# Code Implementation

The chapter describes the step-by-step procedure for developing a peculiar region recognition system in Python. The implementation procedure is divided into the following sections: image database creation, Python environment setup, development of convolutional neural network with VGG16 architecture, and web application development for the specific region.

## 4.1 Creating Image Database

The Image database is required for the development of the Vgg16 model was obtained from the Kaggle dataset repositories, which is provided by the link: <https://www.kaggle.com/datasets/shriaman/periocular>

The image database is represented in two folders, which are named LEFT and RIGHT that contained peculiar region of left and right eye respectively. Each of the folders are further divided into subfolders that are named with person identifier. The figures below describe the structure of the dataset folders

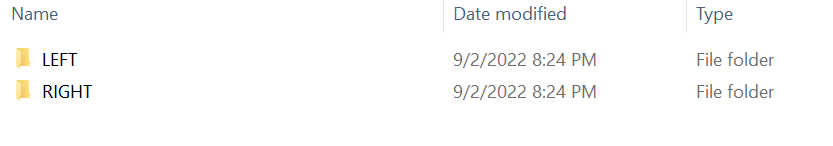


Figure 1 : Description of the dataset folder

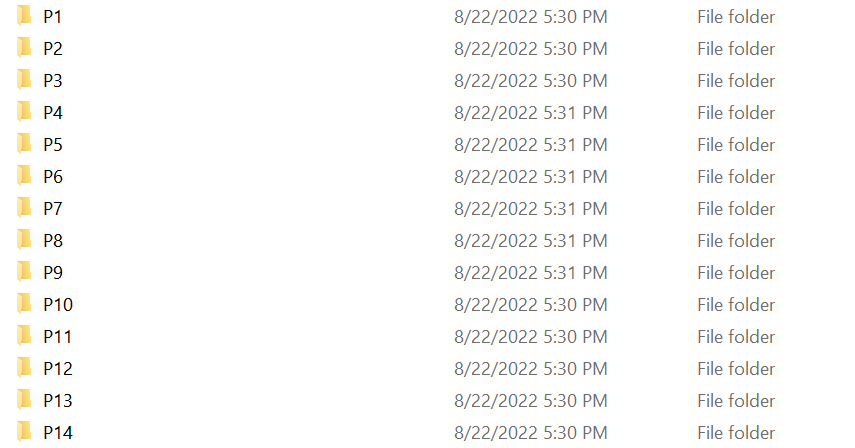


Figure 2 : Description of the sub folders in LEFT and RIGHT

## 4.2 Setting up environment

The first step carried out is to install the necessary programming tools, which includes the installation of the Anaconda software and the necessary libraries. The anaconda software provides the Jupyter Notebook application and Spyder application, which is used to write the python codes for this project. The important libraries for this project such as numpy, pandas and matplotlib, tensorflow, keras, dlib and flask were installed using the Pip tools and conda tools. The figure 3 shows the libraries imported into the environment that are used for training the model

## 4.3 Development of Peculiar Region Recognition Model

The model used for the classification of the peculiar region was developed using the convolutional neural network, a crucial deep learning algorithm. The algorithm is implemented in the Python environment using Python functions, libraries, and codes. The figure 3 displays the significant libraries used.

