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

PROJECT REPORT

Submitted by

Abiisek M	193002003
Girish Babu B	193002032
Kamya H	193002045
Keerthana Reddy R	193002047

EEC ELECTIVE

Department of Electronics and Communication Engineering

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

Rajiv Gandhi Salai (OMR), Kalavakkam – 603 110

ODD SEM 2022 – 2023

TEAM ID: PNT2022TMID52880

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

SYMBOLS	ABBREVIATION
DFD	Data Flow Diagram
ECG	Electro Cardio Gram
CNN	Convolutional Neural Networks
DF-WKNN	Difference, Weighted k-nearest neighbor
WT	Wavelet Transform
ICA	Independent Component Analysis

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:

Making a correct diagnosis of the type of arrhythmia is conducive to the prevention and treatment of heart disease, abnormal heart rhythm may be caused by underlying heart disease, and in some circumstances can even give rise to life-threatening. In today's era, the computer technology develops by leaps and bounds, which provides technical conditions for identifying types of arrhythmia. For instance, when determining the type of arrhythmia, electrocardiogram (ECG) data can be automatically classified according to machine learning algorithms. Identifying arrhythmia as early as possible helps the patient in choosing appropriate treatment. Classification of ECG arrhythmia with high accuracy is a challenging problem. Arrhythmia classification requires pre-processing ECG Signal, extraction of features, and optimization of the features and classification of arrhythmia.

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

Previous studies on arrhythmia were used to diagnose the abnormally fast, slow, or irregular heart rhythm through ECG (Electrocardiogram), which is one of the biological signals. ECG has the form of P-QRS-T wave, and many studies have been done to extract the features of QRS-complex and R-R interval. However, in the conventional method, the P-QRS- T wave must be accurately detected, and the feature value is extracted through the P-QRS-T wave. If an error occurs in the peak detection or feature extraction process, the accuracy becomes very low. Therefore, in this paper, we implement a system that can perform PVC (Premature Ventricular Contraction) and PAC (Premature Atrial Contraction) classification byusing P-QRS-T peak value without feature extraction process using deep neural network. The parameters were updated for PVC and PAC classification in the learning process usingP-QRS-T peak without feature value. As a result of the performance evaluation, we could confirm higher accuracy than the previous studies and omit the process of feature extraction, and the time required for the pre-processing process to construct the input data set is relatively reduced.

2.2 REFERENCES

[1] 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.

In this paper Difference, Weighted k-nearest neighbor (DF-WKNN) classifier is presented to recognize unbalanced UCI cardiac arrhythmia data from the UCI arrhythmia data set. This method incorporates the Scorrelation of K neighbours into the classification.

[2] Ullah, A., Anwar, S.M., Bilal, M. and Mehmood, R.M., 2020. Classification of arrhythmia by using deep learning with 2-D ECG spectral image representation. *Remote Sensing*, *12*(10), p.1685.

In this study, they proposed a two-dimensional (2-D) convolutional neural network (CNN) model for the classification of ECG signals into eight classes; namely, normal beat, premature ventricular contraction beat, paced beat, right bundle branch block beat, left bundle branch block beatpremature , atrial contraction beat, ventricular flutter wave beat, and ventricular escape beat.The one-dimensional ECG time series signals are transformed into 2-D spectrograms through a short-time Fourier transform. The 2-D CNN model consisting of four convolutional layers and four pooling layers is designed for extracting robust features and testing was done.

[3] 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.

This paper presents survey issues concerned in ECG denoising, feature extraction, optimization and classification. Mainly methods used to analyze the

performance.

[4] Avanzato, Roberta, and Francesco Beritelli. "Automatic ECG diagnosis using convolutional neural network." *Electronics* 9, no. 6 (2020): 951.

For the "atrial premature beat" class, ECG segments were correctly classified 100% of the time. Finally, for the "premature ventricular contraction" class, ECG segments were correctly classified 96% of the time. In total, there was an average classification accuracy of 98.33%. The sensitivity (SNS) and the specificity (SPC) were, respectively, 98.33% and 98.35%.

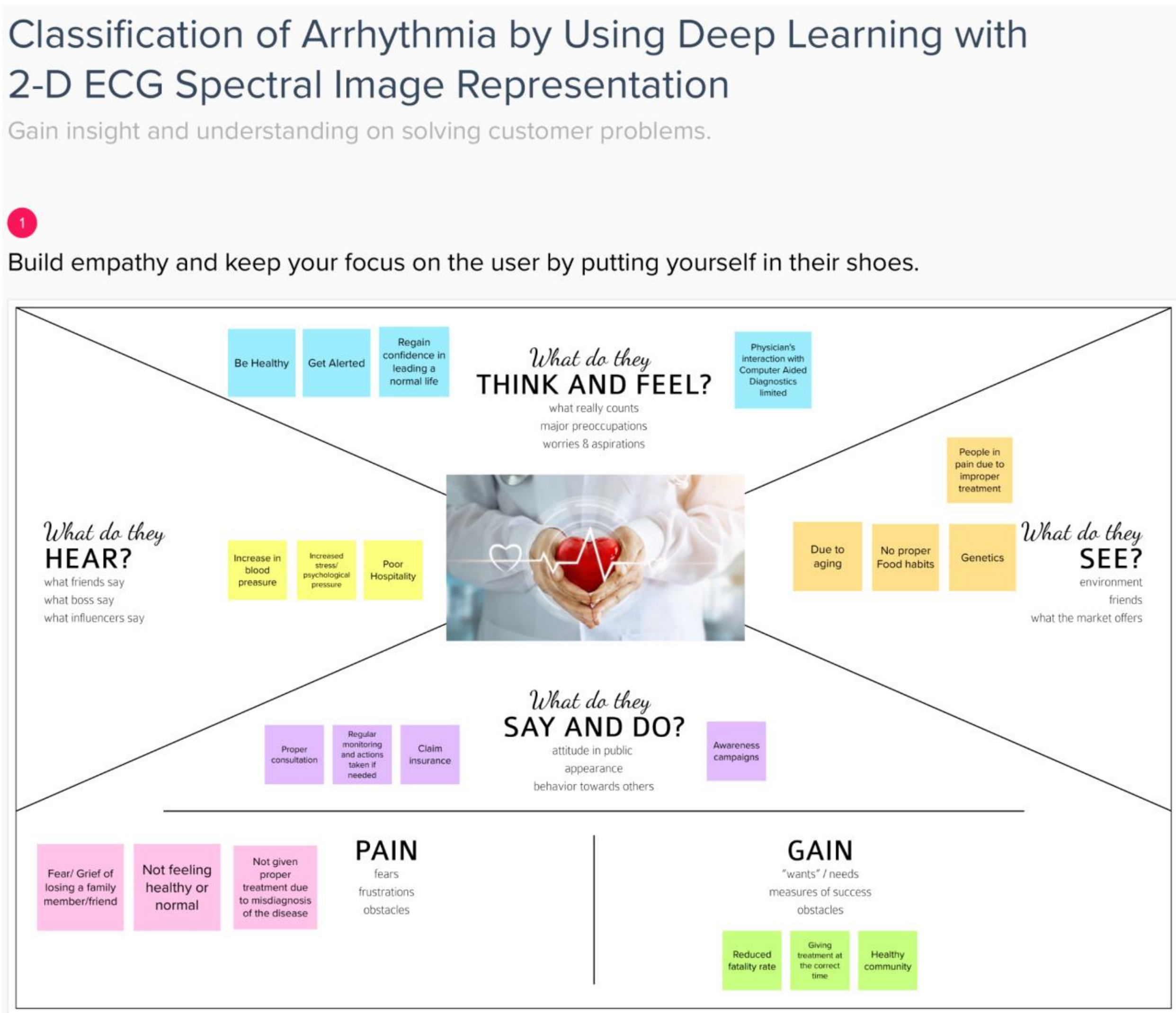
[5] C. Ye, M. T. Coimbra and B. V. K. Vijaya Kumar, "Arrhythmia detection and classification using morphological and dynamic features of ECG signals," 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology, 2010, pp. 1918-1921, doi: 10.1109/IEMBS.2010.5627645.
In this paper, a new approach is proposed for arrhythmia classification based on a combination of morphological and dynamic features. Wavelet Transform (WT) and Independent Component Analysis (ICA) are applied separately to each heartbeat to extract corresponding coefficients, which are categorized as 'morphological' features.

2.3 PROBLEM STATEMENT DEFINITION

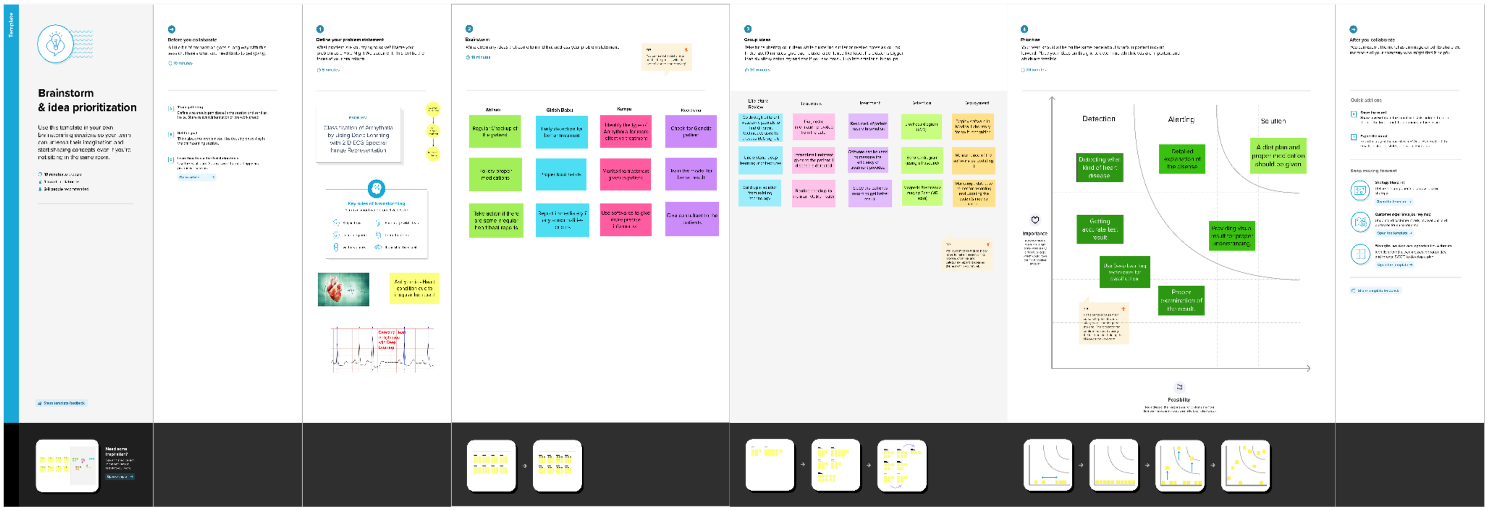
Arrhythmia is a representative type of CVD that refers to any irregular change from the normal heart rhythms, There are several types of arrhythmia including atrial fibrillation, premature contraction, ventricular fibrillation, and tachycardia. Although a single arrhythmia heartbeat may not have a serious 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 convolutional neural network (CNN), in which we classify ECG into seven categories, one being normal and the other side 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.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP:



3.2 BRAIN STORMING:



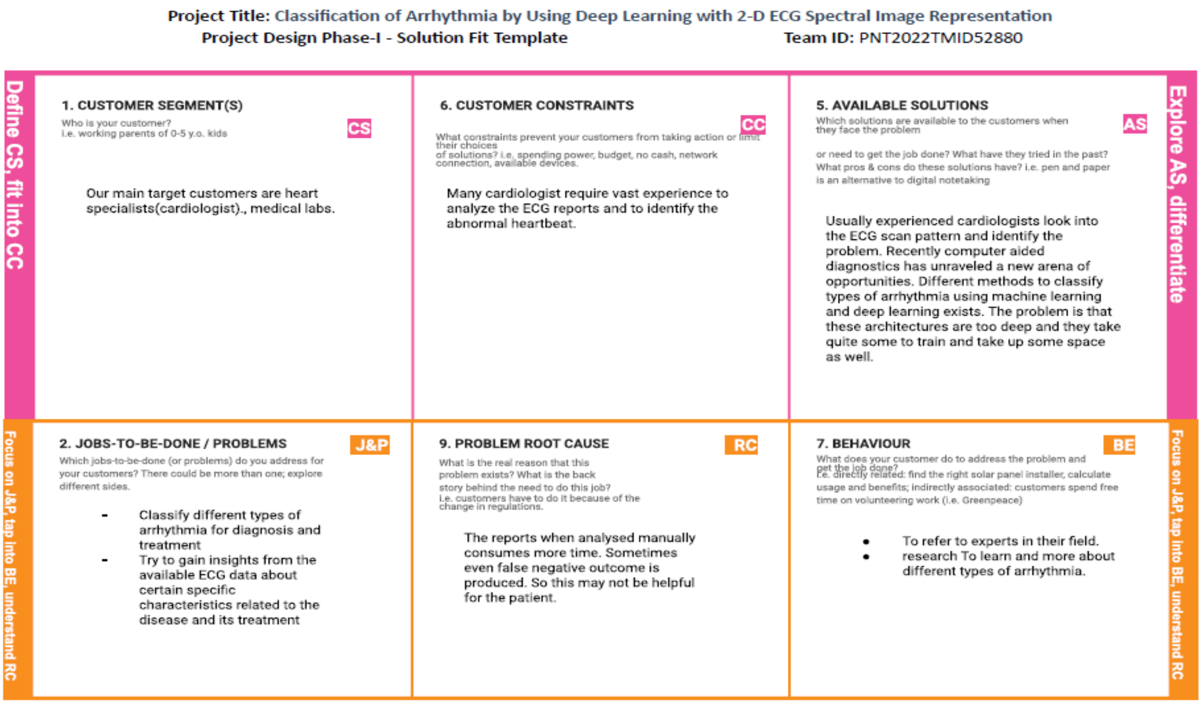
3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Classification of different types of Arrhythmia using deep learning and feature selection methods.
2.	Idea / Solution description	Arrhythmia is a problem with the rate or rhythm of the heartbeat. There are different types of arrhythmia, and it is usually identified by experts through ECG signals. This manual intervention takes quite some time and experience on the physician's side to understand the intricate patterns of the signal and classify it properly to provide

		appropriate treatment. In this project, we try to reduce the burden of the physician and build a CNN based deep learning model which takes the ECG reports as the input. CNN usually acquires excessive features when implemented. So, this causes the model more time to process the data and sometimes even false detection of the disease. When we use feature selection methods, the number of features used for training the model is drastically reduced. Hence, this improves the efficiency of the model and henceforth it helps the medical society. So, in this work we plan to extract a set of features from the CNN, pass it through different feature selection methods and finally classify it using machine learning classifiers like SVM, Random Forest, etc.
3.	Novelty / Uniqueness	Till now feature selection methodologies have been directly implemented on the ECG signal. Here we try to implement feature selection method in the features extracted by a deep neural network and improve the efficiency of the model.

