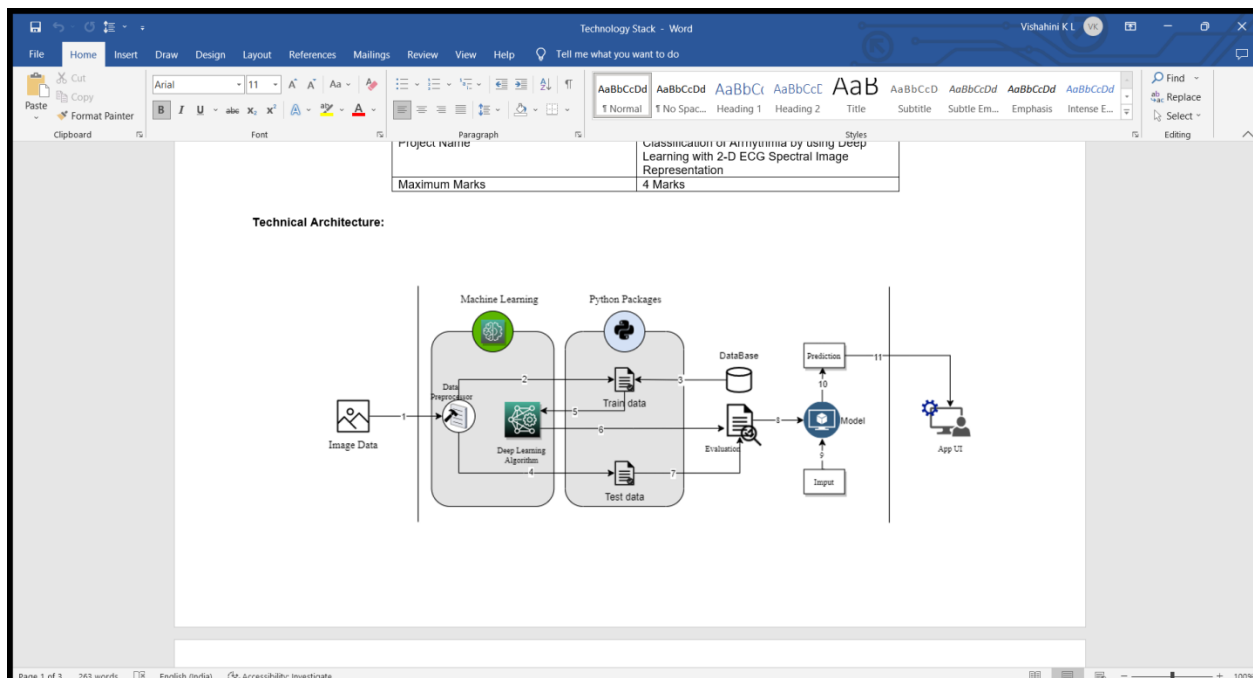


Team ID	PNT2022TMID24237
Project Name	Classification of arrhythmia by using deep learning with 2-d ECG Spectral Image Representation

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

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 convolutional 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.

TECHNICAL ARCHITECTURE:



PROJECT OBJECTIVES

- Know fundamental concepts and techniques of the Artificial Neural Network and Convolution Neural Networks
- Gain a broad understanding of image data.
- Work with Sequential type of modeling
- Work with Keras capabilities
- Work with image processing techniques
- know how to build a web application using the Flask framework.

PROJECT FLOW

- User interacts with User interface to upload image
- Uploaded image is analyzed by the model which is integrated

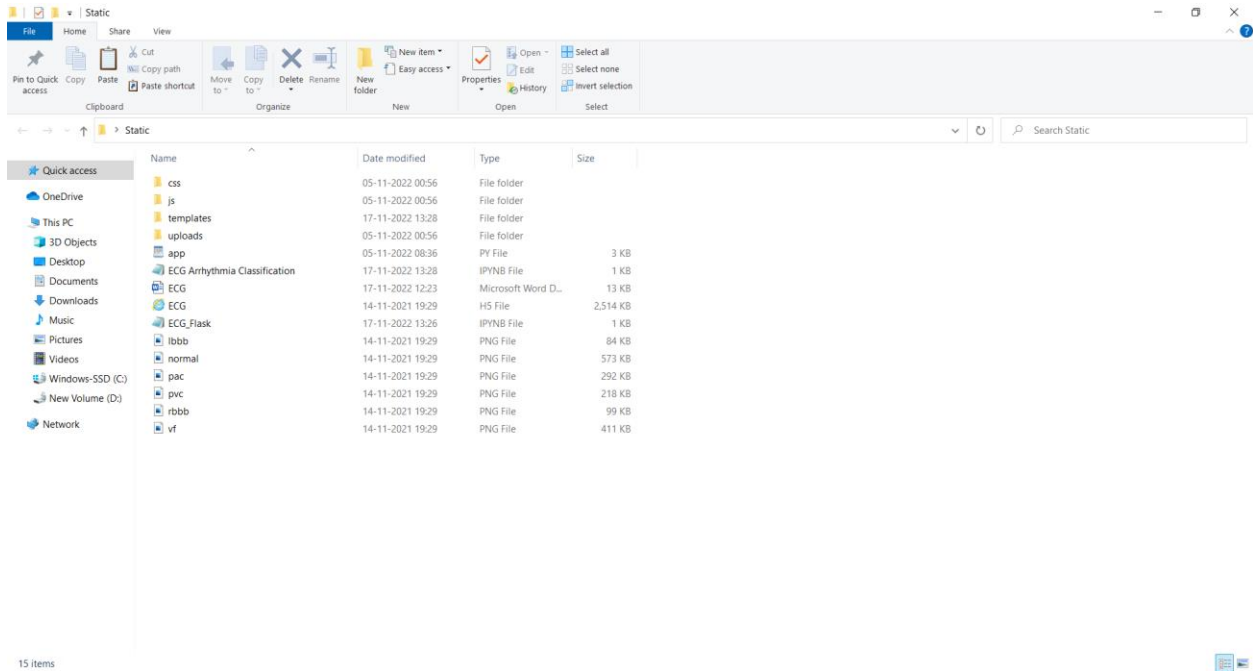
- Once model analyses the uploaded image, the prediction is showcased on the UI