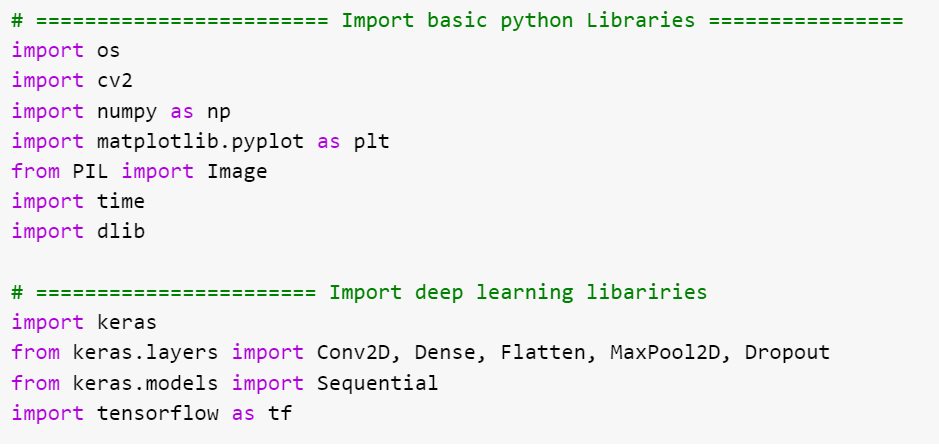


Figure 3 : Importing necessary libraries into the environment

This section described the procedure for the development of the deep learning can be described in the following stages: preprocessing stage, training stage, evaluation stage and saving the model.

### 4.3.1 Preprocessing Stage

As seen in figures 4 and 5, the first step in the preprocessing stage entails importing the image folders and files using the OS library into the Python environment (Jupyter notebook). Then, as shown in figure 6, the image files were read as image arrays using the OpenCV library, and the labels (person identifiers) were sorted to correspond with each image array.

The array that was extracted as shown in figure 7 was used to identify the peculiar regions using the 68 facial landmark predictor algorithm.

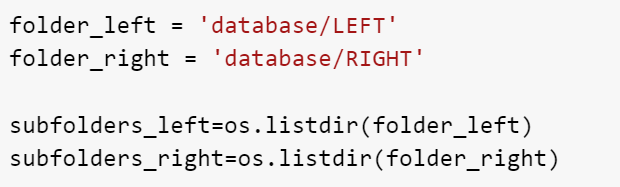


Figure 4 : Loading image folders in the python environment



Figure 5 : Loading image files to python environment

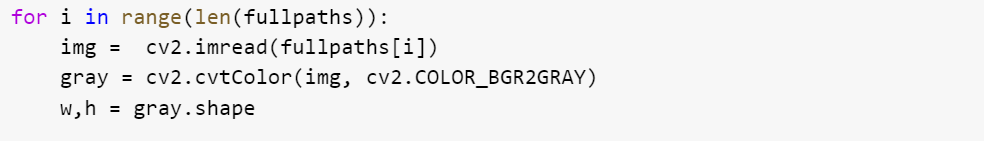


Figure 6 : Reading the image files as image array

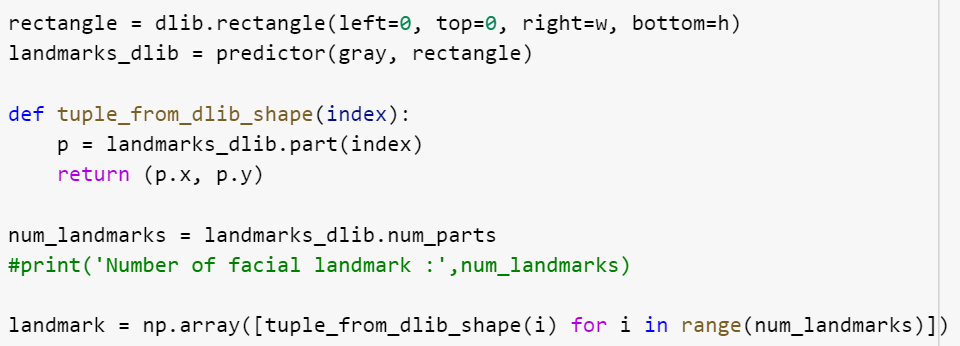


Figure 7 Detection of facial landmark from image array

The image array is normalized by dividing by the highest grayscale value, which is 255, to produce an array between the range of 0-1 as shown in figure 8. This prepares the image array for the training stage. The accuracy of training and validation is anticipated to increase with normalization.