4.	Social Impact / Customer Satisfaction	This when implemented in real time reduces the number of false detections, improves the efficiency and more importantly helps the medical society in various ways.
5.	Business Model (Revenue Model)	When developed as an application, this model can be used from any location and can be given as a paid subscription to the users.
6.	Scalability of the Solutio	Using different types of feature selection methodologies, we can observe various efficiency levels of those methods. This model can be implemented in a web interface/ web application mode, which various clinicians can access to classify the data as well as try to get more insights from the same.

3.4 PROPOSED SOLUTION FIT:



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

Project Design Phase-I - Solution Fit Template **Team ID:** PNT2022TMID52880

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

Our main target customers are heart specialists(cardiologist)., medical labs.

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

Many cardiologist require vast experience to analyze the ECG reports and to identify the abnormal heartbeat.

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Usually experienced cardiologists look into the ECG scan pattern and identify the problem. Recently computer aided diagnostics has unraveled a new arena of opportunities. Different methods to classify types of arrhythmia using machine learning and deep learning exists. The problem is that these architectures are too deep and they take quite some to train and take up some space as well.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

- Classify different types of arrhythmia for diagnosis and treatment
- Try to gain insights from the available ECG data about certain specific characteristics related to the disease and its treatment

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations.

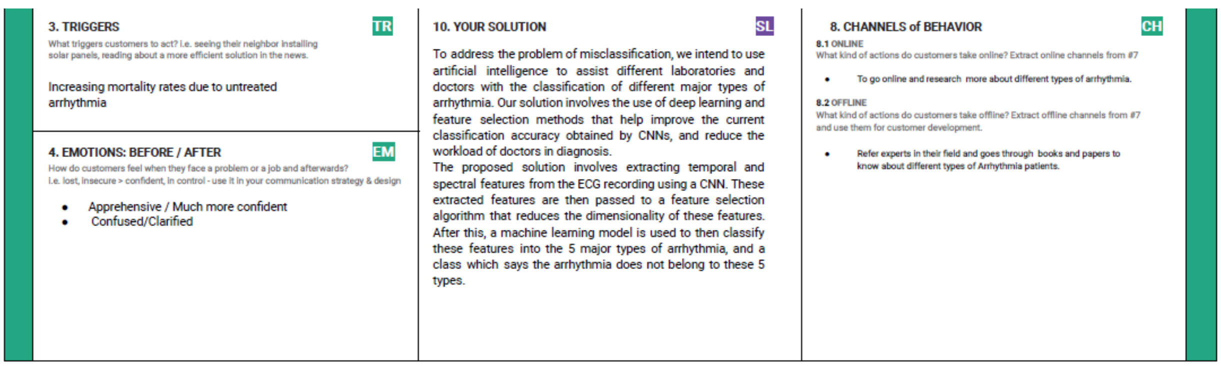
The reports when analysed manually consumes more time. Sometimes even false negative outcome is produced. So this may not be helpful for the patient.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

- To refer to experts in their field.
- research To learn and more about different types of arrhythmia.

Focus on J&P, tap into BE, understand RC



3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbor installing solar panels, reading about a more efficient solution in the news.

Increasing mortality rates due to untreated arrhythmia

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards? i.e. lost, insecure > confident, in control - use it in your communication strategy & design

- Apprehensive / Much more confident
- Confused/Clarified

10. YOUR SOLUTION SL

To address the problem of misclassification, we intend to use artificial intelligence to assist different laboratories and doctors with the classification of different major types of arrhythmia. Our solution involves the use of deep learning and feature selection methods that help improve the current classification accuracy obtained by CNNs, and reduce the workload of doctors in diagnosis.
The proposed solution involves extracting temporal and spectral features from the ECG recording using a CNN. These extracted features are then passed to a feature selection algorithm that reduces the dimensionality of these features. After this, a machine learning model is used to then classify these features into the 5 major types of arrhythmia, and a class which says the arrhythmia does not belong to these 5 types.

8. CHANNELS of BEHAVIOR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

- To go online and research more about different types of arrhythmia.

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

- Refer experts in their field and goes through books and papers to know about different types of Arrhythmia patients.

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS:

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 interface and input	Check your profile and choose your file Upload image in jpeg/ png format
FR-4	Data processing	Evaluating the model using test data
FIR-5	Report generation	Image will be shown as output

4.2 NON-FUNCTIONAL REQUIREMENTS:

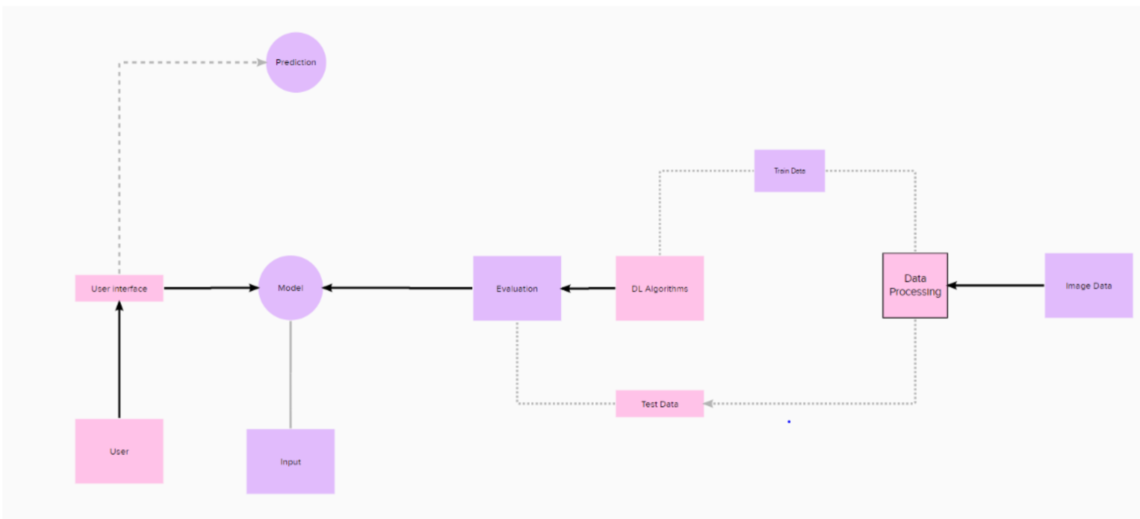
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Long term heart rhythm monitoring system with Deep learning model.
NFR-2	**Security**	Data acquired for training and testing must be protect by security models.
NFR-3	**Reliability**	Error free. Must be very accurate for every scenario.
NFR-4	**Performance**	· Accurate classification · Inclusive model
NFR-5	**Availability**	Cloud based web application will improve the availability of solution built.
NFR-6	**Scalability**	Accuracy must not be affected. Fast and quick classification will improve the scalability of the solution.

5. PROJECT DESIGN:

5.1 DATA FLOW DIAGRAMS:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



5.2 SOLUTION & TECHNICAL ARCHITECTURE:

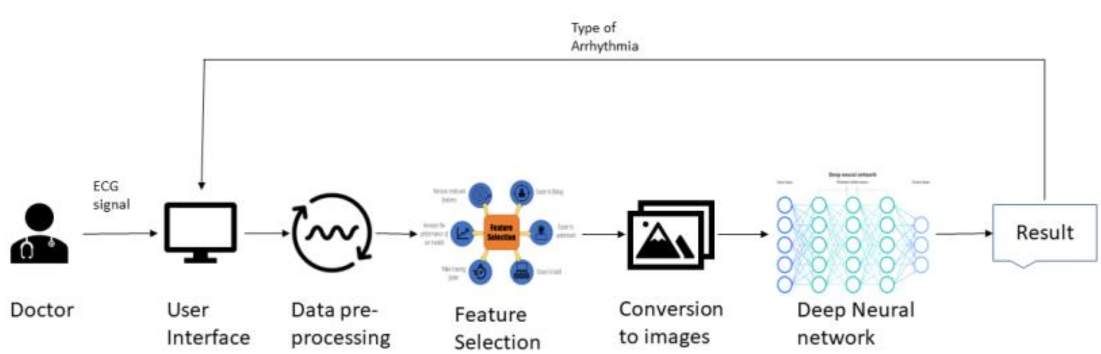


Fig. 5.1 Solution Architecture

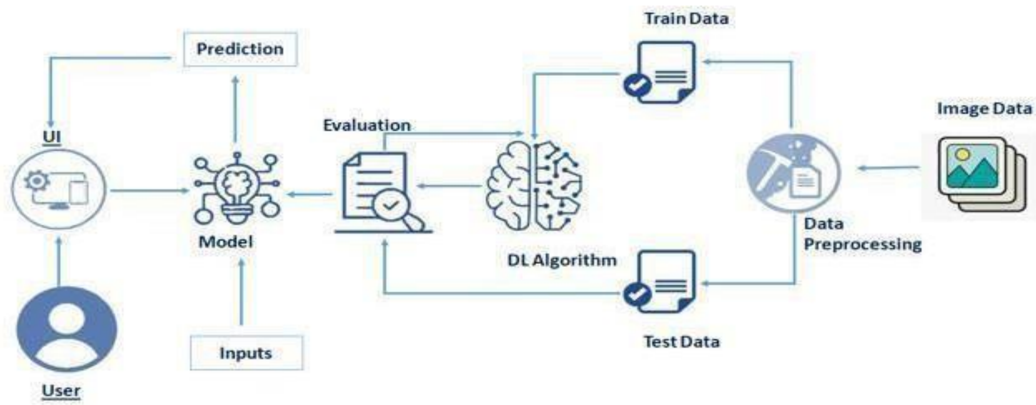


Fig 5.2 Technical Architecture

TABLE-1 : COMPONENTS & TECHNOLOGIES:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with User interface to upload image	Anaconda, Jupyter notebooks, Spyder, Python.
2.	Model analyses	Once model analyses the uploaded image, theprediction is showcased on the UI	Kaggle.com, data.gov, UCI
3.	Data collection	Create the dataset	Python, Keras, Numpy
4.	Data Preprocessing-1	Import the ImageDataGenerator library	Python, Keras, Numpy
5.	Data Preprocessing-2	Configure ImageDataGenerator class	Python, Numpy, Keras
6.	Data Preprocessing-3	Apply ImageDataGenerator	Python, Numpy, Keras

		functionality to Trainset and Testset	
7.	Model Building-1	Import the model building libraries and Initializing The model	Python, Numpy, Keras,
8.	Model Building-2	Adding layers and configure	Python, Numpy, Keras
9.	Model Building-3	Training and testing the model, Optimize and save the model	Python, Numpy, Keras
10.	Application Building	Purpose of create an HTML file and BuildingPython code	HTML, Python
11.	Train the model on IBM	CNN Development and integrate it with the flask Application	IBM Watson

TABLE-2: APPLICATION CHARACTERISTICS:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Open source software is that by which the source code or the base code is usually available for modification or enhancement.	Flask(Python)

2.	Security Implementations	By placing a filtration barrier between the targetedserver and the attacker, the WAF is able to protectagainst attacks like cross site forgery, cross site scripting and SQL injection.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Does not affect the performance even though usedby many users.	Technology used
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Technology used
5.	Performance	Design consideration for the performance of theapplication (number of requests per sec, use of Cache, use of CDN's) etc.	Technology used

5.3 USER STORIES:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
			makes it easy to see	through dashboard		
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	Conformation	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2

		USN-4	As a user, I can register for the application through Gmail	Register access with gmail	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	Access the dashboard with email and passwords	High	Sprint-1

User Type	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Acceptance criteria**	**Priority**	**Release**
	Available		official website	Which prompts reminders about your heart rhythms		
	Login	USN-3	As a Healthcare staff, I can log in to the application by entering the right credentials like phone number and password		Low	Sprint-1

	Dashboard	USN-4	Enables healthcare professionals to access important patient statistics in real-time		High	Sprint-1

User Type	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Acceptance criteria**	**Priority**	**Release**
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have	I can receive confirmation email & click confirm	High	Sprint-1

			registered for the application			
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	Access gmail and dashboard	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	Enter and access with password and email	High	Sprint-1
	Dashboard	USN-6	As a web user grid format which	Seeing information	low	Sprint-2