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
 - Collect the dataset or Create the dataset
- Data Preprocessing.
 - Import the ImageDataGenerator library
 - Configure ImageDataGenerator class
 - Apply ImageDataGenerator functionality to Trainset and Testset
- Model Building
 - Import the model building Libraries
 - Initializing the model
 - Adding Input Layer
 - Adding Hidden Layer
 - Adding Output Layer
 - Configure the Learning Process
 - Training and testing the model
 - Optimize the Model
 - Save the Model
- Application Building
 - Create an HTML file
 - Build Python Code

PROJECT STRUCTURE

Create a Project folder which contains files as shown below



- We are building a Flask Application that needs HTML pages stored in the templates folder and a python script app.py for serverside scripting
- we need the model which is saved and the saved model in this content is ECG.h5
- The static folder will contain js and CSS files.
- Whenever we upload an image to predict, those images are saved in the uploads folder.

PREREQUISITES

REQUIRED SOFTWARE AND PACKAGES

Anaconda Navigator :

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupiter notebook and spyder

PACKAGES REQUIRED TO BUILD DEEP LEARNING MODELS

Tensor flow: TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML powered applications.

Keras : Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features:

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.

- Highly scalability of computation.

Flask: Web frame work used for building Web applications

PRIOR KNOWLEDGE

SUPERVISED AND UNSUPERVISED LEARNING:

DATASET COLLECTION

Artificial Intelligence is a data hunger technology, it depends heavily on data, without data, it is impossible for a machine to learn. It is the most crucial aspect that makes algorithm training possible. In Convolutional Neural Networks, as it deals with images, we need training and testing data set. It is the actual data set used to train the model for performing various actions. In this activity lets focus of gathering the dataset

DOWNLOAD THE DATASET

You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository, etc.

The dataset used for this project is in this [Link](#) Please refer to the link to download the data set

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



<https://drive.google.com/file/d/16SURk6lMaakmVf4axGNDub3joHl-XdBT/view?usp=sharing>

IMAGE PREPROCESSING

Image Pre-processing includes the following main tasks

- Import ImageDataGenerator Library.
- Configure ImageDataGenerator Class.
- Applying ImageDataGenerator functionality to the trainset and test set.

IMPORT THE 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.

Let us import the ImageDataGenerator class from Keras

```
Anaconda Prompt (anaconda3) - python

(base) C:\Users\Welcome>python
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import keras
2022-11-05 12:46:44.779459: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlderror: cudart64_110.dll
not found
2022-11-05 12:46:44.779711: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
>>> import keras
>>> from 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.


```
Anaconda Prompt (anaconda3) - python
(base) C:\Users\Welcome>python
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import keras
2022-11-05 12:46:44.779459: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlderror: cudart64_110.dll
not found
2022-11-05 12:46:44.779711: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
>>> import keras
>>> from keras.preprocessing.image import ImageDataGenerator
>>>
>>> train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
>>> test_datagen=ImageDataGenerator(rescale=1./255)
>>>
```

APPLY IMAGEDATAGENERATOR FUNCTIONALITY TO TRAINSET AND TESTSET

Let us apply ImageDataGenerator functionality to Train set and Test set 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}