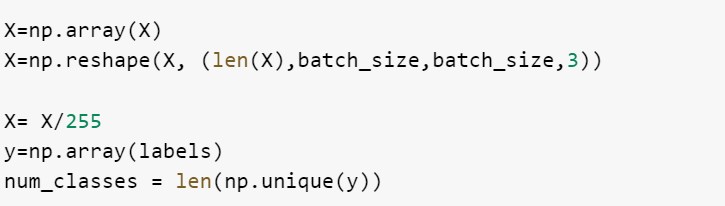


Figure 8 : Scaling the image array to improve accuracy

Furthermore, the preprocessing stage also includes cross validation method, which is used to split the image array into training set and testing set. This is achieved by creating training array and testing array which is made up of 80% and 20% of the image array. In other to ensure that the splitting well balanced, 20% data is taken from each person folder to form the 20% testing set while keeping 80% for training set. The figure 9 describes the python code implementation for splitting the dataset into training set and testing set.

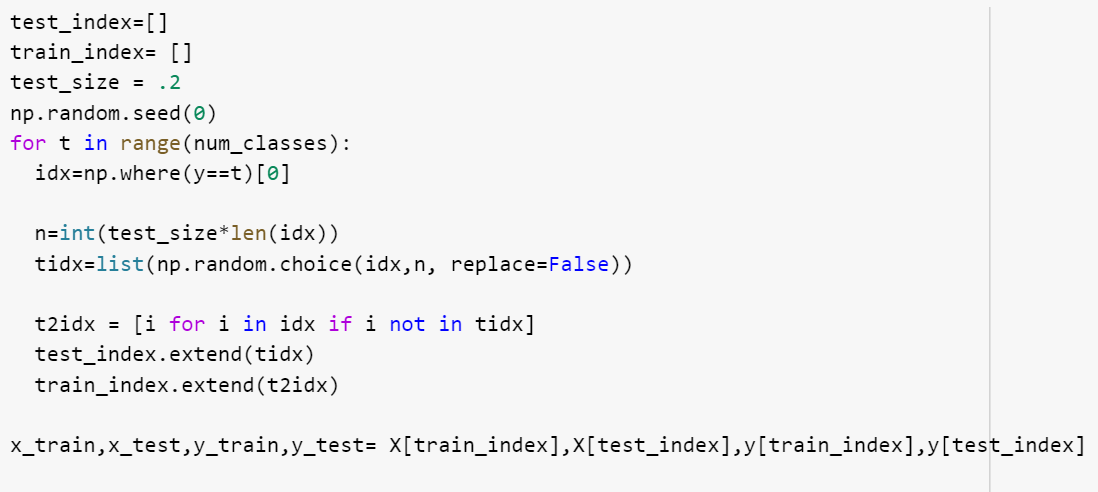


Figure 9 : Splitting the dataset into 80% training and 20% testing

## 4.3.2 Training Stage

A simple CNN architecture consisting of a sequential layer, a maxpooling layer, and a dense layer was found to be efficient for the classification of unusual regions. The image array prepared in the preprocessing stage is fed into a convolutional neural network code in this training stage. The model was built with the Adam optimizer and the loss function sparse categorical cross entropy. The CNN model can now be configured for multi-class classification. Figure 10 depicts the code implementation in the Python environment step by step.



Figure 10 : Training the model for classification of peculiar region using CNN

### 4.3.3 Evaluation Model Performance

The model was evaluated by plotting the result from training history, this includes training accuracy, visualization accuracy, training loss and validation loss.

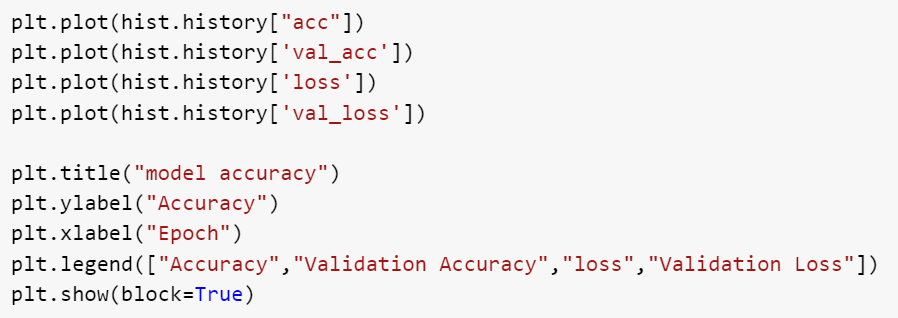


Figure 11 : Presentation model evaluation performance in plots

### 4.3.4 Saving the Model

In this step, the convolutional neural network model (cnn) developed in 4.3.2 is saved as h5 weight for subsequent usage. The model is required for the recognition of peculiar region in the web application.

