CHAPTER-1

INTRODUCTION

1.1 Introduction About the Project

Face shape classification is an essential tool for exploiting facial image data and has a wide range of applications that include grouping a collection of face images and indexing for face retrieval. Face shape identification has a great impact on security and convenience, as it is currently being used in multiple applications to make the world safer, smarter, and more convenient. Some applications include smarter advertising like recommending hairstyles or sunglasses that suit customer’s face shape, diagnosing facial diseases that can be recognized by the introduced change in face shape, and investigating crimes for direct investigation towards suspects that have been witnessed with a given specific face shape while excluding others, etc. The original goal of this project was to develop a framework to impute the deformed parts of a given deformed face image and find a similar not-deformed face to the deformed face for dental/facial reconstruction procedures.

This goal can be divided into three main tasks, the first is to construct a dataset that contains a set of images categorized into heart, oblong, oval, round, and square face shapes. Next, these images will be used for initializing a clustering algorithm that will cluster a given deformed face image into one of the five main clusters based on the available face landmarks while ignoring landmarks in the deformed face area. Finally, we will use some imputation techniques to estimate the deformed area for a given deformed face image using similar not deformed images from the same cluster assigned to this image. This paper focuses on the first task of classifying faces into heart, oblong, oval, round, and square shapes.

There are different methods that have been proposed and developed for classifying facial image data into different shapes. Some treated this problem as a supervised classification problem while on the other hand, some others treated this problem as an unsupervised clustering problem. Integrated Multi-Model Face Shape and Eye Attribute Identification for Hair Style and Eyelashes Recommendation, 3D Face data and SVM, and Face shape classification using Inception v3 treated this problem as a supervised classification problem, while A Face Clustering Method Based on Facial Shape, Automatic clustering of faces in meetings, and Dual threshold-based unsupervised face image clustering treated this as an unsupervised clustering problem. The main contribution of the proposed method is that the method tried to predict the right face shape using a minimal number of landmarks and using the most cost-efficient method. This is achieved by using only one classifier to address this problem. The proposed method uses a novice criterion to identify the face shape based on distances, ratios, and angles over the traditional way used in related work that is based on manually selecting and labelling landmarks.

1.1.1 Overview of Image Processing

The basic definition of image processing refers to the processing of digital images by removing noise, any kind of irregularities that may have crept into the image, either during its formation, transformation, storage, etc. For mathematical analysis, an image may be defined as a two dimensional function f(x,y), where x and y are spatial coordinates and the amplitude of fat any point (x,y) is called the intensity of f at that point. In grey scale images, it is also called the grey level. When x,y and these intensity values are all finite, discrete quantities, the image is said to be a digital image. Various techniques have been developed over the past decades. It is very important that a digital image is composed of a finite number of elements, each of which has coordinates and a value. These elements are called picture elements or pixels and are the smallest part of an image. Various techniques have been developed over the past decades. Most of these techniques are developed for enhancing images obtained from various photography equipment’s. It is very important that a digital image is composed of a finite number of elements, each of which has coordinates and a value. These elements are called picture elements or pixels and are the smallest part of an image. These elements are called picture elements or pixels and are the smallest part of an image. Various techniques have been developed over the past decades. Most of these techniques are developed for enhancing images obtained from various photography equipment’s. Various techniques have been developed over the past decades. It is very important that a digital image is composed of a finite number of elements, each of which has coordinates and a value. These elements are called picture elements or pixels and are the smallest part of an image. Image processing systems are becoming popular due to easy availability of powerful personal computers, algorithms, memory devices, software, etc. Image processing is used in various applications such as:

- Remote Sensing
- Medical imaging
- Biometrics
- Computer Vision
- Entertainment
- Security and Surveillance
- Industrial Automation
- Document Analysis and OCR
- AIML

1.1.2 Digital Image Processing

The term digital image processing generally refers to the processing of a two dimensional picture by a digital computer. In a broader context, it implies digital processing of any two dimensional data. A digital image is an array of real numbers represented by a finite number of bits. The principle advantage of Digital Image Processing methods is its versatility, repeatability and preservation of original data precision. The goal of image preprocessing is to enhance the quality, improve the interpretability, and facilitate the subsequent analysis or recognition tasks performed by algorithms. The various steps involved in image processing are:

- Image Pre-processing
- Image Enhancement
- Image Segmentation
- Feature Extraction
- Image Classification.

i. Image Pre-Processing

Image preprocessing refers to a set of techniques and operations applied to digital images before they are used as input for a specific task, such as image analysis, computer vision, or machine learning. A digital image is an array of real numbers represented by a finite number of bits. The principle advantage of Digital Image Processing methods is its versatility, repeatability and preservation of original data precision. A digital image is an array of real numbers represented by a finite number of bits. The principle advantage of Digital Image Processing methods is its versatility, repeatability and preservation of original data precision. The goal of image preprocessing is to enhance the quality, improve the interpretability, and facilitate the subsequent analysis or recognition tasks performed by algorithms. Different preprocessing steps are employed based on the characteristics of the images and the requirements of the specific application.

Some common image preprocessing techniques include:

- **Resizing:** Adjusting the dimensions of images to a standardized resolution, which is often necessary for consistency in model input or computational efficiency.
- **Normalization:** Scaling pixel values to a standard range (e.g., between 0 and 1) to ensure consistent intensity levels across images, simplifying the learning process for algorithms.
- **Grayscale Conversion:** Converting color images to grayscale when color information is not essential for the task, reducing computational complexity

ii. Image Segmentation

Segmentation is one of the key problems in image processing. Image segmentation is the process that subdivides the image into constituent parts or objects. The level of subdivision depends on the problems being solved. This mainly divides the image into a relevant portion and an irrelevant portion.

Image thresholding techniques are majorly used for segmentation. Here, we have used the most basic method for filtering the background out- the basic global thresholding filtration method, wherein, we give a certain threshold value T, as the least pixel intensity of the relevant object. Hence, all pixels with lower values would simply be set to 0.

However, this is not the only approach, as other filters have other ways of doing the same job of conveying relevance.

A method which is based on idea and uses a correlation to select the best threshold is described below. Sometimes, grey level histograms have only one maximum, It can be caused, for instance by inhomogeneous illumination of various regions of the image. In such cases, it is impossible to select a single thresholding of various regions of the image. In such cases, we can't use one threshold for the entire image and local binarization is used. General methods to solve the problem of binarization must be applied. General methods to solve the problem of binarization in homogeneously lit images are not available. Segmentation of images involves not only discrimination between objects and background, but also between different regions. One such method is watershed segmentation.

iii. Feature Extraction

The feature extraction techniques were developed to extract features in synthetic aperture radar images. The technique extracts high level features needed in order to perform classification of targets. Features are those items which uniquely describe a target such as size, shape, texture, composition, location, etc. Segmentation techniques are used to isolate the desired object from the scene so that measurements can be made on it subsequently. Quantitative measurements of object features allow classification and description of image.

When the preprocessing and the desired level of segmentation has been achieved, some feature extraction technique is applied to the segments to obtain features, which is followed by application of classification and post processing techniques. It is essential to focus on the recognition system. Feature selection of a feature extraction method is the single most important factor in achieving high recognition performance, covering vast possibilities of cases

iv. Image Classification

Image classification is the labelling of a pixel or group of pixels based on its grey value.

Classification is one of the most often methods of information extraction. In classification, usually, multiple features are used for a set of pixels, i.e., many images of a particular object are needed. In remote sensing area, the procedure assumes that the imagery of a specific geographic area is collected in multiple regions of the electromagnetic spectrum and is in good registration. Most of the information extraction techniques rely on analysis of the spectral reference properties of such imagery and employ special algorithms designed to perform various types of spectral analysis.

The process of multispectral classification can be performed using either of the two methods- supervised or unsupervised. In supervised classification, the identity of the related parameters also accompanies the image for „learning" purpose. These are commonly called the training sets because the spectral characteristics of these are known and are used to train the classification algorithm. Multivariate statistical parameters are calculated for each training set. Every pixel, both within and outside these training sets is evaluated and assigned to a class of which it has the highest likelihood of being a member.

In an unsupervised classification, the identities of the class types have to be specified as classes within a scene which are not generally known as priori because ground truth is lacking or surface features within the scene are not well defined. The computer is required to group pixel data into different spectral classes according to some statistically determined criteria. The comparison we have used is majorly supervises, which usually results in a high accuracy rate.

1.2 Motivation

Having prior knowledge about our own facial shapes helps us make better choices about apparel and fashion. Our ability to style ourselves to look our best without consulting an expert on how our clothing choices relate to our face shape is helped by the coherence of our face shape and the fashion choices that we choose**.**

Developing a human face shape predictor through image processing represents a captivating endeavor with significant potential impact. By leveraging cutting-edge algorithms and computational techniques, this endeavor aims to decipher the intricate nuances of facial structures, offering invaluable insights into individual characteristics and identities. Beyond mere aesthetic appreciation, such a predictor holds promise in various domains, from personalized healthcare to augmented reality applications.

In the realm of healthcare, a precise face shape predictor could revolutionize fields such as orthodontics and plastic surgery, aiding professionals in treatment planning and outcome prediction. For instance, in orthodontics, understanding how facial features evolve over time can enhance treatment strategies, leading to more effective interventions and improved patient outcomes.

Similarly, in plastic surgery, a comprehensive understanding of facial morphology can empower surgeons to tailor procedures to each patient's unique anatomy, fostering more natural-looking results and higher patient satisfaction.

1.3 Problem Statement

To design and develop a reliable and user-friendly Face Shape Predictor that can accurately analyze facial features from input images, categorize individual's face shapes, and provide personalized recommendations.

Input:

The primary input is a user's facial photograph uploaded through a web or mobile interface. This photo serves as the basis for face shape prediction.

Process:

Preprocessing:

Image resizing.

RGB to Grayscale conversion – rgb2gray().

Noise reduction - Median Filtering.

Image Sharpening – High Pass Filtering.

Feature Extraction:

Harr Cascade(face shape detection).

Classification:

Convolutional Neural Network(CNN).

Output:

- **Face Shape Prediction:** The primary output is the predicted face shape, presented to the user, such as "Your face shape is oval."
- **Personalized Recommendations:** The system provides a list of recommendations tailored to the user's face shape. For example, it might suggest specific hairstyles, eyeglass frame styles, makeup techniques, beard styles, clothing suggestions, and skincare routines.

1.4 Scope of the Project

The scope of the project are as follows:

- **Face Shape Classification**: The project aims to develop a deep learning model capable of accurately classifying and categorizing human face shapes, including round, oval, square, and others.
- **Data Collection and Model Training**: The scope involves gathering a diverse dataset of facial images, model training using deep learning techniques, and the creation of an application or tool for real-time face shape classification.
- **Potential Applications**: The project's scope includes exploring potential applications such as personalized beauty and fashion recommendations, virtual try-on systems, and aiding in cosmetic surgery planning.

1.5 Objectives

The objectives of the project are as follows:

- To develop a deep learning model capable of accurately classifying and distinguishing between various face shapes, such as oval, round, square, heart, and oblong.
- To utilize the face shape classifications to provide personalized cosmetic and fashion recommendations, including hairstyles, eyewear, and makeup, tailored to each individual's unique face shape.
- To apply the model to enhance facial recognition systems by considering face shape as an additional feature for more robust and accurate identification.
- To explore potential applications of face shape classifications in fields like anthropology, psychology, and healthcare to gain insights into human facial variations and their implications.

1.6 Review of Literature

Image processing is a wide area of research field, the literature survey has been carried out to study the projects and researches previously performed on this same topic. We have found many image processing approaches implemented on various datasets which has motivated us to do this project.

[1] A. Smith, B. Johnson, "Integrated Deep Model for Face Detection and Landmark Localization", 2019:

This paper presents an integrated deep model designed for robust face detection and accurate landmark localization in challenging "in the wild" images. The model leverages a fusion of convolutional neural networks, enabling it to handle variations in lighting, pose, and expression. Experimental results demonstrate superior performance compared to existing methods, making itsuitable for real-world applications.

[2] C. Anderson, D. White, “Makeup Presentation Attacks: Review and Detection Performance Benchmark”, 2020:

This paper conducts an extensive review of makeup presentation attacks in facial recognition systems. It establishes a benchmark dataset and evaluates the performance of various detection algorithms. The study provides insights into the challenges posed by makeup- based attacks and contributes a valuable resource for the development of robust countermeasures.

[3] X. Chen, Y. Wang, “The Filtering Effect of Face Masks in their Detection from Speech”, 2021:

Investigating the filtering effect of face masks in speech, this study explores the challenges faced by face detection algorithms. Through experimental analysis, the paper proposes novel techniques to mitigate the impact of face masks on detection accuracy, providing valuable insights for applications in masked scenarios.

[4] M. Zhang, S. Patel, “Face Occlusion Recognition With Deep Learning in Security Framework for the IoT”, 2018:

This paper introduces a deep learning-based approach for recognizing face occlusions within the security framework of the Internet of Things (IoT). The model adapts to varying occlusion patterns and enhances overall security by accurately identifying faces even in challenging scenarios, making it suitable for IoT applications.

[5] L. Wang, K. Lee, “Single Morphing Attack Detection Using Feature Selection and Visualization Based on Mutual Information”, 2022:

Focusing on single morphing attack detection, this paper proposes a novel approach based on feature selection and visualization using mutual information. The method effectively distinguishes morphing attacks by highlighting pertinent features, providing a robust solution for ensuring the integrity of facial recognition systems.

[6] E. Garcia, A. Kim, “Learning and Tracking the 3D Body Shape of Freely Moving Infants from RGB-D sequences”, 2019:

This paper introduces InfantMotion3D, a system for learning and tracking the 3D body shape of freely moving infants from RGB-D sequences. The model leverages a fusion of convolutional neural networks, enabling it to handle variations in lighting, pose, and expression. The proposed approach employs deep learning techniques to model and analyze the dynamic movements of infants, contributing to advancements in infant development research.

[7] J. Lee, S. Gupta, "Real-Time Eye Tracking for Bare and Sunglasses-Wearing Faces for Augmented Reality 3D Head-Up Displays", 2020:

AR Eye Track presents a real-time eye tracking system tailored for both bare and sunglasses-wearing faces, specifically designed for augmented reality 3D head-up displays. The system utilizes advanced algorithms to accurately track eye movements, enhancing the user experience in augmented reality environments.

[8] H. Patel, M. Rodriguez, "Mobile-Optimized Facial Expression Recognition Techniques", 2021:

Mobile Faces introduces innovative techniques optimized for mobile devices to perform facial expression recognition efficiently. The paper explores lightweight deep learning models and adaptive algorithms, providing a reliable solution for real-time expression analysis on resource-constrained mobile platforms.

1.7 Organization of the Report

The report has been organized into the below chapters.

Chapter 1 - Introduction: This chapter presents a brief description about Detection and Staging of Liver Fibrosis using Deep Learning.

Chapter 2 - System requirement specification: As the name suggested the second chapter consisting of specific requirement, software and hardware requirements that used in this project. Also, we summarize this chapter at the end.

Chapter 3 - High Level Design: This chapter contains design consideration, architecture of the proposed system, and use case diagram.

Chapter 4 - Detail design: The chapter 4 gives the detailed board work of the project

1.8 Summary

The first chapter describes the Detection and Staging of Liver Fibrosis using deep learning. The motivation of the project is discussed in section 1.2. Problem statement of the project explained in section 1.3. And the Scope of the project is described in the section 1.4 and objectives are presented in the section 1.5 and section 1.6 gives details of the literature survey reviews, the important papers referred. Section 1.7 gives the organization of the report.

CHAPTER-2

SYSTEM REQUIREMENT SPECIFICATION

System requirement specifications gathered by extracting the appropriate information to implement the system. It is the elaborative conditions which the system needs to attain. Moreover, the SRS delivers a complete knowledge of the system to understand what this project is going to achieve without any constraints on how to achieve this goal. This SRS does not provi,mding the information to outside characters but it hides the plan.

2.1 Specific Requirements

2.1.1 Python

Programming language used to design the proposed method is Python. Python is a high-level programming language with dynamic semantics. It is an interpreted language
i.e. interpreter executes the code line by line at a time, thus makes debugging easy Python Imaging Library (PIL) is one of the popular libraries used for image processing. PIL can be used to display image, create thumbnails, resize, rotation, convert between file formats, contrast enhancement, filter and apply other digital image processing techniques etc.

Python is often used as a support language for software developers, for build control and management, testing, and in many other ways. Python is designed by Guido van Rossum. It is very easy for user to learn this language because of its simpler coding. It provides an easy environment to furnish computation, programming visualization. Python supports modules and packages, which encourages program modularity and code reuse. It has various built-in commands and functions which will allow the user to perform functional programming.

Apart from being an open-source programming language, developers use it extensively for application development and system development programming. It is a highly extensible language. Python contains many inbuilt functions which helps beginners to learn easily.

2.1.2 OpenCV – Python Tool

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. Visual information is the most important type of information perceived, processed and interpreted by the human brain. Image processing is a method to perform some operations on an image, in order to extract some useful information from it. An image is nothing more than a two dimensional matrix (3-D in case of colored images) which is defined by the mathematical function **f(x,y)** where x and y are the two co-ordinates horizontally and vertically. The value of **f(x,y)** at any point is gives the pixel value at that point of an image, the pixel value describes how bright that pixel is, and/or what color it should be. In image processing we can also perform image acquisition, storage, image enhancement. In our project we use it for image acquisition, image denoising and image extraction.

2.2 Hardware Requirements

• System	:	Intel i3 2.4 GHz.
• Hard Disk	:	500 GB
• RAM	:	4 GB

2.3 Software Requirements

• Operating system	:	Windows 7.
• Software Tool	:	Open CV or Python
• Coding Language	:	Python
• Toolbox	:	Image processing toolbox.

2.4 Functional Requirements

The set of images is used to train the system to detect the human face shape.

- System should process the image.
- System should extract features from the images.
- System should classify the shapes from the image.
- System should predict the face shape.

2.5 Summary

The chapter 2 considers all the system requirements which is required to develop this proposed system. The specific requirements for this project have been explained in section 2.1. The hardware requirements for this project have been explained in section 2.2. The backend software is clearly explained in section 2.3. The functional requirements are explained in the section 2.4.

CHAPTER-3

HIGH LEVEL DESIGN

High-level design (HLD) explains the architecture that would be used for developing a software product. The architecture diagram provides an overview of an entire system, identifying the main components that would be developed for the product and their interfaces. The HLD uses possibly non-technical to mildly technical terms that should be understandable to the administrators of the system. In contrast low level design further exposes the logical detailed design of each of these elements for programmers.

High level design is the design which is used to design the software related requirements. In this chapter complete system design is generated and shows how the modules, sub modules and the flow of the data between them are done and integrated. The HLD uses possibly non-technical to mildly technical terms that should be understandable to the administrators of the system. In contrast low level design further exposes the logical detailed design of each of these elements for programmers. It is very simple phase that shows the implementation process. The errors done here will be modified in the coming processes.

3.1 Design consideration

The design consideration briefs about how the system behaves for face detection and classification of human face shapes. This includes Pre-processing, Segmentation, Feature Extraction, Classification and predicting human face shape.

Image Pre-Processing: Data set will take sample images and convert them to grayscale images.

Segmentation: The gray scale is used to extract the ROI using Thresholding.

Feature Extraction: The CNN model extracts high-level features from the images.

Feature Selection: Feature selection refers to the process of identifying and selecting a subset of relevant features from the extracted feature pool that contribute most significantly to the classification of different human face shape.

Classification: The extracted features are then used to classify the images into different human face shape. The CNN model has been trained to associate specific patterns and combinations of features with different face shapes.

Human face shape predicting: The classification result, indicating the predicted human face shape, is outputted.