```
Anaconda Prompt (anaconda3) - python
(base) C:\Users\Welcome>python
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import keras
2022-11-05 12:51:16.444120: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlderror: cudart64_110.dll
not found
2022-11-05 12:51:16.444438: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
>>> import keras
>>> from keras.preprocessing.image import ImageDataGenerator
>>> train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
>>> test_datagen=ImageDataGenerator(rescale=1./255)
>>> import sklearn
>>> from sklearn.model_selection import train_test_split
>>> x_train=train_datagen.flow_from_directory(directory=r'C:\Users\Welcome\Desktop\Pava\IBM Project\mit arrhythmia database\extracted db\data\train',target_size=(64,64),batch_size=32,class_mode='categorical')
Found 15341 images belonging to 6 classes.
>>> x_test=test_datagen.flow_from_directory(directory=r'C:\Users\Welcome\Desktop\Pava\IBM Project\mit arrhythmia database\extracted db\data\test',target_size=(64,64),b
Found 6825 images belonging to 6 classes.
>>> _
```

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.

Arguments:

- **directory:** Directory where the data is located. If labels is "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
- **batch_size:** Size of the batches of data. Default: 32.
- **target_size:** Size to resize images to after they are read from disk.
- **class_mode:**
 - 'int': means that the labels are encoded as integers (e.g. for `sparse_categorical_crossentropy` loss).

- 'categorical' means that the labels are encoded as a categorical vector (e.g. for `categorical_crossentropy` loss).
- 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for `binary_crossentropy`).
- None (no labels).

MODEL BUILDING

We are ready with the augmented and pre-processed image data, Lets begin our model building, this activity includes the following steps

- Import the model building Libraries
- Initializing the model
- Adding CNN Layers
- Adding Hidden Layer
- Adding Output Layer
- Configure the Learning Process
- Training and testing the model
- Saving the model

IMPORT THE LIBRARIES

This is a very crucial step in our deep learning model building process. We have to define how our model will look and that requires.

```
Anaconda Prompt (anaconda3) - python

(base) C:\Users\Welcome>python
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import keras
2022-11-05 13:02:40.239726: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlderror: cudart64_110.dll
not found
2022-11-05 13:02:40.239978: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
>>> import keras
>>> from keras.preprocessing.image import ImageDataGenerator
>>> train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
>>> test_datagen=ImageDataGenerator(rescale=1./255)
>>> import sklearn
>>> from sklearn.model_selection import train_test_split
>>> x_train=train_datagen.flow_from_directory(directory=r'C:\Users\Welcome\Desktop\Pava\IBM Project\mit arrhythmia database\extracted db\data\train',target_size=(64,64)
Found 15341 images belonging to 6 classes.
>>> x_test=test_datagen.flow_from_directory(directory=r'C:\Users\Welcome\Desktop\Pava\IBM Project\mit arrhythmia database\extracted db\data\test',target_size=(64,64),b
atch_size=32,class_mode='categorical')
Found 6825 images belonging to 6 classes.
>>> import numpy as np
>>> import tensorflow
>>> 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
>>> _
```

INITIALIZE 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.

```
Anaconda Prompt (anaconda3) - python
>>> test_datagen=ImageDataGenerator(rescale=1./255)
>>> import sklearn
>>> from sklearn.model_selection import train_test_split
>>> x_train=train_datagen.flow_from_directory(directory=r'C:\Users\Welcome\Desktop\Pava\IBM Project\mit arrhythmia database\extracted db\data\train',target_size=(64,64),
Found 15341 images belonging to 6 classes.
>>> x_test=test_datagen.flow_from_directory(directory=r'C:\Users\Welcome\Desktop\Pava\IBM Project\mit arrhythmia database\extracted db\data\test',target_size=(64,64),b
atch_size=32,class_mode='categorical')
Found 6825 images belonging to 6 classes.
>>> import numpy as np
>>> import tensorflow
>>> from tensorflow.keras.models import Sequential
>>> from tensorflow.keras.layers import layers
>>> from tensorflow.keras.layers import Dense,Flatten
>>> from tensorflow.keras.layers import Conv2D,MaxPooling2D
>>> model=Sequential()
2022-11-05 13:08:25.907786: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll
not found
2022-11-05 13:08:25.909120: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cublas64_11.dll'; dlerror: cublas64_11.dll n
ot found
2022-11-05 13:08:25.910242: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cublaslt64_11.dll'; dlerror: cublaslt64_11.d
ll not found
2022-11-05 13:08:25.911360: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cuffft64_10.dll'; dlerror: cufft64_10.dll not
found
2022-11-05 13:08:25.912451: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'curand64_10.dll'; dlerror: curand64_10.dll n
ot found
2022-11-05 13:08:25.913615: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cusolver64_11.dll'; dlerror: cusolver64_11.d
ll not found
2022-11-05 13:08:25.914864: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cusparse64_11.dll'; dlerror: cusparse64_11.d
ll not found
2022-11-05 13:08:25.916041: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudnn64_8.dll'; dlerror: cudnn64_8.dll not f
ound
2022-11-05 13:08:25.916261: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1934] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned
above are installed properly if you would like to use GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required librari
es for your platform.
Skipping registering GPU devices...
2022-11-05 13:08:25.927556: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to
use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
>>> 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())
>>>
```

ADDING DENSE LAYERS

Dense layer is deeply connected neural network layer. It is most common and frequently used layer. We have 6 neurons in op layer as we have considered 6 classes. Understanding the model is a very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers

```
Anaconda Prompt (anaconda3) - python
ll not found
2022-11-05 13:08:25.916041: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudnn64_8.dll'; dlopen: cudnn64_8.dll not f
ound
2022-11-05 13:08:25.916261: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1934] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned
above are installed properly if you would like to use GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required librari
es for your platform.
Skipping registering GPU devices...
2022-11-05 13:08:25.927556: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to
use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
>>> 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'))
>>> 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 (MaxPooling2D)	(None, 14, 14, 32)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 32)	200736
dense_1 (Dense)	(None, 6)	198

```

Total params: 211,078
Trainable params: 211,078
Non-trainable params: 0
>>>
```