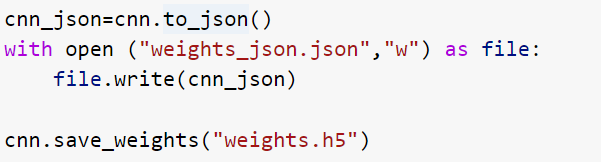


Figure 12 : Saving Model for the peculiar region recognition

## 4.4 Development of Web Application for the Peculiar region Recognition

This section discusses the step by step implementation of the peculiar region recognition web application. The web application provides an interface for the detection and recognition of peculiar region. The development was done with the flask web development framework using python programming language. The web application includes two dashboards or pages, which is described below:

### 4.4.1 Dashboard for Adding images to database

This dashboard is designed to provide the features for the extraction of peculiar region of a person image to the database. It provides two options for providing input images, which includes uploading the person image from the local storage and capturing the person image using the camera. The data processing pipeline in this section aids the detection and extraction of region of interest from face in person image and the extraction of peculiar region, which includes image array of left eye and the right eye. The peculiar region is cropped and dumped into the database, the dashboard also provides an input area to specify the person identifier, which is used to save the image in the database.

### 4.4.2 Dashboard for recognition of peculiar region

The dashboard for face recognition was designed with the same features in the dashboard for adding images to the database such as detection and extraction of peculiar region from image of a person uploaded and live camera. However, the recognition features was also added to classify the peculiar region detected to a person id in the database.