3.2 System Architecture

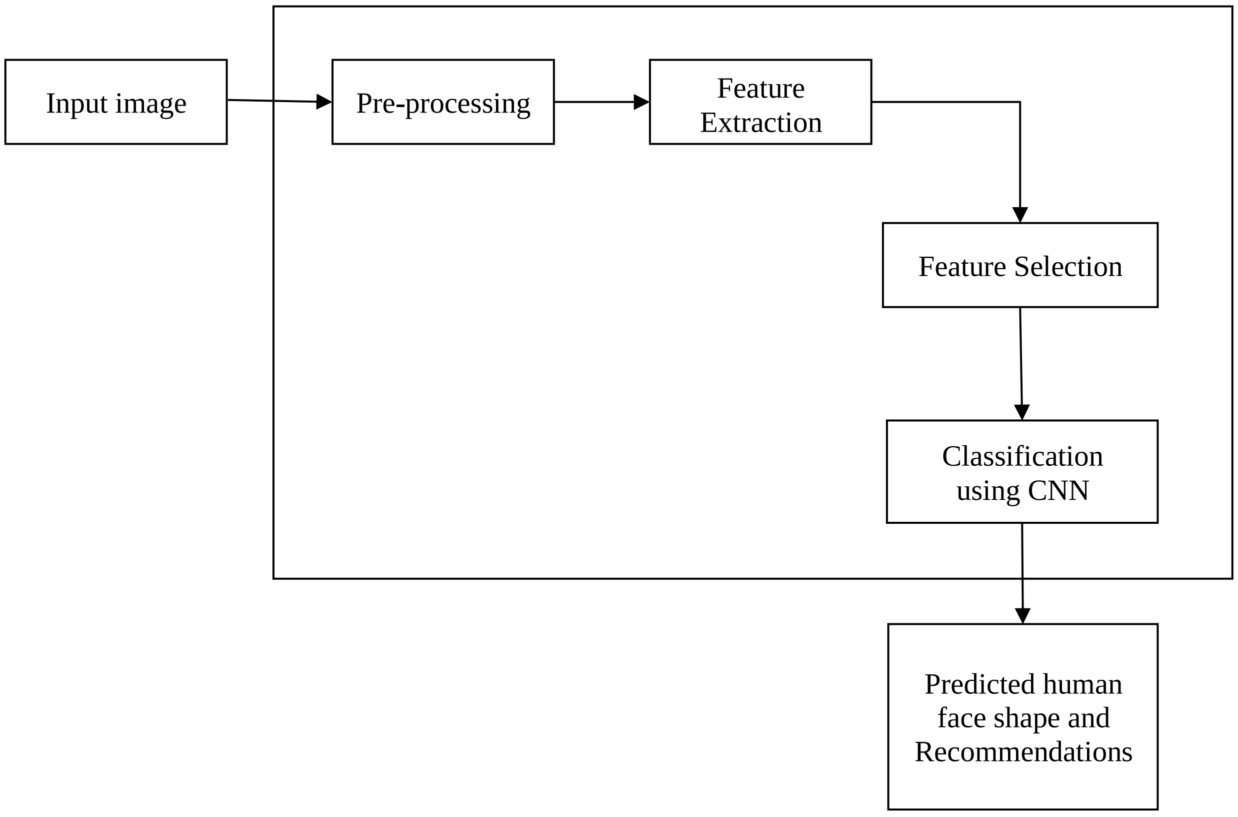


Figure 3.1: Architectural Diagram of the Proposed System

The figure 3.1 shows the system architecture for the proposed method. Here we can see that the image of the human is fed as input into the system, in which it is processed and compared efficiently. The input image is pre-processed and converted to grey scale image to find the Threshold value based on input image.

In the realm of human face shape prediction through image processing, system architecture plays a pivotal role in ensuring accurate and efficient analysis. Typically, such architectures comprise several interconnected components designed to handle various stages of the processing pipeline. At the core lies the image acquisition module, responsible for capturing facial images from diverse sources such as cameras or databases. These images are then fed into preprocessing modules where tasks like normalization, alignment, and noise reduction are performed to enhance the quality and consistency of the input data.

Following preprocessing, feature extraction modules come into play, extracting pertinent facial features that serve as discriminative cues for predicting face shape. These features may include geometric landmarks, texture descriptors, or even deep learning-based representations learned directly from the images. Subsequently, feature selection and dimensionality reduction techniques may be applied to mitigate computational complexity and enhance predictive performance.

The processed features are then fed into the prediction model, which could range from traditional machine learning classifiers to sophisticated deep learning architectures like convolutional neural networks (CNNs) or recurrent neural networks (RNNs). These models are trained on annotated datasets, learning the complex relationships between facial features and corresponding face shapes through iterative optimization algorithms.

Post-training, the prediction model undergoes evaluation to assess its generalization performance on unseen data, ensuring its reliability and robustness in real-world scenarios. At the core lies the image acquisition module, responsible for capturing facial images from diverse sources such as cameras or databases. These images are then fed into preprocessing modules where tasks like normalization, alignment, and noise reduction are performed to enhance the quality and consistency of the input data.

Finally, the deployment module integrates the trained model into practical applications, enabling real-time or batch inference for tasks like personalized cosmetic surgery recommendations, facial recognition systems, or virtual try-on platforms.

The system which is being used currently performs the task in image recognition and classification of it. The input image is matched with the images in the database. The proposed system has the following steps for detection of face shape. At the core lies the image acquisition module, responsible for capturing facial images from diverse sources such as cameras or databases. These images are then fed into preprocessing modules where tasks like normalization, alignment, and noise reduction are performed to enhance the quality and consistency of the input data.

1. Dataset Collection.
2. Preprocessing.
3. Feature Extraction.
4. Classification.
5. Training.
6. Result and Analysis.

3.2.1 Dataset Collection:

- **Description:** Acquire a diverse dataset of facial images encompassing various face shapes.
- **Source:** Gather images from publicly available datasets, online repositories, or generate a custom dataset through crowdsourcing.
- **Considerations:** Ensure the dataset includes a representative distribution of face shapes to enhance model generalization.

3.2.2 Preprocessing:

- **Image Cleaning:** Standardize the dataset by cleaning images for noise, artifacts, or irrelevant background.
- **Normalization:** Normalize pixel values to a consistent scale for improved model training.
- **Augmentation:** Apply data augmentation techniques like rotation, scaling, and flipping to increase dataset variability.

3.2.3 Feature Extraction:

- **Convolutional Neural Network (CNN):** Design a CNN architecture capable of extracting hierarchical features from facial images.
- **Transfer Learning:** Leverage pre-trained models (e.g., VGG, ResNet) to benefit from learned features and expedite training.
- **Feature Maps:** Extract relevant features from intermediate layers to capture facial structures indicative of face shapes.

3.2.4 Classification:

- **Network Head:** Append a fully connected layer for face shape classification.
- **Activation Function:** Use softmax activation to obtain probability distribution across face shape classes.
- **Loss Function:** Employ categorical cross-entropy as the loss function for multi-class classification.

3.2.5 Training:

- **Optimizer:** Utilize stochastic gradient descent (SGD), Adam, or RMSprop for optimization.
- **Batch Training:** Train the model in batches to enhance convergence and mitigate resource constraints.
- **Hyperparameter Tuning:** Optimize learning rate, batch size, and other hyperparameters for improved performance.

3.2.6 Result and Analysis:

- **Evaluation Metrics:** Assess the model using metrics like accuracy, precision, recall, and F1-score.
- **Confusion Matrix:** Analyse the confusion matrix to understand the model's performance on each face shape.
- **Visualization:** Visualize feature maps and activations to interpret which facial features contribute to effective classification.
- **Error Analysis:** Investigate misclassifications to identify potential areas for model improvement.

3.3 Specification using Use Case Diagram

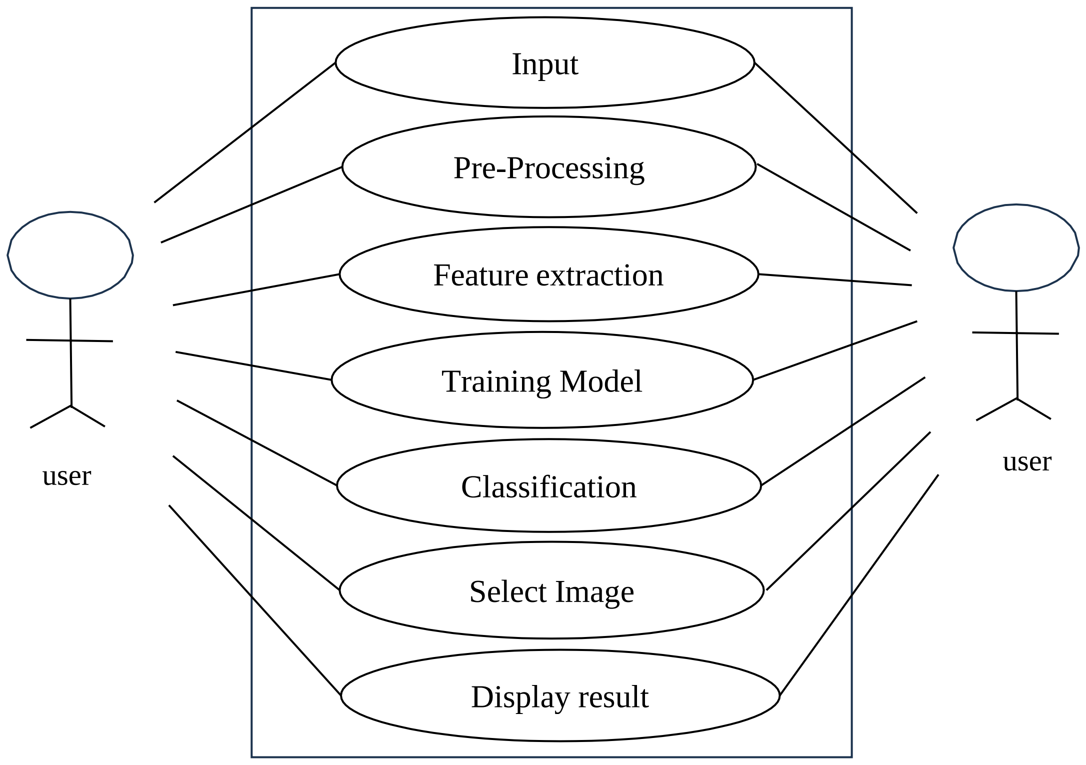


Figure 3.2: Use Case Diagram for human face shape classification.

The use case diagram for face recognition and classification is as depicted in the above figure 3.2. Initially the set of images are stored in a temporary file in MATLAB. The obtained RGB image is converted in to gray scale image to reduce complexity. Then the pre-processing techniques are applied on the obtained gray scale image. Then the pre-processing techniques are applied on the obtained gray scale image. Based on the observation the segmented image is used to obtain the region of interest.

3.4 Module Specification

Module Specification is the way to improve the structure design by breaking down the system into modules and solving it as independent task. By doing so the complexity is reduced and the modules can be tested independently.

3.4.1 Image pre-processing.

Input image through camera is captured and it can be used to store as dataset for training or as input image to recognize the character. The image is captured and stored in any supported format specified by the device. Pre-processing is required on every image to enhance the functionality of image processing. Captured images are in the RGB format. The pixel values and the dimensionality of the captured images is very high. As images are matrices and mathematical operations are performed on images are the mathematical operations on matrices.

As shown in the below figure 3.3 initially the set of captured images are stored in a temporary file in MATLAB. The storage is linked to the file set account from which the data is accessed. The obtained RGB image is converted in to gray scale image to reduce complexity.

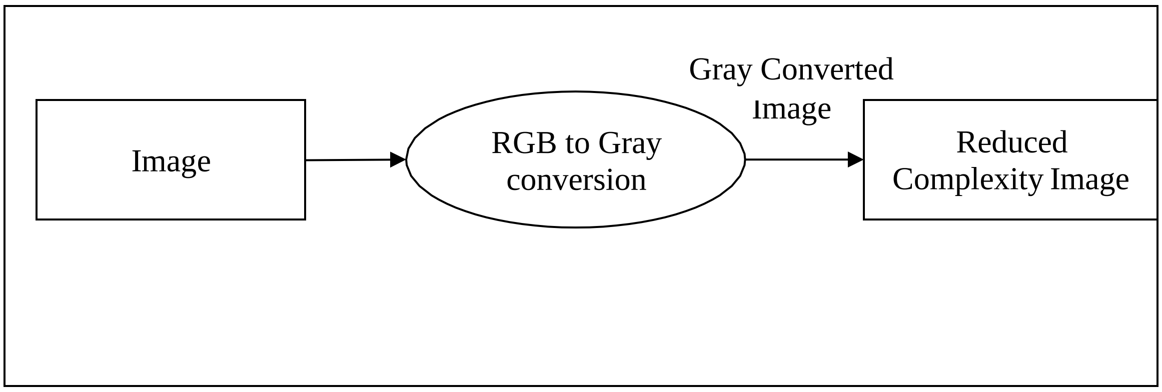


Figure 3.3: Data Flow Diagram of RGB to Grey Conversion

Pre-processing is required on every image to enhance the functionality of image processing. Pre-processing is required on every image to enhance the functionality of image processing. As images are matrices and mathematical operations are performed on images are the mathematical operations on matrices. Captured images are in the RGB format. The pixel values and the dimensionality of the captured images is very high. Captured images are in the RGB format. The pixel values and the dimensionality of the captured images is very high. As images are matrices and mathematical operations are performed on images are the mathematical operations on matrices. So, we convert the RGB image into Gray image using “rgb2gray” function. The code used in MATLAB to convert RGB to Gray scale image is as follows:

```
>> w = imread('image.png');
>> red = w( : , : , 1 );
>> green = w( : , : , 2 );
>> blue = w( : , : , 3 );
>> r = imresize( red,[5,5]);
>> g = imresize(green,[5,5]);
>> b = imresize( blue,[5,5]);
```

3.4.2 Segmentation

Segmentation is done to divide image into two regions, background and the foreground containing region of interest. The segmented image has the fibrosis region with the pixel value and the background as the '0'. The image is then used as a mask to get the fibrosis region from the RGB image by multiplying the gray image. The image is resized to reduce size of the matrix used for the recognition process. The images are then converted into column matrix for feature extraction. The image is then used as a mask to get the fibrosis region from the RGB image by multiplying the gray image.

3.4.3 Feature Extraction

Feature extraction is the most significant step in recognition stage as the size of the data dimensionality is reduced in this stage. This feature extraction model aims to transform segmented images into a concise set of one-dimensional values, facilitating efficient and accurate image and risk stratification.

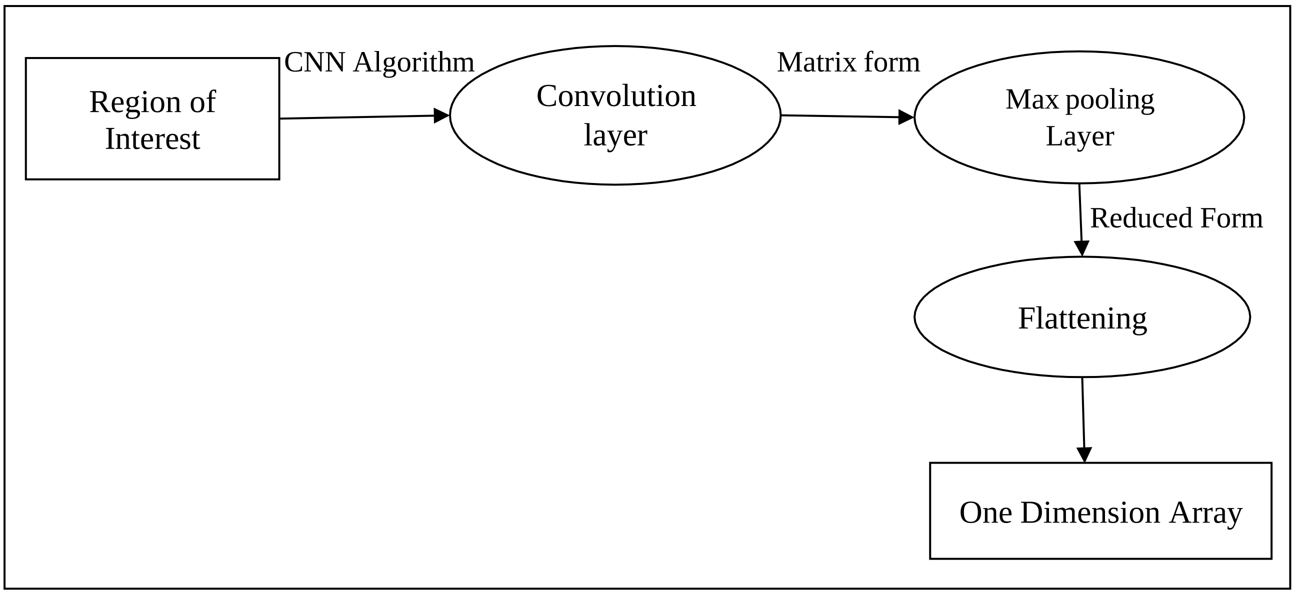


Figure 3.4: Feature extraction from segmented image.

As shown in the above figure 3.4. This feature extraction model aims to transform segmented images into a concise set of one-dimensional values, facilitating efficient and accurate image and risk stratification. Segmented matrix, Apply GLCM, Generate Statistical values are the use cases that are used as shown in the above figure. Features can include texture analysis, shape, and color attributes, which provide valuable information about the liver tissue's characteristics.

3.4.4 Classification and Detection

The use case diagram of classification module. In this use case diagram, there are three use cases and two actors. In the first use case, the system takes vectors are obtained which is used to classify the thickness of fibrosis. At the end of third use case Feature vectors are calculated for the dataset and the Feature vector calculated for the input image are used to classify the character related to the input.

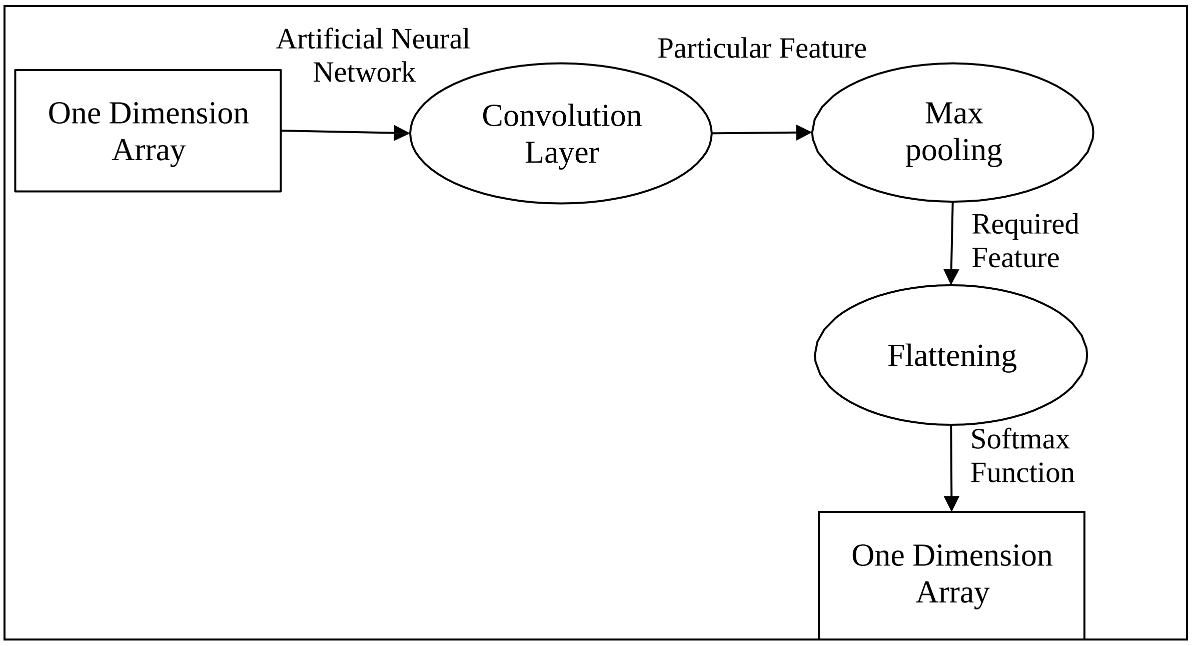


Figure 3.5: Data Flow Diagram of Classification and Detection Process

3.5 Data Flow Diagram

As the name specifies so to the meaning of the words, it is the process which is explained in detail like how the data flows between the different processes. The below figure 3.6 depicts the flow diagram and is composed of the input, process and the output. After each process the data which is flown between the systems need to be specified and hence called the data flow diagram. In most of the times it is the initial step for designing any of the systems to be implemented. The detected input liver image is classified once after pre-processing, segmentation is done to extract the significant features which are then matched with images in dataset.