CONFIGURE THE LEARNING PROCESS

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

```
Anaconda Prompt (anaconda3) - python
2022-11-05 13:08:25.916041: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudnn64_8.dll'; dlerror: cudnn64_8.dll not found
2022-11-05 13:08:25.916261: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1934] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are installed properly if you would like to use GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...
2022-11-05 13:08:25.927556: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
>>> 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'))
>>> 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 (MaxPooling2D)	(None, 14, 14, 32)	0
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
>>> model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
>>>
```

TRAIN THE MODEL

Now, let us train our model with our image dataset.

`fit_generator` functions used to train a deep learning neural network

`steps_per_epoch`: it specifies the total number of steps taken from the generator as soon as one epoch is finished and next epoch has started. We can calculate the value of `steps_per_epoch` as the total number of samples in your train dataset divided by the batch size.

Epochs: an integer and number of epochs we want to train our model.

validation_data can be either:

- an inputs and targets list
- a generator
- an inputs, targets, and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.

validation_steps: only if the **validation_data** is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your test dataset divided by the validation batch size.

```
Anaconda Prompt (anaconda3) - python
max_pooling2d_1 (MaxPooling (None, 14, 14, 32) 2D) 0

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

>>> model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
>>> model.fit_generator(generator=x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_
... test,validation_steps=len(x_test))
File "<stdin>", line 2
    test,validation_steps=len(x_test))
    ^
SyntaxError: invalid syntax

>>> model.fit_generator(generator=x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
<stdin>:1: UserWarning: 'Model.fit_generator' is deprecated and will be removed in a future version. Please use 'Model.fit', which supports generators.
Epoch 1/10
480/480 [=====] - ETA: 0s - loss: 1.1948 - accuracy: 0.5696
```

SAVE THE MODEL

The model is saved with .h5 extension as follows

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.


```
Anaconda Prompt (anaconda3) - python
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

>>> model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
>>> model.fit_generator(generator=x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_
... test,validation_steps=len(x_test))
File "<stdin>", line 2
    test,validation_steps=len(x_test)
    ^
SyntaxError: invalid syntax
>>> model.fit_generator(generator=x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
<stdin>:1: UserWarning: 'Model.fit_generator' is deprecated and will be removed in a future version. Please use 'Model.fit', which supports generators.
Epoch 1/10
480/480 [=====] - 243s 504ms/step - loss: 1.1948 - accuracy: 0.5696 - val_loss: 1.3013 - val_accuracy: 0.6164
Epoch 2/10
480/480 [=====] - 82s 171ms/step - loss: 0.5324 - accuracy: 0.8350 - val_loss: 0.6060 - val_accuracy: 0.8321
Epoch 3/10
480/480 [=====] - 81s 168ms/step - loss: 0.3445 - accuracy: 0.8982 - val_loss: 0.4317 - val_accuracy: 0.8697
Epoch 4/10
480/480 [=====] - 81s 169ms/step - loss: 0.2855 - accuracy: 0.9136 - val_loss: 0.4400 - val_accuracy: 0.8879
Epoch 5/10
480/480 [=====] - 81s 168ms/step - loss: 0.2522 - accuracy: 0.9301 - val_loss: 0.4163 - val_accuracy: 0.8838
Epoch 6/10
480/480 [=====] - 80s 167ms/step - loss: 0.2370 - accuracy: 0.9321 - val_loss: 0.3774 - val_accuracy: 0.8829
Epoch 7/10
480/480 [=====] - 80s 167ms/step - loss: 0.2182 - accuracy: 0.9360 - val_loss: 0.3582 - val_accuracy: 0.8911
Epoch 8/10
480/480 [=====] - 115s 240ms/step - loss: 0.2147 - accuracy: 0.9377 - val_loss: 0.3954 - val_accuracy: 0.8881
Epoch 9/10
480/480 [=====] - 102s 212ms/step - loss: 0.1938 - accuracy: 0.9430 - val_loss: 0.3519 - val_accuracy: 0.8964
Epoch 10/10
480/480 [=====] - 103s 215ms/step - loss: 0.1896 - accuracy: 0.9439 - val_loss: 0.3672 - val_accuracy: 0.9008
<keras.callbacks.History object at 0x000001078006C4C0>
>>>
>>> model.save('ECG.h5')
>>>
```

TEST THE MODEL

Open the other jupyter file and write the code mentioned below.

Load necessary libraries, Load the saved model using load_model.

Taking an image as input and checking the results .