6. PROJECT PLANNING & SCHEDULING

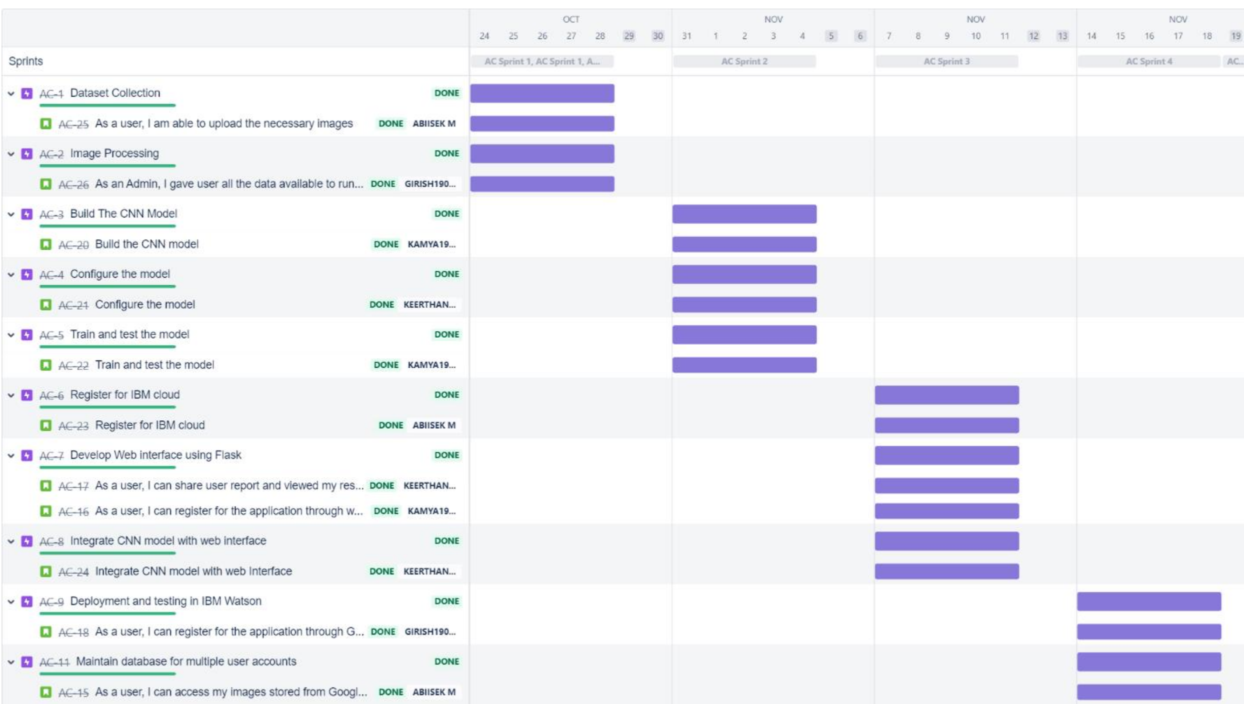
6.1 SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-4	Model Deployment	USN-10	The final model is deployed for the end user to use it.	2	High	Abiisek M Kamya H Girish babu B Keerthana Reddy R
Sprint-4	Storage	USN-9	As a user, I can access my images stored from Google drive.	1	Medium	Abiisek M Kamya H Girish babu B Keerthana Reddy R
Sprint-3	Registration	USN-8	As a user, I can upload my images to be analyzed in the website.	1	Low	Abiisek M Girish Babu B
Sprint-3	Registration	USN-7	As a user, I can access the application through website.	1	Medium	Keerthana Reddy R Kamya H
Sprint-2	Testing & Evaluation	USN-6	As a developer, we tested the trained model using the provided dataset andmodel will be evaluated for accurate results.	2	High	Kamya H Abiisek M
Sprint-2	Train the model	USN-5	As a developer, the dataset will be uploaded and trained by developed algorithm.	2	High	Girish Babu B Keerthana Reddy R
Sprint-2	Initialize the Model	USN-4	Initializing the Image recognition model	1	Low	Abiisek M Kamya H
Sprint-1	Registration	USN-3	As a user, I am able to upload the necessary images.	2	High	Keerthana Reddy R
Sprint-1	Dashboard	USN-2	As an Admin, I gave user all the data available to run the test.	2	High	Abiisek M Girish babu B
Sprint-1	Dashboard	USN-1	As an Admin, I can manage the Arrhythmia Classification details. If normal or abnormal the UI model will share the result for the dashboard.	2	High	Abiisek M Girish babu B

6.2 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	5 Days	24 Oct 2022	28 Oct 2022	20	28 Oct 2022
Sprint-2	20	5 Days	31 Oct 2022	04 Nov 2022	20	04 Nov 2022
Sprint-3	20	5 Days	07 Nov 2022	11 Nov 2022	20	11 Nov 2022
Sprint-4	20	5 Days	14 Nov 2022	18 Nov 2022	20	18 Nov 2022

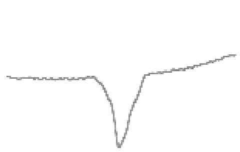
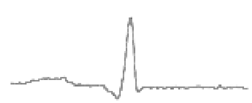
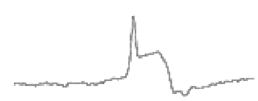
6.3 REPORTS FROM JIRA



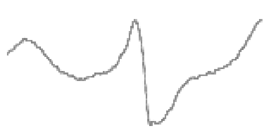
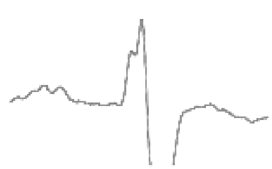
7. CODING AND SOLUTIONING

To implement our proposed solution, we coded the proposed methodology using the Python language. We decided to use python as it is an open-source language and contains multiple libraries and frameworks to implement our methodology with ease. We decided to use the tensorflow and keras framework to code our CNN model.

The first process of our methodology is to download the dataset. A link was provided in the IBM resource dashboard to the dataset (Link). The dataset provided contained 2 main directories called the train and test directories. Within these 2 directories, there were 6 sub folders named: Left Bundle Branch Block, Normal, Premature Atrial Contraction, Premature Ventricular Contraction, Right Bundle Branch Block, Ventricular Fibrillation. Each of these subfolders contained segmented images from the MIT-BIH arrhythmia database. The segmented images represented a heartbeat cycle. The difference in the patterns from the normal ones can be clearly seen in case of these 5 types of arrhythmias. The images are shown below in the figure.

Left Bundle Branch Block Normal Premature Atrial Contraction


Premature Ventricular Contraction Right Bundle Branch Block Ventricular Fibrillation

These are the input images to our deep learning model. There are a total of 15,341 training images and 6825 testing images available in the dataset. There is a fair bias as the number of normal images is high but this just mimics a real-world problem where most of the ECG's received are of normal people compared to the less percentage in our population with the possibility of arrhythmia.

7.1 FEATURE SELECTION:

After going through the dataset, different image augmentations were considered for bettering the performance of our deep learning model. As ECG images represent a pattern, rotation is not a suitable image augmentation technique that can be used. The augmentations techniques we used were: rescaling, shear range, horizontal flip and zoom range. These were found to work good with ECG images and are the ones that made the most sense. These techniques were then implemented using ImageDataGenerator function available in the keras module.

At first, a transformer was created with the function:

```
In [7]: train_datagen = ImageDataGenerator(rescale=1./255, shear_range = 0.2, zoom_range=0.2, horizontal_flip = True)
        test_datagen = ImageDataGenerator(rescale=1./255)
```

Then it was fit to the images present in the directories:

```
In [8]: x_train = train_datagen.flow_from_directory(directory = train_path, target_size=(64,64), batch_size=32, class_mode= "categorical")
        x_test = train_datagen.flow_from_directory(directory =test_path, target_size=(64,64), batch_size=32, class_mode= "categorical")

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

Thus, the images are pre-processed using keras augmentation techniques.

After the image augmentation techniques are done, it is time to construct the deep learning model we have proposed. The deep learning technique we are using here is a Convolutional Neural Network (CNN). The CNN is a type of deep learning architecture that is used extensively in image identification, classification and recognition problems. A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

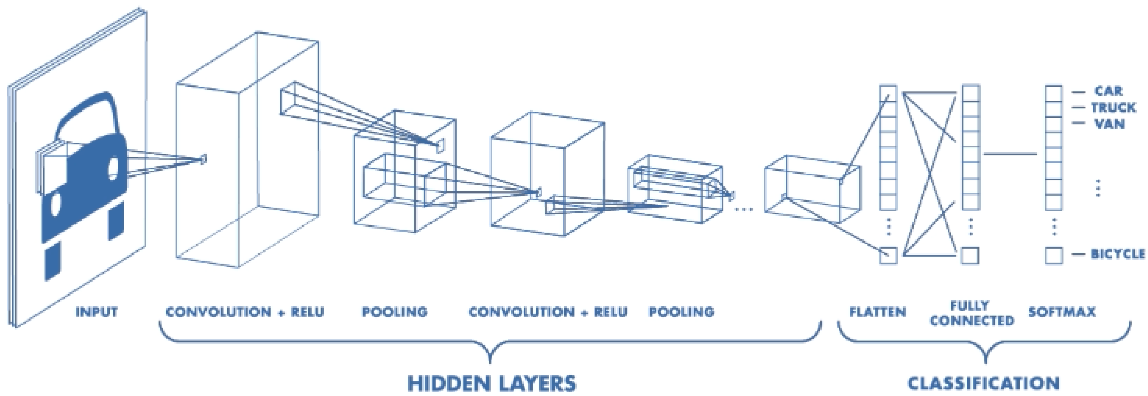


Fig: CNN Architecture

Convolution Layer: The convolution layer is the core building block of the CNN. It carries the main portion of the network's computational load.

This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels. During the forward pass, the kernel slides across the height and width of the image-producing the image representation of that receptive region. This produces a two-dimensional representation of the image known as an activation map that gives the response of the kernel at each spatial position of the image. The sliding size of the kernel is called a stride.

Pooling layer: The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually. The most popular process is max pooling, which reports the maximum output from the neighbourhood.

Fully Connected Layer: Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN. This is why it can be computed as usual by a matrix multiplication followed by a bias effect. The FC layer helps to map the representation between the input and the output.

So, using these background information we built our CNN models in keras using Sequential().

```
In [9]: model=Sequential()
```

The model layers were defined as:

```
In [10]: 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'))
```

A summary of our CNN model is shown below. It consists of a total of 2,11,078 trainable parameters.

```
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 (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
_________________________________________________________________
```

While training the deep learning model, we need to modify each epoch's weights and minimize the loss function. An optimizer is a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rate. Thus, it helps in reducing the overall loss and improve the accuracy. The problem of choosing the right weights for the model is a daunting task, as a deep learning model generally consists of millions of parameters. As a result, a suitable optimization algorithm must be chosen in order to obtain better results. In this proposed method, we have used an Adam optimizer. The name adam is derived from adaptive moment estimation. This optimization algorithm is a further extension of stochastic gradient descent to update network weights during training. Adam optimizer updates the learning rate for each network weight individually. The adam optimizer has several benefits, due to which it is used widely. It is adapted as a benchmark for deep learning papers and recommended as a default optimization algorithm. Moreover, the algorithm is straightforward to implement, has faster running time, low memory requirements, and requires less tuning than any other optimization algorithm.

Since our model has a multiclass classification, it is important to monitor the loss obtained in each category to ensure proper change in the weights with the adam optimizer. Hence, we have used categorial crossentropy as our loss function while training the model. Initially, the most important metric we wanted to monitor was the accuracy of both the training and validation set, so these are the 3 parameters passed to the model.compile() function in keras.

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

Finally, the model was trained using the model.fit() function in keras for a total of 5 epochs.

