CLASSIFICATION OF ARRHYTHMIA BY USING DEEP LEARNING WITH 2-D ECG SPECTRAL IMAGE REPRESENTATION

PROJECT REPORT PRESENTED FOR HX5001 - PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITYAND ENTREPRENEURSHIP

Submitted by

CHARAAN S (2019504511)

ROHIT JACOB GEORGE (2019504572)

YUVARAJ S (2019504610)

KOUSHIK M (2019504540)

DEPARTMENT OF ELECTRONICS ENGINEERING

MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI - 600 044

PROJECT REPORT

1. INTRODUCTION:

a. Project 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 arrhythmias 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 convolutional neural network (CNN), in which we classify ECG into seven categories, one being normal and the other six being different types of arrhythmias 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.

b. Purpose:

In this project, we intend to develop a web interface that is capable of predicting the type of Arrhythmia, that a person is diagnosed with, based on the ECG image given to the interface by the user. This project will be useful for people who are at high-risks of getting diagnosed with heart ailments, and by using this application, they might be able to undertake necessary preventive measures to tackle the ailment at the earliest. We have made the interface as smooth and as user-friendly as possible, and we have removed several bloatware that take up space and hinder the performance. For the prediction, we are using a custom Convolutional Neural Network Model with 2 million trainable parameters, that yields an accuracy of over 98%.

2. LITERATURE SURVEY

a. Existing Problem:

In the existing methods proposed for Arrhythmia classification the techniques used for ECG denoising, Feature Extraction and Classification can be improved. ECG pre-processing methods are not discussed in detailed fashion, in some papers. The variability of other ECG waves, such as P waves, and T waves could be studied based on the proposed feature extraction algorithm to further improve the performance of the whole system. Also, in a few papers, Support-Vector - Machine algorithm is used, which is inefficient when compared to Artificial Neural Networks. In summary, the techniques discussed in the existing methods are quite outdated.

b. References:

1. Mohebbanaaz, Y. P. Sai and L. V. R. kumari, "A Review on Arrhythmia Classification Using ECG Signals," 2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS), 2020, pp. 1-6, doi: 10.1109/SCEECS48394.2020.9.

2. J. Lang and F. Yang, "An improved classification method for arrhythmia electrocardiogram dataset," 2019 IEEE 2nd International Conference on Information Communication and Signal Processing (ICICSP), 2019, pp. 338-341, doi: 10.1109/ICICSP48821.2019.8958499.

3. H. Yang and Z. Wei, "Arrhythmia Recognition and Classification Using Combined Parametric and Visual Pattern Features of ECG Morphology," in IEEE Access, vol. 8, pp. 47103-47117, 2020, doi: 10.1109/ACCESS.2020.2979256.

4. P. Varalakshmi and A. P. Sankaran, "Classification of Arrhythmia Based on Machine Learning Algorithms Using ECG Signals," 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), 2022, pp. 1-7, doi: 10.1109/ACCAI53970.2022.9752565.

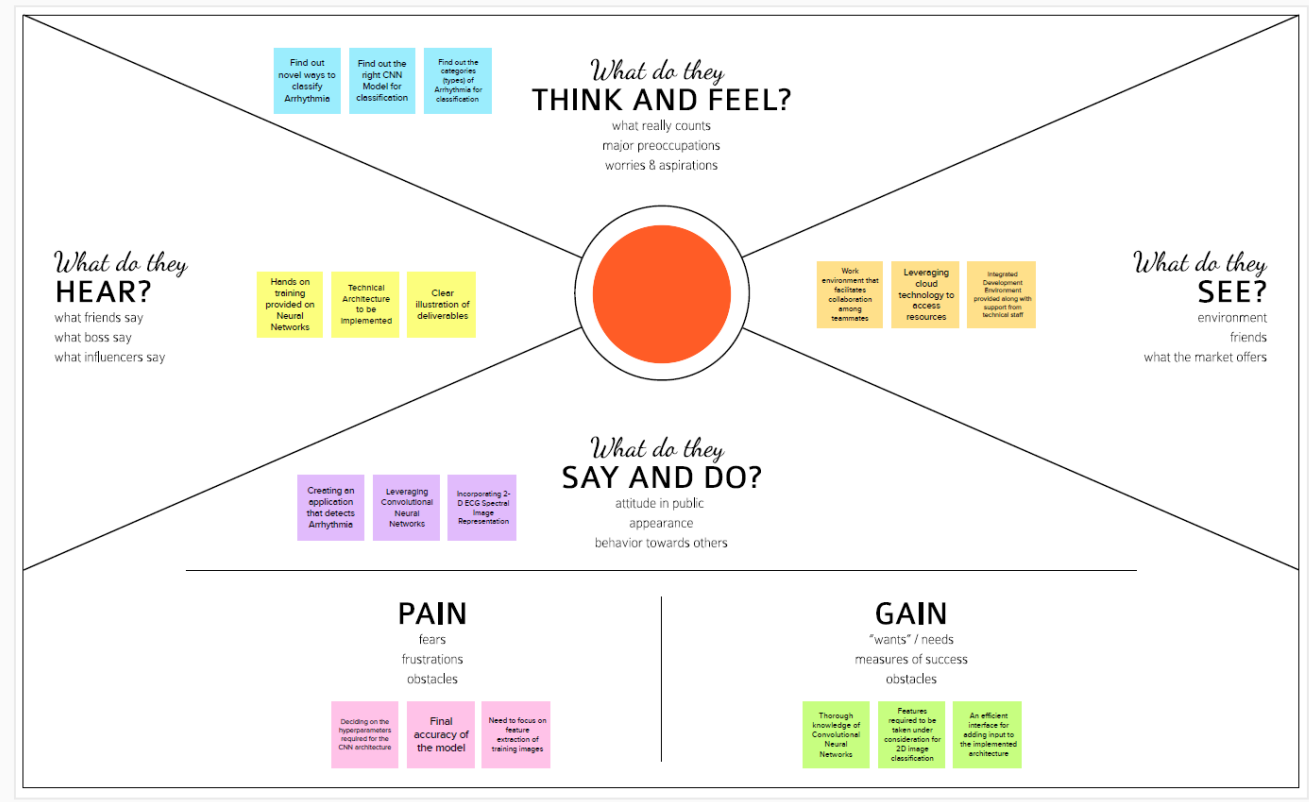
5. C. Ye, M. T. Coimbra and B. V. K. Vijaya Kumar, "Arrhythmia detection and classification using morphological and dynamic features of ECG signals," 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology, 2010, pp. 1918-1921, doi: 10.1109/IEMBS.2010.5627645.