The predicted class index name will be printed.

```
Anaconda Prompt (anaconda3) - python
>>> image=urllib.request.urlopen(url)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'urllib' is not defined
>>> img=image.load_img("C:\Users\Welcome\Desktop\Pava\IBM Project\project structure\uploads\PAC.png",target_size=(64,64))
File "<stdin>", line 1
  img=image.load_img("C:\Users\Welcome\Desktop\Pava\IBM Project\project structure\uploads\PAC.png",target_size=(64,64))
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \U0000XXXX escape
>>> img=image.load_img(r"C:\Users\Welcome\Desktop\Pava\IBM Project\project structure\uploads\PAC.png",target_size=(64,64))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: module 'keras.preprocessing.image' has no attribute 'load_img'
>>> img=keras.utils.load_img(r"C:\Users\Welcome\Desktop\Pava\IBM Project\project structure\uploads\PAC.png",target_size=(64,64))
>>> x=keras.utils.img_to_array(img)
>>> x=np.expand_dims(x,axis=0)
>>> pred=model.predict_classes(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Sequential' object has no attribute 'predict_classes'
>>> pred=model.predict_classes(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Sequential' object has no attribute 'predict_classes'
>>> predict_x=model.predict(x)
1/1 [=====] - 2s 2s/step
>>> classes_x=np.argmax(predict_x,axis=1)
>>> predict_x
array([[0., 0., 1., 0., 0., 0.]], dtype=float32)
>>> index=[ 'Left Bundle Branch Block', 'Normal', 'Premature Atrial Contraction', 'Premature Ventricular Contractions', 'Right Bundle Branch Block', 'Ventricular Fibrillation'
]
>>> result=str(index[predict_x[0]])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: only integer scalar arrays can be converted to a scalar index
>>> result=str(index[classes_x[0]])
>>> result
'Premature Atrial Contraction'
>>>
```

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

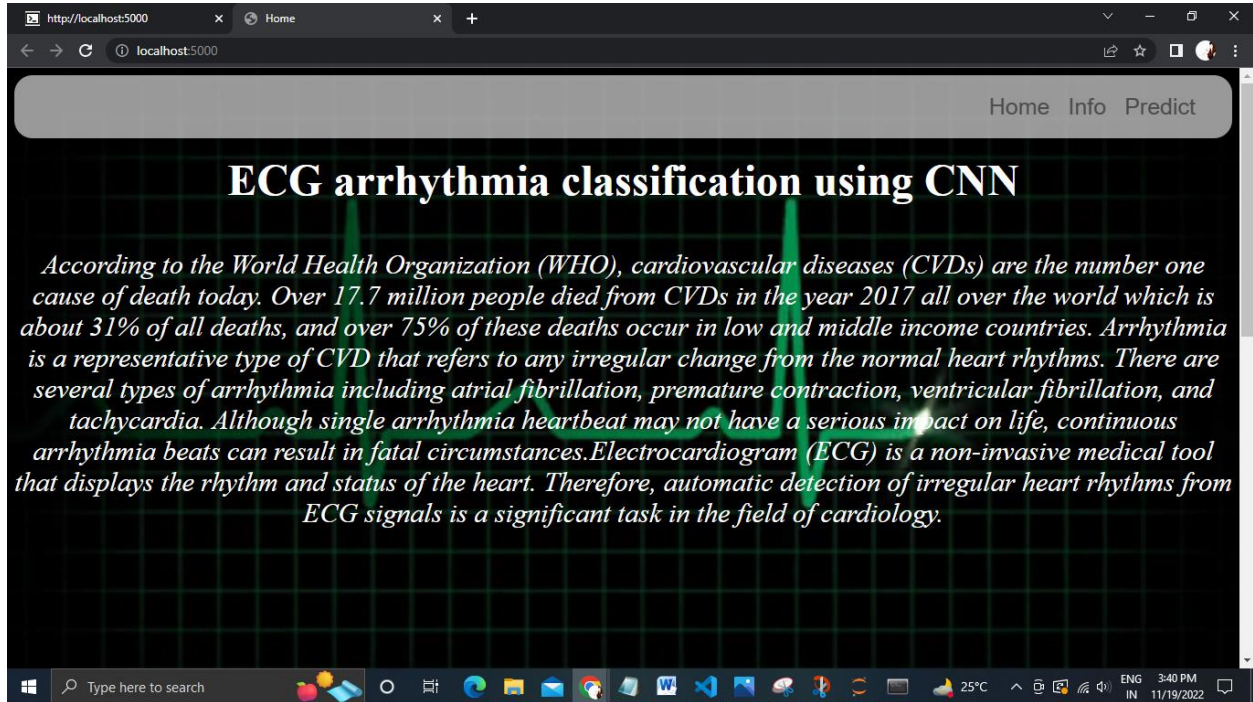
- Building HTML Pages
- Building server-side script

CREATE HTML FILES

- We use HTML to create the front end part of the web page.
- Here, we created 4 html pages- about.html, base.html, index6.html, info.html.
- about.html displays the home page.

- Info.html displays all important details to be known about ECG.
- base.html and index6.html accept input from the user and predicts the values.

HOME PAGE



When the “Info” button is clicked, localhost redirects to “info.html”

INFO.HTML PAGE

http://localhost:5000 Info


localhost:5000/info

Home Info Predict

ECG

NORMAL

Note that the heart is beating in a regular sinus rhythm between 60 - 100 beats per minute (specifically 82 bpm). All the important intervals on this recording are within normal ranges.



localhost:5000/info

Type here to search

25°C

ENG 3:41 PM IN 11/19/2022

http://localhost:5000 Info

localhost:5000/info

in adults.