```
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>
```

The training of the model takes up substantial amount of time due to the large database we have provided as our input, as well as the high number of trainable parameters in the CNN. Once the model is trained, we save the last epoch as a file to access the model later for our website implementation.

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

Once the model is saved, it is time for testing the model, which will be discussed in the next section.

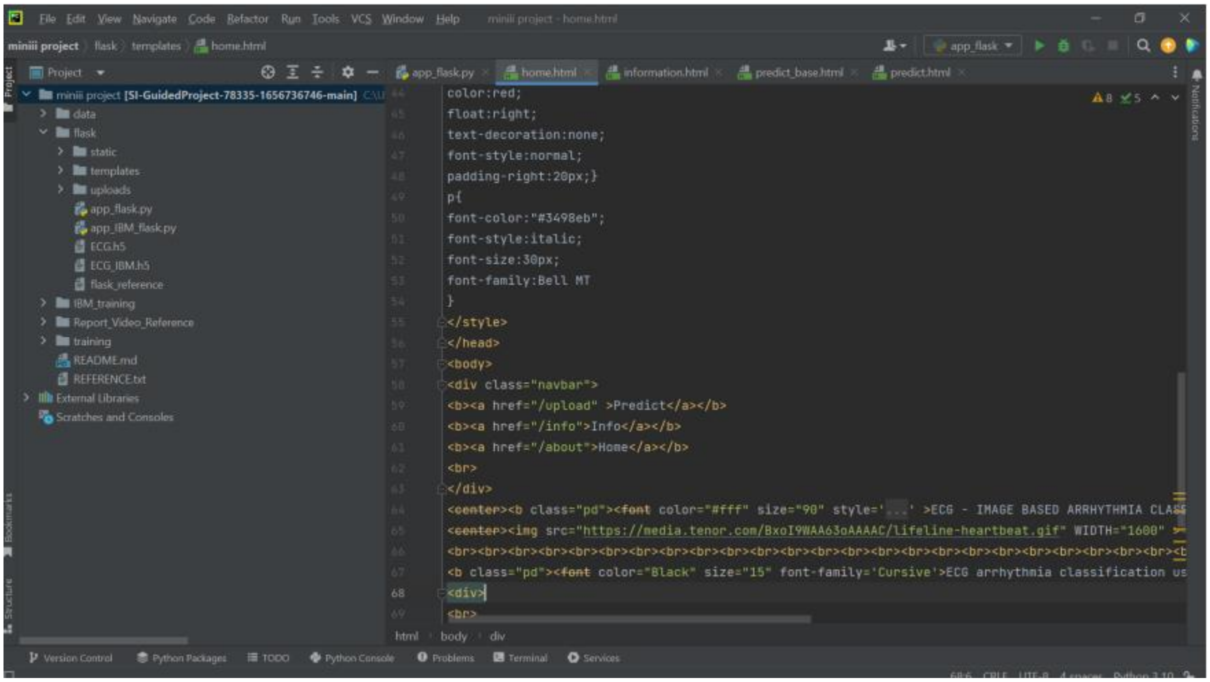
7.2 FLASK APP:

After the deep learning model is trained and saved, it is time to implement the said model in a user friendly and appealing manner. For this, we create a web application using the python library flask, and use html, css, javascript to create the webpages.

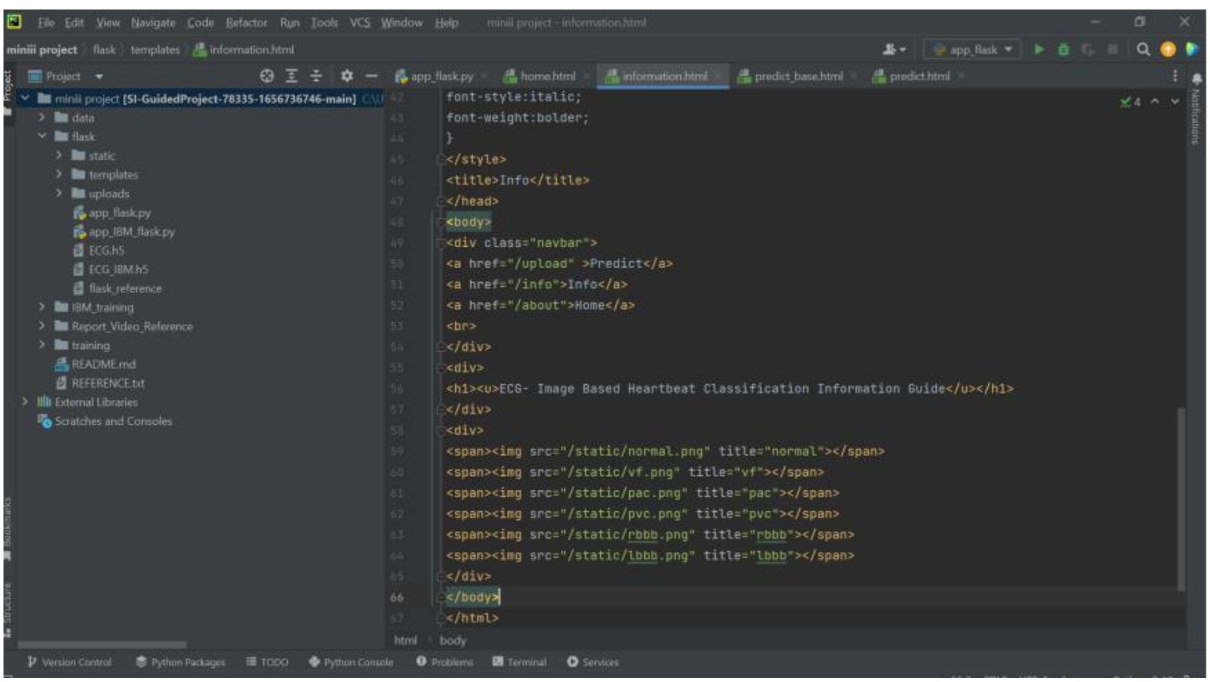
The webpage is designed in a user-friendly manner so that anyone on the internet is capable of using the web application. The idea for our website was as follows. We created a home page, which gives the users basic information about Arrhythmia. The second page contains information about the different types of arrhythmias we are predicting using our model. The last page contains the option for choosing the test file and predicting the type of arrhythmia found using the given input image.

The initial design for the webpages are created using html.