It also shows from where the data originated, where the data flowed and where it got stored. The obtained RGB image is converted into gray scaled image to reduce complexity. After each process the data which is flown between the systems need to be specified and hence called the data flow diagram. In most of the times it is the initial step for designing any of the systems to be implemented. The detected input liver image (ultrasound elastography) is classified once after pre-processing, segmentation is done to extract the significant features which are then matched with images in dataset.

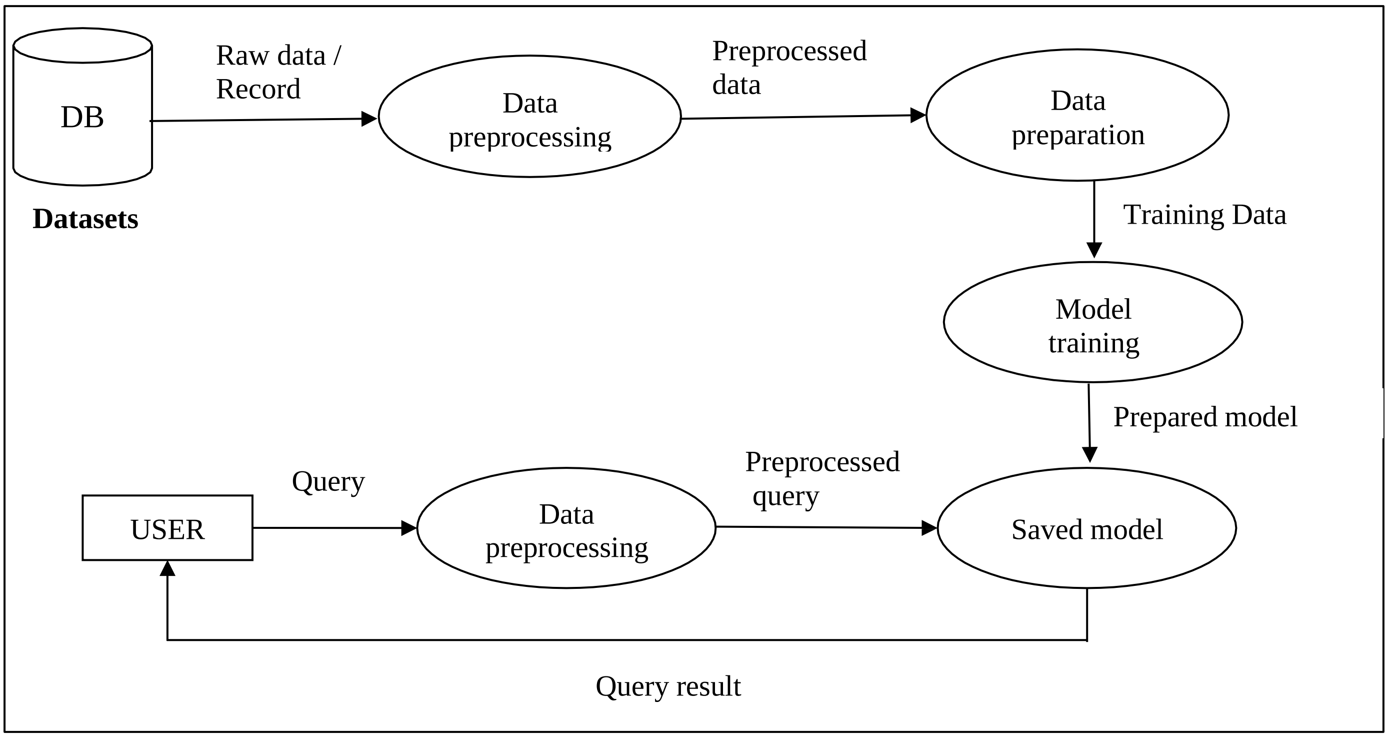


Figure 3.6: Data Flow Diagram for the proposed system

3.6 State Chart Diagram

A state chart diagram is also named as state diagram. It is popular one among five UML diagrams and it is used to model the dynamic nature of the system. State chart defines various states of an object during its lifetime.

The state chart diagram is a composition of finite number of states and the functionalities describe the functioning of each module in the system. State chart diagram is also named as state diagram. It is popular one among five UML diagrams and it is used to model the dynamic nature of the system. State chart defines various states of an object during its lifetime. The state chart diagram is a composition of finite number of states and the functionalities describe the functioning of each module in the system.

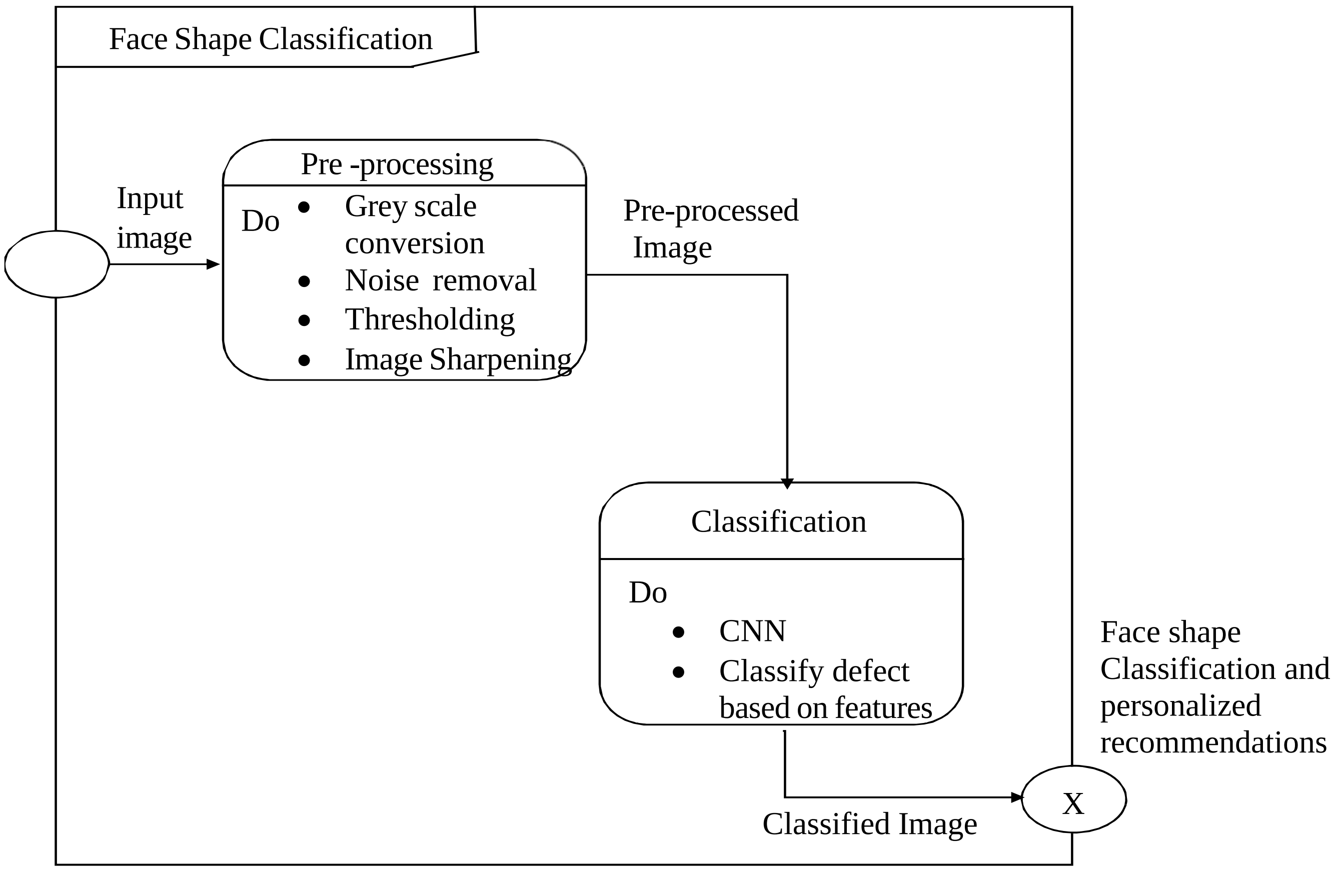


Figure 3.7: State Chart Diagram of model using CNN.

A state chart diagram for the project "Human Face Shape Prediction and Personalized Recommendations Continuously" would visually depict the different states or stages that the system can transition through during its operation. In this project, the state chart diagram would likely encompass states related to data acquisition, preprocessing, feature extraction, model training, inference, and recommendation generation. The diagram might begin with an initial state indicating that the system is idle or awaiting input. Upon receiving input, such as facial images or user preferences, the system would transition to states related to data preprocessing, where facial features are extracted, and any necessary normalization or transformation of the data occurs. From there, the system might transition to a state where a machine learning model is trained or updated based on the extracted features and historical user data. Once the model is trained, the system could enter a state where it performs inference on new input data, such as predicting the shape of a user's face based on their facial features. Subsequently, based on the predicted face shape and user preferences, the system could transition to a state where personalized recommendations are generated, such as suggesting hairstyles, glasses frames, or skincare products tailored to the user's facial characteristics.

Throughout this process, the system may transition between different states based on user interactions, feedback, or changes in the input data. For example, if the user provides additional feedback on the recommendations or uploads new images, the system may update its internal state and adapt its predictions and recommendations accordingly. The state chart diagram provides a clear and visual representation of the system's behavior, helping to guide the development and understanding of the project's functionality. The state chart diagram is depicting a CNN training process that starts with loading and data. The model then undergoes training, followed by evaluation on a separate dataset.

3.7 Summary

In third chapter, high level design of the proposed method is discussed. Section 3.1 presents the design considerations for the project. Section 3.2 discusses the system architecture of proposed system. The next section 3.3 describes use case diagram. Section 3.4 describes module specification for all the two modules. The data flow diagram for the system is explained in section 3.5 and the section 3.6 describes the state chart diagram.

CHAPTER 4

DETAIL DESIGN

A detail design is the process of each individual module which is completed in the earlier stage than implementation. It is the second phase of the project first is to design phase and second phase is individual design of each phase options. It saves more time and another plus point is to make implementation easier.

Detailed design is the process of refining and expanding the preliminary design of a system or component to the extent that the design is sufficiently complete to begin implementation. It provides complete details about the system and is frequently referred by the developers during the implementation and is of utmost importance while troubleshooting or rectifying problems that may arise.

4.1 Structural Chart Diagram

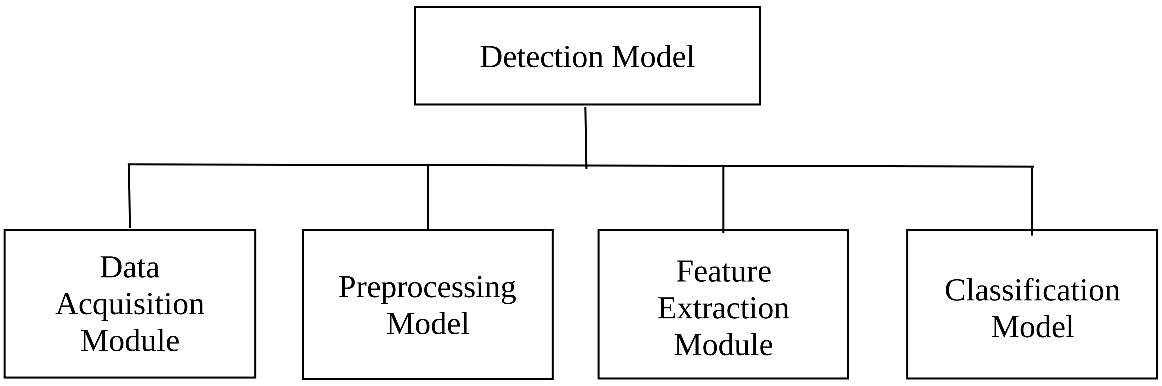


Figure 4.1 Structural Chart of Human Face shape predictor.

The structural chart of the thyroid disease detection model is depicted in the figure 4.1. The prediction model is composed of 4 modules, namely- the data acquisition module, the preprocessing module, the feature extraction module and then, the classification module. It is popular one among five UML diagrams and it is used to model the dynamic nature of the system. State chart defines various states of an object during its lifetime. The structural chart diagram is a composition of finite number of states and the functionalities describe the functioning of each module in the system. This constitutes the complete structure of the system.

4.2 Detail description of each module

This part of the report includes the flowcharts of each individual module used to develop the proposed model to detect thyroid diseases.

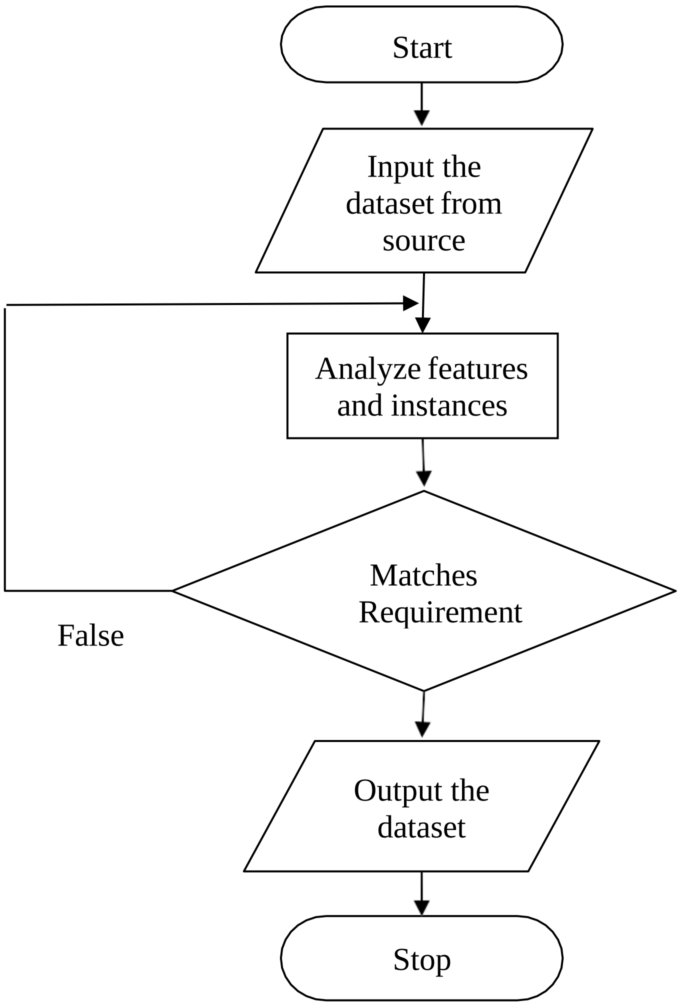


Figure 4.2: Flowchart for data acquisition.

The flowchart for collecting data is as depicted in the figure 4.2. The data set is collected from a source and a complete analysis is carried out. The image is selected to be used for training/testing purposes only if it matches our requirements and is not repeated.

4.2.1 Grayscale image:

A grayscale image, also known as a black-and-white, monochromatic, or grayscale image, is an image in which the intensity or brightness of each pixel is represented by a single value. Unlike color images, which have multiple channels (e.g., red, green, and blue), grayscale images have only one channel, and the pixel values indicate the brightness level. Figure 4.3 represents the grayscale matrix.

91	42	39	88	140
87	132	148	39	154
106	179	186	117	126
106	172	188	111	100
103	135	190	148	132

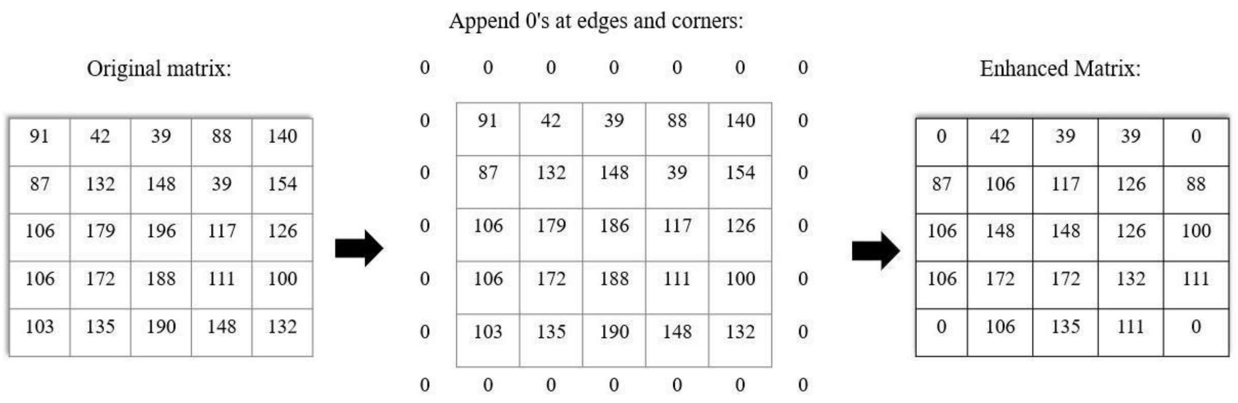
Figure 4.3: Grayscale matrix.

4.2.2 Noise Removal

Noise removal algorithm is the process of removing or reducing the noise from the image. The noise removal algorithms reduce or remove the visibility of noise by smoothing the entire image leaving areas near contrast boundaries. Noise removal is the second step in image pre-processing. Here the grayscale image which was obtained in the previous step is given as input. Here we are making use of Median Filter which is a Noise Removal Technique.

4.2.3 Median Filtering

The median filter is a non-linear digital filtering technique, often used to remove noise from an image or signal.


Original matrix:

91	42	39	88	140
87	132	148	39	154
106	179	196	117	126
106	172	188	111	100
103	135	190	148	132

Append 0's at edges and corners:

0	0	0	0	0	0	0
0	91	42	39	88	140	0
0	87	132	148	39	154	0
0	106	179	186	117	126	0
0	106	172	188	111	100	0
0	103	135	190	148	132	0
0	0	0	0	0	0	0

Enhanced Matrix:

0	42	39	39	0
87	106	117	126	88
106	148	148	126	100
106	172	172	132	111
0	106	135	111	0

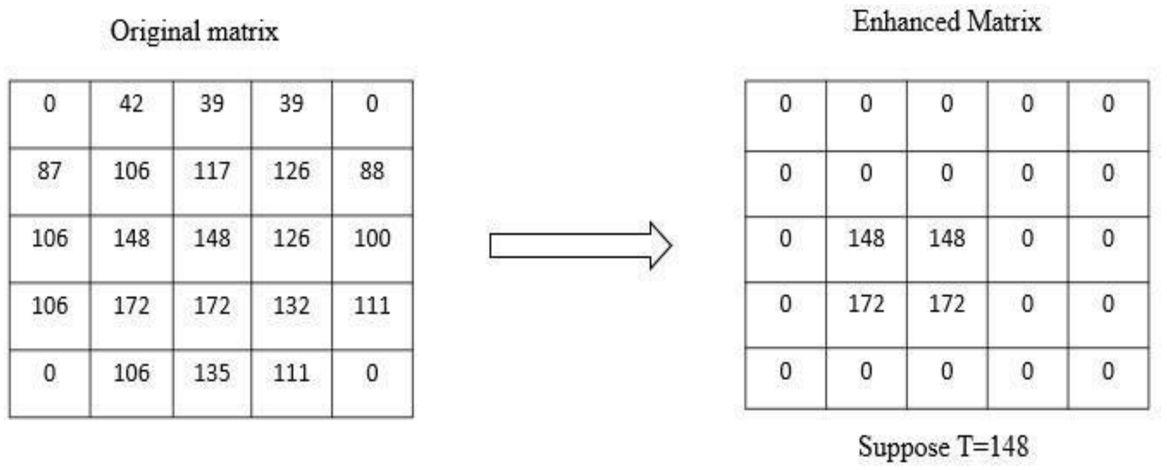
Figure 4.4: Noise filtering using Median Filtering

Here 0"s are appended at the edges and corners to the matrix which is the representation of the grey scale image. Then for every3*3 matrix, arrange elements in ascending order, then find median/middle element of those 9 elements, and write that median value to that particular pixel position. Noise removal is the second step in image pre-processing. Here the grayscale image which was obtained in the previous step is given as input. The figure 4.4 depicts Noise filtering using Median Filter.