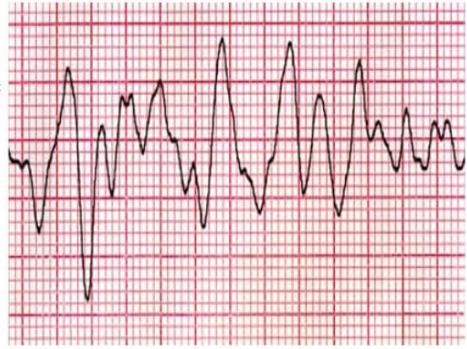
VENTRICULAR FIBRILLATION

A life-threatening heart rhythm that results in a rapid, inadequate heartbeat.

Ventricular fibrillation (VF) is a rapid, life-threatening heart rhythm starting in the bottom chambers of the heart. It can be triggered by a heart attack.

Because the heart doesn't pump adequately during ventricular fibrillation, sustained VF can cause low blood pressure, loss of consciousness or death.

Emergency treatment includes immediate defibrillation with an automated external defibrillator (AED) and cardiopulmonary resuscitation (CPR). Long-term therapy includes implantable defibrillators and medications to prevent recurrence.



Type here to search

25°C

ENG 3:42 PM IN 11/19/2022

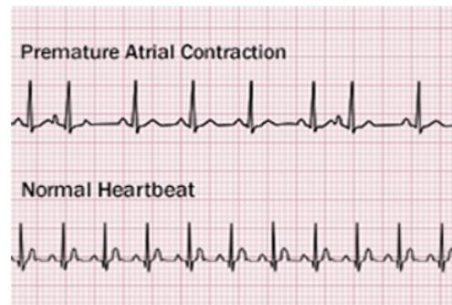
PREMATURE ATRIAL CONTRACTION

Usually, premature atrial contractions have no clear cause and no health risks. In most cases, premature atrial contractions aren't a sign of heart disease and just happen naturally.

But some people who have PACs turn out to have related heart conditions, such as:

- *Cardiomyopathy (a weakened heart muscle)*
- *Coronary heart disease (fatty deposits in your blood vessels)*

If your doctor finds that you have a condition related to the premature heartbeats, you'll work together to make a treatment plan.



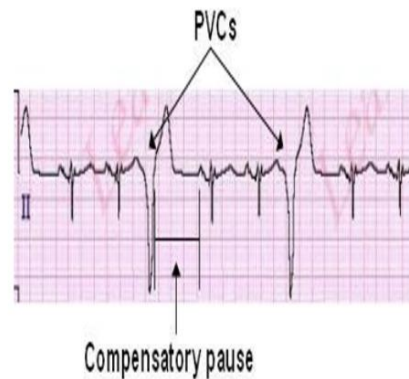
PREMATURE VENTRICULAR CONTRACTIONS

Extra, abnormal heartbeats that begin in one of the heart's two lower chambers.

Premature ventricular contractions (PVCs) occur in most people at some point. Causes may include certain medication, alcohol, some illegal drugs, caffeine, tobacco, exercise or anxiety.

PVCs often cause no symptoms. When symptoms do occur, they feel like a flip-flop or skipped-beat sensation in the chest.

Most people with isolated PVCs and an otherwise normal heart don't need treatment. PVCs occurring continuously for longer than 30 seconds is a potentially serious cardiac condition known as ventricular tachycardia.



http://localhost:5000 x Info x +
localhost:5000/info

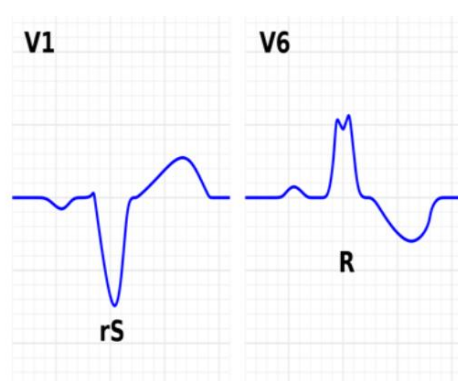
LEFT BUNDLE BRANCH BLOCK

A delay or blockage of electrical impulses to the left side of the heart.

Left bundle branch block sometimes makes it harder for the heart to pump blood efficiently through the circulatory system.

Most people don't have symptoms. If symptoms occur, they include fainting or a slow heart rate.

If there's an underlying condition, such as heart disease, that condition needs treatment. In patients with heart failure, a pacemaker can also relieve symptoms as well as prevent death.



V1 V6

rS R

Type here to search 25°C ENG IN 3:42 PM 11/19/2022

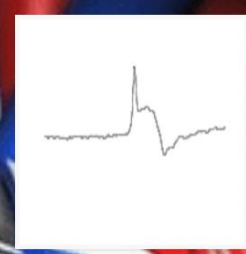
PREDICTION PAGE

http://localhost:5000 x Predict x +
localhost:5000/upload

Home Info Predict

ECG Arrhythmia Classification

Choose...