c. Problem Statement Definition:

In this project, an effective electrocardiogram (ECG) arrhythmia classification method using a convolutional neural network (CNN) needs to be built, in which ECG images are to be classified into seven categories, one being normal and the other six being different types of arrhythmias using deep two-dimensional CNN with grayscale ECG images. Later, a web application is created, 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 should be displayed on the webpage.

3. IDEATION AND PROPOSED SOLUTION:

a. Empathy Map Canvas



b. Ideation and Brainstorming

Multiple iterations of solution were discussed at first. Based on the stakeholder's interests, features were added and updated to the proposed solution. Demographics and percentage of at-risk people were taken into account. The metrics proposed by reliable survey reports and estimates, were considered while working on developing a novel solution.

c. Proposed Solution

1. Problem Statement (Problem to be solved):
   To create a Deep Learning Model that classifies various types of Arryhythmia with 2-dimensional ECG Spectral Image representation.
2. Idea / Solution Description
   In the pre-processing phase - the electromyographic noise present in the ECG signals are removed using wavelet based thresholding technique. Next the ECG signal is transformed into a 2-D representation using a 2D CNN Model. In order to perform this transformation, an efficient CNN model is implemented after analysing various architectures. The core idea is to make this CNN model classify different kinds of arrhythmia such as LBB, PVC, RBB etc.
3. Novelty / Uniqueness
   We will attempt to create an API that is capable of handling inputs and producing the corresponding Arrhythmia class for the given ECG signal. This API will help to simulate an interactive user environment for gaining a seamless experience.
4. Social Impact / Customer Satisfaction:
   The major stakeholders of this project are the individuals aged more than 50. By getting to know about irregularities in the ECG signals of heartbeats, customers can greatly benefit from early diagnosis of heart ailments.
5. Business model / Revenue Model:
   There isn't a single application out there in the market that performs detection of Arrhythmia. By integrating our novel product with a smart wearable, we can launch a new series of health tracking smart devices.
6. Scalability of the solution:
   By pitching our idea to Angel Investors and Venture Captialists, we can extend this idea into a reality by launching a fully-fledged startup that markets this product.

d. Problem Solution Fit:

4. REQUIREMENT ANALYSIS

a. Functional Requirement:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User selection	Knowledge about ECG images select the image to be classified
FR-4	User input	Images need to be uploaded
FR-5	Save image	Images are saved in uploaded folder

FR-6	Predict ECG image	User ECG images in our web application Collection of data sets Database read ECG images

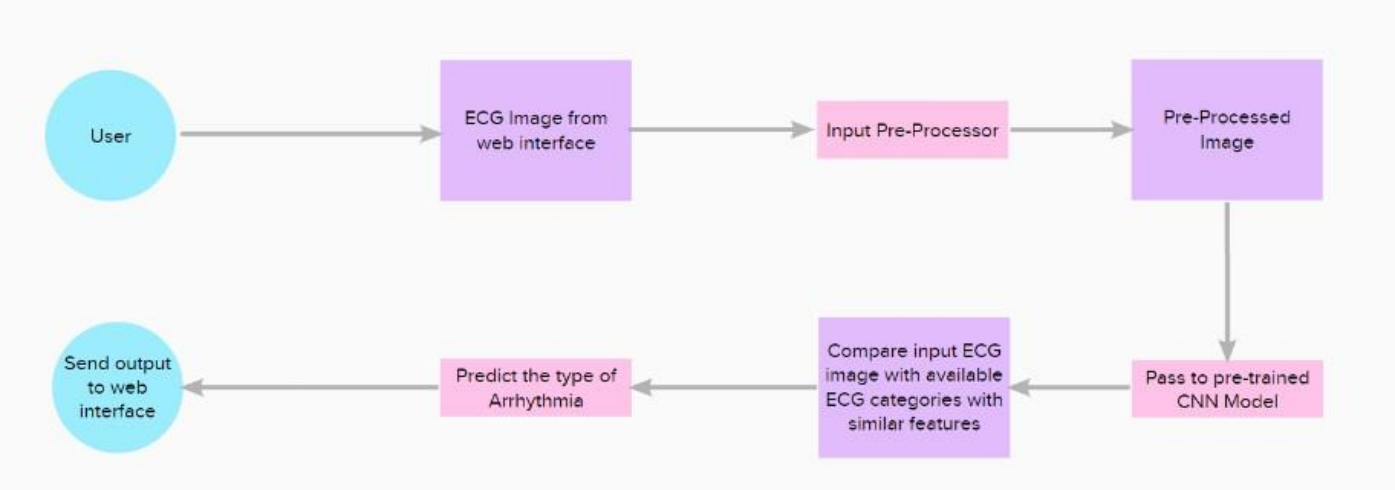
b. Non-Functional Requirement:

Following are the non-functional requirements of the proposed solution.