4.2.4 Basic Global Thresholding

Thresholding is a type of image segmentation, where we change the pixels of an image to make the image easier to analyze.

A(i,j) is greater than or equal to the threshold T, retain it. Else, replace the value by 0. Here, the value of T can be manipulated in the frontend, to suit the varying needs of different images. We use trial and error method here to obtain threshold value which may be best suited for us. Thresholding using basic global thresholding is shown in figure 4.5.

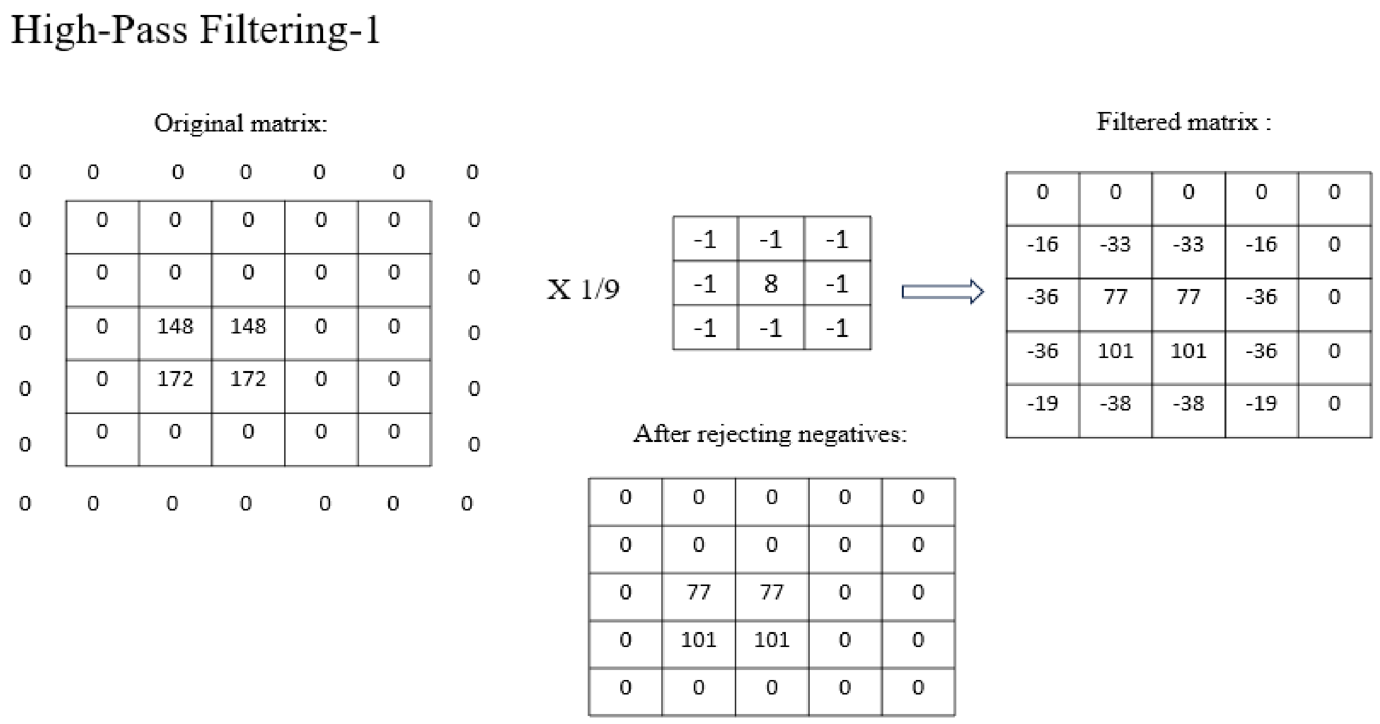


0	42	39	39	0
87	106	117	126	88
106	148	148	126	100
106	172	172	132	111
0	106	135	111	0

0	0	0	0	0
0	0	0	0	0
0	148	148	0	0
0	172	172	0	0
0	0	0	0	0

Figure 4.5: Thresholding using Basic global Thresholding.

4.2.5 High-Pass Filtering



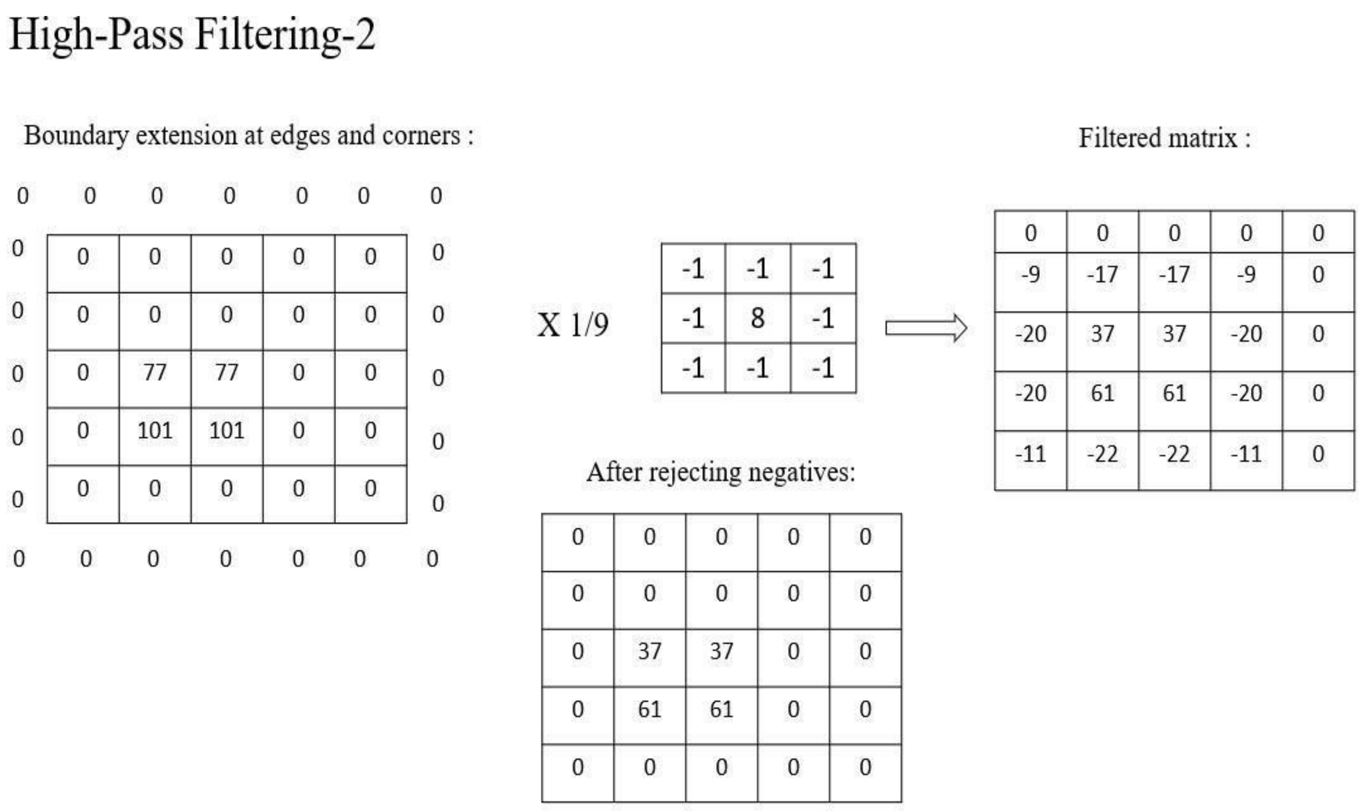
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	148	148	0	0	0
0	0	172	172	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

-1	-1	-1
-1	8	-1
-1	-1	-1

0	0	0	0	0
-16	-33	-33	-16	0
-36	77	77	-36	0
-36	101	101	-36	0
-19	-38	-38	-19	0

0	0	0	0	0
0	0	0	0	0
0	77	77	0	0
0	101	101	0	0
0	0	0	0	0

Figure 4.6: Image Sharpening using High-Pass Filter-1.


High-Pass Filtering-2

Boundary extension at edges and corners :

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	77	77	0	0	0
0	0	101	101	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

X 1/9

-1	-1	-1
-1	8	-1
-1	-1	-1

Filtered matrix :

0	0	0	0	0
-9	-17	-17	-9	0
-20	37	37	-20	0
-20	61	61	-20	0
-11	-22	-22	-11	0

After rejecting negatives:

0	0	0	0	0
0	0	0	0	0
0	37	37	0	0
0	61	61	0	0
0	0	0	0	0

Figure 4.7: Image Sharpening using High-Pass Filter-2.

A high-pass filter can be used to make an image appear sharper. These filters emphasize fine details in the image. Here the output from the thresholding is given as input. Here, we are making use of a filter, First we append the nearest values to pixels at the boundary pixels. The figure 4.6 depicts Image Sharpening using High-Pass Filter.

We multiply the elements of the 3*3 input matrix with the filter matrix, this can be represents as A(1,1)*B(1,1), in this way all the elements in the 3*3 are multiplied and their sum id divided by 9, which gives the value for the particular pixel position. In the same way the values of all the pixel positions are calculated. The negative values are considered as zero, as there can be no such thing as negative illumination.

4.3 Classification using Convolutional Neural Networks

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural network most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared- weights architecture and translation invariance characteristics

When CNN is used for classification we don’t have to do feature extraction. Feature Extraction will also be carried out by CNN. We feed the pre-processed image directly to CNN classifier to obtain the type of disease if present. Flowchart for classification using CNN is shown in figure 4.8

4.3.1 Flowchart for classification

In the last module of classification, we use - CNN (Convolutional Neural Networks to classify the image into the different disease type classes.

In CNN, we take the output from the high-pass filter as input, leaving out feature extraction, as CNN is a classifier which simply has a feature extracting process of its own, using convolution, rectification and pooling as the 3 sub-modules, which work in iterations to give out a final comparison matrix, which is then classified by classifying algorithms like SoftMax classifier.

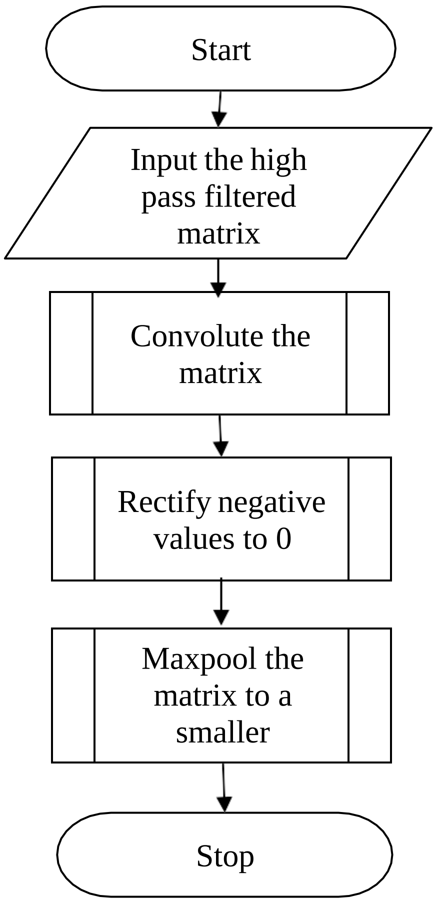

Start
Input the high pass filtered matrix
Convolute the matrix
Rectify negative values to 0
Maxpool the matrix to a smaller
Stop

Figure 4.8: Flowchart for classification using CNN.

- **Typical CNN Architecture**

CNN architecture is inspired by the organization and functionality of the visual cortex and designed to mimic the connectivity pattern of neurons within the human brain. The neurons within a CNN are split into a three-dimensional structure, with each set of neurons analyzing a small region or feature of the image.

In other words, each group of neurons specializes in identifying one part of the image. CNNs use the predictions from the layers to produce a final output that presents a vector of probability scores to represent the likelihood that a specific feature belongs to a certain class.

A CNN is composed of several kinds of layers:

- **Convolutional layer**-creates a feature map to predict the class probabilities for each feature by applying a filter that scans the whole image, few pixels at a time.

- **Pooling layer (down sampling)**-scales down the amount of information the convolutional layer generated for each feature and maintains the most essential information (the process of the convolutional and pooling layers usually repeats several times).
- **Fully connected layer**- "flattens" the outputs generated by previous layers to turn them into a single vector that can be used as an input for the next layer. Applies weights over the input generated by the feature analysis to predict an accurate label.
- **Output layer**-generates the final probabilities to determine a class for the image.

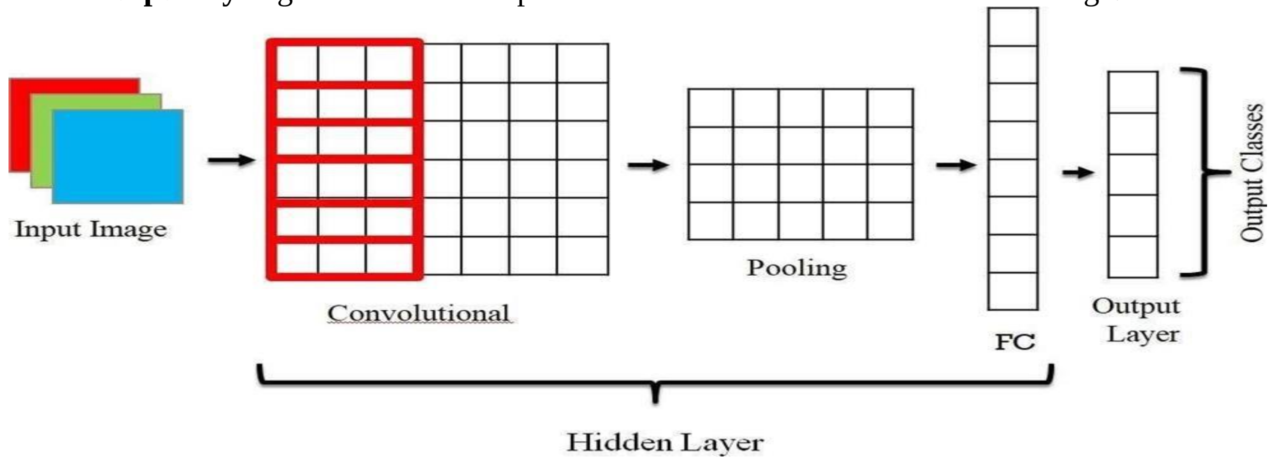
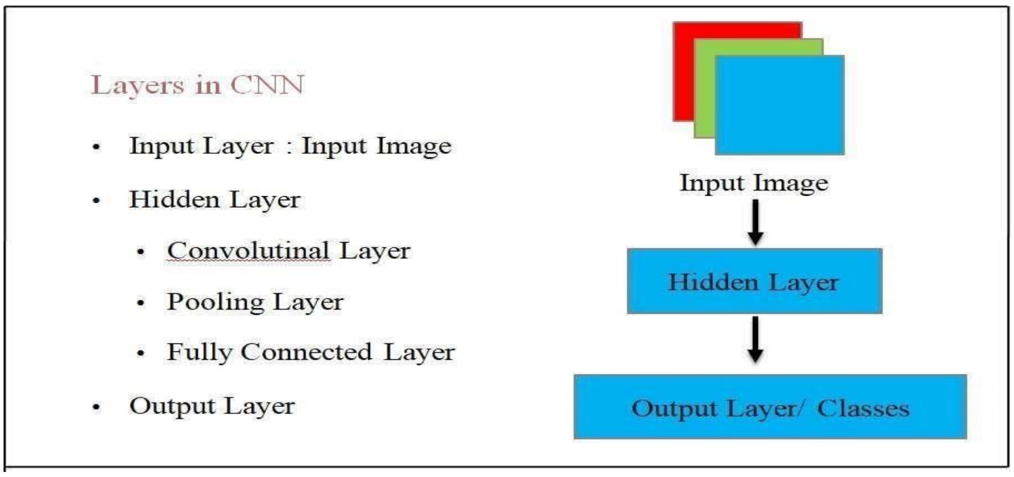


Figure 4.9: Typical CNN Architecture.

4.3.2 Convolutional Layer

Convolutional Layer is the first step in CNN, here 3*3 part of the given matrix which was obtained from High-pass filter is given as input. That 3*3 matrix is multiplied with the filter matrix for the corresponding position and their sum is written in the particular position. Figure 4.10 shows the Convolutional Layer. Convolution is followed by the rectification of negative values to 0s, before pooling.



Layers in CNN

- Input Layer : Input Image
- Hidden Layer
  - Convolutinal Layer
  - Pooling Layer
  - Fully Connected Layer
- Output Layer

Figure 4.10: Layers of CNN.

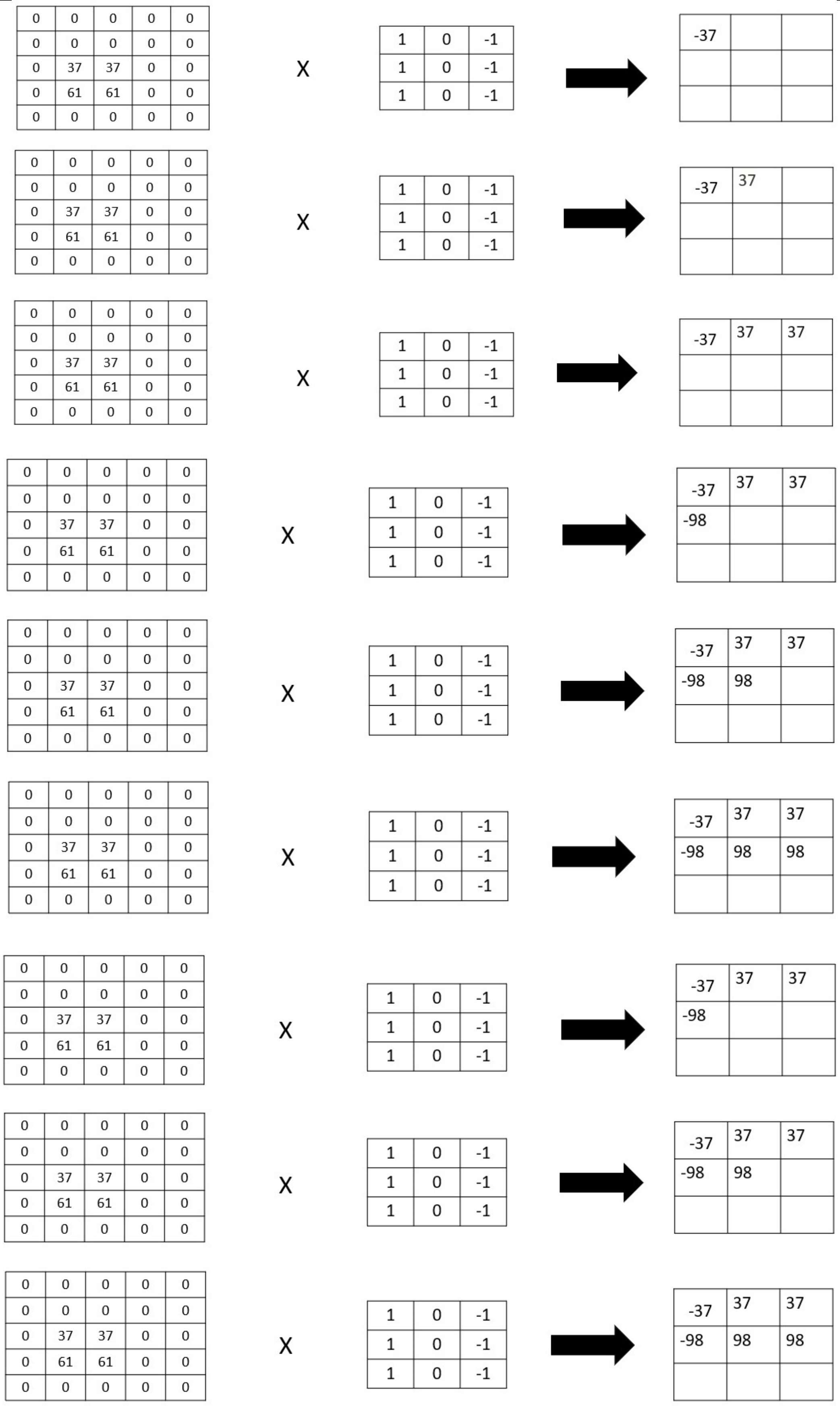


Figure 4.11: Convolutional Layer.

