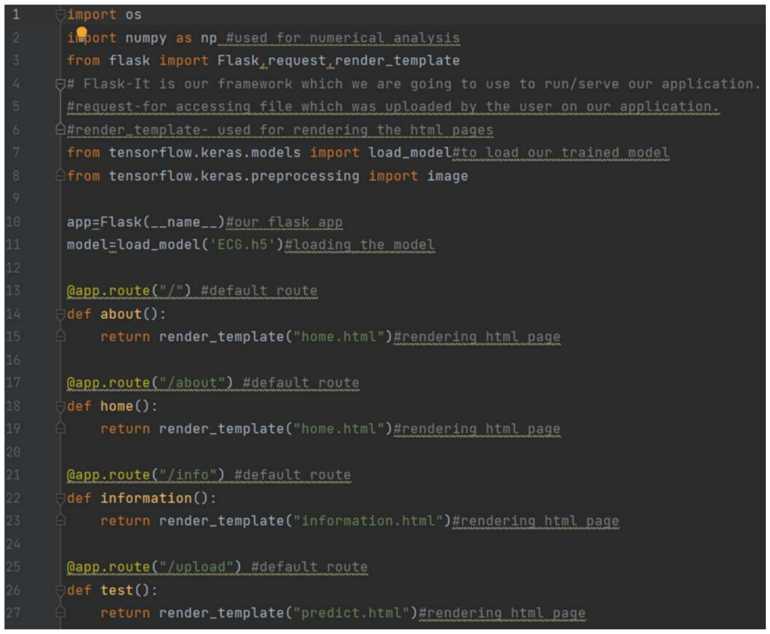
Premature ventricular contraction

Extra abnormal heartbeats that begin in one of the heart's lower chambers. Premature ventricular contraction(PVCs)occur in most people at some point. Causes may include certain medication,alcohol,some illegal drugs,tabacco,anxiety.

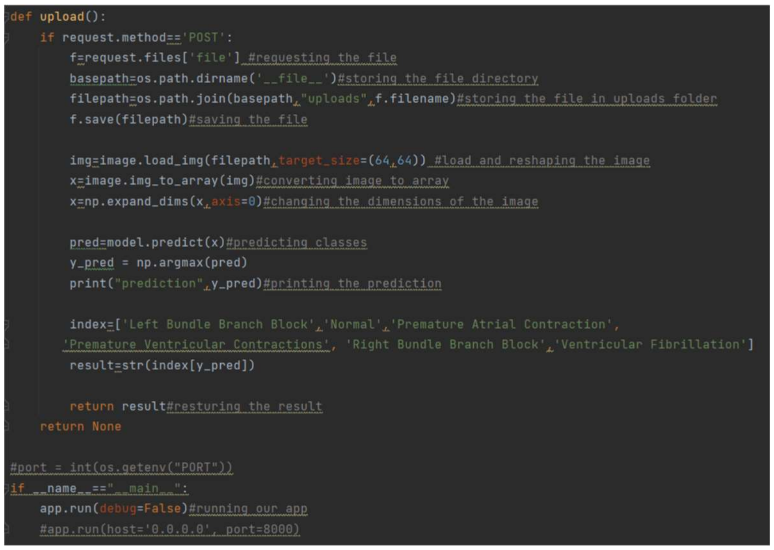
● **Building server-side script:**

We will build the flask file 'app.py' which is a web framework written in python for server-side scripting.

- The app starts running when the "__name__" constructor is called in main.
- render_template is used to return HTML file.
- "GET" method is used to take input from the user.
- "POST" method is used to display the output to the user.