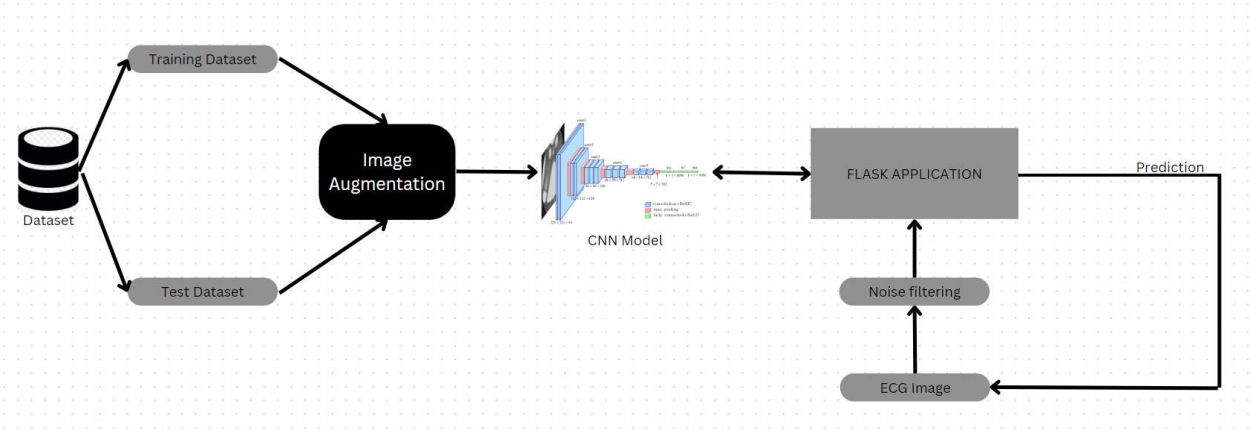
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	A user friendly and simple UI Web application. Easy drag and drop options
NFR-2	**Security**	No third-party web and UI is used for prediction of data Details about user interaction with the web application are protected
NFR-3	**Reliability**	Higher accuracy rate Defect free
NFR-4	**Performance**	Fast and quick classification of the required class is done
NFR-5	**Availability**	Availability describes how likely the system is accessible to a user at a given point in time and the periodically for a solution.
NFR-6	**Scalability**	The ability of the user problem in arrhythmia disease to handle an increase in workload without performance degradation

5. PROJECT DESIGN

a. Data Flow Diagrams:



b. Solution and Technical Architecture:



c. User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Patient/Doctor (Web User)	Web interface	USN-1	As a user, I can access the web interface	I can login to my account	High	Sprint-1
Patient/Doctor (Web User)	Dashboard	USN-2	As a user, I can access the dashboard/homepage	I can view the homepage	High	Sprint-1
Patient/Doctor (Web User)	Types of Arrhythmias	USN-3	As a user, I can view various articles about different kinds of Arrhythmia	I can view the articles	Low	Sprint-1
Patient/Doctor (Web User)	Page Navigation	USN-4	As a user, I can access several tabs and pages on the interface	I can view different pages and navigate	Medium	Sprint-2
Patient/Doctor (Web User)	Info and About Page	USN-5	As a user, I can see the info and about page for the web interface	I can view the info and about page	Medium	Sprint-2
Patient/Doctor (Web User)	Page to send input	USN-6	As a user, I can see an option to upload input image of ECG	I can view the input page	High	Sprint-3
Patient/Doctor (Web User)	Prediction result page	USN-7	As a user, I can see the predicted result	I can view the prediction	High	Sprint-3

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
			for the given ECG image			
Patient/Doctor (Web User)	Type of Arrhythmia	USN-8	As a user, I can see the type of Arrhythmia	I can view the type of Arrhythmia page	High	Sprint-3
Patient/Doctor (Web User)	Side-effects page	USN-9	As a user, I can see the various side effects of the predicted Arrhythmia	I can view the side effects page	Low	Sprint-4
Patient/Doctor (Web User)	Prediction history page	USN-10	As a user, I can see the various predictions done in the past	I can view the prediction history page	Medium	Sprint-4
Patient/Doctor (Web User)	Type of CVD page	USN-11	As a user, I can see the predicted type of CVD based on predicted arrhythmia	I can view the type of CVD page	High	Sprint-4
Administrator	Performance metrics	USN-12	As an administrator, I can see the number of people who are using the developed interface	I can view the performance metrics	Medium	Sprint-4

