MAR BASELIOS INSTITUTE OF TECHNOLOGY AND SCIENCE

Nellimattom, Kothamangalam

(Affiliated to APJ Abdul Kalam Technological University, TVM)

Department of Computer Applications

Main Project Report

FACE SHAPE DETECTION

Done by

Aswin G Nair

Reg No: MBI23MCA-2015

Under the guidance of

Prof. KESIYA SAJU

2023-2025

CERTIFICATE

FACE SHAPE DETECTION

Certified that this is the bonafide record of project work done by

Aswin G Nair

Reg No: MBI23MCA-2015

During the academic year 2023-2025, in partial fulfillment of requirements foraward of the degree,

Master of Computer Applications of
APJ Abdul Kalam Technological University
Thiruvananthapuram

Faculty Guide

Prof. Kesiya Saju

Head of the Department

Prof. Reshma S

Project Coordinator

Prof. Merin Joy M

External Examiner

ACKNOWLEDGEMENT

First and foremost, I thank God Almighty for his divine grace and blessings in making all this possible. May he continue to lead me in the years to come. No words can express my humble gratitude to my beloved parents who have been guiding me in all walks of my journey.

I am also grateful to Prof. Reshma S, Head of Computer Applications Department, Prof. Merin Joy M, Project Coordinator for their valuable guidance and constant supervision as well as for providing necessary information regarding the project & also for their support.

I would like to express my special gratitude and thanks to my Project Guide Prof. Kesiya Saju, Assistant Professor, Department of Computer Applications for giving me such attention and time.

I profusely thank other Professors in the department and all other staffs of MBITS, for their guidance and inspirations throughout my course of study. My thanks and appreciations also goes my friends and people who have willingly helped me out with their abilities.

ABSTRACT

Face shape detection is essential in many applications, such as augmented reality, virtual try-on systems, beauty advice personalized to individuals, and medical diagnosis. Conventional face shape classification techniques are based on handcrafted features and geometric computation, which are not robust enough in real-world situations because of lighting, pose, and facial expression variations. In this work, I suggest a deep learning method based on Convolutional Neural Networks (CNNs) to classify face shapes into given categories like oval, round, square, heart, and oblong.

The CNN model is trained on a heterogeneous dataset of labeled facial images, utilizing data augmentation methods to promote generalization. Feature extraction is done automatically by deep learning layers without the requirement of manual intervention. The model is tested based on common performance measures including accuracy, precision, recall, and F1-score, achieving remarkable improvements compared to existing machine learning-based classifiers. The designed system offers a robust, automated face shape detection approach that can be easily integrated into real-time applications like virtual beauty advisors, AI-based styling assistants, and facial recognition systems. Future efforts will be on generalizing the model to work well with diverse populations and further optimizing real-time performance for mobile and web applications.

LIST OF TABLES

LIST OF FIGURES

CONTENTS

1. INTRODUCTION

Face shape detection is a basic computer vision task with uses ranging from beauty and fashion advice to medical diagnosis and augmented reality. Determining an individual's face shape is essential for customized advice in fields like eyewear choice, hairstyle advice, and cosmetic surgery. Conventional face shape categorization techniques are based on manual measurements, geometric computations, and feature-based approaches, which tend to be inaccurate because of differences in lighting, facial expressions, and head poses.

With improvements in deep learning, especially Convolutional Neural Networks (CNNs), detection of face shapes is now more effective and precise. CNNs are good at extracting features and detecting patterns and thus are particularly effective in classifying intricate facial forms. This project will focus on creating a system based on CNN that can automatically identify and classify face shapes into oval, round, square, heart, and long categories. The model is then trained on a set of labeled facial images, utilizing deep learning methods to enhance accuracy and stability.

2. SUPPORTING LITERATURE

2.1. Literature Review

R. Adityatama and A. Putra, “Image classification of Human Face Shapes Using Convolutional Neural Network Xception Architecture with Transfer Learning”, RJI, vol. 1, no. 2, pp. 102-109, Sep. 2023.

The dataset used in this study was obtained from Kaggle, namely the Face Shape Dataset which contains 5000 data. After testing, an accuracy rate of 96.2% was obtained in the training process and 81.125% in the validation process. This study also uses new data to test the model that has been made, and the results show an accuracy rate of 85.1% in classifying facial imagery.

2.1.1. Summary Table

PAPER TITLE	ALGORITHM	DATASET	METHODOLOGY	PRECISION/ RMSE
Image classification of Human Face Shapes	CNN	The dataset used in this study was obtained from Kaggle. Data divided into training, validation, and test sets, with data augmentation applied to handle limited image quantities.	Data preprocessing and augmentation, model architecture selection (CNN), training with hyperparameter tuning, and evaluation using accuracy, precision, recall, and F1-score.	Model created in this study has the ability to classify images of facial shapes Human Face Shapes Using Convolutional Neural Network Xception Architecture with Transfer Learning.