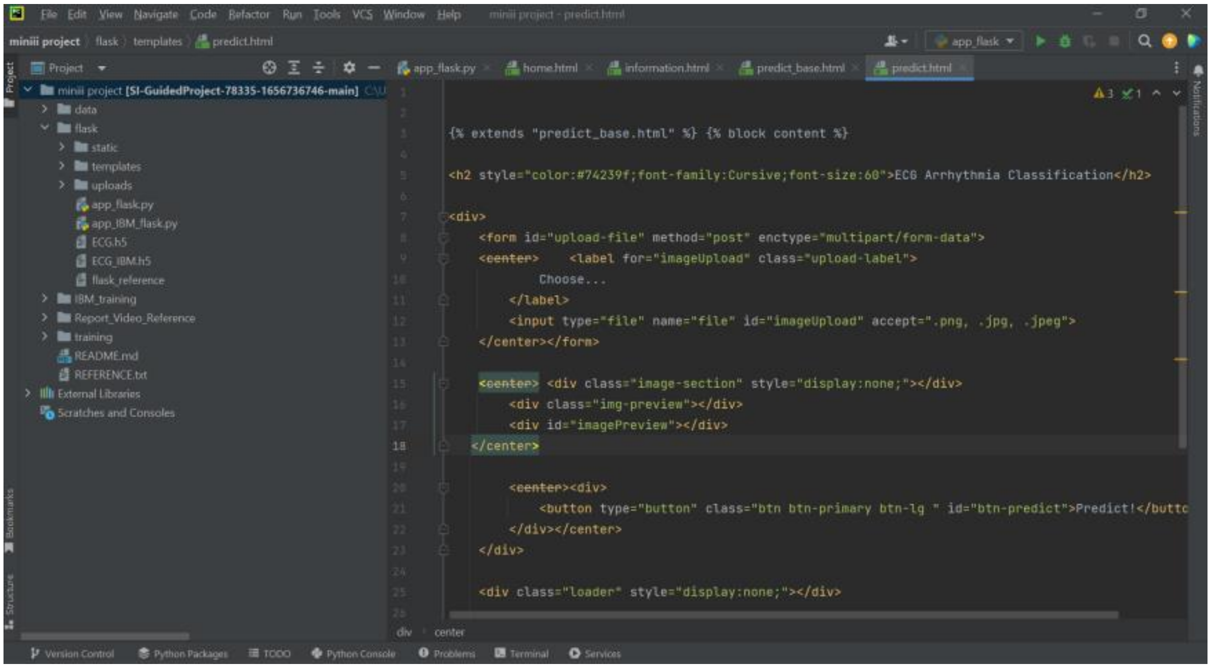
The HTML file used to build the Home.html:



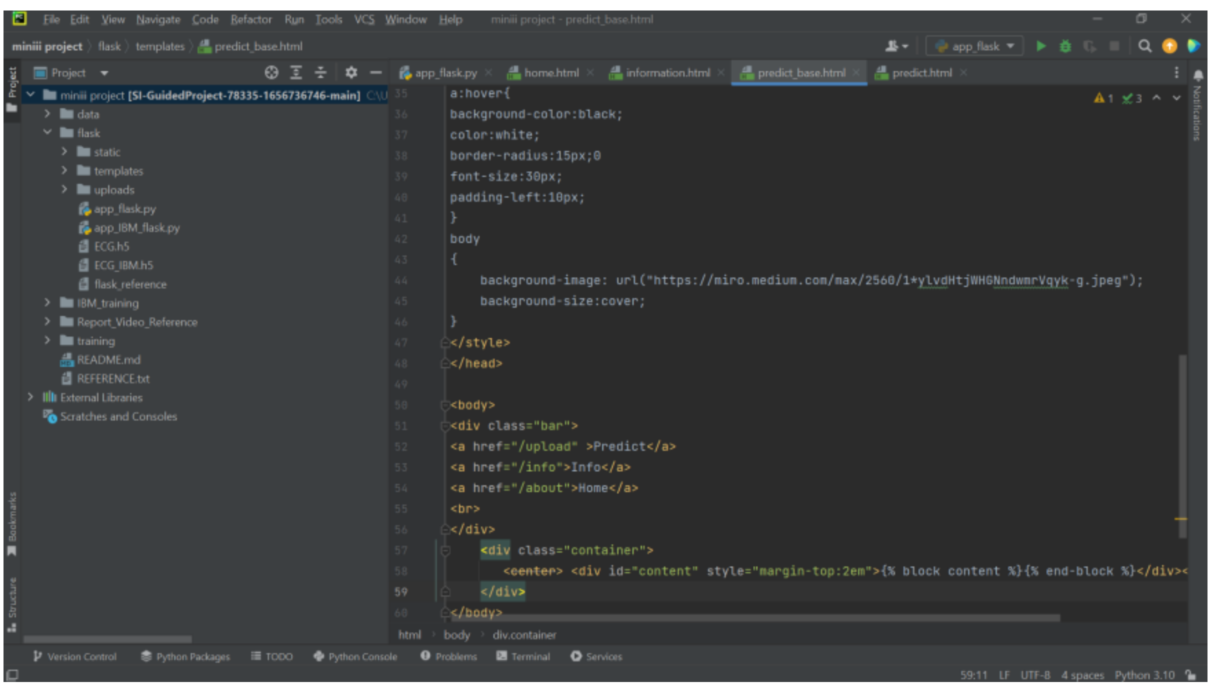
The HTML file used to build the information.html:



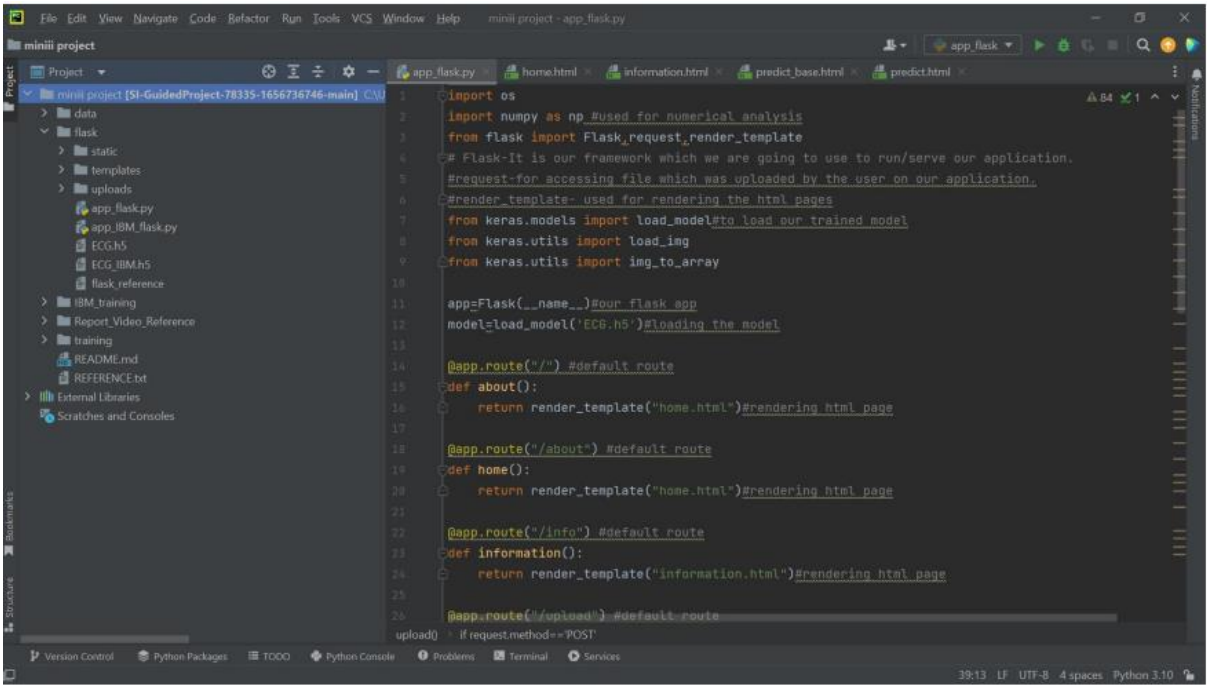
The HTML file used to build the predict.html:



The HTML file used to build the predict_base.html:

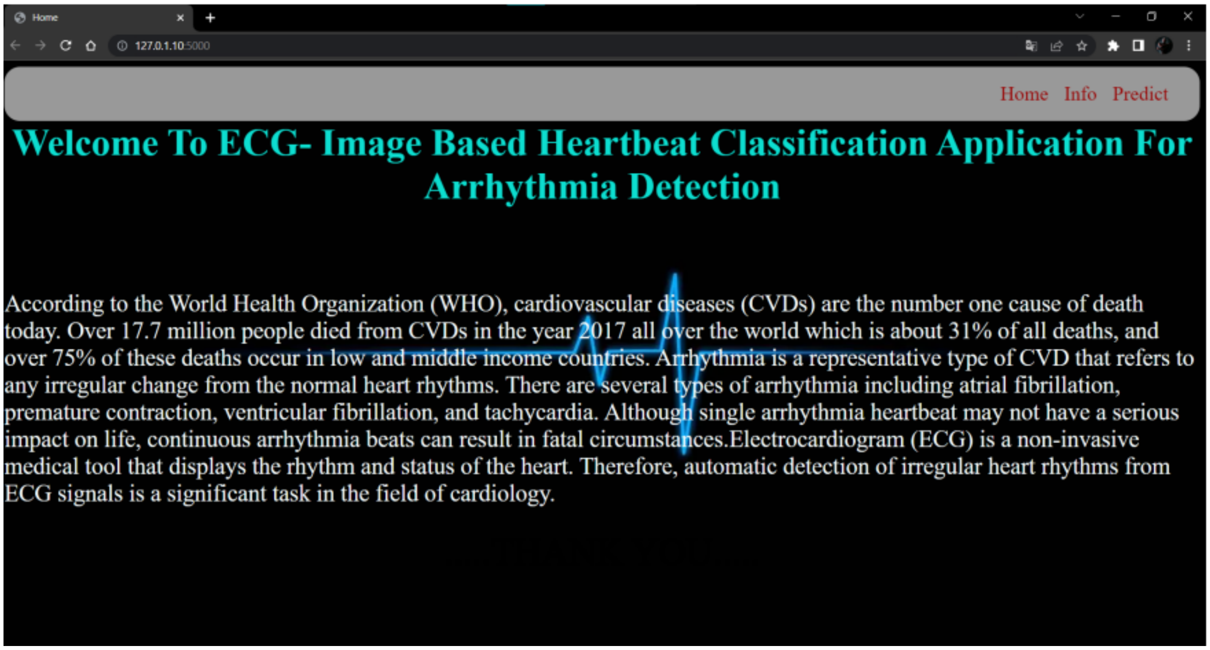


The python flask app where the deep learning model is called to predict the type of arrhythmia found in the image:

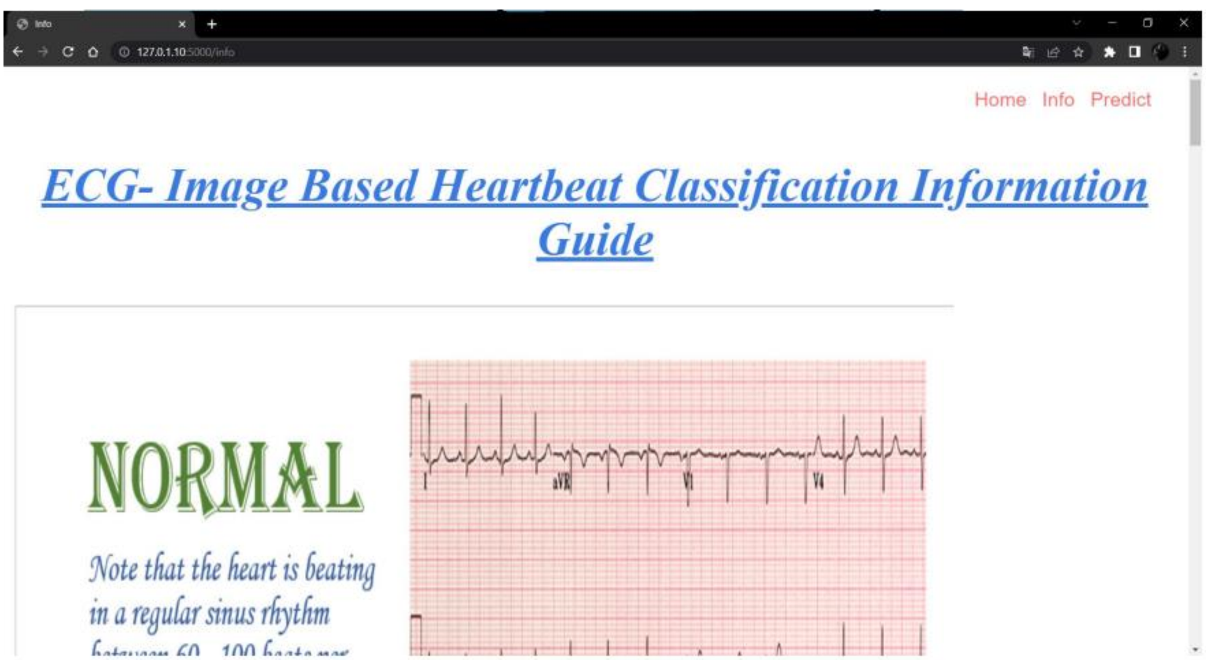


An example implementation of our web app is shown below:

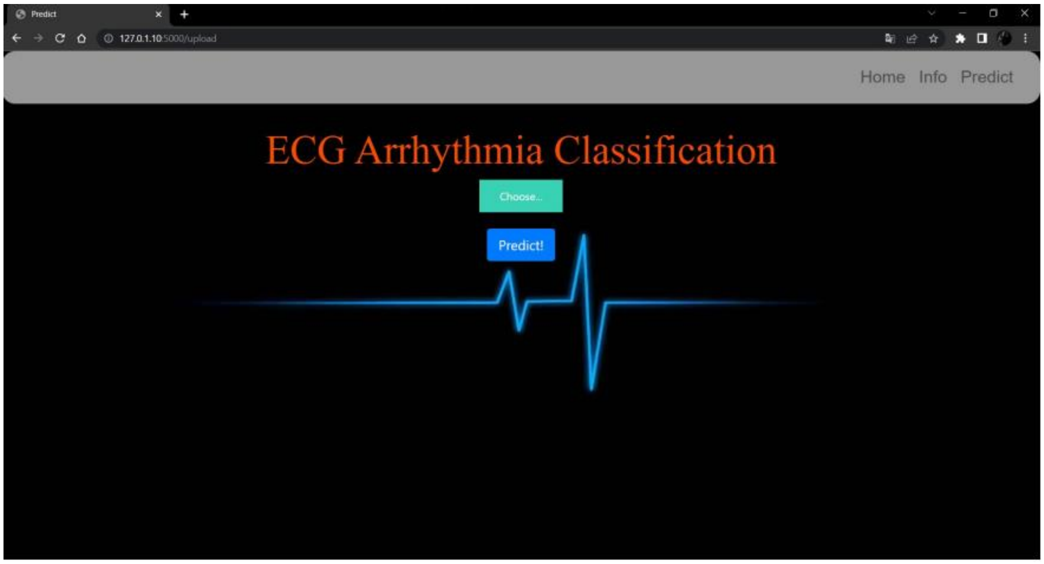
Running the Web application:



Info tab to allow the user to study about several classes of Arrythmia:



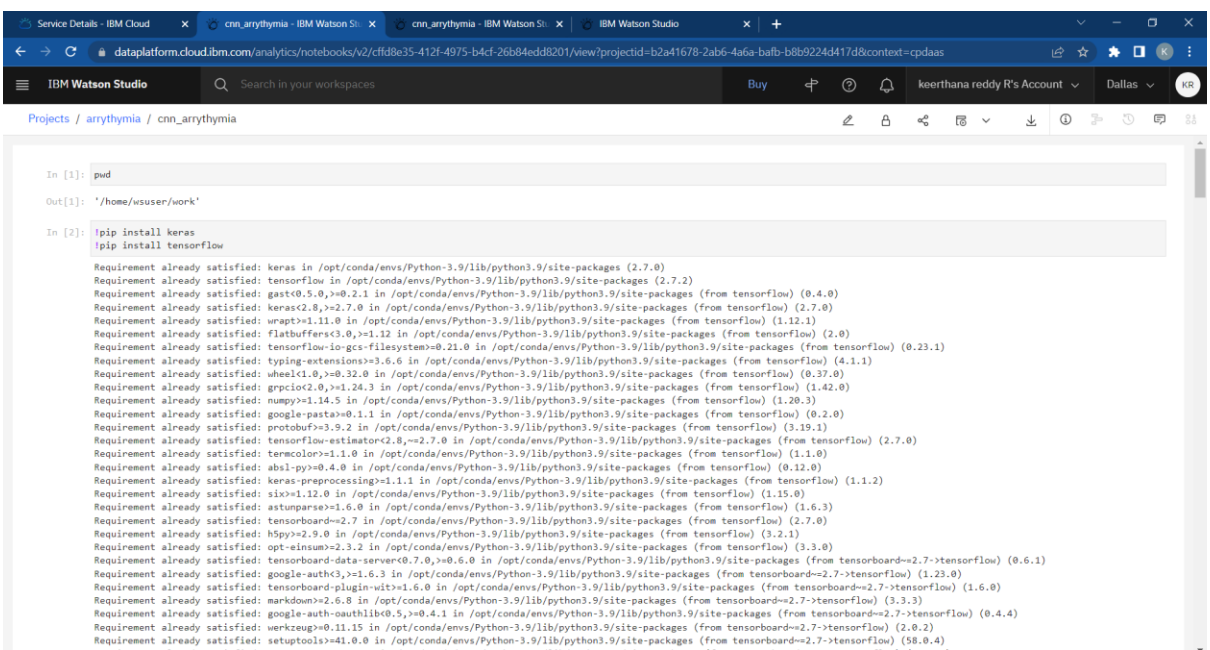
Predicting window:

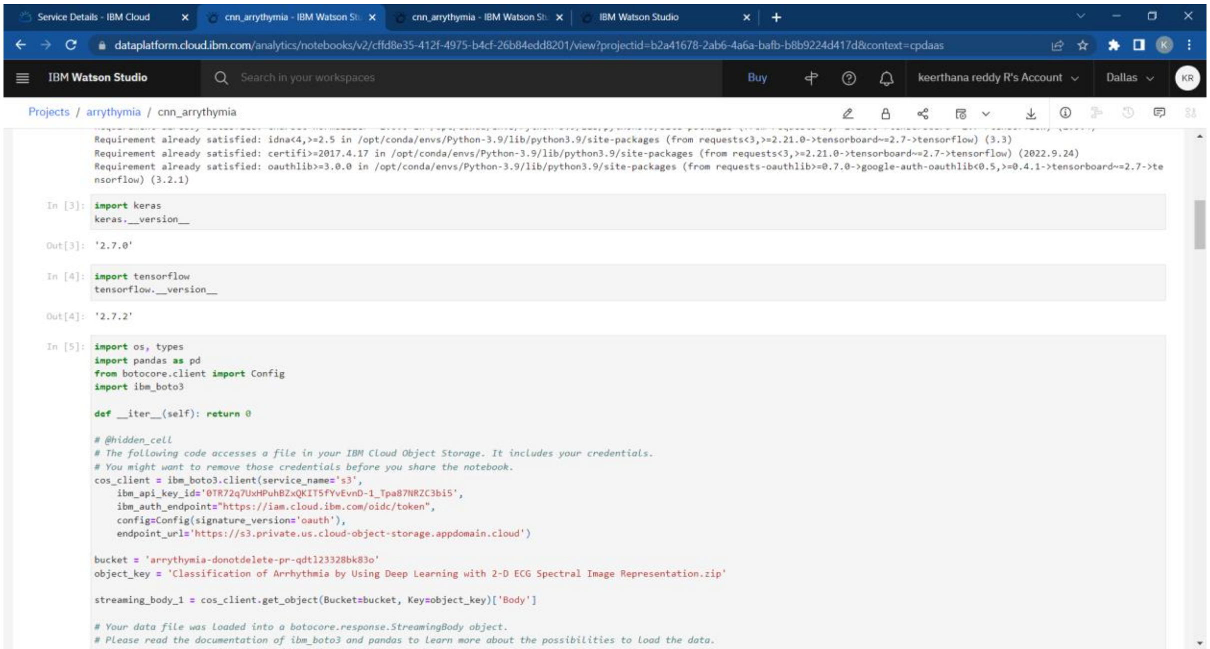


The solution was implemented also using IBM Watson. The images from the implementation can be found here:

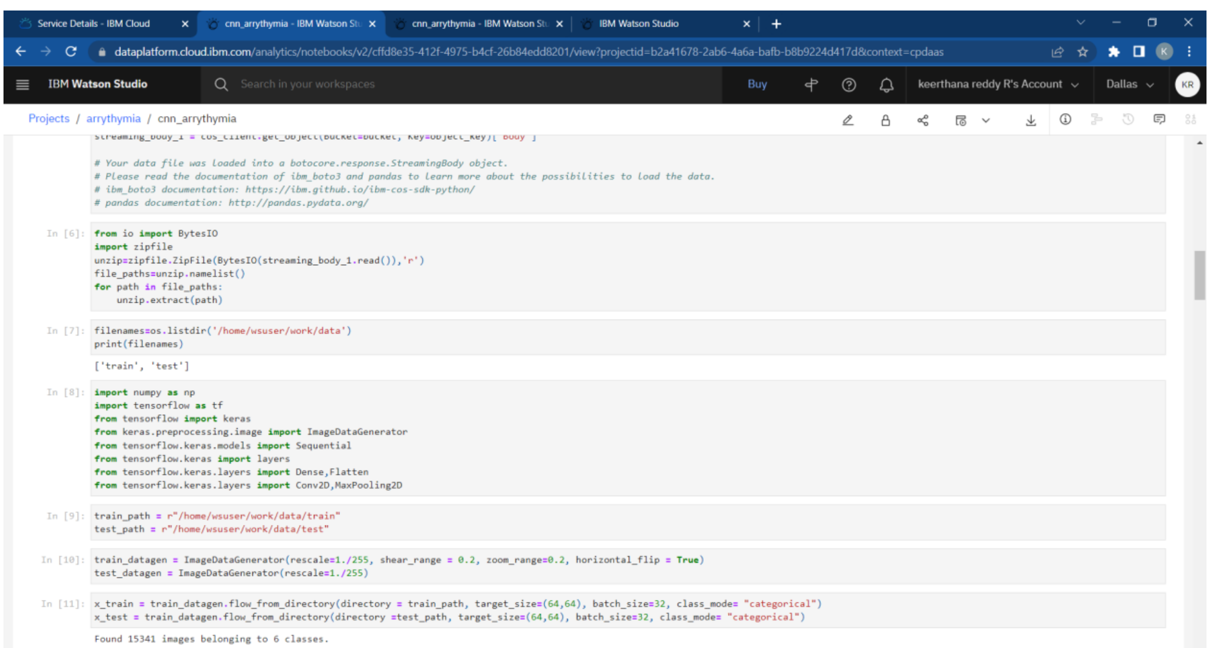
7.3 MODEL

LOADING DATA AND IMAGE PREPROCESSING:



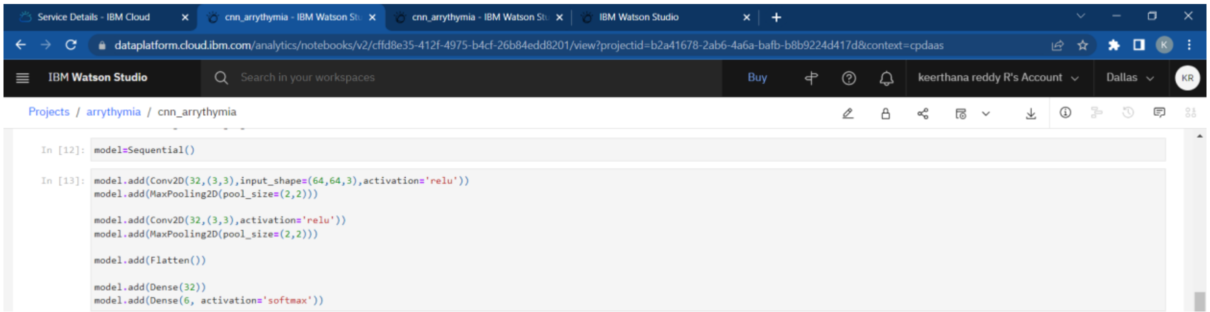


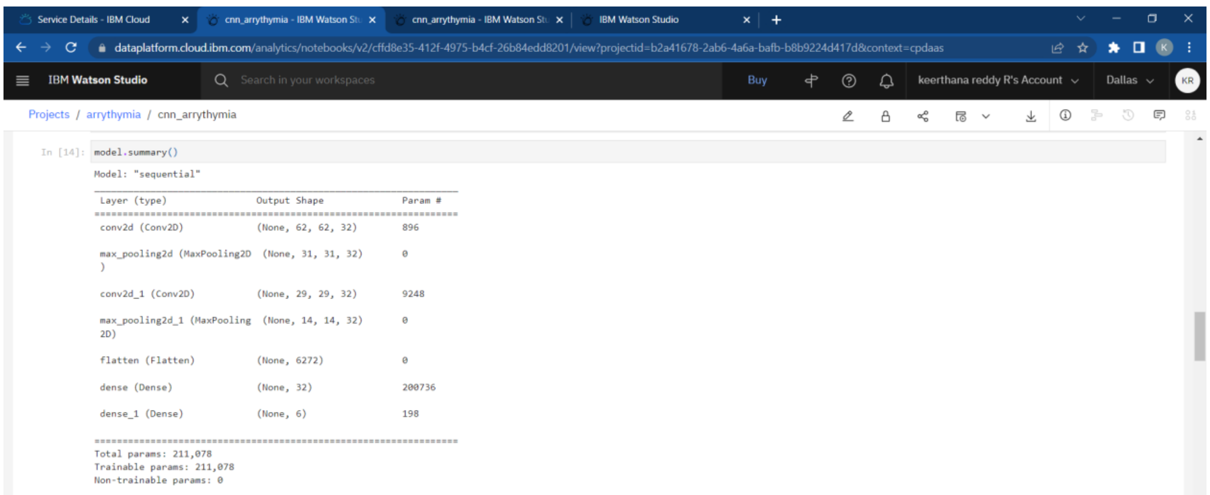
```
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensorboard~=2.7->tensorflow) (3.3)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensorboard~=2.7->tensorflow) (2022.9.24)
Requirement already satisfied: oauthlib>=3.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.7->te
nsorflow) (3.2.1)