6. PROJECT PLANNING AND SCHEDULING

a. Sprint Planning and Estimation

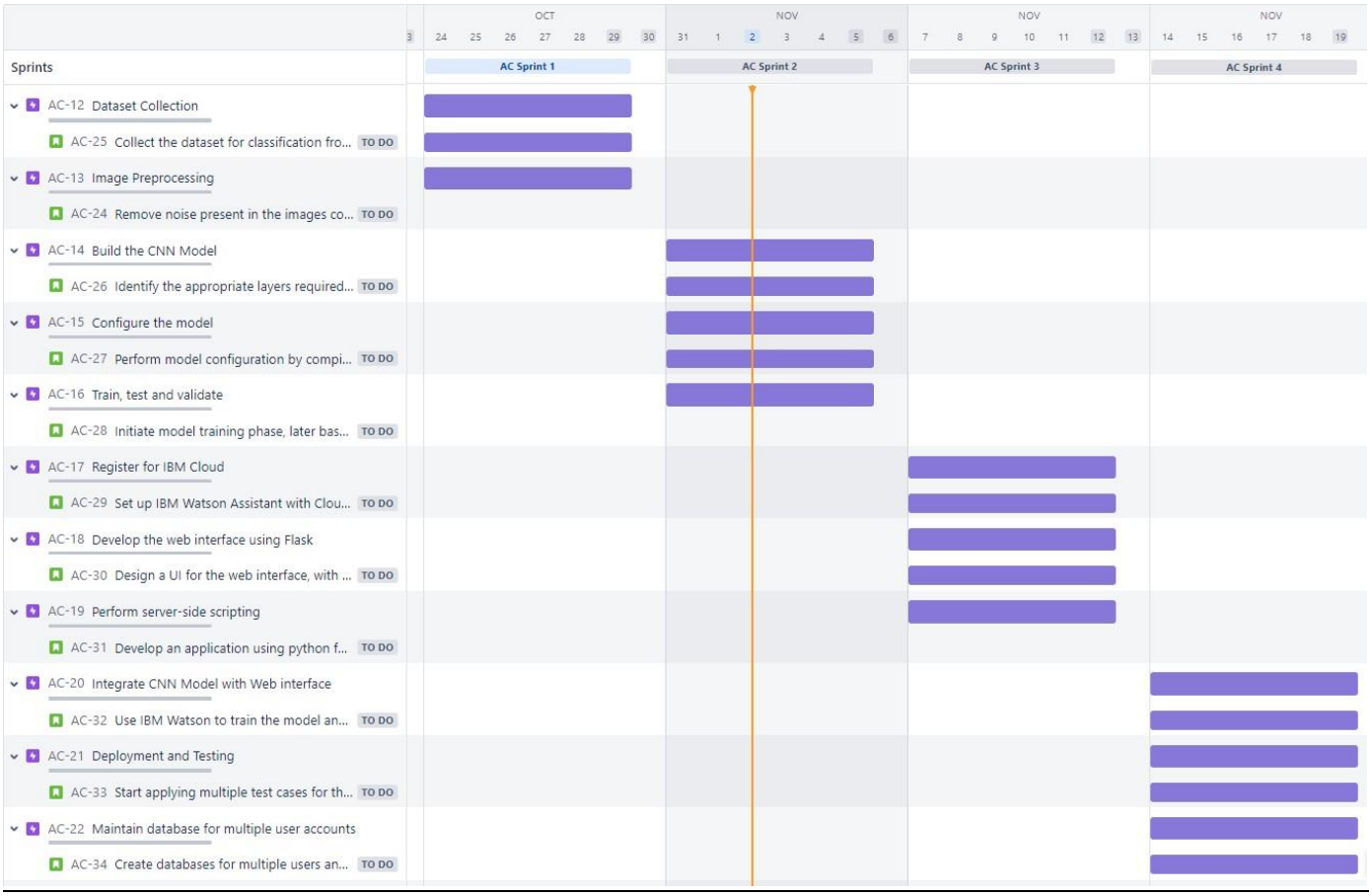
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Dataset Collection	USN-1	Collect the dataset for classification from sources available on the internet.	10	High	Charaan, Yuvaraj
Sprint-1	Image Preprocessing	USN-2	Remove noise present in the images collected and perform data pre-processing	10	High	Rohit Jacob George, Koushik
Sprint-2	Build the CNN Model	USN-3	Identify the appropriate layers required for the model and determine the model parameters	2	High	Charaan, Rohit Jacob George
Sprint-2	Configure the model	USN-4	Perform model configuration by compiling it and implement techniques for loss reduction	5	Medium	Yuvaraj, Koushik
Sprint-2	Train, test and validate	USN-5	Initiate model training phase, later based on model and validation loss values, start test phase	13	High	Charaan, Koushik
Sprint-3	Register for IBM Cloud	USN-6	Set up IBM Watson Assistant with Cloud Service	2	High	Rohit Jacob George, Yuvaraj
Sprint-3	Develop the web interface using Flask	USN-7	Design a UI for the web interface, with login,	5	High	Charaan, Rohit