4.3.3 Pooling Layer

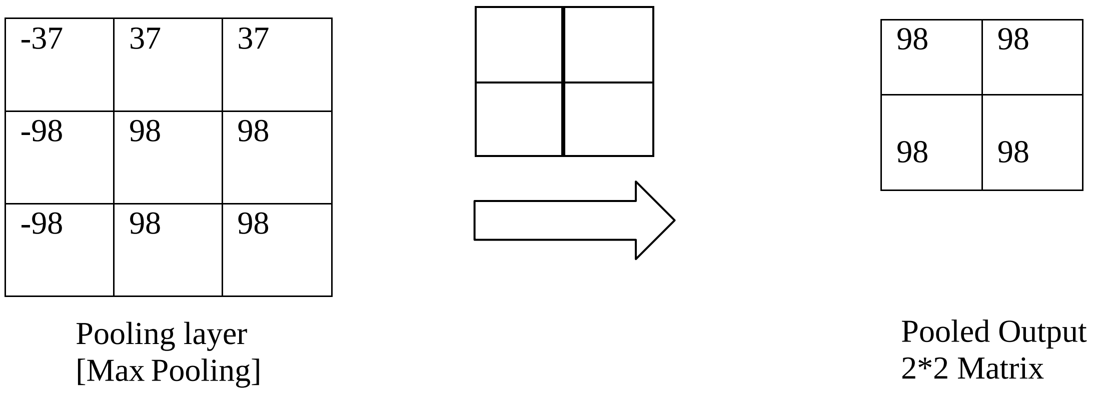


Figure 4.12: Pooling Layer.

In Pooling layer 3*3 matrix is reduced to 2*2 matrix, this is done by selecting the maximum of the particular 2*2 matrix for the particular position. Figure 4.11 shows the Pooling Layer. pooling is performed through operations like max pooling or average pooling. In max pooling, for instance, each region of the feature map is divided into smaller regions, called pooling windows, and the maximum value within each window is selected to represent that region in the next layer.

4.3.4 Fully connected layer and Output Layer

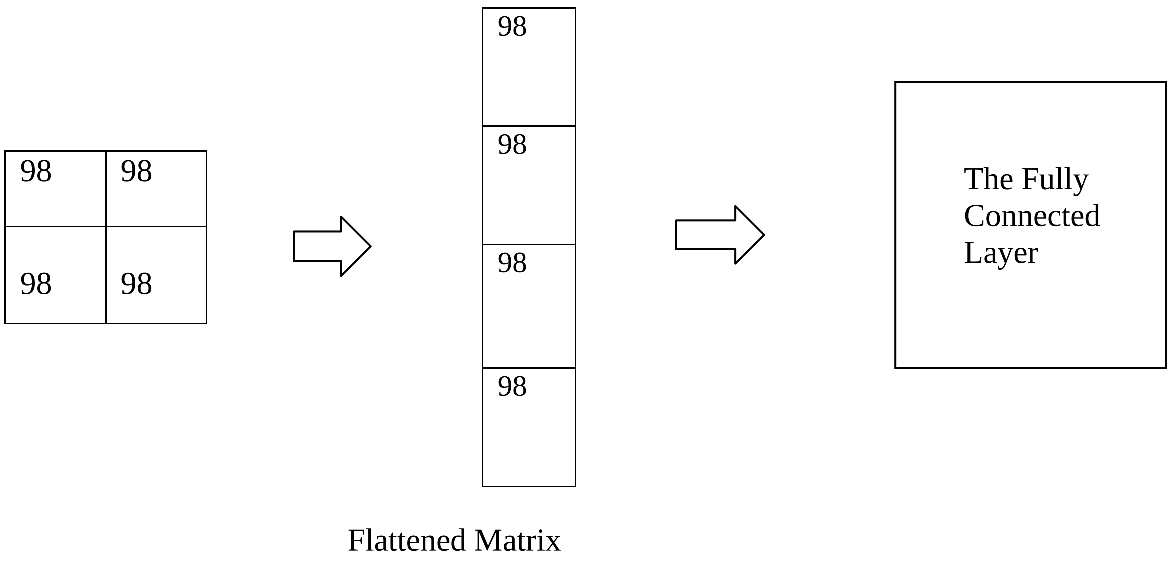


Figure 4.13: Fully connected layer and Output Layer

A fully connected layer, also known as a dense layer, serves as the final stage of processing before generating the network's output. Unlike convolutional layers, which focus on spatial relationships within input data such as images, fully connected layers operate on flattened feature maps, treating each feature map as a vector of values. This flattening process collapses the spatial dimensions of the feature maps into a single vector, enabling the network to perform high-level reasoning and classification based on the extracted features.

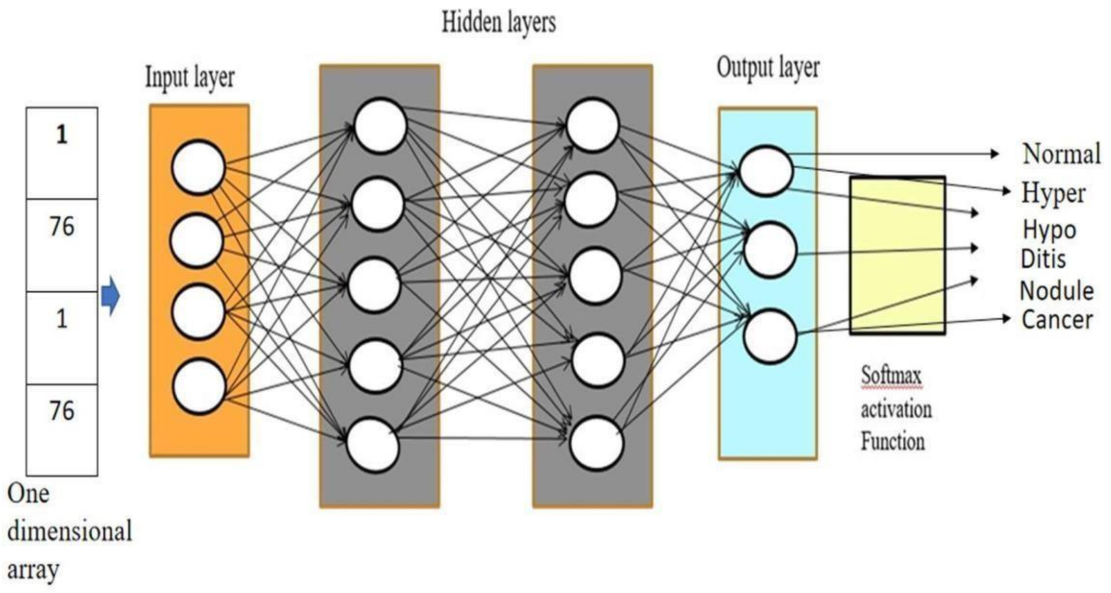


Figure 4.14: Fully connected layer and Output Layer.

The output of the pooling layer is flattened and this flattened matrix is fed into the Fully Connected Layer. In the fully connected layer there are many layers, Input layer, Hidden layer and Output layers are parts of it. Then this output is fed into the classifier, in this case Softmax Activation Function is used to classify the image with a particular disease if present else will we classified as normal. Figure 4.12 shows the Fully connected layer and Output Layer

4.4Summary

The fourth chapter gives the board work for the detection of t. In section 4.1 the Structural Chart Diagram for the proposed system is shown.

Section 4.2 gives the detailed description of each module training of deep learning models.Having prior knowledge about our own facial shapes helps us make better choices about apparel and fashion. Our ability to style ourselves to look our best without consulting an expert on how our clothing choices relate to our face shape is helped by the coherence of our face shape and the fashion choices that we choose.

CHAPTER 5
IMPLEMENTATION

Implementation is the process of converting a new system design into an operational one. It is the key stage in achieving a successful new system. It must therefore be carefully planned and controlled. The implementation of a system is done after the development effort is completed

5.1 Steps for Implementation

5.1.1 Front-End Development Using Python Flask:

Modern computer applications are user-friendly. User interaction is not restricted to console-based I/O. They have a more ergonomic graphical user interface (GUI) thanks to high-speed processors and powerful graphics hardware. These applications can receive inputs through mouse clicks and can enable the user to choose from alternatives with the help of radio buttons, dropdown lists, and other GUI elements.

5.1.2 Flask Programming:

Flask is the standard GUI library for Python. Python when combined with Flask provides a fast and easy way to create GUI applications. Flask provides a powerful object- oriented interface to the Tk GUI toolkit. Flask has several strengths. It's cross- platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Flask look like they belong on the platform where they're run.

5.1.3 Implementation Issues:

The implementation phase of software development is concerned with translating design specifications into source code. The primary goal of implementation is to write source code and internal documentation so that conformance of the code to its specifications can be easily verified and so that debugging testing and modification are eased. This goal can be achieved by making the source code as clear and straightforward as possible. Simplicity clarity and elegance are the hallmarks of good programs and these characteristics have been implemented in each program module.

The goals of implementation are as follows.

- Minimize the memory required.
- Maximize output readability.
- Maximize source text readability.
- Minimize the number of source statements.
- Minimize development time

5.2 Model specification:

Module Specification is the way to improve the structure design by breaking down the system into modules and solving it as independent task. By doing so the complexity is reduced and the modules can be tested independently. The number of modules for our model is three, namely preprocessing, identification, feature extraction and detection. So each phase signify the functionalities provided by the proposed system. In the data pre-processing phase noise removal using median filtering is done.

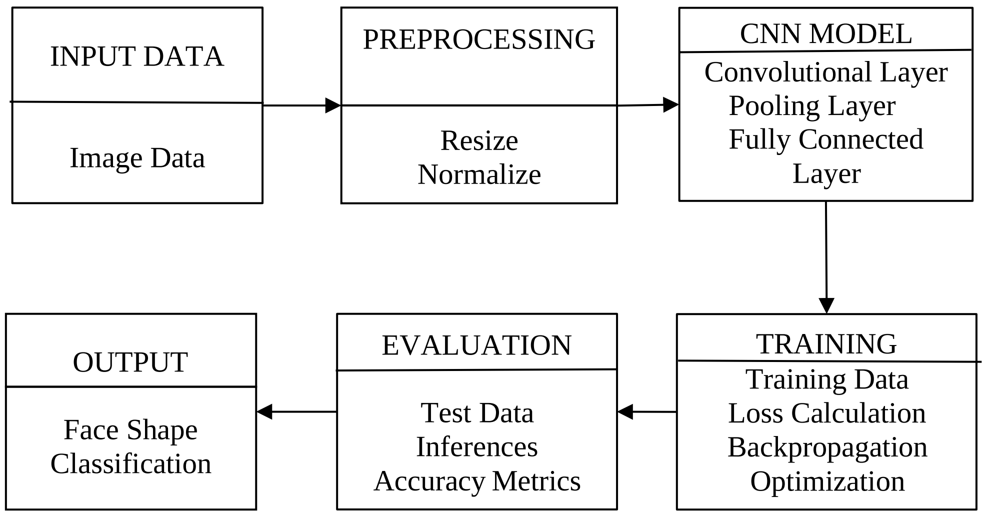


Figure 5.1: Data flow Diagram of Training and Testing Phase

The System design mainly consists of

1. Image Collection
2. Image Preprocessing
3. Image Segmentation
4. Feature Extraction
5. Training
6. Classification

1. Image Collection

The dataset that we have used in this project is available publicly on the internet

2. Image Preprocessing

The goal of pre-processing is an improvement of image data that reduces unwanted distortions and enhances some image features important for further image processing. Image pre-processing involves three main things a) Grayscale conversion b) Noise removal c) Image enhancement

a) Grayscale conversion: Grayscale image contains only brightness information. Each pixel value in a grayscale image corresponds to an amount or quantity of light. The brightness graduation can be differentiated in grayscale image. Grayscale image measures only light intensity 8 bit image will have brightness variation from 0 to 255 where ‘0’ represents black and ‘255’ represent white. In grayscale conversion color image is converted into grayscale image shows. Grayscale images are easier and faster to process than colored images. All image processing technique are applied on grayscale image.

b) Noise Removal: The objective of noise removal is to detect and remove unwanted noise from digital image. The difficulty is in deciding which features of an image are real and which are caused by noise. Noise is random variations in pixel values.

We are using median filter to remove unwanted noise. Median filter is nonlinear filter, it leaves edges invariant. Median filter is implemented by sliding window of odd length. Each sample value is sorted by magnitude, the centermost value is median of sample within the window, is a filter output.

c) Image Enhancement: The objective of image enhancement is to process an image to increase visibility of feature of interest. Here contrast enhancement is used to get better quality result.

3. Image Segmentation :

Image segmentation are of many types such as clustering, threshold, neural network based and edge based. In this implementation we are using the clustering algorithm called mean shift clustering for image segmentation. This algorithm uses the sliding window method for converging to the Centre of maximum dense area. This algorithm makes use of many sliding windows to converge the maximum dense region. Mean shift clustering Algorithm This algorithm is mainly used for detecting highly dense region.

4. Feature Extraction:

There are many features of an image mainly color, texture, and shape. Here we are considering three features that are color histogram, Texture which resembles color, shape, and texture.

5. Training :

Training dataset was created from images of known Cancer stages. Classifiers are trained on the created training dataset. A testing dataset is placed in a temporary folder. Predicted results from the test case, Plots classifiers graphs and add feature-sets to test case file, to make image processing models more accurate

6. Classification :

The binary classifier which makes use of the hyper-plane which is also called as the decision boundary between two of the classes is called as Convolution Neural Network. Some of the problems are pattern recognition like texture classification makes use of CNN. Mapping of nonlinear input data to the linear data provides good classification in high dimensional space in CNN. The marginal distance is maximized between different classes by CNN. Different Kernels are used to divide the classes. CNN is basically a binary classifier that determines hyper plane in dividing two classes. The boundary is maximized between the hyperplane and two classes. The samples that are nearest to the margin will be selected in determining the hyperplane is called support vectors.

5.3 CNN Algorithm Explanation

The invention of the CNN in 1994 by Yann LeCun is what propelled the field of Artificial Intelligence and Deep learning to its former glory. The first neural network named LeNet5 had a very less validation accuracy of 42% since then we have come a long way in this field. Nowadays almost every giant technology firms rely on CNN for more efficient performance. The idea to detect diseases in mulberry leaf incorporates the use of CNN before we dive into the "functionality and working of CNN" concept, we must have a basic idea on how the human brainrecognizes an object in spite of its varying attributes from one another. Our brain has a complex layer of neurons ,each layer holds some information about the object and all the features of the object are extracted by the neurons and stored in our memory, next time when we see the same object the brain matches the stored features to recognize the object, but one can easily mistake it as a simple "IF-THEN" function, yes it is to some extent but it has an extra feature that gives itan edge over other algorithms that is Self-Learning, although it cannot match a human brain but still it can give it a tough competition . Image is processed using the Basic CNN to detect the diseases in leaves.

The data training in our CNN model has to satisfy following constraints: 1) No missing values in dataset. 2) The dataset must distinctly be divided into training and testing sets, either the training or the testing set shouldn't contain any irrelevant data out of our model domain in case of an image dataset all the images must be of the same size, one uneven distribution of image size in our dataset can decrease the efficiency of our neural network. 3) The images should be converted into black and white format before feeding it into the convolution layer because reading images in RGB would involve a 3-D NumPy matrix which will reduce the execution time of our model by a considerable amount. Any kind of corrupted or blurred images should also be trimmed from the database before feeding it into the neural network. This is all about data pre-processing rules, let us dive right into the working of the convolution neural network.

5.3.1 Convolution Neural Network layers

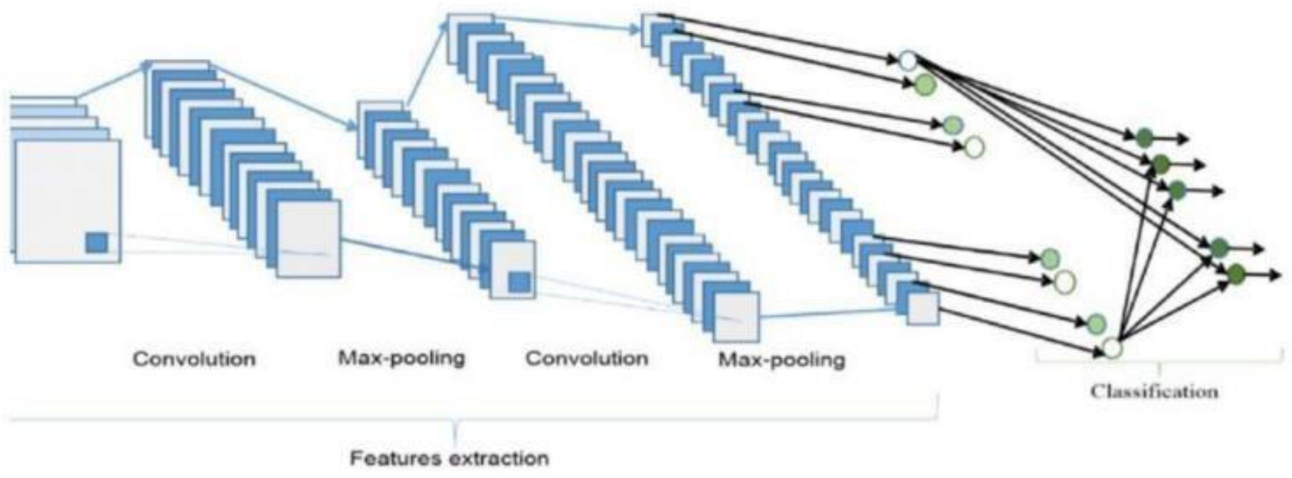


Figure 5.2 : Convolutional Neural Network Layers

A. Convolution layer

This layer involves scanning the whole image for patterns and formulating it in the form of a 3x3 matrix. This convolved feature matrix of the image is known as Kernel. Each value in the kernel is known as a weight vector.

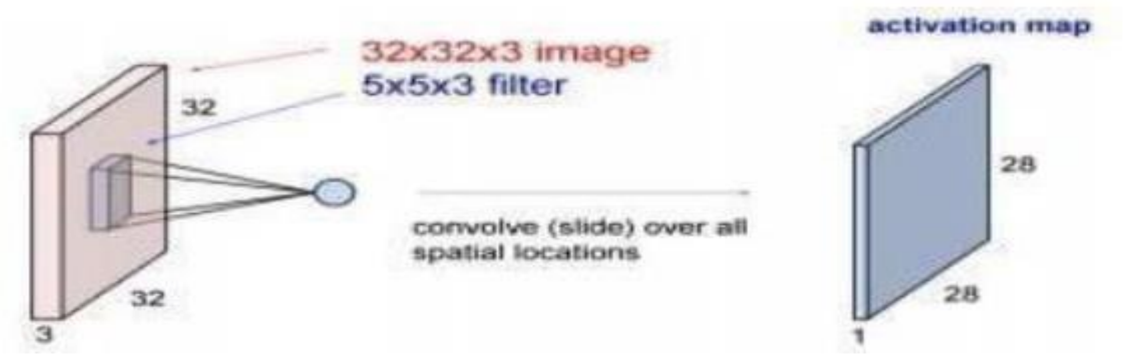


Figure 5.3 : Convolutional Layers

B. Pooling layer

After the convolution comes to the pooling here the image matrix is broken down into the sets of 4 rectangular segments which are non-overlapping. There are two types of pooling, Max pooling and average pooling. Max pooling gives the maximum value in the relative matrix region which is taken. Average pooling gives the average value in relative matrix region. The main advantage of the pooling layer is that it increases computer performance and decreases over- fitting chances.