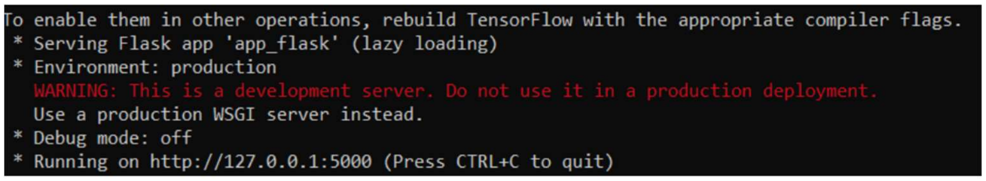


```python
import os
import numpy as np #used for numerical analysis
from flask import Flask,request,render_template
# Flask-It is our framework which we are going to use to run/serve our application.
#request-for accessing file which was uploaded by the user on our application.
#render_template- used for rendering the html pages
from tensorflow.keras.models import load_model#to load our trained model
from tensorflow.keras.preprocessing import image

app=Flask(__name__)#our flask app
model=load_model('ECG.h5')#loading the model

@app.route("/") #default route
def about():
    return render_template("home.html")#rendering html page

@app.route("/about") #default route
def home():
    return render_template("home.html")#rendering html page

@app.route("/info") #default route
def information():
    return render_template("information.html")#rendering html page

@app.route("/upload") #default route
def test():
    return render_template("predict.html")#rendering html page
```



```python
def upload():
    if request.method=='POST':
        f=request.files['file'] #requesting the file
        basepath=os.path.dirname('__file__')#storing the file directory
        filepath=os.path.join(basepath,"uploads",f.filename)#storing the file in uploads folder
        f.save(filepath)#saving the file

        img=image.load_img(filepath,target_size=(64,64)) #load and reshaping the image
        x=image.img_to_array(img)#converting image to array
        x=np.expand_dims(x,axis=0)#changing the dimensions of the image

        pred=model.predict(x)#predicting classes
        y_pred = np.argmax(pred)
        print("prediction",y_pred)#printing the prediction

        index=['Left Bundle Branch Block','Normal','Premature Atrial Contraction',
       'Premature Ventricular Contractions', 'Right Bundle Branch Block','Ventricular Fibrillation']
        result=str(index[y_pred])

        return result#resturing the result
    return None

#port = int(os.getenv("PORT"))
if __name__=="__main__":
    app.run(debug=False)#running our app
    #app.run(host='0.0.0.0', port=8000)
```

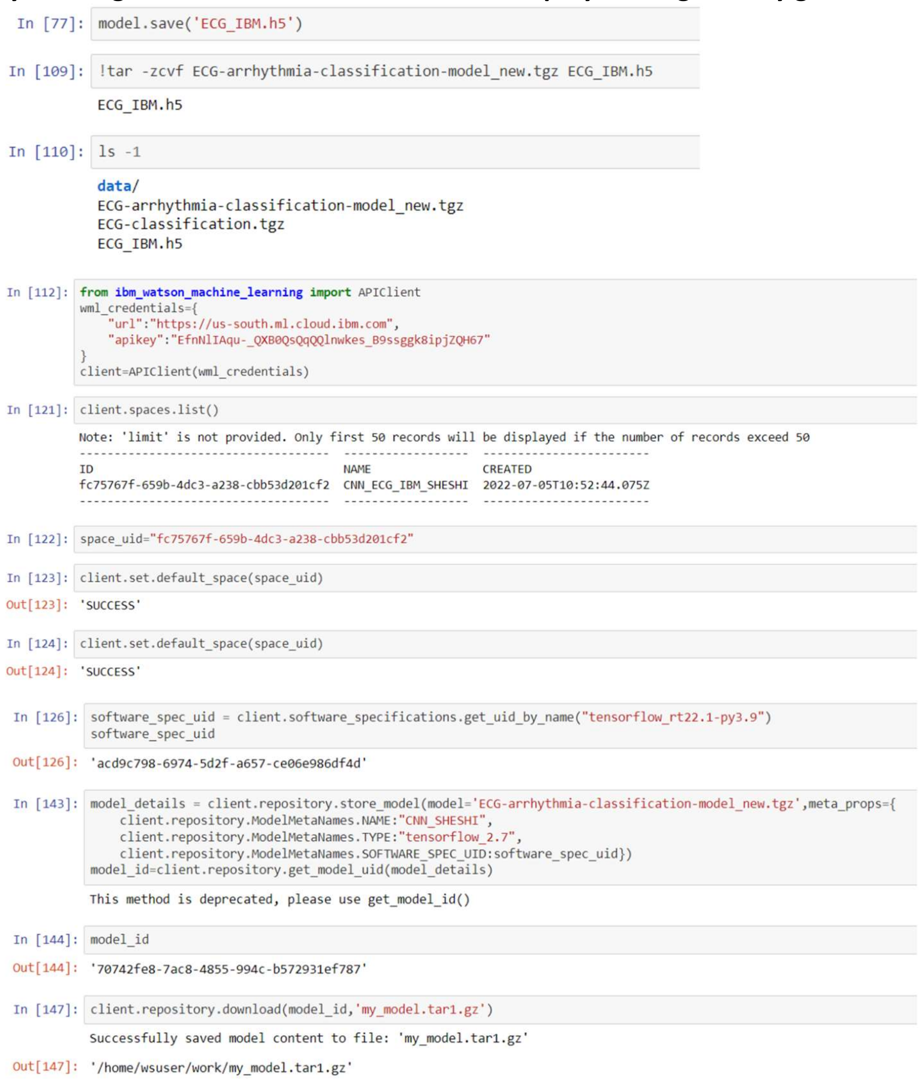
● **Running The App:**



```
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Serving Flask app 'app_flask' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
```