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
			registration and input adding features			Jacob George
Sprint-3	Perform server-side scripting	USN-8	Develop an application using python for back-end functions	13	Medium	Rohit Jacob George, Yuvaraj
Sprint-4	Integrate CNN Model with Web interface	USN-9	Use IBM Watson to train the model and integrate it with the Flask application	2	High	Charaan, Koushik
Sprint-4	Deployment and Testing	USN-10	Start applying multiple test cases for the developed application	5	Medium	Rohit Jacob George, Yuvaraj
Sprint-4	Maintain database for multiple user accounts	USN-11	Create databases for multiple users and maintain history of cases for each individual user	13	Low	Charaan, Rohit Jacob George

b. Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	06 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	13 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

c. Reports from JIRA



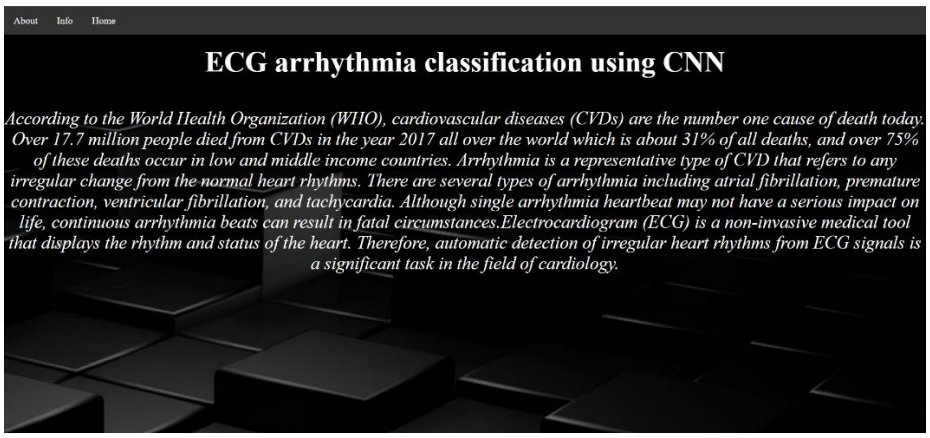
7. CODING AND SOLUTIONING

a. Feature 1

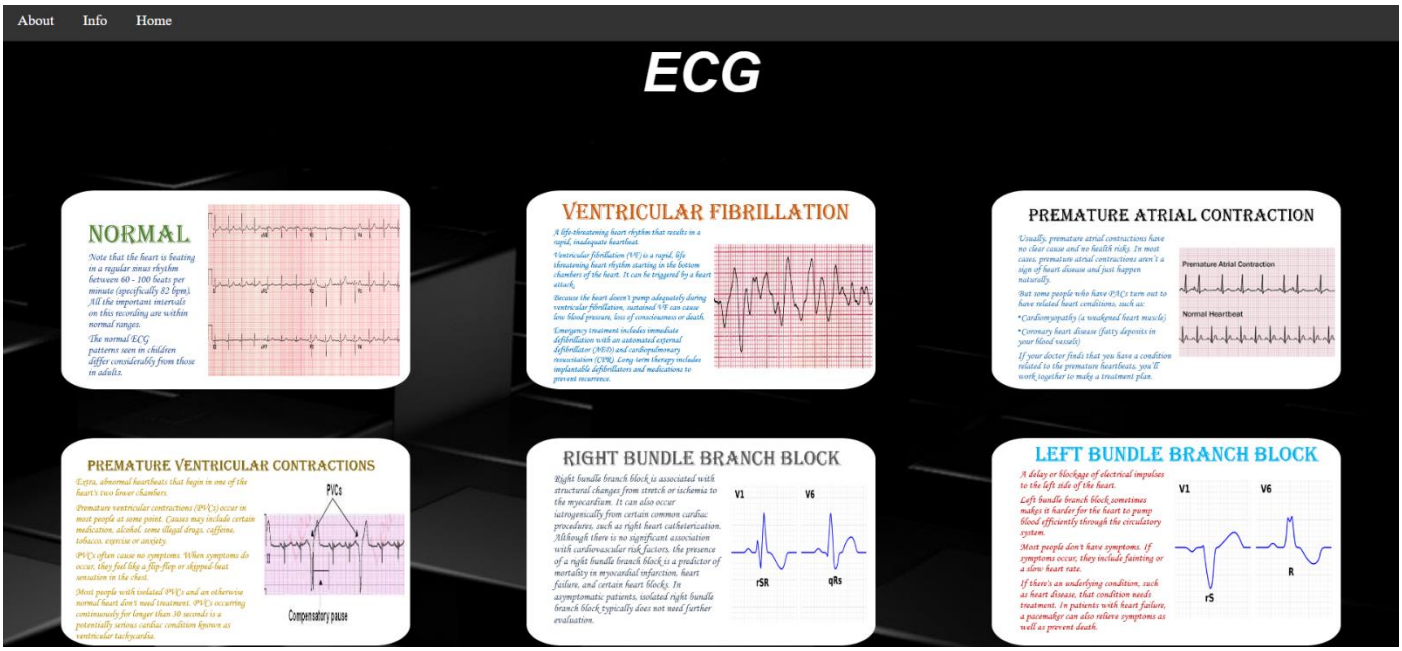
Interactive Web – based User Interface:

A web-application was created exclusively using Flask and Python. The web-application is made in such as way that it eliminated bloatware and keeps the web-page as lite and minimalist as possible, thereby increasing the optimization at the same time making it to serve it's purpose. The web-application consists of a dash-board that contains three tabs – About, Info and Home.