Result: Premature Atrial Contraction

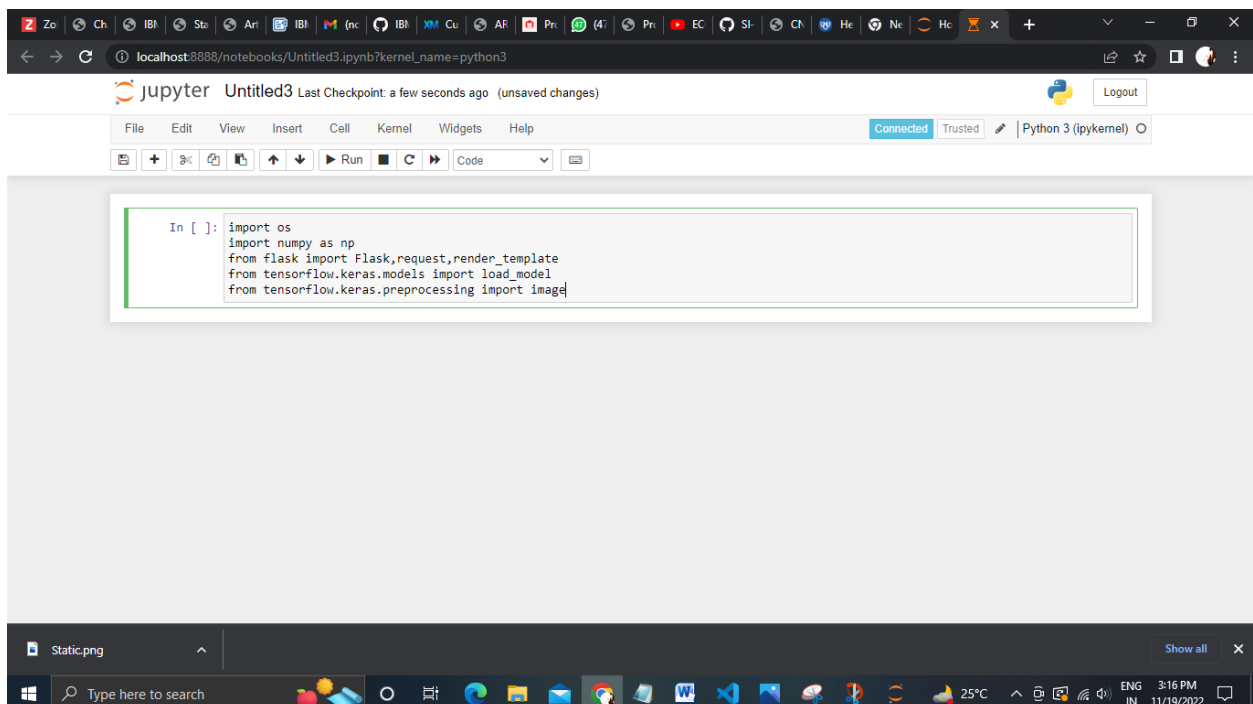
Type here to search 25°C ENG IN 3:45 PM 11/19/2022

Upload the image and click on the Predict button to view the result on the “base.html” page on the localhost.

BUILD PYTHON CODE

- Let us build the flask file ‘app.py’ which is a web framework written in python for server-side scripting. Let’s see step by step procedure for building the backend application.
- 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.