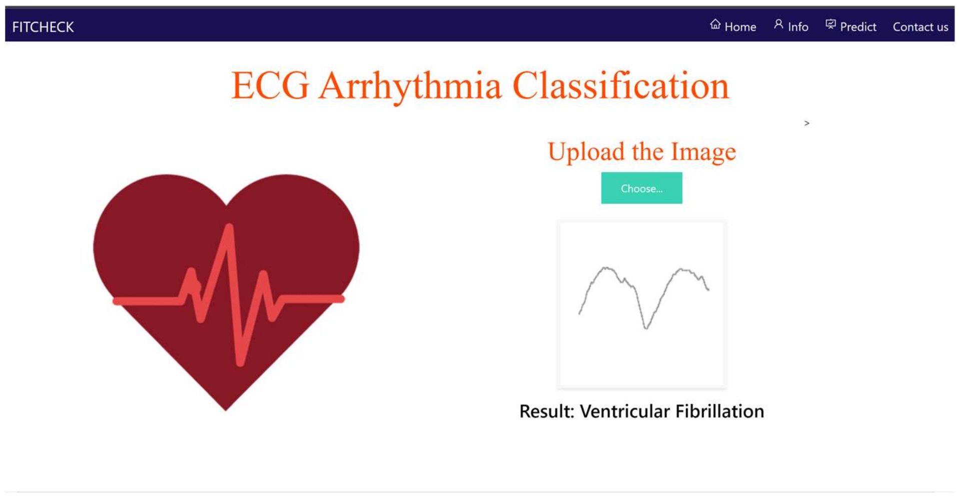
Navigate to the localhost (http://127.0.0.1:5000/)where you can view your web page.

e. Apply CNN algorithm and save the model and deploy it using API key generated:



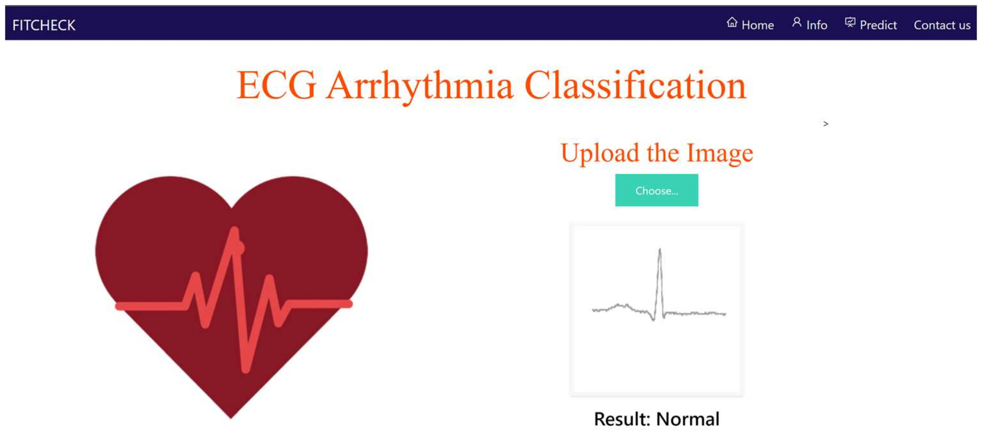
```
In [77]: model.save('ECG_IBM.h5')

In [109]: !tar -zcvf ECG-arrhythmia-classification-model_new.tgz ECG_IBM.h5

          ECG_IBM.h5

In [110]: ls -1

          data/
          ECG-arrhythmia-classification-model_new.tgz
          ECG-classification.tgz
          ECG_IBM.h5

In [112]: from ibm_watson_machine_learning import APIClient
          wml_credentials={
              "url":"https://us-south.ml.cloud.ibm.com",
              "apikey":"EfnNlIAqu-_QXB0QsQqQQlnwkes_B9ssggk8ipjZQH67"
          }
          client=APIClient(wml_credentials)

In [121]: client.spaces.list()

          Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
          ------------------------------------  ------------------  ------------------------
          ID                                    NAME                CREATED
          fc75767f-659b-4dc3-a238-cbb53d201cf2  CNN_ECG_IBM_SHESHI  2022-07-05T10:52:44.075Z
          ------------------------------------  ------------------  ------------------------

In [122]: space_uid="fc75767f-659b-4dc3-a238-cbb53d201cf2"

In [123]: client.set.default_space(space_uid)

Out[123]: 'SUCCESS'

In [124]: client.set.default_space(space_uid)

Out[124]: 'SUCCESS'

In [126]: software_spec_uid = client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
          software_spec_uid

Out[126]: 'acd9c798-6974-5d2f-a657-ce06e986df4d'

In [143]: model_details = client.repository.store_model(model='ECG-arrhythmia-classification-model_new.tgz',meta_props={
              client.repository.ModelMetaNames.NAME:"CNN_SHESHI",
              client.repository.ModelMetaNames.TYPE:"tensorflow_2.7",
              client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid})
          model_id=client.repository.get_model_uid(model_details)

          This method is deprecated, please use get_model_id()

In [144]: model_id

Out[144]: '70742fe8-7ac8-4855-994c-b572931ef787'

In [147]: client.repository.download(model_id,'my_model.tar1.gz')

          Successfully saved model content to file: 'my_model.tar1.gz'

Out[147]: '/home/wsuser/work/my_model.tar1.gz'
```