In [3]: import keras
        keras.__version__

Out[3]: '2.7.0'

In [4]: import tensorflow
        tensorflow.__version__

Out[4]: '2.7.2'

In [5]: import os, types
        import pandas as pd
        from botocore.client import Config
        import ibm_boto3

        def __iter__(self): return 0

        # @hidden_cell
        # The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
        # You might want to remove those credentials before you share the notebook.
        cos_client = ibm_boto3.client(service_name='s3',
            ibm_api_key_id='0TR72q7UxHPuhBZxQKIT5fYvEvnD-1_Tpa87NRZC3bi5',
            ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
            config=Config(signature_version='oauth'),
            endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

        bucket = 'arrythymia-donotdelete-pr-qdtl23328bk83o'
        object_key = 'Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation.zip'

        streaming_body_1 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

        # Your data file was loaded into a botocore.response.StreamingBody object.
        # Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
```



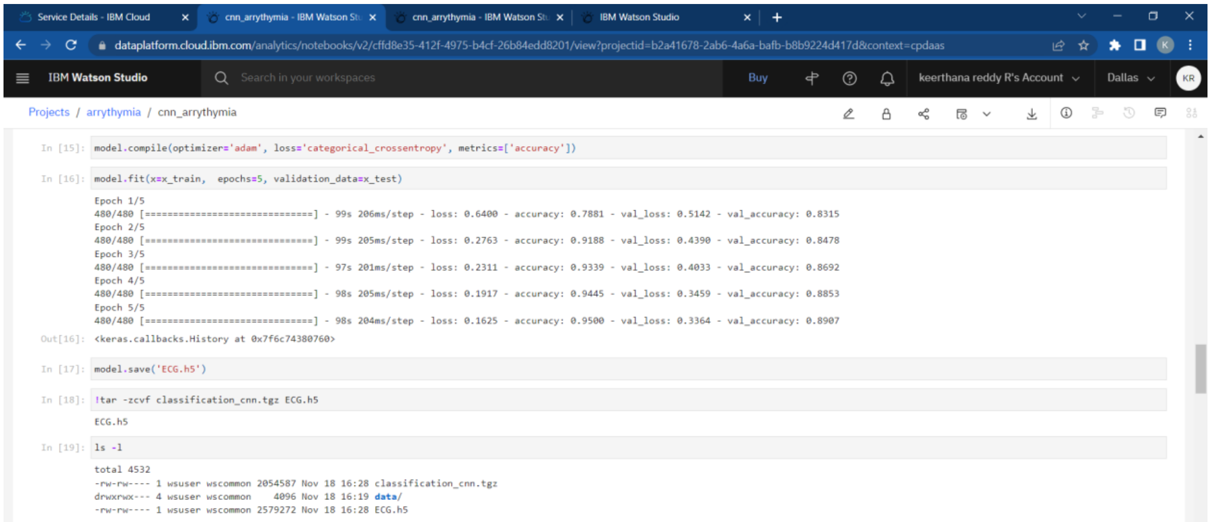
```
        # Your data file was loaded into a botocore.response.StreamingBody object.
        # Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
        # ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
        # pandas documentation: http://pandas.pydata.org/

In [6]: from io import BytesIO
        import zipfile
        unzip=zipfile.ZipFile(BytesIO(streaming_body_1.read()),'r')
        file_paths=unzip.namelist()
        for path in file_paths:
            unzip.extract(path)

In [7]: filenames=os.listdir('/home/wsuser/work/data')
        print(filenames)

        ['train', 'test']

In [8]: import numpy as np
        import tensorflow as tf
        from tensorflow import keras
        from keras.preprocessing.image import ImageDataGenerator
        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

In [9]: train_path = r"/home/wsuser/work/data/train"
        test_path = r"/home/wsuser/work/data/test"

In [10]: train_datagen = ImageDataGenerator(rescale=1./255, shear_range = 0.2, zoom_range=0.2, horizontal_flip = True)
         test_datagen = ImageDataGenerator(rescale=1./255)

In [11]: x_train = train_datagen.flow_from_directory(directory = train_path, target_size=(64,64), batch_size=32, class_mode= "categorical")
         x_test = train_datagen.flow_from_directory(directory =test_path, target_size=(64,64), batch_size=32, class_mode= "categorical")

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

MODEL BUILDING:

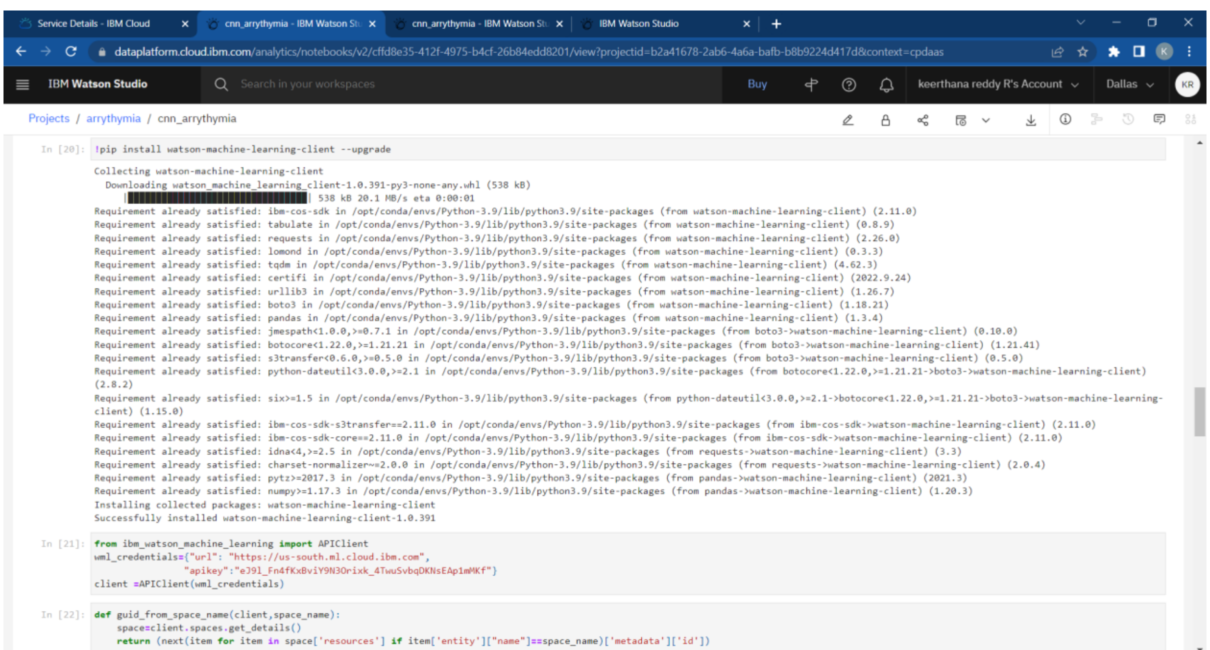


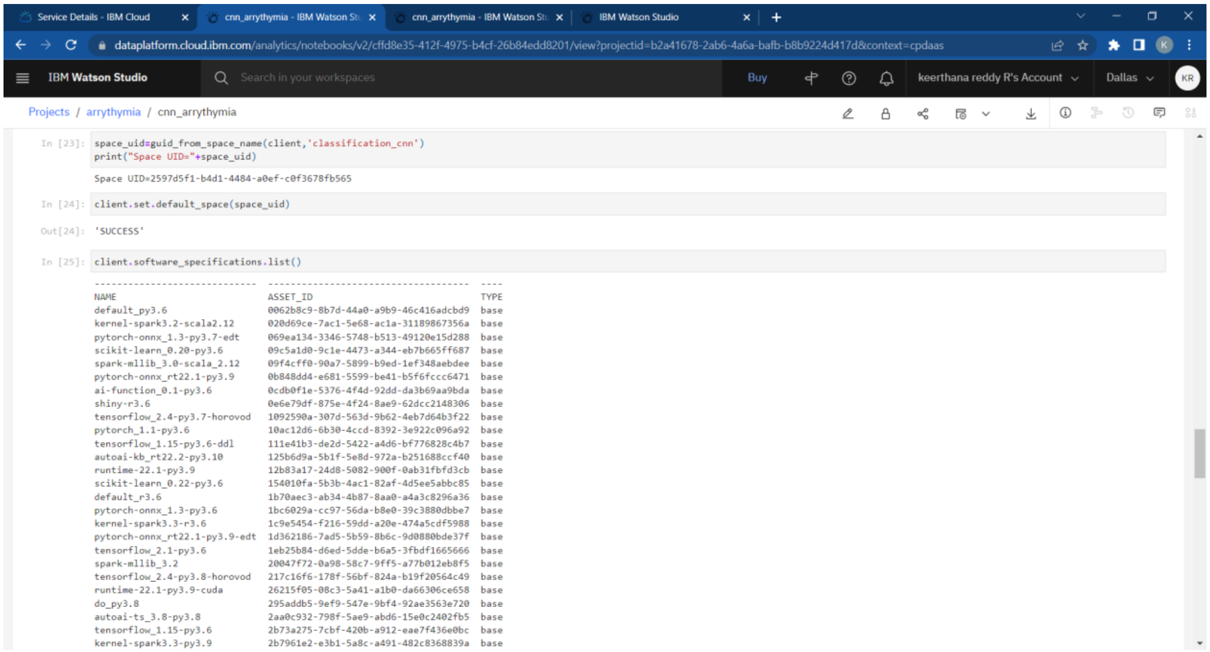


TRAINING THE MODEL IN CLOUD AND SAVING IT:



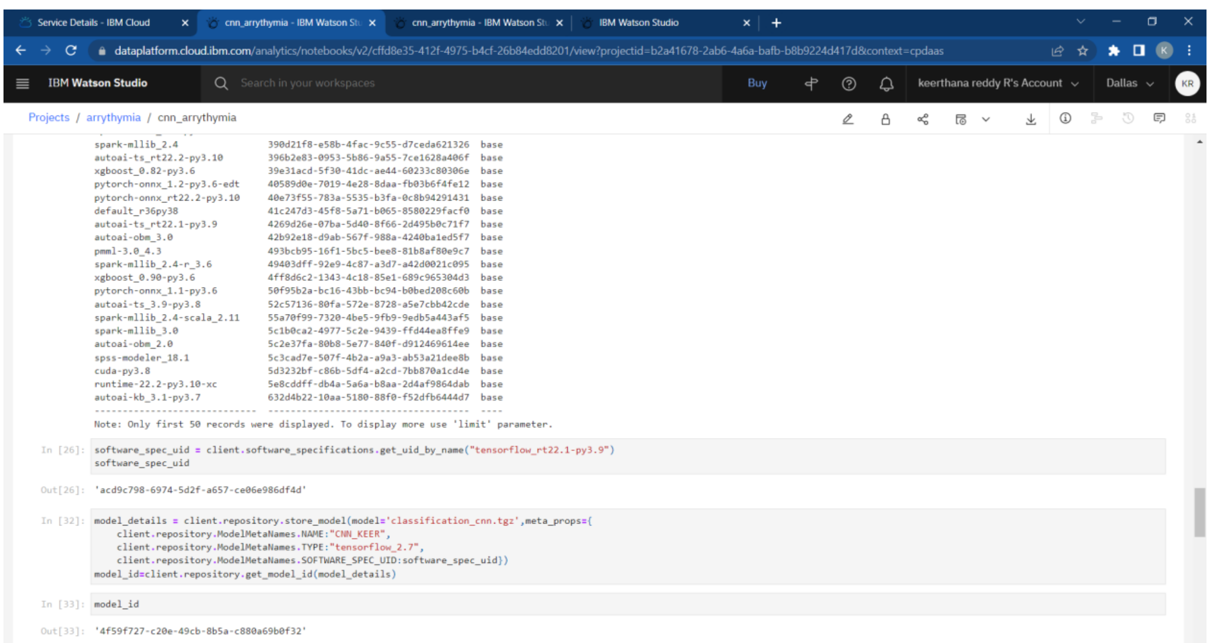
DEPLOYING THE MODEL IN IBM WATSON:





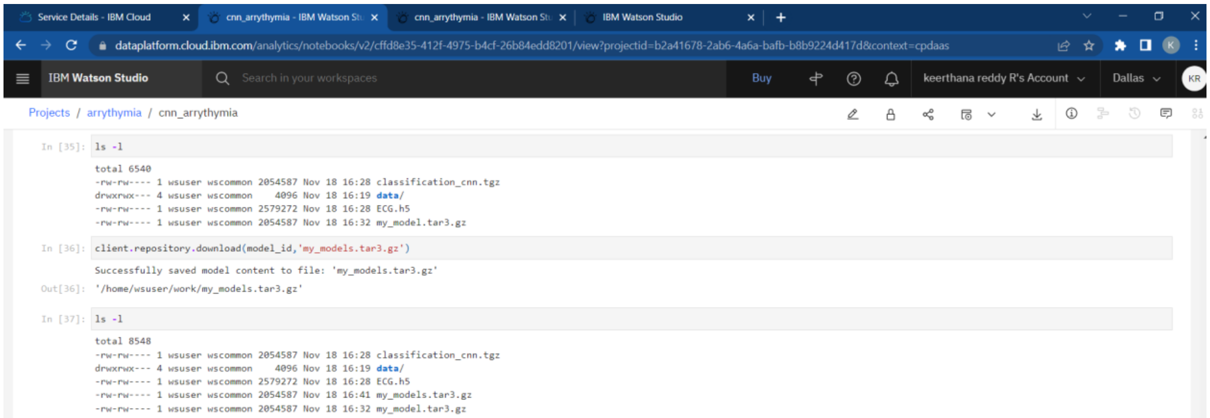
Projects / arrythymia / cnn_arrythymia

```
In [23]: space_uid=guid_from_space_name(client,'classification_cnn')
         print("Space UID="+space_uid)

         Space UID=2597d5f1-b4d1-4484-a0ef-c0f3678fb565

In [24]: client.set.default_space(space_uid)

Out[24]: 'SUCCESS'

In [25]: client.software_specifications.list()

         ----------------------------  ------------------------------------  ----
         NAME                          ASSET_ID                              TYPE
         default_py3.6                 0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
         kernel-spark3.2-scala2.12     020d69ce-7ac1-5e68-ac1a-31189867356a  base
         pytorch-onnx_1.3-py3.7-edt    069ea134-3346-5748-b513-49120e15d288  base
         scikit-learn_0.20-py3.6       09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
         spark-mllib_3.0-scala_2.12    09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
         pytorch-onnx_rt22.1-py3.9     0b848dd4-e681-5599-be41-b5f6fccc6471  base
         ai-function_0.1-py3.6         0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
         shiny-r3.6                    0e6e79df-875e-4f24-8ae9-62dcc2148306  base
         tensorflow_2.4-py3.7-horovod  1092590a-307d-563d-9b62-4eb7d64b3f22  base
         pytorch_1.1-py3.6             10ac12d6-6b30-4ccd-8392-3e922c096a92  base
         tensorflow_1.15-py3.6-ddl     111e41b3-de2d-5422-a4d6-bf776828c4b7  base
         autoai-kb_rt22.2-py3.10       125b6d9a-5b1f-5e8d-972a-b251688ccf40  base
         runtime-22.1-py3.9            12b83a17-24d8-5082-900f-0ab31fbfd3cb  base
         scikit-learn_0.22-py3.6       154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
         default_r3.6                  1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base
         pytorch-onnx_1.3-py3.6        1bc6029a-cc97-56da-b8e0-39c3880dbbe7  base
         kernel-spark3.3-r3.6          1c9e5454-f216-59dd-a20e-474a5cdf5988  base
         pytorch-onnx_rt22.1-py3.9-edt 1d362186-7ad5-5b59-8b6c-9d0880bde37f  base
         tensorflow_2.1-py3.6          1eb25b84-d6ed-5dde-b6a5-3fbdf1665666  base
         spark-mllib_3.2               20047f72-0a98-58c7-9ff5-a77b012eb8f5  base
         tensorflow_2.4-py3.8-horovod  217c16f6-178f-56bf-824a-b19f20564c49  base
         runtime-22.1-py3.9-cuda       26215f05-08c3-5a41-a1b0-da66306ce658  base
         do_py3.8                      295addb5-9ef9-547e-9bf4-92ae3563e720  base
         autoai-ts_3.8-py3.8           2aa0c932-798f-5ae9-abd6-15e0c2402fb5  base
         tensorflow_1.15-py3.6         2b73a275-7cbf-420b-a912-eae7f436e0bc  base
         kernel-spark3.3-py3.9         2b7961e2-e3b1-5a8c-a491-482c8368839a  base
```

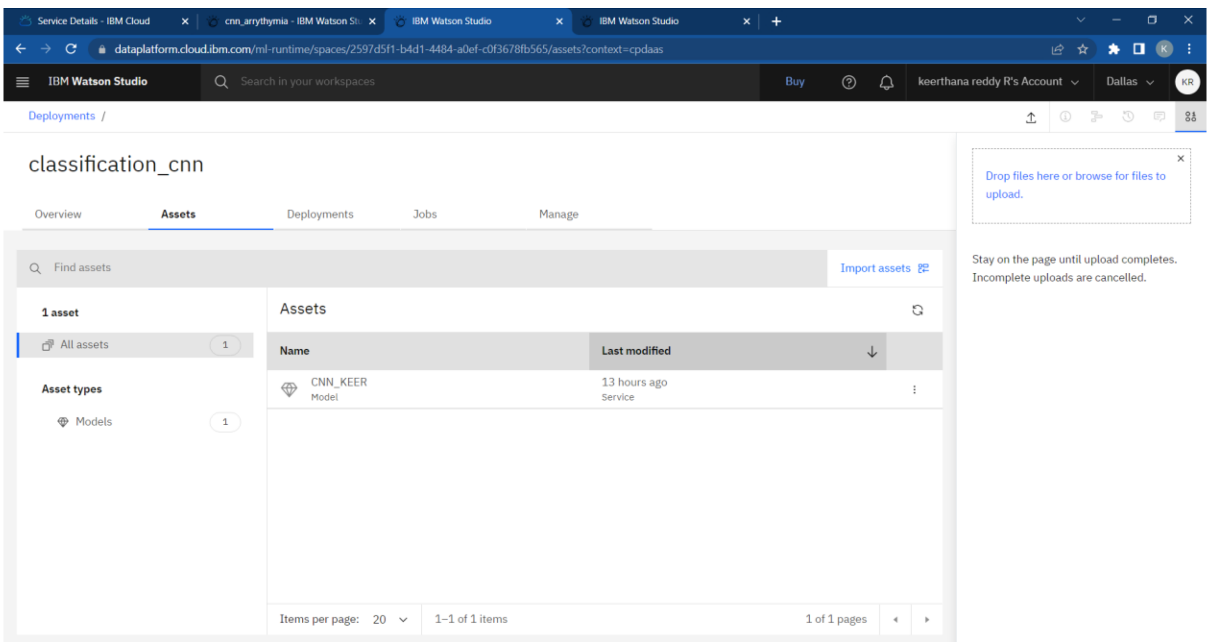


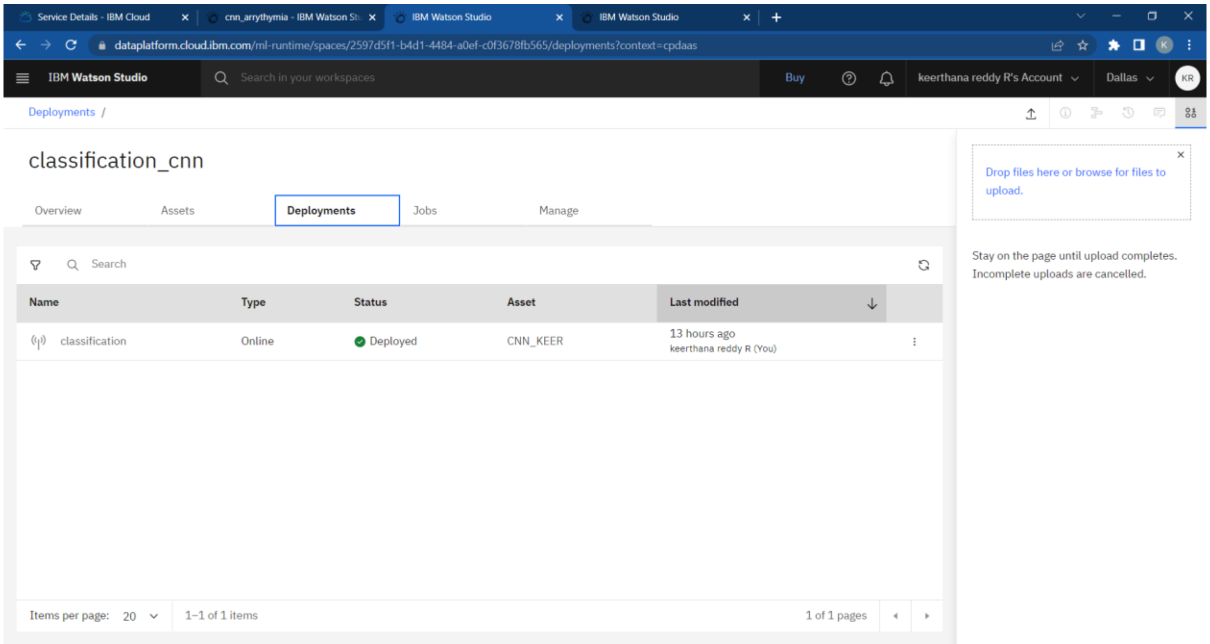
Projects / arrythymia / cnn_arrythymia

```
         spark-mllib_2.4               390d21f8-e58b-4fac-9c55-d7ceda621326  base
         autoai-ts_rt22.2-py3.10       396b2e83-0953-5b86-9a55-7ce1628a406f  base
         xgboost_0.82-py3.6            39e31acd-5f30-41dc-ae44-60233c80306e  base
         pytorch-onnx_1.2-py3.6-edt    40589d0e-7019-4e28-8daa-fb03b6f4fe12  base
         pytorch-onnx_rt22.2-py3.10    40e73f55-783a-5535-b3fa-0c8b94291431  base
         default_r36py38               41c247d3-45f8-5a71-b065-8580229facf0  base
         autoai-ts_rt22.1-py3.9        4269d26e-07ba-5d40-8f66-2d495b0c71f7  base
         autoai-obm_3.0                42b92e18-d9ab-567f-988a-4240ba1ed5f7  base
         pmml-3.0_4.3                  493bcb95-16f1-5bc5-bee8-81b8af80e9c7  base
         spark-mllib_2.4-r_3.6         49403dff-92e9-4c87-a3d7-a42d0021c095  base
         xgboost_0.90-py3.6            4ff8d6c2-1343-4c18-85e1-689c965304d3  base
         pytorch-onnx_1.1-py3.6        50f95b2a-bc16-43bb-bc94-b0bed208c60b  base
         autoai-ts_3.9-py3.8           52c57136-80fa-572e-8728-a5e7cbb42cde  base
         spark-mllib_2.4-scala_2.11    55a70f99-7320-4be5-9fb9-9edb5a443af5  base
         spark-mllib_3.0               5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9  base
         autoai-obm_2.0                5c2e37fa-80b8-5e77-840f-d912469614ee  base
         spss-modeler_18.1             5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b  base
         cuda-py3.8                    5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e  base
         runtime-22.2-py3.10-xc        5e8cddff-db4a-5a6a-b8aa-2d4af9864dab  base
         autoai-kb_3.1-py3.7           632d4b22-10aa-5180-88f0-f52dfb6444d7  base
         ----------------------------  ------------------------------------  ----
         Note: Only first 50 records were displayed. To display more use 'limit' parameter.

In [26]: software_spec_uid = client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
         software_spec_uid

Out[26]: 'acd9c798-6974-5d2f-a657-ce06e986df4d'

In [32]: model_details = client.repository.store_model(model='classification_cnn.tgz',meta_props={
             client.repository.ModelMetaNames.NAME:"CNN_KEER",
             client.repository.ModelMetaNames.TYPE:"tensorflow_2.7",
             client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid})
         model_id=client.repository.get_model_id(model_details)

In [33]: model_id

Out[33]: '4f59f727-c20e-49cb-8b5a-c880a69b0f32'
```



DEPLOYMENTS:





Testing was also done using the deployed model.

8. TESTING

Now that the model is successfully trained and saved, it is time to test the model using different test cases.

8.1 TEST CASES

First and foremost, the model was tested in the model building code. The keras utils module was imported to convert the input image to tensors and was given as input to the stored model. A Left bundle branch block image was given for testing, and the model yielded the correct results.

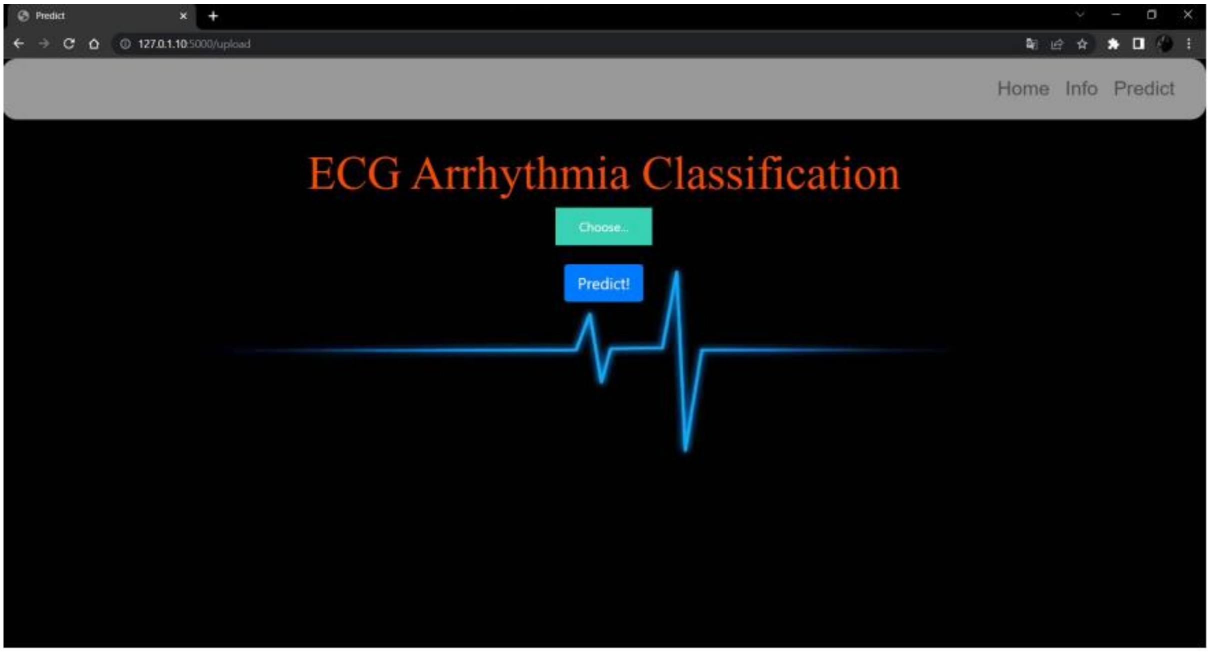
```
from tensorflow.keras.models import load_model
from keras.utils import load_img
from keras.utils import img_to_array
```