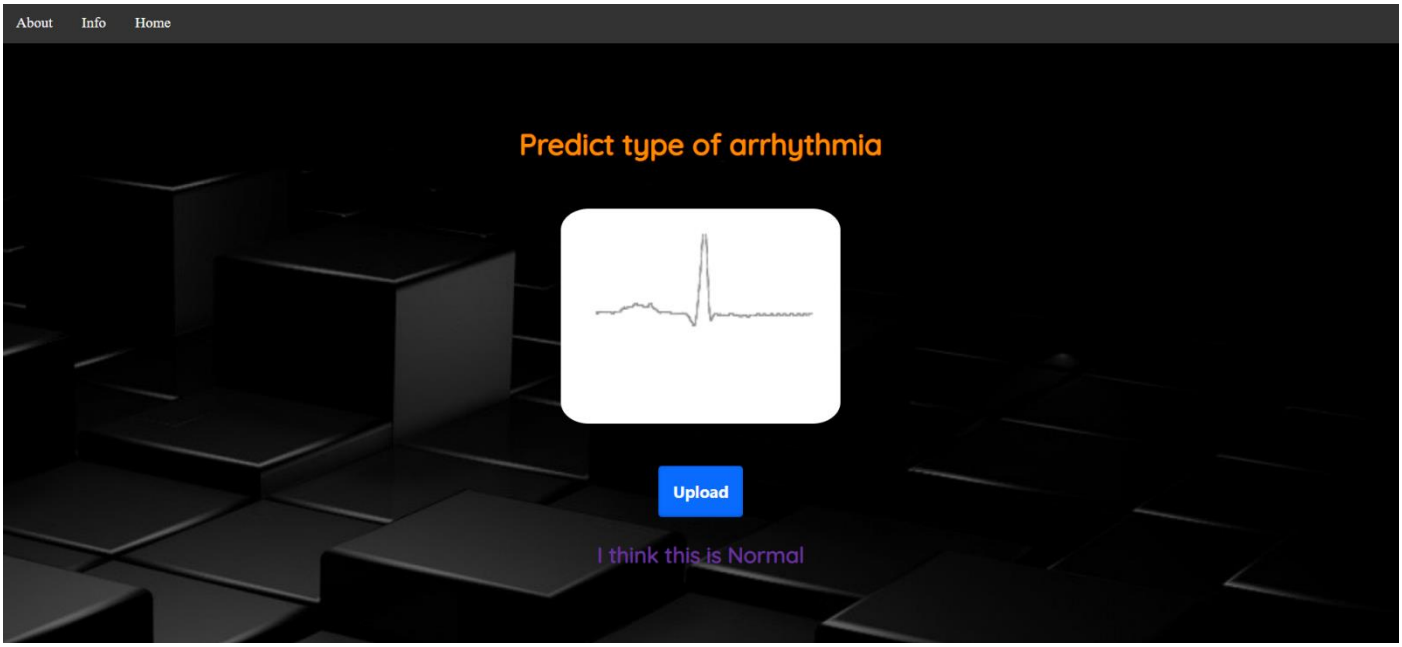
The About Page consists of basic information about the website and it's purpose:



The Info Page explains the different types of Arrhythmia present and gives a brief summary about each of them.

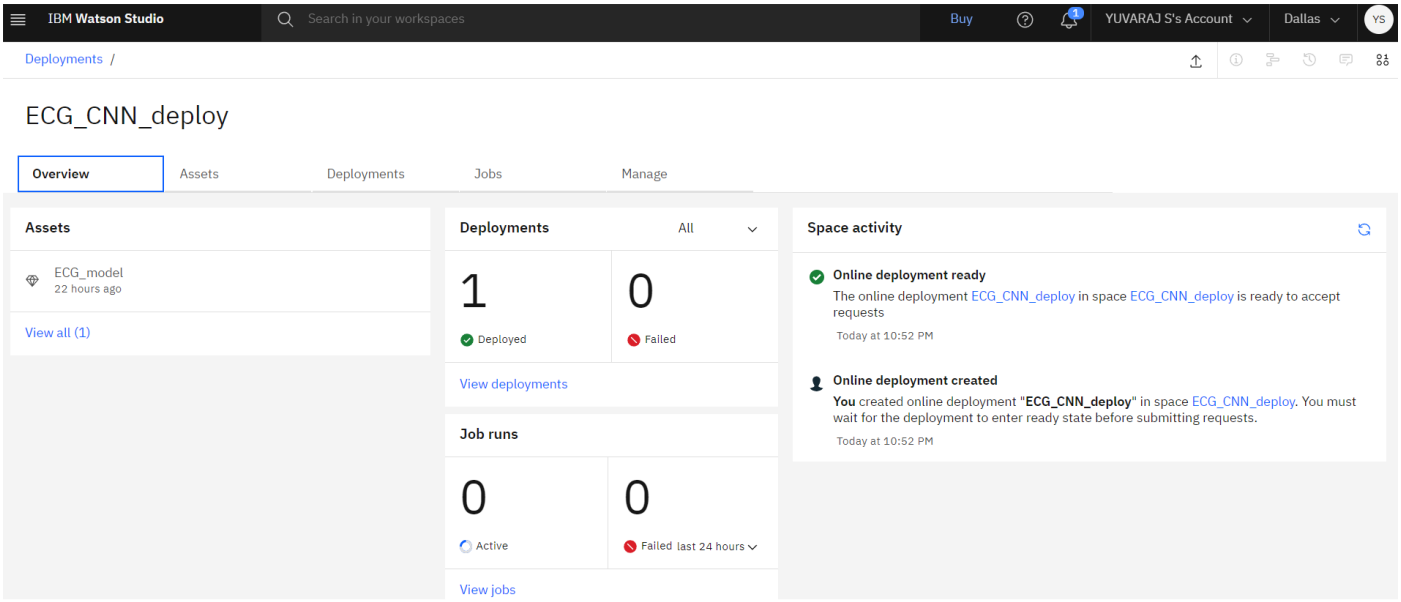


Finally, we have the Home Page where the user can upload the necessary ECG input(s) and get them classified.



b. <u>Real-Time prediction generator by CNN Model trained on IBM Watson:</u>

The developed CNN Model is integrated with the web-application by deploying it on IBM Watson platform. By doing so, we were able to generate prediction results in real-time.