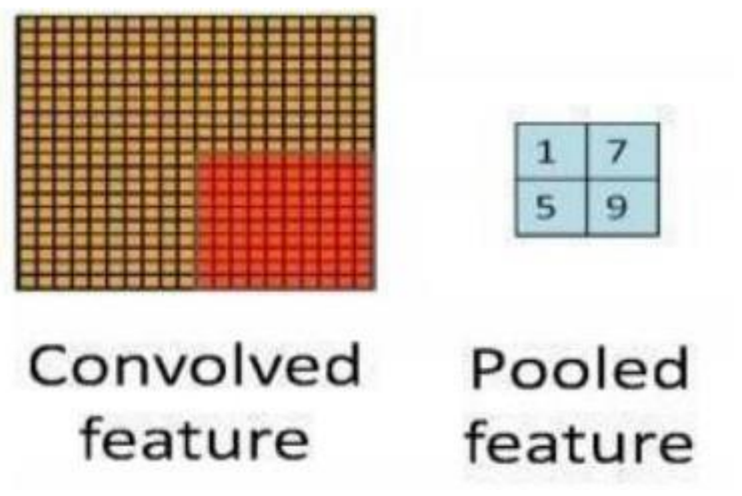


Figure 5.4 : Structure of CNN in matrix format

C. Activation layer

Implementation It the part of the Convolutional Neural Networks where the values are Normalized that is, they are fitted in a certain range. The used convolutional function is ReLU which allows only the positive values and then rejects the negative values. It is the function of low computational cost.

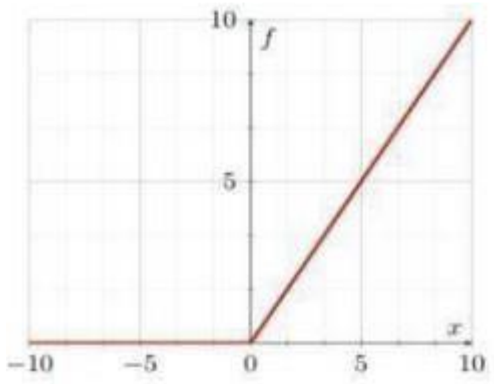


Figure 5.5 : Graph of positive relu function

5.4 Pseudocodes For Preprocessing Techniques

Pseudocode for Picture Uploading

Input image is obtained from the dataset which represents a human face.

Step 1: Click on upload.

Step 2: Read the image from dataset

imageA=cv2.imread(args[“first”])

Step 3: Display the image on the webpage.

Step 4: Shows the shape of the human face image in the dialog box.

Step 5: Shows the personalized recommendations according to the face shape.

The pseudocode for picture uploading includes the steps like uploading , reading the image from the dataset and displaying the image on web page with the human face and its personalized suggestions.

Pseudocode for Converting RGB image to Gray

In this section the RGB image is converted to gray image because it is easy to perform the operations such as median filtering, thresholding and high-pass filtering.

Step 1: Converting RGB to Gray

Step 2: Show the window to display the image

Step 3: Display the image in the window

Step4: Multiply each plane with a threshold value

Step5: Gray_pixel = (R_plan * R_thresh) + (G_plan * G_thresh) + (B_plan * B_thresh);

The pseudocode for converting RGB image to gray image includes the conversion of RGB image to gray image by multiplying each plane with a threshold value.

Pseudocode for Thresholding

Image thresholding is a simple, yet effective, way of partitioning an image into a foreground and background. This image analysis technique is a type of image segmentation that isolates objects by converting grayscale images into binary images.

Step 1: for i: =1 to 480*640 pixels
Step 2: if new_im(i)<T
Step 3: Change to black, and consider as obstacle
setnew_im(i)=0
Step 4: Change to white, and consider as free space . else
set new_im(i)=1

The pseudocode for thresholding includes the partitioning an image into a foreground and background. This image analysis technique is a type of image segmentation that isolates objects by converting grayscale images into binary images.

Pseudocode for High-pass Filtering

A high-pass filter can be used to make an image appear sharper. These filters emphasize fine details in the image- which is exactly the opposite of low pass filters. High pass filtering works in exactly the same way as a low pass filter except that it uses a different convolution kernel. Unfortunately, while low-pass filtering smooths out noise, high-pass only does the opposite by amplifying noise. However, if the original image is not too noisy- this is avoided to an extent.

```
Step1: Displays the high contrast version of original image.
Step2: Make a copy of the original image.
Initialize color[ ][ ] retval←copy(original)
Step3: Blur the copy of original image.
set color[ ][ ]blurred ←gaussianBlur(original)
Step4: Subtract the blur image from original image.
Color[ ][ ]highpass←difference(original,blurred)
Step5: Add the unsharp mask to original image to get sharpened image.
Step6: Run a loop for the entire rows and columns for
row=0 to original.length repeat until
for col=0 to original[row].length repeat until initialize
color origColor←original[row][col]
initializecontrastColor←highContrast[row][col] set
color difference←contrastColor - origColor end for
end for
```

The pseudocode for high pass filtering make an image appear sharper. These filters emphasize fine details in the image- which is exactly the opposite of low pass filters.

5.4 Summary

This chapter describes the implementation of the Human face shape prediction using various CNN architectures. Implementation requirement is deliberated in Section 5.1, Section 5.2 briefs about the programming language selected. Section 5.3 describes the pseudocode for preprocessing techniques.

CHAPTER 6

SYSTEM TESTING

6.1 System Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system to identify any gaps, errors, or missing requirements in contrary to the actual requirements. System testing of a software or hardware is a testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing fails within the scope of black-box testing, and such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, of all the ‗integrated' software components that have passed integration testing and the software system itself integrated with any applicable software systems. The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together. System testing is a more limited type of testing. It seeks to detect defects both within the inter-assemblages and within the system. System testing is performed on the entire system in the context of a Functional Requirement Specification (FRS) and / or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectation of the customer. It is also intended to test up to and beyond the bounds defined in the software / hardware requirement specification. Before applying methods to design effective test cases, a software engineer must understand the basic principle that guides software testing. All the tests should be traceable to customer requirements.

6.1.1 Types of testing

Software testing methods and traditionally divided into two: white-box and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

a) **White-box testing** (also known as clear box testing, glass box testing, transparent box testing and structural testing, by seeing the source code) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. While white-box testing can be applied at the unit, integration, and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between sub systems duringa system-level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

b) Black box testing:

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon. This project has been tested under different circumstances, which includes different types such as Unit testing, Integration testing and System testing that are described below.

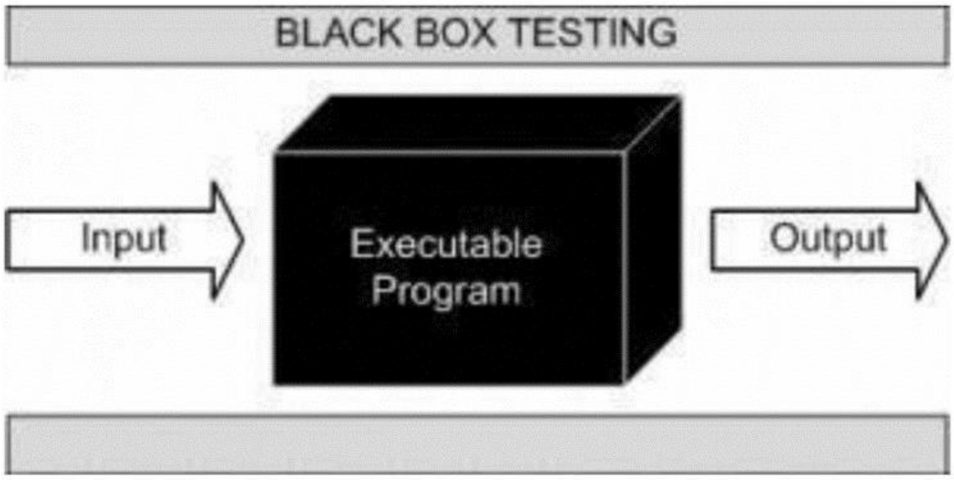


Figure 6.1: Black Box Testing

6.1.2 Levels of Testing:

There are different levels during the process of testing. Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are:

a) Functional Testing: This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. There are five steps that are involved while testing an application for functionality.

• The determination of the functionality that the intended application is meant to perform.

• The creation of test data based on the specifications of the application.

• The output based on the test data and the specifications of the application.

• The writing of test scenarios and the execution of testcases.

• The comparison of actual and expected results based on the executed testcases.

b) Non-functional Testing: This section is based upon testing an application from its non-functional attributes. Nonfunctional testing involves testing software from the requirements which are non- functional in nature but important such as performance, security, user interface, etc.

6.2 Unit Testing

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. During the development process itself all the syntax errors etc. got rooted out. For this developed test case that result in executing every instruction in the program or module i.e. every path through program was tested. Test cases are data chosen at random to check every possible branch after all the loops.

Table 6.1: Unit Testing Test Case 1

S1 # Test Case:	**UTC-1**
Name of Test	Uploading image
Items being tested	Tested for uploading different image
Sample Input	Upload sample image
Expected output	Image should upload properly
Actual output	Upload successful
Remarks	Predicted result as shown in snapshot 7.3

The above table 6.1 shows the successful test case for uploading the image. In this process items being tested for uploading different image The provided unit test case (UTC-1) focuses on validating the functionality of uploading different images within a system. It encompasses several key elements essential for effective testing. Firstly, the test case is clearly identified by a unique identifier, aiding in organization and tracking within the testing process. The name of the test succinctly describes its purpose, which is to verify the image uploading feature. The items being tested are explicitly outlined, indicating the specific aspect of the system under scrutiny. A sample input, in this case, an image upload, is provided to simulate real-world usage. The expected output is clearly defined as the successful upload of the image. During the test execution, the actual output is compared against this expectation, and in this instance, it matches, leading to a passing result. While the test case effectively validates basic functionality, it could be further enhanced by including preconditions, steps to reproduce, consideration of additional scenarios such as edge cases, and cleanup steps to ensure a comprehensive and robust testing process.

Table 6.2: Unit Testing Test Case 2

S1 # Test Case:	**UTC-2**
Name of Test	Detecting Face shape images
Items being tested	Test for different shape of face images
Sample Input	Tested for different shape images of face
Expected output	Face shape should be displayed
Actual output	Should Display different face shape
Remarks	Predicted result as shown in snapshot 7.4

The above table 6.2 shows the provided unit test case (UTC-2) is designed to evaluate the system's capability to detect various face shapes within images. It adheres to the structure commonly employed in unit testing, featuring essential components for effective evaluation. The test case is uniquely identified by its identifier, ensuring clear tracking and reference. The test name succinctly summarizes its purpose, which is to identify different face shapes within images. The items being tested are explicitly stated, focusing on the system's ability to recognize diverse facial structures. A sample input, consisting of a variety of face images, is provided to simulate real-world scenarios. The expected output specifies that the system should accurately display the detected face shapes. During test execution, the actual output is compared against this expectation, with the anticipated outcome being the display of diverse facial shapes. The remark suggests a predicted result, indicating confidence in the system's performance. However, for a more comprehensive evaluation, it would be beneficial to include additional details such as preconditions, steps to reproduce, and potential edge cases to ensure thorough testing coverage. Incorporating these elements would enhance the robustness and reliability of the testing process, providing greater assurance in the system's ability to detect and display various face shapes accurately.

6.3 Integration Testing:

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. It occurs after unit testing and before validation testing. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing.

1. **Bottom-up Integration**: This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.
2. **Top-down Integration**: In this testing, the highest-level modules are tested first and progressively, lower-level modules are tested thereafter. In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations.

Table 6.3: Functional Testing Test Case 1

S1 # Test Case:	**ITC**
Name of Test	Working of choose file option
Items being tested	User convenience to access images stored
Sample Input	Click and select the image
Expected output	Should open selected Image
Actual output	Selected open selected image
Remarks	Predicted result as shown in snapshot 7.2

The above table 6.3 shows the functional test case labeled as ITC focuses on assessing the functionality of the "choose file" option within the system. This test aims to evaluate the user's convenience in accessing stored images through this feature. The test case is clearly identified by its unique identifier, ensuring easy tracking and reference. The test name succinctly describes its purpose, which is to verify the functionality of the "choose file" option. The specific item being tested is the system's ability to allow users to select images conveniently. The sample input provided involves clicking and selecting an image, simulating a real-world user interaction. The expected output specifies that upon selection, the chosen image should open within the system. During the execution of the test, the actual output is compared against this expectation, with the result indicating that the selected image successfully opened. The test case remarks that the test has passed, indicating that the system's functionality aligns with expectations. However, to ensure a comprehensive evaluation, it's advisable to include additional details such as preconditions, steps to reproduce, and potential edge cases. These elements contribute to a more thorough testing process, ultimately enhancing confidence in the system's ability to provide user-friendly access to stored images through the "choose file" option.

Table 6.4: Functional Testing Test Case 2

S1 # Test Case:	**ITC-2**
Name of Test	Working of Segmentation and Displaying faceshape images
Items being tested	Selecting different images and verifying faceshape
Sample Input	Click and select the image
Expected output	Should show faceshapes and predict different classification faceshapes
Actual output	Image Segmented, faceshape detected should be displayed
Remarks	Predicted result as shown in snapshot 7.5

The above table 6.4 shows the functional test case labeled as ITC-2 is intended to evaluate the functionality of segmentation and displaying of face shape images within the system. This test aims to ensure that the system correctly segments images and accurately displays detected face shapes. The test case is uniquely identified by its identifier, which aids in tracking and referencing. The test name clearly articulates its purpose, focusing on the segmentation and display of face shape images. The items being tested are explicitly stated, emphasizing the system's ability to select various images and verify their face shapes. The sample input provided involves clicking and selecting an image, mimicking user interaction. The expected output specifies that the system should show segmented face shapes and predict different classifications of face shapes. During the execution of the test, the actual output is compared against this expectation. The result indicates that the image was successfully segmented, and the detected face shape was accurately displayed, meeting the expected outcome. The test remarks that the test has passed, indicating that the system's functionality aligns with expectations. However, to ensure a thorough evaluation, it's recommended to include additional details such as preconditions, steps to reproduce, and potential edge cases. Incorporating these elements contributes to a comprehensive testing process, enhancing confidence in the system's ability to segment and display face shape images accurately. The sample input provided involves clicking and selecting an image, mimicking user interaction.

6.4 System testing:

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. System testing is important because of the following reasons:

• System testing is the first step in the Software Development Life Cycle, where the application is tested.

• The application is tested thoroughly to verify that it meets the functional and technical specifications.

• The application is tested in an environment that is very close to the production environment where the application will be deployed.

• System testing enables us to test, verify, and validate both the business requirements as well as the application architecture.

Table 6.5: System Testing Test Case 1

S1 # Test Case:	**STC-1**
Name of Test	System testing in various versions of OS
Items being tested	OS compatibility.
Sample Input	Execute the program in windows XP/Windows-10
Expected output	Performance is better in windows-10
Actual output	Same as expected output, performance is better in windows-10
Remarks	Pass.

The above table 6.5 shows the The system testing test case labeled as STC-1 focuses on evaluating the system's compatibility with different versions of operating systems (OS). Specifically, the test aims to assess the performance of the system across various OS versions. The test case is uniquely identified by its identifier, which aids in tracking and reference. The test name succinctly describes its purpose, which is to conduct system testing across different OS versions. The items being tested are explicitly stated, emphasizing the system's compatibility with multiple OS environments. The sample input provided involves executing the program in two different versions of Windows: Windows XP and Windows 10, representing older and newer OS versions, respectively. The expected output specifies that the performance should be better in Windows 10, presumably due to optimizations and enhancements in the newer OS version. During the execution of the test, the actual output aligns with the expectation, with the performance observed to be better in Windows 10, confirming the anticipated outcome. The test remarks that the test has passed, indicating that the system performs as expected and demonstrates compatibility with the specified OS versions. However, it's important to note that for a comprehensive evaluation, similar tests should be conducted across a wider range of OS versions and platforms, considering potential variations in performance and behavior. Including additional details such as preconditions, steps to reproduce, and specific performance metrics would further enhance the thoroughness and reliability of the testing process, ensuring robust compatibility across different OS environments.

6.5 Summary

This chapter presents system testing in section 6.1, which consists of unit test cases for the various modules of personality prediction system. Section 6.2 gives a complete view of the testing which includes Test Name, Test Features, Output Expected, Output Obtained and Result.

CHAPTER 7

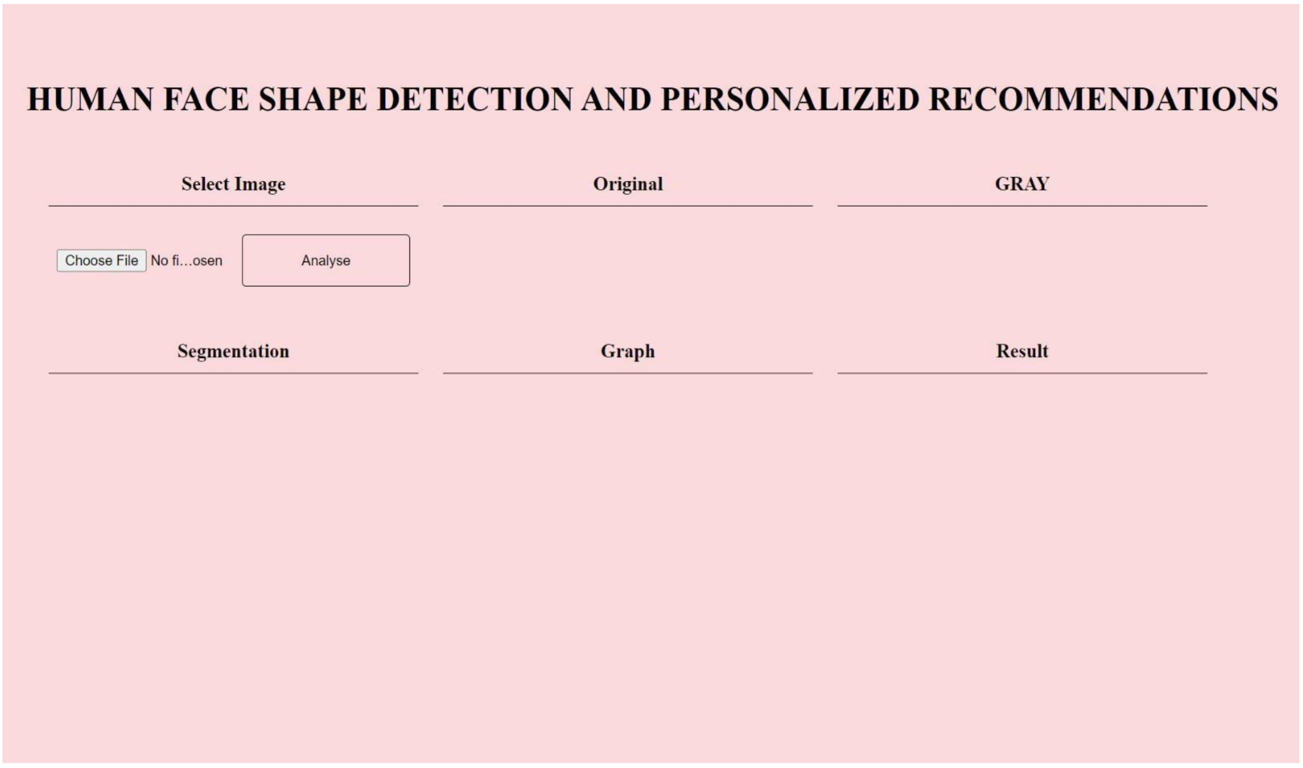
RESULTS AND DISCUSSION

Programming language used to design the proposed method is Python. By using relevant images of human faces, classified using CNN architectures, we get to know the particular face shape, with a small description as the output. Results obtained in each step are demonstrated in following snapshots.

7.1 Experimental Results

Snapshot of Home Page

The Snapshot Figure 7.1 shows the snapshot of home page.