6. RESULT

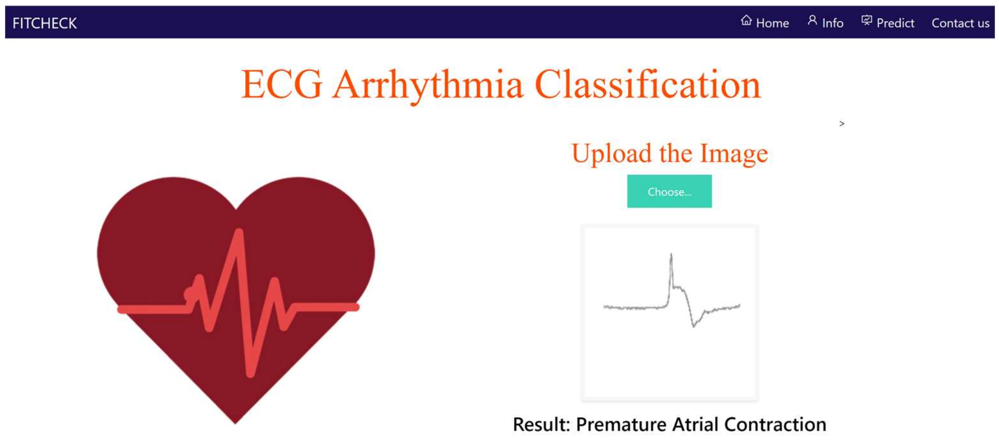


ECG Arrhythmia Classification

Result: Ventricular Fibrillation



Result: Normal



Result: Premature Atrial Contraction

7. ADVANTAGES & DISADVANTAGES

7.1 Advantages:

- The proposed model predicts Arrhythmia in images with a high accuracy rate of nearly 96%
- The early detection of Arrhythmia gives better understanding of disease causes, initiates therapeutic interventions and enables developing appropriate treatments.

7.2 Disadvantages:

- Not useful for identifying the different stages of Arrhythmia disease.
- Not useful in monitoring motor symptoms

8. APPLICATIONS

- It is useful for identifying the arrhythmia disease at an early stage.
- It is useful in detecting cardiovascular disorders .

9. CONCLUSION

- Cardiovascular disease is a major health problem in today's world. The early diagnosis of cardiac arrhythmia highly relies on the ECG.

● Unfortunately, the expert level of medical resources is rare, visually identify the ECG signal is challenging and time-consuming.

● The advantages of the proposed CNN network have been put to evidence.

● It is endowed with an ability to effectively process the non-filtered dataset with its potential anti-noise features. Besides that, ten-fold cross-validation is implemented in this work to further demonstrate the robustness of the network.

10. FUTURE SCOPE

For future work, it would be interesting to explore the use of optimization techniques to find a feasible design and solution. The limitation of our study is that we have yet to apply any optimization techniques to optimize the model parameters and we believe that with the implementation of the optimization, it will be able to further elevate the performance of the proposed solution to the next level.

11. BIBLOGRAPHY

○ https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/
○https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.convolution2dlayer.html;jsessionid=0a7e3bc26fabda07a5032030294b

APPENDIX

Github Repositories Link:

https://github.com/nimishsara12/ECG-AI-Model