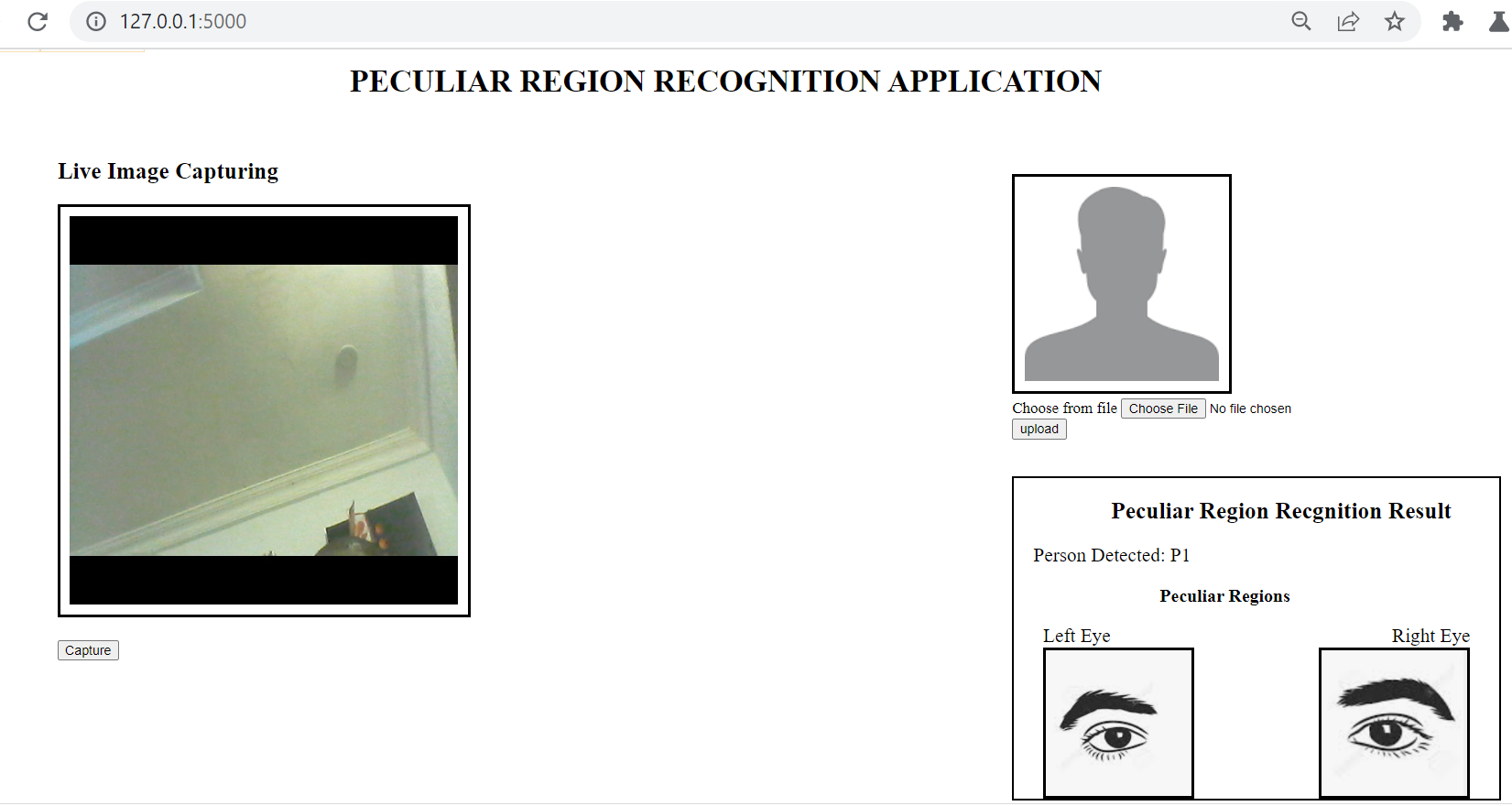
### 4.4.3 Architecture of the Web Application