IMPORT THE LIBRARIES

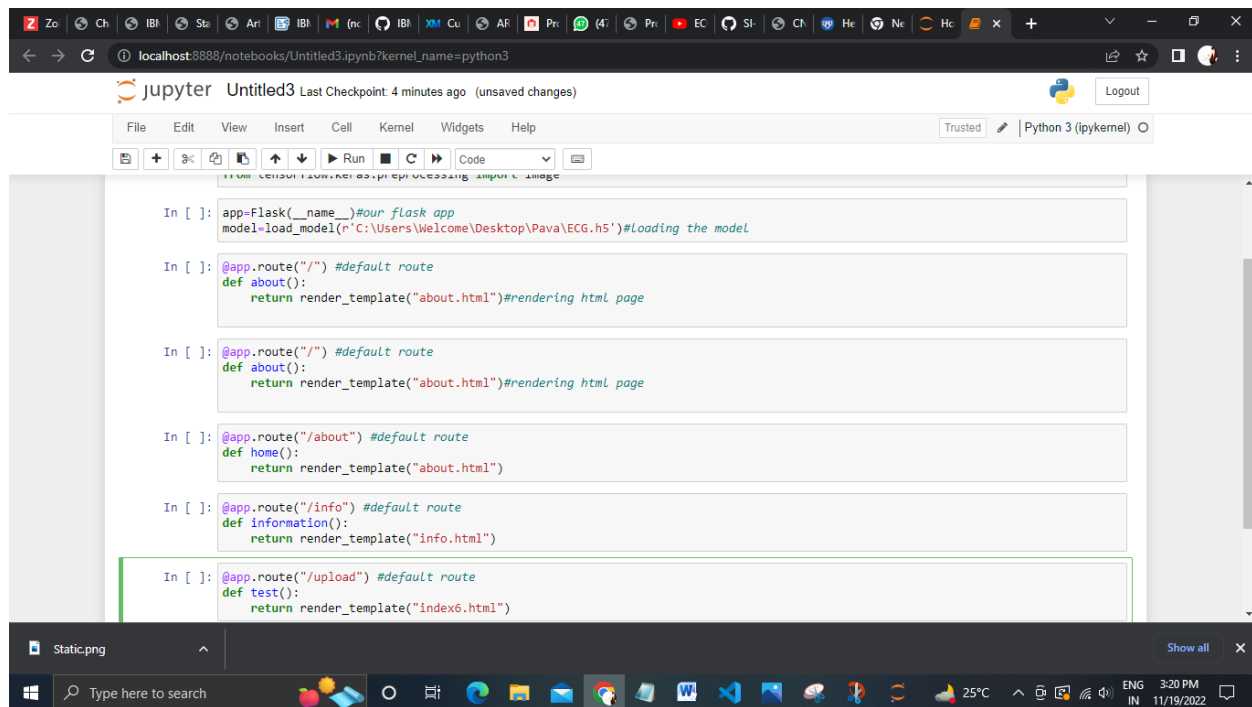


The screenshot shows a Jupyter Notebook interface in a web browser. The browser address bar shows 'localhost:8888/notebooks/Untitled3.ipynb?kernel_name=python3'. The Jupyter interface has a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The main area displays a code cell with the following Python code:

```
In [ ]: import os
import numpy as np
from flask import Flask, request, render_template
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

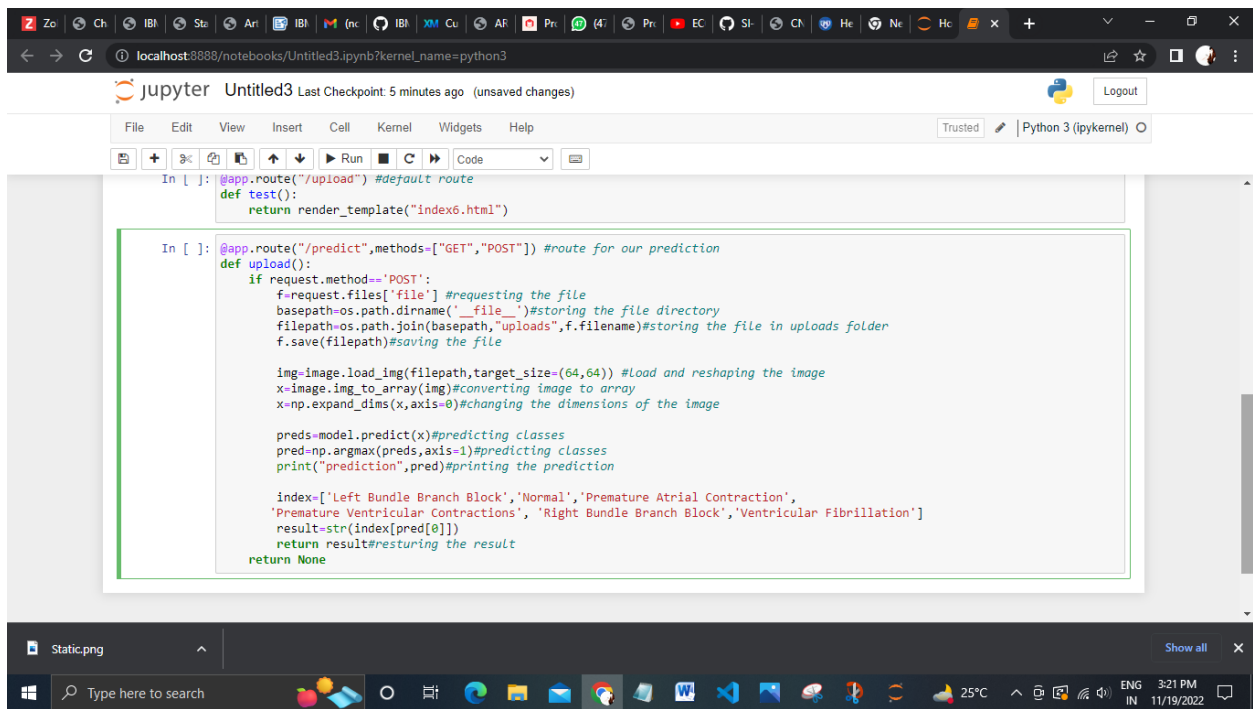
The bottom of the image shows a Windows taskbar with various application icons, a search bar, and system tray information including temperature (25°C), time (3:16 PM), and date (11/19/2022).

ROUTING TO THE HTML PAGE



Showcasing prediction on UI

When the image is uploaded, it predicts the category of uploaded the image is either 'Left Bundle Branch Block', 'Normal', 'Premature Atrial Contraction', 'Premature Ventricular Contractions', 'Right Bundle Branch Block', 'Ventricular Fibrillation'. If the image predicts value as 0, then it is displayed as “Left Bundle Branch”. Similarly, if the predicted value is 1, it displays “Normal” as output and so on.



RUN THE APP