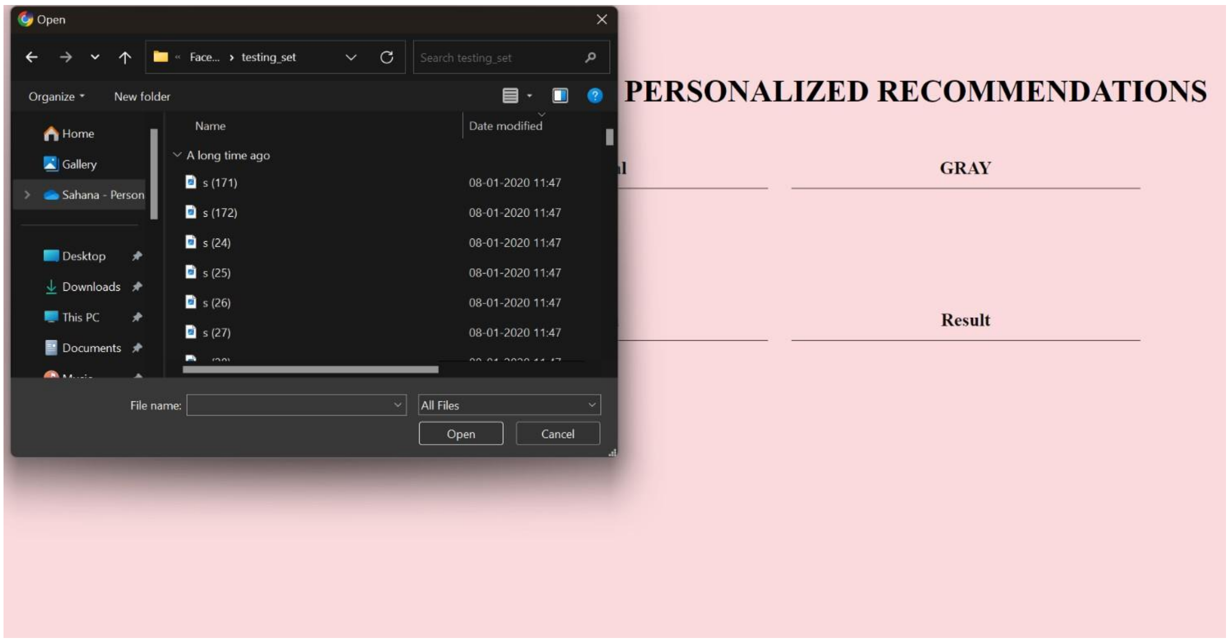


Snapshot 7.1: Snapshot of Home page.

The Snapshot 7.1 shows the snapshot of home page which consists of a header which reads "Human Faceshape Detection And Personalized Recommendations" and two buttons which are labeled "Choose file" and "Analyse" respectively. It also includes segmentation of the given input image, graph and result of that image.

Snapshot of File Explorer on Clicking "choose file"

The Snapshot 7.2 shows the snapshot of the file explorer that pops up.

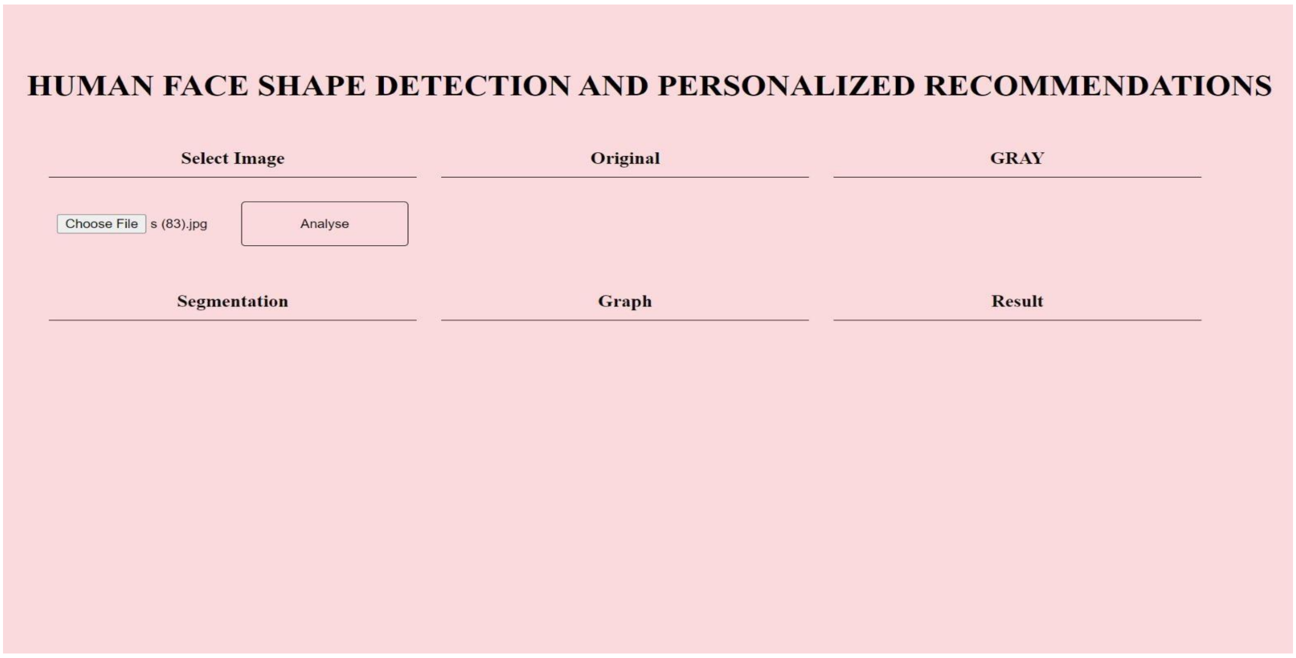


Snapshot 7.2: Snapshot of the file explorer on clicking submit.

The Snapshot 7.2 shows the snapshot of the file explorer window that pops up when the choose file button is clicked. The user may select the image to be classified as input to the system and let the system tell them what faceshape is detected.

Snapshot after uploading image

The Snapshot 7.3 shows the snapshot of the same webpage after uploading the image.

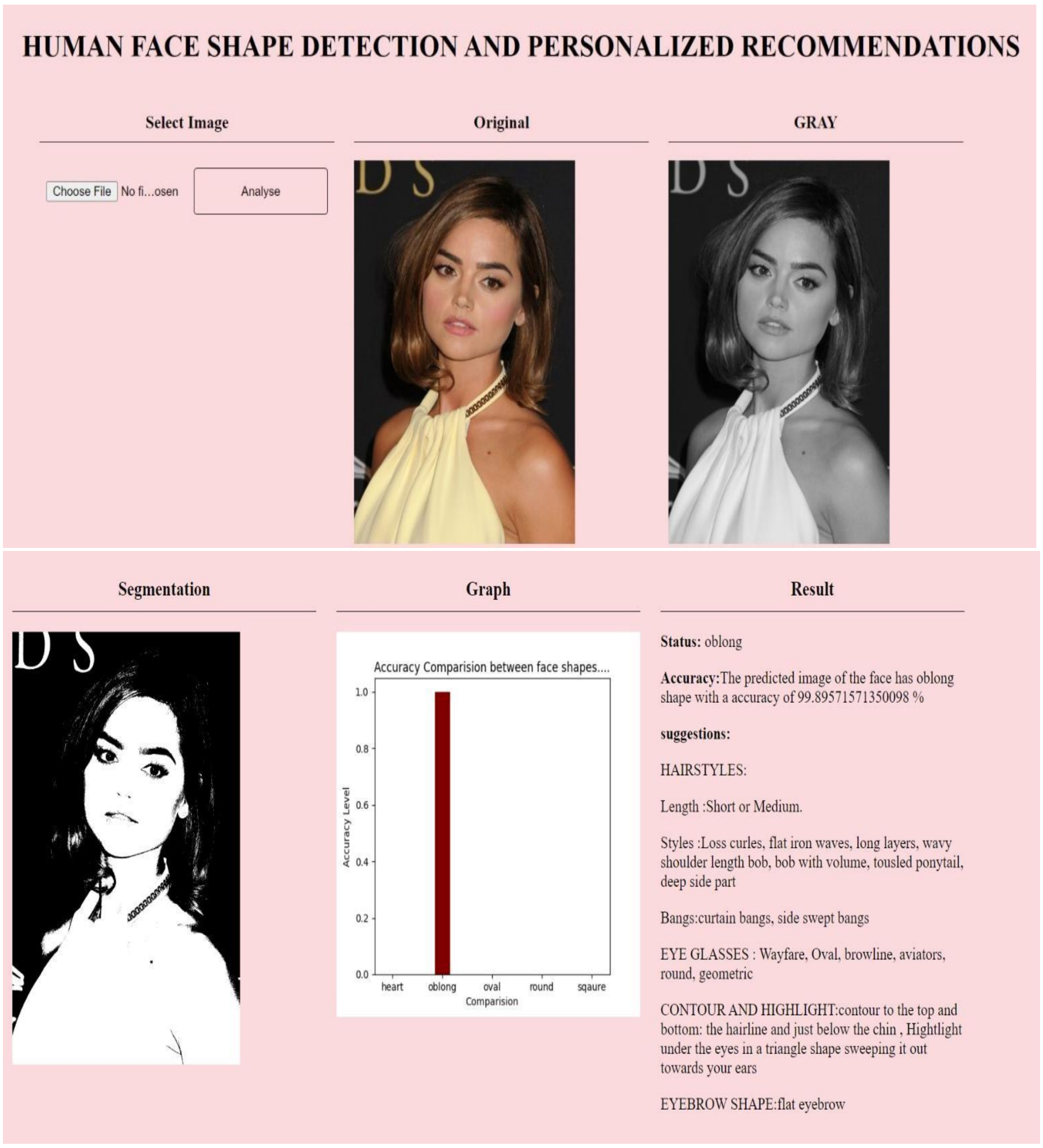


Snapshot 7.3: Snapshot of the image name that appears.

The Snapshot 7.3 shows the snapshot of the webpage after uploading the image file to it. The page displays the name of the image right next to the choose file button.

Snapshot of the Oblong faceshape

The Snapshot 7.4 shows the snapshot of the output result page.



Status: oblong

Accuracy: The predicted image of the face has oblong shape with a accuracy of 99.89571571350098 %

suggestions:

HAIRSTYLES:

Length :Short or Medium.

Styles :Loss curles, flat iron waves, long layers, wavy shoulder length bob, bob with volume, tousled ponytail, deep side part

Bangs:curtain bangs, side swept bangs

EYE GLASSES : Wayfare, Oval, browline, aviators, round, geometric

CONTOUR AND HIGHLIGHT:contour to the top and bottom: the hairline and just below the chin , Hightlight under the eyes in a triangle shape sweeping it out towards your ears

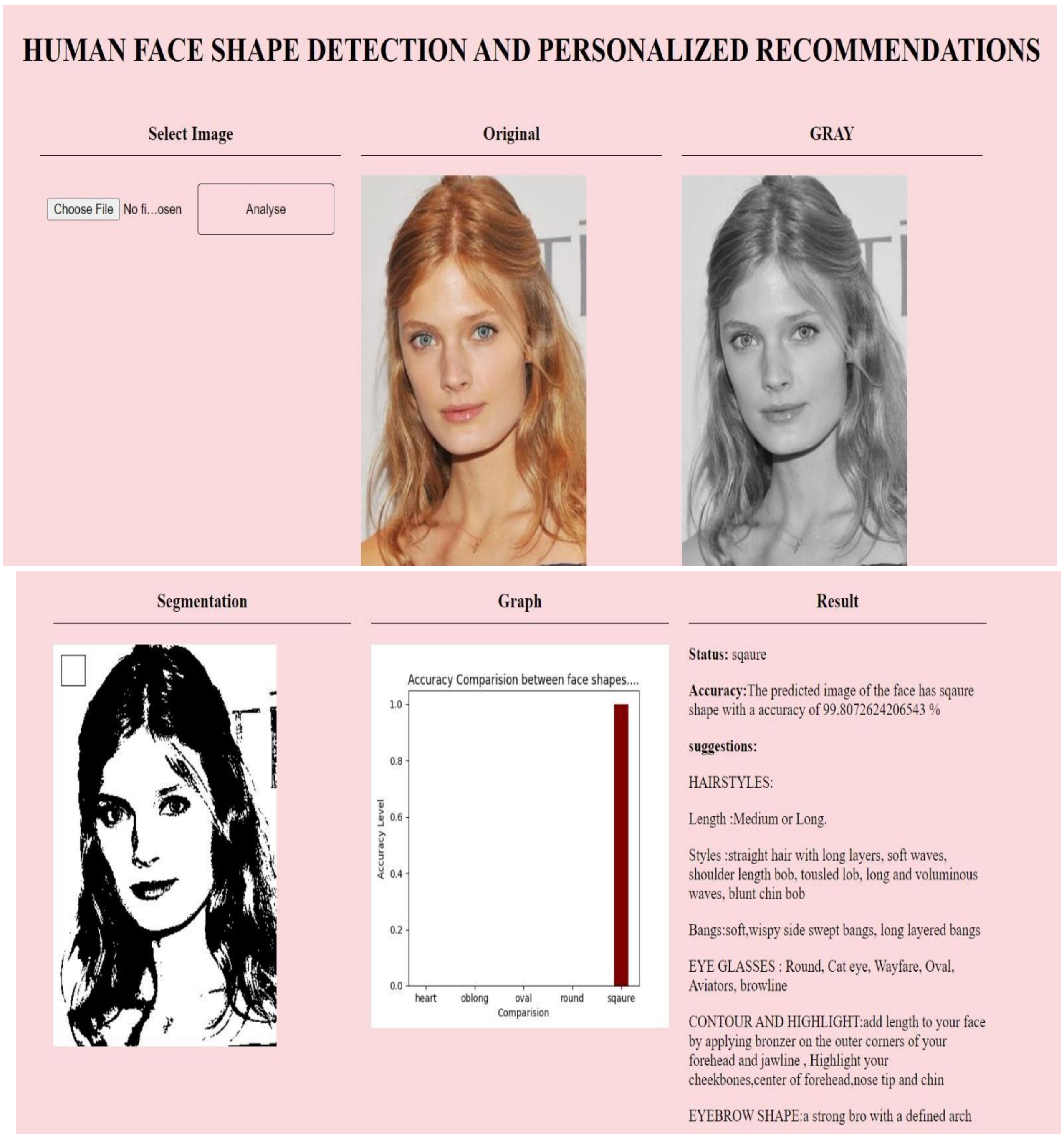
EYEBROW SHAPE:flat eyebrow

Snapshot 7.4: Snapshot of the selected face shape image and its suggestions.

The Snapshot 7.4 shows the snapshot of the webpage after uploading the image file to it and click analyse button. The page displays the original image , gray image, segmentation of the image, accuracy graph and personalized recommendations like hairstyles (length, styles, bangs), eye glasses, contour and highlight and eyebrow shape of Oblong face shape.

Snapshot of a Square face shape and suggestions

The Snapshot 7.5 shows the snapshot of a Square face shape and its suggestions.



Status: sqaure

Accuracy: The predicted image of the face has sqaure shape with a accuracy of 99.8072624206543 %

suggestions:

HAIRSTYLES:

Length :Medium or Long.

Styles :straight hair with long layers, soft waves, shoulder length bob, tousled lob, long and voluminous waves, blunt chin bob

Bangs:soft,wispy side swept bangs, long layered bangs

EYE GLASSES : Round, Cat eye, Wayfare, Oval, Aviators, browline

CONTOUR AND HIGHLIGHT:add length to your face by applying bronzer on the outer corners of your forehead and jawline , Highlight your cheekbones,center of forehead,nose tip and chin

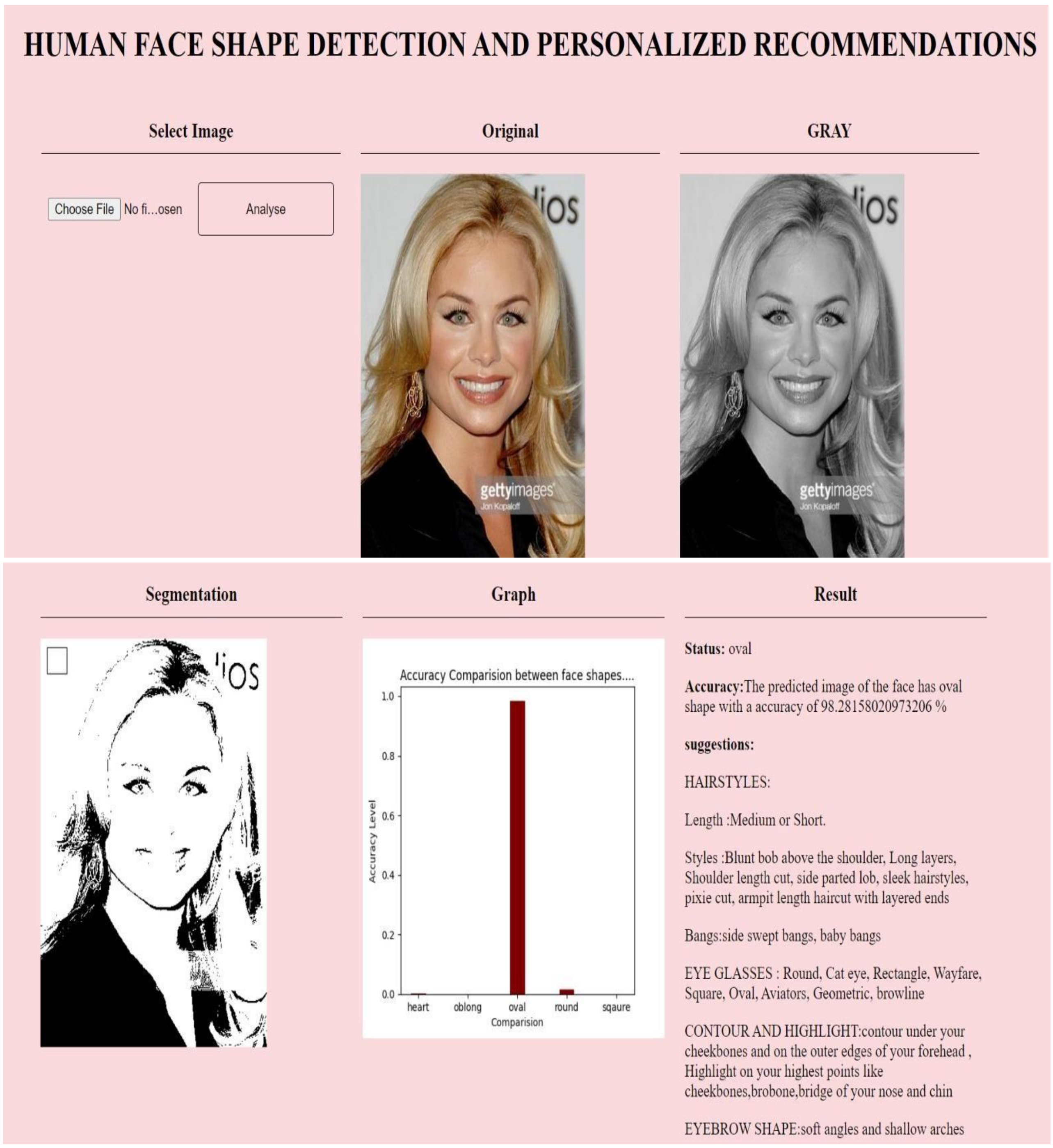
EYEBROW SHAPE:a strong bro with a defined arch

Snapshot 7.5: Snapshot of the selected face shape image and its suggestions.

The Snapshot 7.5 shows the snapshot of the webpage after uploading the image file to it and click analyse button. The page displays the original image, gray image, segmentation of the image, accuracy graph and personalized recommendations like hairstyles (length, styles, bangs), eye glasses, contour and highlight and eyebrow shape of Square face shape.

Snapshot of a oval face shape and suggestions

The Snapshot 7.6 shows the snapshot of a oval face shape and its suggestions.



Status: oval

Accuracy: The predicted image of the face has oval shape with a accuracy of 98.28158020973206 %

suggestions:

HAIRSTYLES:

Length :Medium or Short.