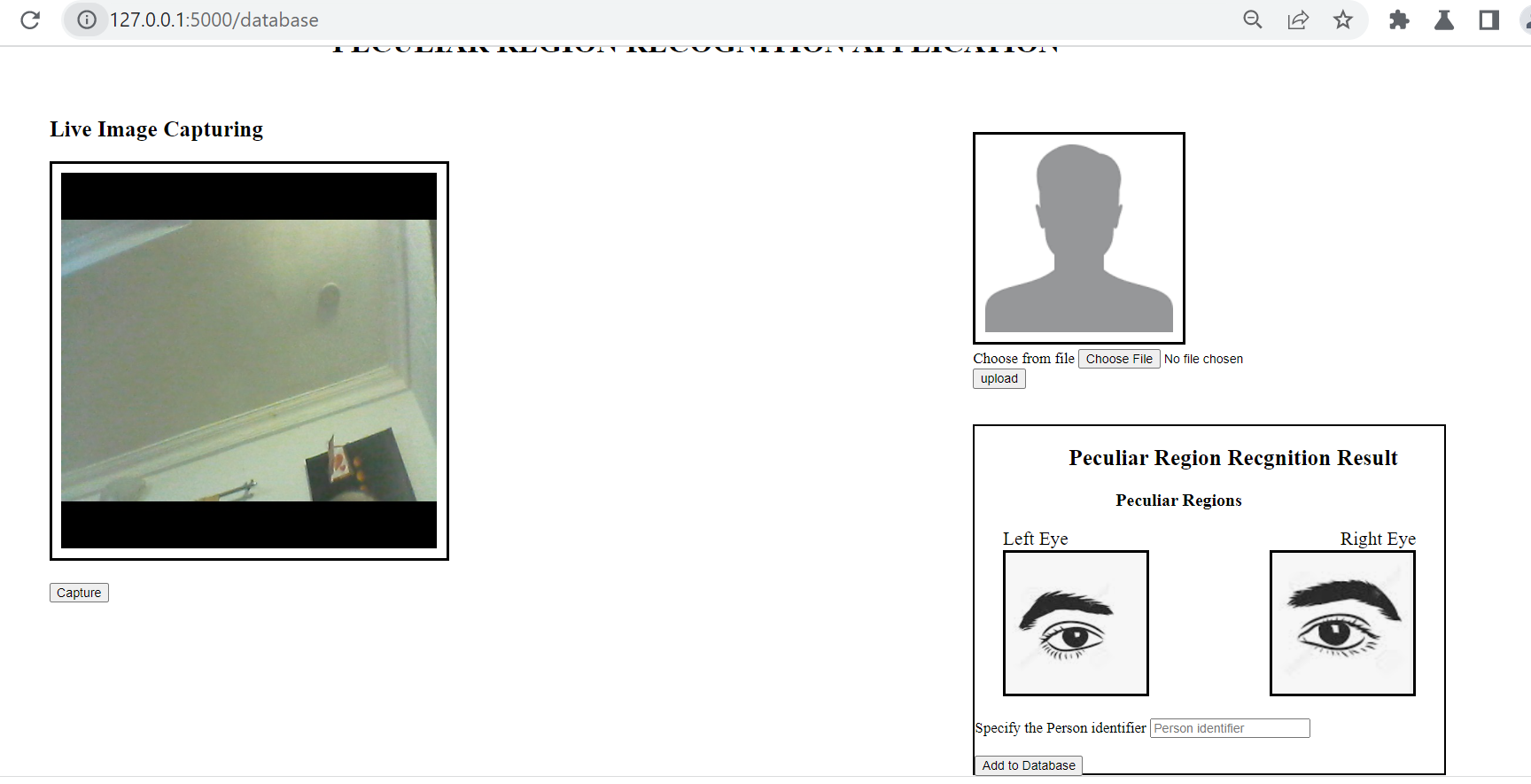
The web application is developed with the Flask in python, which is a micro web framework that provides tools and technologies for creating web pages. The architecture of the web application was designed to ensure it perform all the functions efficiently. The necessary files and folder and that formed the web application and their functions are described below :

1. app.py : This is a python file that contain the backend python codes that controls the web application for the peculiar region detection and recognition
2. facecaptured.py : This is the python file that provides code that detect person face from image using the Haar cascade algorithm. This file is used by the app.py for the detection of face from image uploaded or captured by the camera
3. haarcascade\_frontalface\_default.xlm : This is the Haar cascade file that detect person face from image captured or uploaded into the web application. The file is required by the Haar cascade algorithm developed by the codes in facecaptured.py
4. shape\_predictor\_68\_face\_landmarks.dat : This file is considered as the model used by the app.py for facial landmark detection on detected face. It is used to produces key points representing the peculiar regions on person face
5. templates : The template folder contains the hypertext markup language (HTML) files used to create the two web pages (the page for adding image to database and the page for recognition)
6. static : The template folder contains all the static images used to design the web application
7. database : This is the folder that contains all images of person peculiar region stored for recognition.
8. pec.py : This is a python file that contain the code used for training convolutional neural network model with VGG16 architecture
9. weight.h5 : This file is the VGG16 model produced by the training code (pec.py), it is used by the app.py for the recognition of the peculiar regions
10. Requirements.txt : This file contains the python dependencies that is required for setting up the web application.

### 4.4.4 Setting Up web application for peculiar region detection and recognition

The web application can be executed by running the app.py file using a python interpreter such as the Anaconda or Pycharm. The figure below shows the two pages in the code





# Experimental Results

The experimental setup was created with the goal of determining the effectiveness of the unique region recognition system. As a result, the CNN model was trained and evaluated using images from the dataset as well as images captured of a specific region of a person's face using the web application.

In the figure below, the experimental results include training accuracy, training loss, validation accuracy, and validation loss. The plots show that the model's accuracy improves for both training and testing data.