8. TESTING:

a. Test Cases:

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite
HomePage_TC_OO1	UI	Home page	Verify user is able to see the home page screen	Enter the website
HomePage_TC_OO2	Functional	Home Page	Verify whether upload button is woking	Upload ECG image
Dashboard_TC_OO3	UI	Dashboard	Verify whether options are visible at Dashboard	Enter the website
AboutPage_TC_OO4	UI	About Page	Verify user is able to view About Page	Enter the website
InfoPage_TC_OO5	UI	Info Page	Verify user is able to view Info Page	Enter the website
InfoPage_TC_OO6	Functional	Info Page	Verify whether all the images are loading without aspect ratio differences	Enter Info Page
PredictionPage_TC_OO7	Functional	Prediction Page	Verify whether ECG image is getting predicted correctly	Upload ECG image
PredictionPage_TC_OO8	Functional	Prediction Page	Verify whether application provides warning for uploading incorrect data	Upload ECG image

WebHost_TC_OO9	Functional	Website	Verify whether website gets displayed when launched locally from server before hosting	Launch website from localhost

b. User Acceptance Testing:

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	4	0	0	0	4
Duplicate	0	0	0	0	0
External	2	0	0	0	2
Fixed	4	2	0	0	6
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	10	2	0	0	12

1. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	100	0	0	100
Client Application	37	0	2	35
Security	5	0	0	5
Outsource Shipping	3	0	0	3
Exception Reporting	15	0	0	15
Final Report Output	10	0	0	10
Version Control	6	0	0	6

9. RESULTS

a. Performance Metrics:

S.No.	Parameter	Screenshot / Values
1.	Data Responsiveness	Normal

2.	Amount Data to Rendered (DB2 Metrics)	No database used
3.	Utilization of Data Filters	No database used
4.	Effective User Story	7
5.	Descriptive Reports	3

10. ADVANTAGES AND DISADVANTAGES:

a. Advantages:
   i. The proposed model predicts Arrhythmia in images with a high accuracy rate of nearly 98%.
   ii. The early detection of Arrhythmia gives better understanding of disease causes, initiates therapeutic interventions and enables developing appropriate treatments.

b. Disadvantages:
   i. Not useful for identifying the different stages of Arrhythmia disease.
   ii. Not useful in monitoring motor symptoms.

11. CONCLUSION:

a. Cardiovascular disease is a major health problem in today's world.
b. The early diagnosis of cardiac arrhythmia highly relies on the ECG.
c. Unfortunately, the expert level of medical resources is rare, visually identify the ECG signal is challenging and time-consuming.
d. The advantages of the proposed CNN network have been put to evidence.
e. 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.

12. 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.

13. APPENDIX:

a. Source Code:

For CNN Model:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense,Convolution2D,MaxPooling2D,Flatten
```

In [2]:

```
import tensorflow
device_name = tensorflow.test.gpu_device_name()
if not device_name:
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
Found GPU at: /device:GPU:0
```

In [3]:

```
def main():
    os.environ['TF_GPU_THREAD_MODE'] = 'gpu_private'
```

In [4]:

```
train_datagen =
ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
```

In [5]:

```
test_datagen = ImageDataGenerator(rescale=1./255)
```

In [6]:

```
x_train =
train_datagen.flow_from_directory("data/train",target_size=(64,64),class_mode='categorical',bat
ch_size=32)
```

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

In [7]:

```
x_test =
test_datagen.flow_from_directory("data/test",target_size=(64,64),class_mode='categorical',batch
_size=32)
```

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

In [8]:

```
x_train.class_indices
```

Out[8]:

```
{'Left Bundle Branch Block': 0,
 'Normal': 1,
 'Premature Atrial Contraction': 2,
 'Premature Ventricular Contractions': 3,
 'Right Bundle Branch Block': 4,
 'Ventricular Fibrillation': 5}
```

In [9]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import load_model
from tensorflow.keras.layers import Dense,Convolution2D,MaxPooling2D,Flatten,Dropout
```

In [10]:

```
model=Sequential()
model.add(Convolution2D(32,(3,3),input_shape=(64,64,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Convolution2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(units=128,kernel_initializer="random_uniform",activation="relu"))
model.add(Dense(units=128,kernel_initializer="random_uniform",activation="relu"))
model.add(Dense(units=128,kernel_initializer="random_uniform",activation="relu"))
model.add(Dense(units=128,kernel_initializer="random_uniform",activation="relu"))
model.add(Dense(units=128,kernel_initializer="random_uniform",activation="relu"))
model.add(Dense(units = 6, kernel_initializer="random_uniform", activation = 'softmax'))
```

In [11]:

```
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 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
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
_________________________________________________________________
```

In [12]:

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

Out[12]:

```
480
```

In [13]:

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

```
Epoch 1/10
480/480 [==============================] - 170s 332ms/step - loss: 1.4493 - accuracy: 0.4780 -
val_loss: 1.6764 - val_accuracy: 0.3193
Epoch 2/10
480/480 [==============================] - 24s 50ms/step - loss: 1.1341 - accuracy: 0.5727 - va
l_loss: 1.3422 - val_accuracy: 0.5194
Epoch 3/10
480/480 [==============================] - 25s 52ms/step - loss: 0.5962 - accuracy: 0.7865 - va
l_loss: 1.1789 - val_accuracy: 0.6796
Epoch 4/10
480/480 [==============================] - 24s 50ms/step - loss: 0.3330 - accuracy: 0.8887 - va
l_loss: 0.7553 - val_accuracy: 0.7952
Epoch 5/10
480/480 [==============================] - 24s 50ms/step - loss: 0.2249 - accuracy: 0.9245 - va
l_loss: 0.5240 - val_accuracy: 0.8503
Epoch 6/10
480/480 [==============================] - 24s 50ms/step - loss: 0.1742 - accuracy: 0.9462 - va
l_loss: 0.5899 - val_accuracy: 0.8492
Epoch 7/10
480/480 [==============================] - 25s 52ms/step - loss: 0.1329 - accuracy: 0.9597 - va
l_loss: 0.4521 - val_accuracy: 0.8772
Epoch 8/10
```

```
480/480 [==============================] - 140s 292ms/step - loss: 0.1133 - accuracy: 0.9653 -
val_loss: 0.5241 - val_accuracy: 0.8495
Epoch 9/10
480/480 [==============================] - 27s 56ms/step - loss: 0.1000 - accuracy: 0.9679 - va
l_loss: 0.4809 - val_accuracy: 0.8876
Epoch 10/10
480/480 [==============================] - 25s 53ms/step - loss: 0.0909 - accuracy: 0.9714 - va
l_loss: 0.4966 - val_accuracy: 0.8624
```

In [14]:

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

In [15]:

```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
import numpy as np
```

In [16]:

```
model = load_model('ECG.h5')
img=image.load_img("data/test/Premature Ventricular
Contractions/fig_5656.png",target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
print(x.shape)
y=np.argmax(model.predict(x),axis=1)
index=['Left Bundle Branch Block', 'Normal', 'Premature Atrial Contraction', 'Premature
Ventricular Contractions', 'Right Bundle Branch Block','Ventricular Fibrillation']
index[y[0]]
```

```
(1, 64, 64, 3)
```

Out[16]:

```
'Normal'
```

For ECG Server:

```
From flask import Flask
from flask import request
from flask import make_response

import numpy as np
import os

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
staticLoad = False #Useful during development
formatError = "Incorrect format"

import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

model = load_model('ECG.h5')
app = Flask(__name__)

def getOptAndType(name):
    type = name[name.find('.') + 1:]
    if type == "css" or type == "html":
        opt = 'r'
    else:
        opt = 'rb'
```

```
    return (opt, type)

staticAssets = {}
assetsDir = 'assets'
for file in os.listdir(assetsDir):
    opt, type = getOptAndType(file)
    staticAssets[file] = open(assetsDir + '/' + file, opt).read()

def createNoResourceResponse():
    res = make_response()
    res.status = "Resource not found"
    res.status_code = 404
    return res

@app.route('/')
def index():
    if staticLoad:
        indexPage = staticAssets['index.html']
    else:
        indexPage = open(assetsDir + '/' + 'index.html').read()
    return indexPage

@app.route('/about')
def about():
    return serveFile("about.html")

@app.route('/info')
def info():
    return serveFile("info.html")

@app.route('/<name>')
def serveFile(name):
    if staticLoad and name not in staticAssets.keys():
        return createNoResourceResponse()
    if not staticLoad and name not in os.listdir(assetsDir):
        return createNoResourceResponse()

    opt, type = getOptAndType(name)
    if staticLoad:
        file = staticAssets[name]
    else:
        file = open(assetsDir + '/' + name, opt).read()
    res = make_response(file)
    ext = {'html': 'text/html', 'css': 'text/css', 'js': 'text/javascript', 'jpg': 'image/jpg',
'png': 'image/png', 'ico': 'image/ico'}
    res.content_type = ext[type]

    return res

@app.post('/image')
```

```
def predictImage():
    imgFile = request.get_data(cache = False)
    try:
        img = tf.io.decode_image(imgFile)
    except:
        return formatError
    imgRGB = img
    if len(imgRGB.shape) != 3 or (imgRGB.shape[2] != 1 and imgRGB.shape[2] != 3):
        return formatError

    if imgRGB.shape[2] == 1:
        imgRGB = tf.image.grayscale_to_rgb(img)

    x = tf.image.resize(imgRGB, [64,64]).numpy()
    x = np.expand_dims(x,axis=0)
    y = np.argmax(model.predict(x),axis=1)
    index = ['Left Bundle Branch Block', 'Normal', 'Premature Atrial Contraction', 'Premature
Ventricular Contractions', 'Right Bundle Branch Block','Ventricular Fibrillation']
    return index[y[0]]


if __name__ == "__main__":
    app.run(debug = False)
```

b. Github and Project Link Demo:

i. Github link: https://github.com/IBM-EPBL/IBM-Project-473-1658303160

ii. Project Link Demo: https://youtu.be/zlYGC4122R4