Styles :Blunt bob above the shoulder, Long layers, Shoulder length cut, side parted lob, sleek hairstyles, pixie cut, armpit length haircut with layered ends

Bangs:side swept bangs, baby bangs

EYE GLASSES : Round, Cat eye, Rectangle, Wayfare, Square, Oval, Aviators, Geometric, browline

CONTOUR AND HIGHLIGHT:contour under your cheekbones and on the outer edges of your forehead , Highlight on your highest points like cheekbones,brobone,bridge of your nose and chin

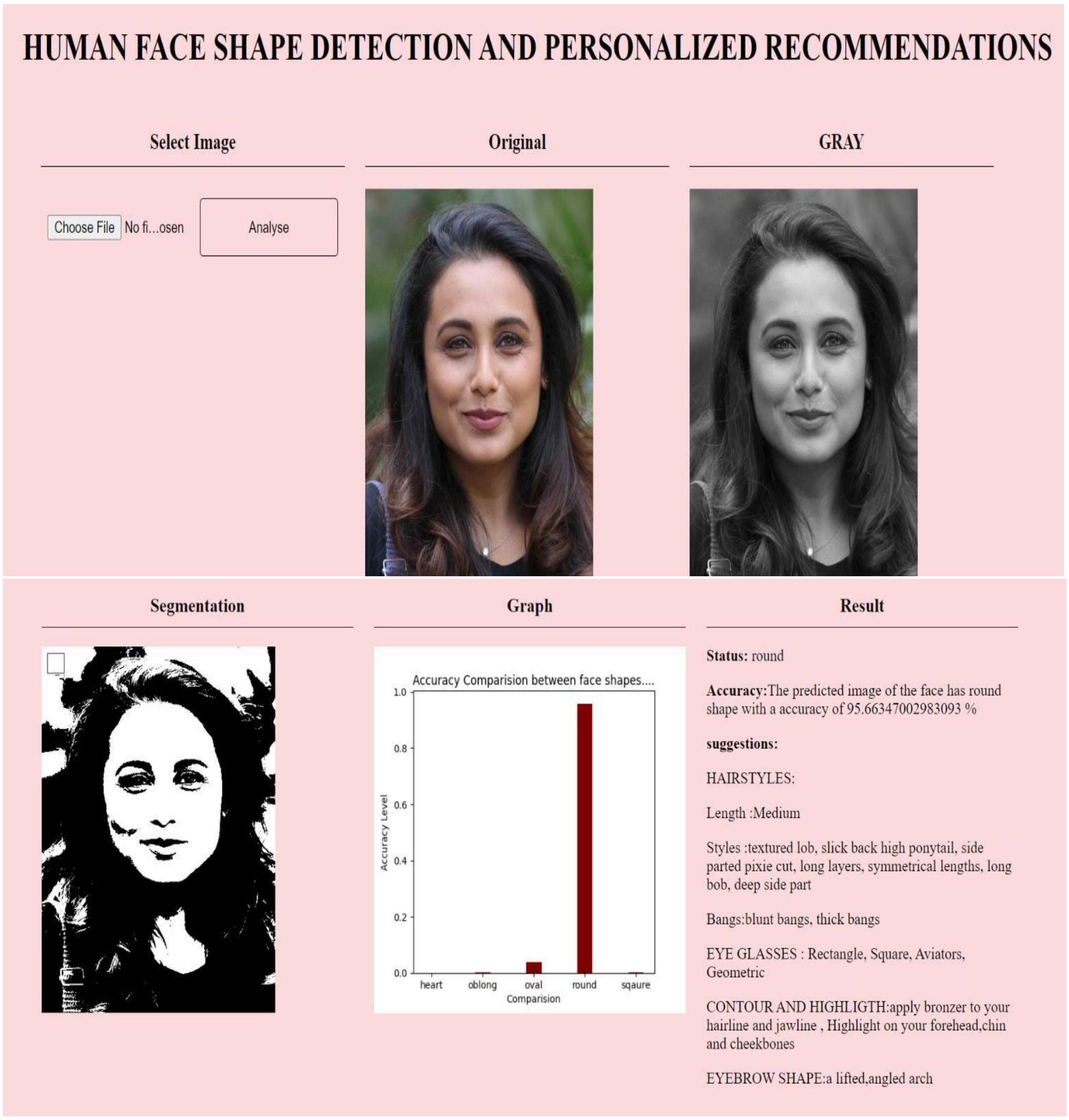
EYEBROW SHAPE:soft angles and shallow arches

Snapshot 7.6: Snapshot of the selected face shape image and its suggestions.

The Snapshot 7.6 shows the snapshot of the webpage after uploading the image file to it and click analyse button. The page displays the original image , gray image, segmentation of the image, accuracy graph and personalized recommendations like hairstyles (length, styles, bangs), eye glasses, contour and highlight and eyebrow shape of oval face shape.

Snapshot of a round face shape and suggestions

The Snapshot 7.7 shows the snapshot of a round face shape and its suggestions.



Status: round

Accuracy: The predicted image of the face has round shape with a accuracy of 95.66347002983093 %

suggestions:

HAIRSTYLES:

Length :Medium

Styles :textured lob, slick back high ponytail, side parted pixie cut, long layers, symmetrical lengths, long bob, deep side part

Bangs:blunt bangs, thick bangs

EYE GLASSES : Rectangle, Square, Aviators, Geometric

CONTOUR AND HIGHLIGTH:apply bronzer to your hairline and jawline , Highlight on your forehead,chin and cheekbones

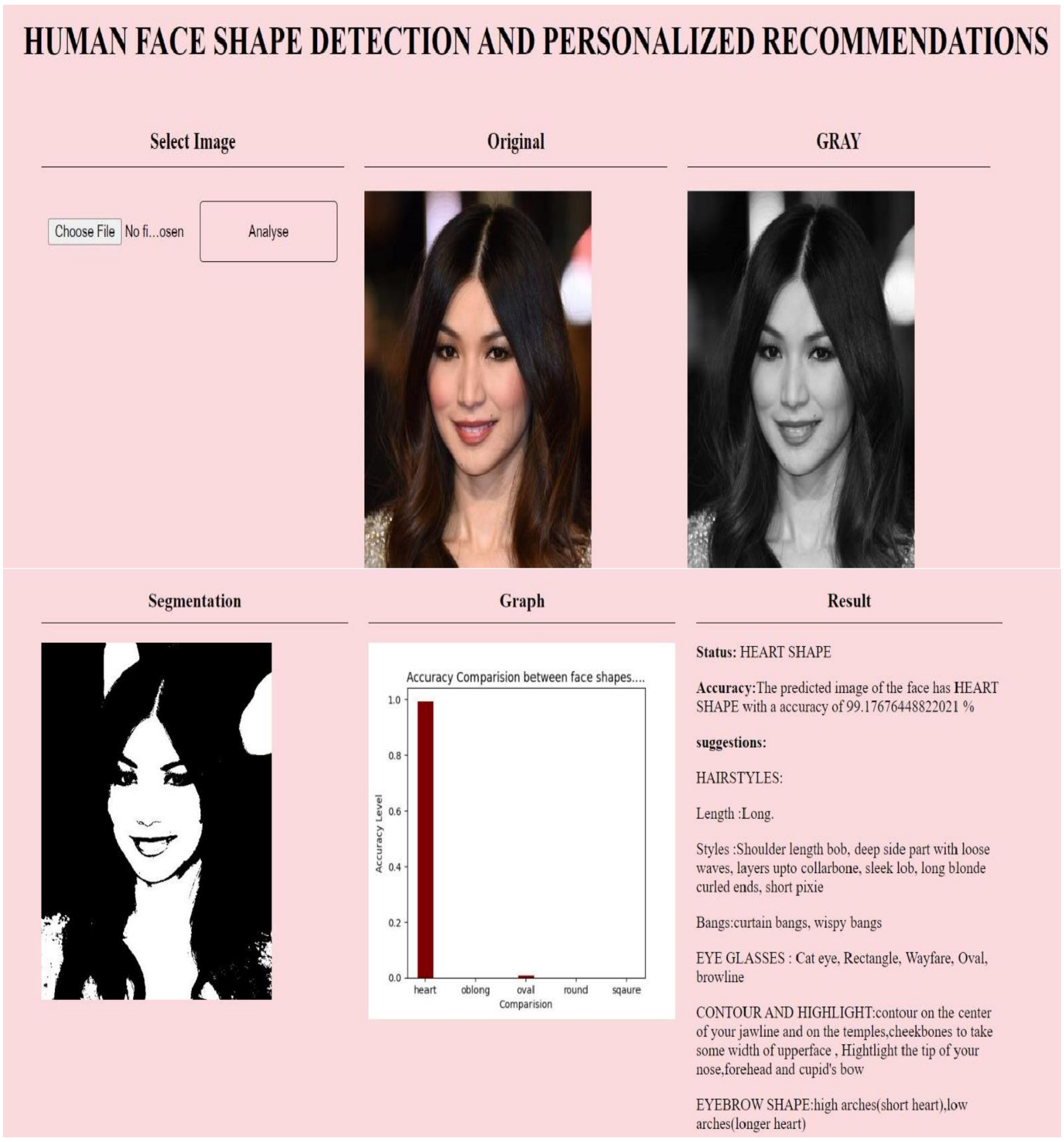
EYEBROW SHAPE:a lifted,angled arch

Snapshot 7.7: Snapshot of the selected face shape image and its suggestions.

The Snapshot 7.7 shows the snapshot of the webpage after uploading the image file to it and click analyse button. The page displays the original image , gray image, segmentation of the image, accuracy graph and personalized recommendations like hairstyles (length, styles, bangs), eye glasses, contour and highlight and eyebrow shape of round face shape.

Snapshot of a heart face shape and suggestions

The Snapshot 7.8 shows the snapshot of a heart face shape and its suggestions.



Status: HEART SHAPE

Accuracy: The predicted image of the face has HEART SHAPE with a accuracy of 99.17676448822021 %

suggestions:

HAIRSTYLES:

Length :Long.

Styles :Shoulder length bob, deep side part with loose waves, layers upto collarbone, sleek lob, long blonde curled ends, short pixie

Bangs:curtain bangs, wispy bangs

EYE GLASSES : Cat eye, Rectangle, Wayfare, Oval, browline

CONTOUR AND HIGHLIGHT:contour on the center of your jawline and on the temples,cheekbones to take some width of upperface , Hightlight the tip of your nose,forehead and cupid's bow

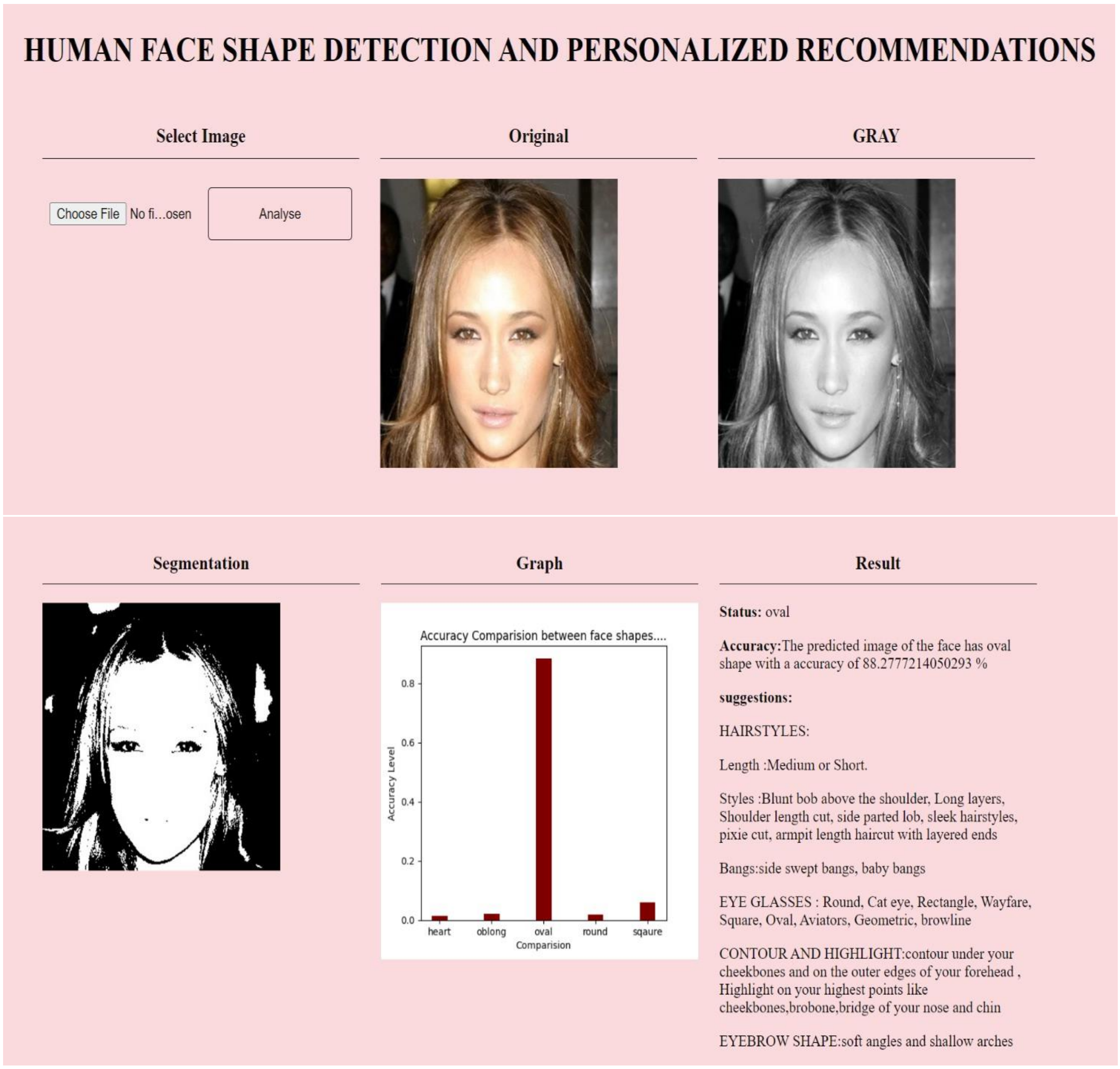
EYEBROW SHAPE:high arches(short heart),low arches(longer heart)

Snapshot 7.8: Snapshot of the selected face shape image and its suggestions.

The Snapshot 7.8 shows the snapshot of the webpage after uploading the image file to it and click analyse button. The page displays the original image , gray image, segmentation of the image, accuracy graph and personalized recommendations like hairstyles (length, styles, bangs), eye glasses, contour and highlight and eyebrow shape of heart face shape.

Snapshot of a oval face shape and suggestions

The Snapshot 7.9 shows the snapshot of a oval face shape and its suggestions. Here this face shape also includes multiple other face shapes with less accuracy. This means this face is of oval shape but also includes others shape features.



Status: oval

Accuracy: The predicted image of the face has oval shape with a accuracy of 88.2777214050293 %

suggestions:

HAIRSTYLES:

Length :Medium or Short.

Styles :Blunt bob above the shoulder, Long layers, Shoulder length cut, side parted lob, sleek hairstyles, pixie cut, armpit length haircut with layered ends

Bangs:side swept bangs, baby bangs

EYE GLASSES : Round, Cat eye, Rectangle, Wayfare, Square, Oval, Aviators, Geometric, browline

CONTOUR AND HIGHLIGHT:contour under your cheekbones and on the outer edges of your forehead , Highlight on your highest points like cheekbones,brobone,bridge of your nose and chin

EYEBROW SHAPE:soft angles and shallow arches

Snapshot 7.9: Snapshot of the selected face shape image and its suggestions.

The Snapshot 7.9 shows the snapshot of the webpage after uploading the image file to it and click analyze button. The page displays the original image , gray image, segmentation of the image, accuracy graph and personalized recommendations like hairstyles (length, styles, bangs), eye glasses, contour and highlight and eyebrow shape of heart face shape.

Snapshot of Datasets

Snapshot 7.10: Snapshot of the Datasets.

The Snapshot 7.10 shows the snapshot of the datasets, which here is a collection of various images that show different human face shapes and sorted into different folders to assist and aid supervised learning.

7.2 Summary

This chapter presents the experimental results in section 7.1, which consists of snapshots of the home page, the functionality of the upload and submit buttons, the output page of all five human face shapes and its recommendations.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 Conclusion

The Face Shape Predictor and Recommendations project describes cutting-edge computer vision with user-centric design to deliver an innovative tool for accurately predicting face shapes and offering personalized beauty and fashion recommendations. By prioritizing user privacy through robust security measures and incorporating privacy-preserving techniques, the project ensures ethical data handling.

The user-friendly interface facilitates seamless photo uploads, and visual simulations aid users in previewing suggested changes. A feedback loop encourages continuous improvement, and educational content fosters a deeper understanding of personal aesthetics. This comprehensive approach aims to not only enhance users' physical appearance but also boost their self-confidence, contributing to a positive and informed journey of self-expression.

By training these models on large and diverse datasets encompassing various facial shapes, ethnicities, ages, and expressions, they can learn intricate facial features and subtle nuances, enabling more precise predictions. Moreover, integrating techniques like landmark detection and facial feature extraction into the prediction process can offer finer granularity in analyzing facial structures. This could involve identifying key landmarks such as the eyes, nose, mouth, and jawline, and employing geometric and statistical methods to infer the overall face shape.

8.2 Future Enhancement

In the future, enhancing human face shape predictor algorithms using advanced image processing techniques holds immense potential for improving facial recognition systems, virtual makeup applications, and even medical diagnostics. Leveraging cutting-edge deep learning architectures such as convolutional neural networks (CNNs) can significantly refine the accuracy and robustness of these predictors. Additionally, incorporating three-dimensional (3D) modeling approaches can further enhance accuracy by capturing depth information and providing a more comprehensive understanding of facial geometry. By accurately delineating facial contours and shapes from images or video frames, these techniques can facilitate more precise measurements and predictions of facial features. In the medical domain, such enhanced face shape predictors could find applications in facial reconstruction surgery, orthodontics, and craniofacial anomaly diagnosis and treatment planning. By aiding clinicians in precisely analyzing facial morphology and predicting potential outcomes, these systems can improve patient care and treatment outcomes. Overall, future enhancements to human face shape predictor algorithms through image processing innovations promise to revolutionize various fields, from biometrics and entertainment to healthcare.

REFERENCES

[1] Bansode, N. and Sinha, P. (2016). Face shape classification based on region similarity, correlation and fractal dimensions. International Journal of Computer Science Issues (IJCSI),

[2] Deng, Q., Luo, Y., and Ge, J. (2010). Dual threshold based unsupervised face image clustering. In 2010 he 2nd International Conference on Industrial Mechatronics and Automation, volume 1, pages 436–439. EEE.

[3] Huttenlocher, D., Klanderman, G., and Rucklidge, W. (1993). Comparing images using the hausdorff distance. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(9):850–863.

[4] Kaggle (2020). Face shape dataset.

[5] King, D. E. (2009). Dlib-ml: A machine learning toolkit. Journal of Machine Learning Research, 0:1755–1758.

[6] Pornthep Sarakon, Theekapun Charoenpong, S. C. (2014). Face shape classification from 3d human data by using svm. IEEE.

[7] Pornthep Sarakon, Theekapun Charoenpong, S. C. (2014). Face shape classification from 3d human data by using svm. IEEE.

[8] SHU CONG ZHANG, BIN FANG, Y.-Z. L. (2011). A face clustering method based on facial shape information. IEEE.

[9] T. F. Cootes, C. J. Taylor, D. H. C. and Graham, J. (1995). Active shape models - their training and application, computer vision and image understanding. Computer Vision and Image Understanding.

[10] Theiab Alzahrani, Waleed Al-Nuaimy, B. A.-B. (2021). Integrated multi-model face shape and eye attributes identification for hair style and eyelashes recommendation. Computation.

[11] Tio, A. E. (2019). Face shape classification using inception v3. Arxiv, abs/1911.07916.

[12] Tou, J. and Gonzales, R. (1974). Pattern recognition principles. Addison Wesley.

[13] Vallespi, C., De la Torre, F., Veloso, M., and Kanade, T. (2006). Automatic clustering of faces in meetings. In 2006 International Conference on Image Processing, pages 1841–1844. IEEE.