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

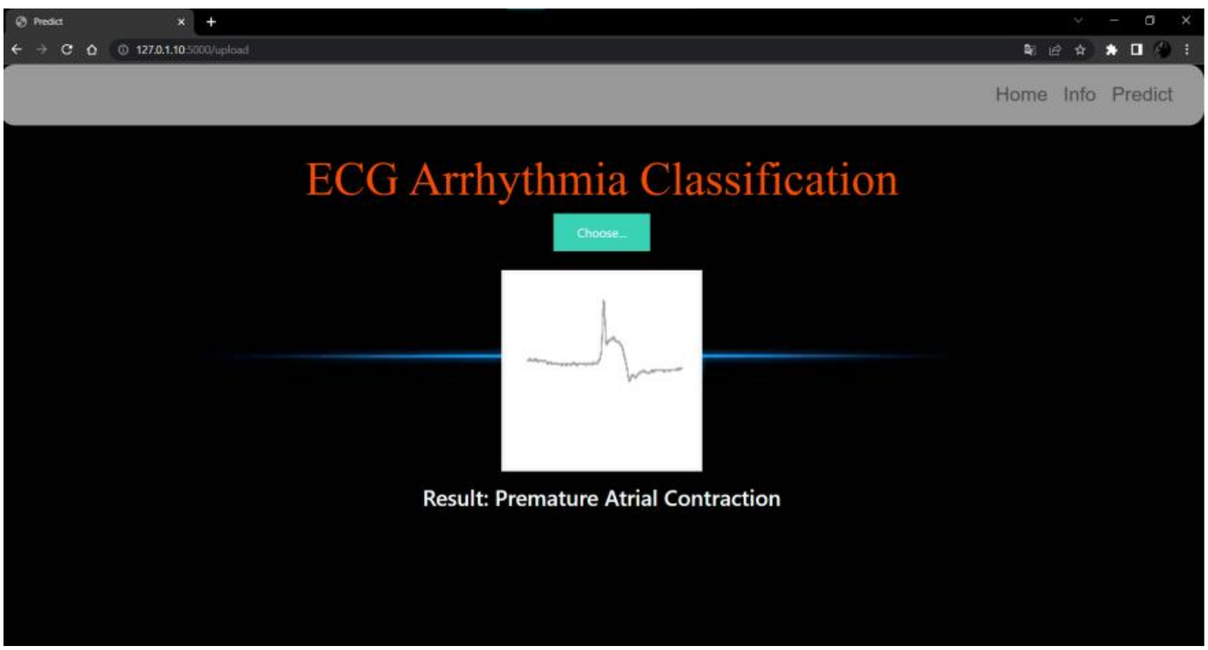
```
img=load_img('/content/drive/MyDrive/IBM project/dataset/test/Left Bundle Branch Block/fig_5906.png', target_size=(64,64))
x=img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=model.predict(x)
pred_classes=['Left bundle branch block', 'Normal', 'Premature atrial contraction', 'Premature ventricular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

```
1/1 [==============================] - 0s 16ms/step
[[1. 0. 0. 0. 0. 0.]]
Left bundle branch block
```

Then the model was tested in the web app, and the test results were obtained. We go to the predict page in the web app.



The image is uploaded, and predict button is pressed.



The correct answer is predicted!

The same was tested in the deployed model in IBM Watson.

```
In [38]: from keras.models import load_model
         from keras.preprocessing import image

In [40]: model=load_model('/home/wsuser/work/ECG.h5')
```

```
-rw-rw---- 1 wsuser wscommon 2054587 Nov 18 16:32 my_model.tar3.gz

In [46]: img=image.load_img('/home/wsuser/work/data/test/Left Bundle Branch Block/fig_5906.png', target_size=(64,64))
         x=image.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. 0. 0. 0. 0. 0.]]
Left bundle branch block
```

8.2 USER ACCEPTANCE TESTING:

1. DEFECT ANALYSIS

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Sub total
By Design	5	1	2	0	8
Duplicate	1	0	1	0	2
External	2	3	0	1	6
Fixed	4	2	4	0	10
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	12	6	7	1	26

2. 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	10	0	0	10
Client Application	20	0	0	20
Security	1	0	0	1
Outsource Shipping	4	0	0	4
Exception Reporting	5	0	0	5
Final Report Output	4	0	0	4
Version Control	2	0	0	2

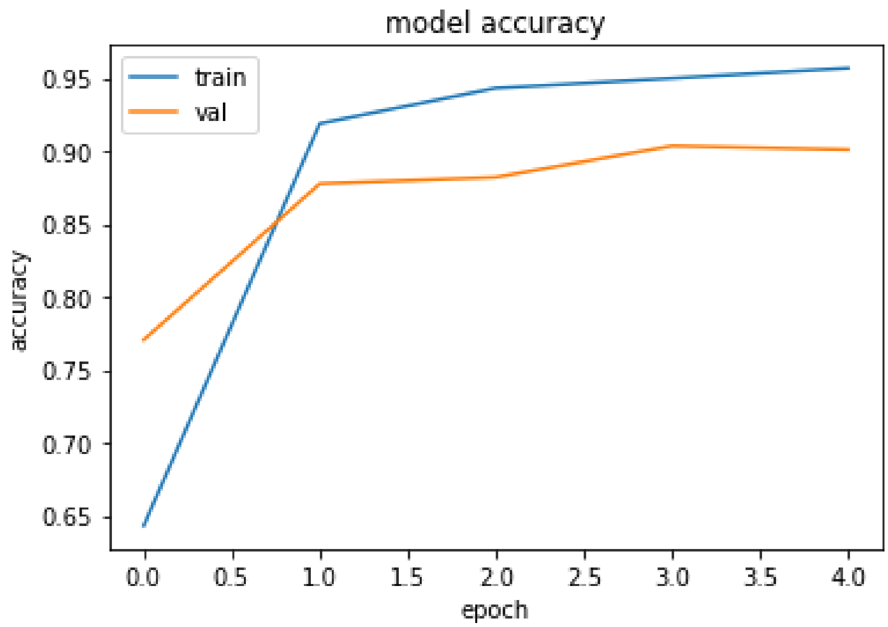
9. PERFORMANCE METRICS

We used 4 main metrics to evaluate our model:

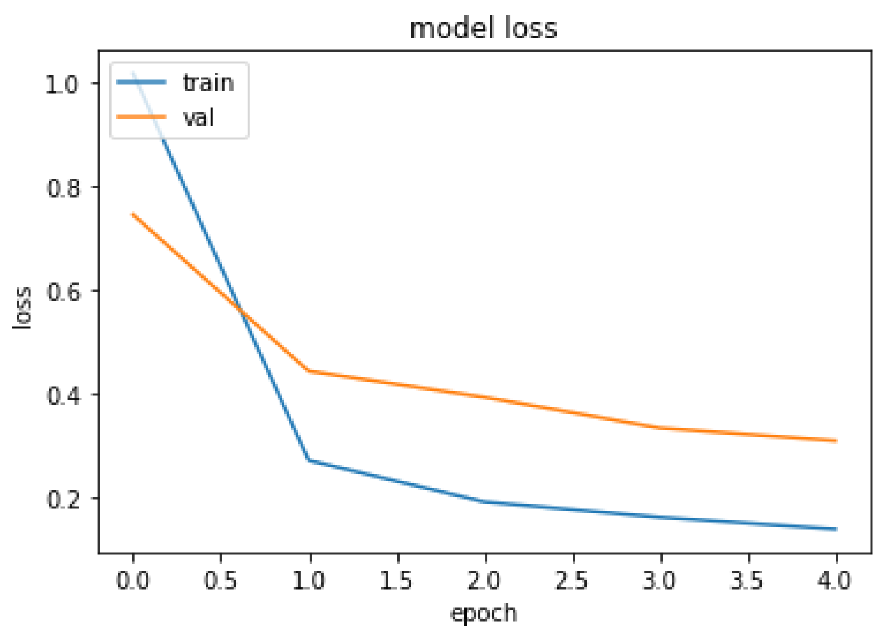
- Categorical accuracy: Categorical Accuracy calculates the percentage of predicted values (yPred) that match with actual values (yTrue) for one-hot labels. For a record: We identify the index at which the maximum value occurs using argmax(). If it is the same for both yPred and yTrue, it is considered accurate. We then calculate Categorical Accuracy by dividing the number of accurately predicted records by the total number of records.
- Loss: We use categorical cross entropy to compute the loss here. Used as a loss function for multi-class classification model where there are two or more output labels. The output label is assigned one-hot category encoding value in form of 0s and 1. The output label, if present in integer form, is converted into categorical encoding using keras.utils to_categorical method.
- Precision: Precision is defined as the ratio of correctly classified positive samples (True Positive) to a total number of classified positive samples (either correctly or incorrectly).
- Recall: The recall is calculated as the ratio between the number of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect Positive samples. The higher the recall, the more positive samples detected.

PLOTS:

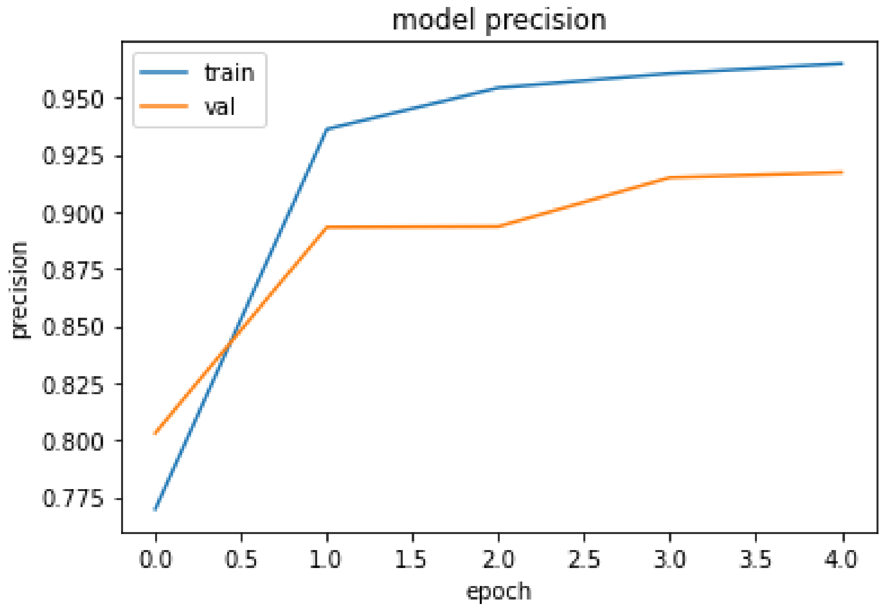
Categorical accuracy:



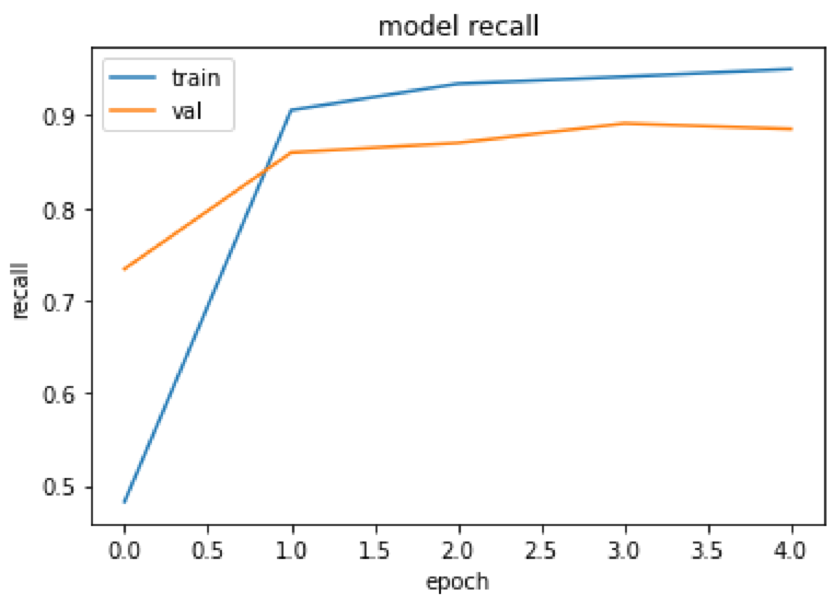
Loss:



Precision:



Recall:



As we can see the model decreases the categorical cross entropy loss as well as very good precision and recall. The final accuracy of the model obtained is: 88.87%.

The accuracy mentioned here can be increased by increasing the number of epochs. As we can see here, an almost convergence criterion is not observed yet. So, by increasing the number of epochs, the accuracy can be increased. The reason why we chose for very small number of epochs for training is because of huge training duration required by the model as the dataset is huge. But a decent accuracy is obtained due to proper pre-processing and the architecture of the CNN model deployed.

10 ADVANTAGES & DISADVANTAGES:

10.1 ADVANTAGES:

i. The proposed model predicts Arrhythmia in images with a high accuracyrate of nearly 96%

ii. The early detection of Arrhythmia gives better understanding of diseasecauses, initiates therapeutic interventions and enables developing appropriate treatments.

10.2 DISADVANTAGES:

i. Not useful for identifying the different stages of Arrhythmia disease.

ii. Not useful in monitoring motor symptoms

11. CONCLUSION:

- Cardiovascular disease is a major health problem in today's world. The early diagnosis ofcardiac arrhythmia highly relies on the ECG.
- Unfortunately, the expert level of medical resources is rare, visually identify the ECGsignal 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 thiswork 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:

Github Link:

- https://github.com/IBM-EPBL/IBM-Project-20142-1659713158

Demo Video Link:

- https://drive.google.com/file/d/19B9eeh1jcTvlo5n1EKiEorGgAcTzpRSc/view?usp=drivesdk

Google Colab Link:

- https://colab.research.google.com/drive/1NlxOYDBgeQR11ZN-QPcqRWYVFYdbhcdC?usp=sharing

REFERENCES:

- https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/