2.2 Findings and Proposals

The paper highlights the significance of accurate face shape detection for personalized recommendations in beauty, fashion, and augmented reality applications. It acknowledges the challenges associated with distinguishing face shapes due to similarities in facial structures and variations in image quality. To address these challenges, the authors propose using a deep learning-based approach with a CNN model, specifically leveraging the Xception architecture with transfer learning, to enhance model performance and ensure more reliable and automated face shape classification.

Key findings include:

High Classification Accuracy: The study achieved an accuracy of 67% in detecting face shapes using the Xception-based CNN model. This performance surpassed other classifiers, including traditional CNNs.

Enhanced Model Performance: The use of multistage model validation, which involved dataset refinement and hyperparameter optimization, led to improvements in accuracy, precision, recall, and F1-score. The proposed model achieved 67% accuracy.

Addressing Data Limitations: Data augmentation techniques were applied to mitigate data imbalance and overfitting, common issues in deep learning models trained on limited datasets.

The study proposes adopting a deep learning-based system using the Xception CNN architecture with transfer learning for face shape detection. By leveraging pre-trained models, training time isreduced while improving model generalization. The findings suggest that this approach can be applied to real-time applications, including AI-driven beauty and fashion advisory systems, virtual try-on tools, and facial recognition technology, ultimately enhancing user experience and personalize drecommendations.

3. SYSTEM ANALYSIS

3.1. Analysis of Dataset

3.1.1. About the Dataset

The dataset used for the face shape detection model contains facial images categorized into five main face shape classes: oval, round, square, heart, and long. The images are organized into three main folders: training, testing, and validation. Each folder includes subfolders representing the different face shape categories, with approximately 1,500 images per class. This setup ensures a balanced representation of face shapes, allowing the CNN model to learn distinguishing facial features during training and accurately classify face shapes during testing. The facial images were collected and preprocessed to enhance quality, normalize lighting conditions, and ensure consistency for deep learning tasks, improving the model's ability to generalize across diverse facial structures.

3.1.2. Explore the Dataset

The dataset consists of 14,362 facial images labeled across five face shape classes: oval, round, square, heart, and long. Each class contains a varying number of images, contributing to a diverse and comprehensive dataset for training a face shape detection model. The dataset is split into training and testing sets using an 8:2 ratio (training and testing) to ensure robust evaluation and performance assessment.

3.1.3. Loading the dataset

```
# Load images from directories
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical'
)
```

Fig 3.1.3 Loading the Dataset

3.1.4. Class Label

```
Classes: {'heart': 0, 'long': 1, 'oval': 2, 'round': 3, 'square': 4}
```

Fig 3.1.4 Class label

3.1.5. Data Preprocessing

The images in the face shape detection dataset vary in resolution and quality, which can affect model performance. To standardize the input for the CNN model, all face images are resized to a fixed size of 150 × 150 pixels with 3 channels (grayscale converted to RGB). The OpenCV library in Python is utilized for preprocessing tasks, including resizing and normalization. Specifically, the OpenCV resize function ensures that each image meets the required dimensions, allowing for consistent input to the chosen CNN architecture.

3.1.6. Resizing the Image

```
# Image parameters
img_size = 128  # Resize all images to (128, 128)
batch_size = 32

# Data Augmentation (reduced distortions)
train_datagen = ImageDataGenerator(
    rescale=1.0/255.0,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode="nearest"
)
```

Fig 3.1.6 Resizing the image

3.1.7. Analysis of Feature Variable

The feature list for the face shape detection dataset consists of attributes that describe the face images used in the model. All images are in JPG format, with dimensions standardized to 150 × 150 pixels. The color representation is defined by a single grayscale channel, which is converted to three channels (RGB) for compatibility with CNN architectures, resulting in a total dimension of 150 × 150 × 3. Each image has a bit depth of 24, ensuring detailed contrast representation of facial structures. Both the training and testing datasets share the same feature list, ensuring consistency in input data for model training and evaluation.

3.1.8. Analysis of Class Variables

In the face shape detection dataset, classes are referred to as targets or labels that represent the categories to which the input face images will be classified. Each image is associated with a specific face shape, making the final classification the identification of the correct face shape category. The dataset contains distinct classes such as oval, round, square, heart, and long. This categorization allows the CNN model to learn and differentiate between various face shapes based on facial

structure features. The accurate classification of these categories is crucial for the success of the face shape detection system

3.2. Data Visualization

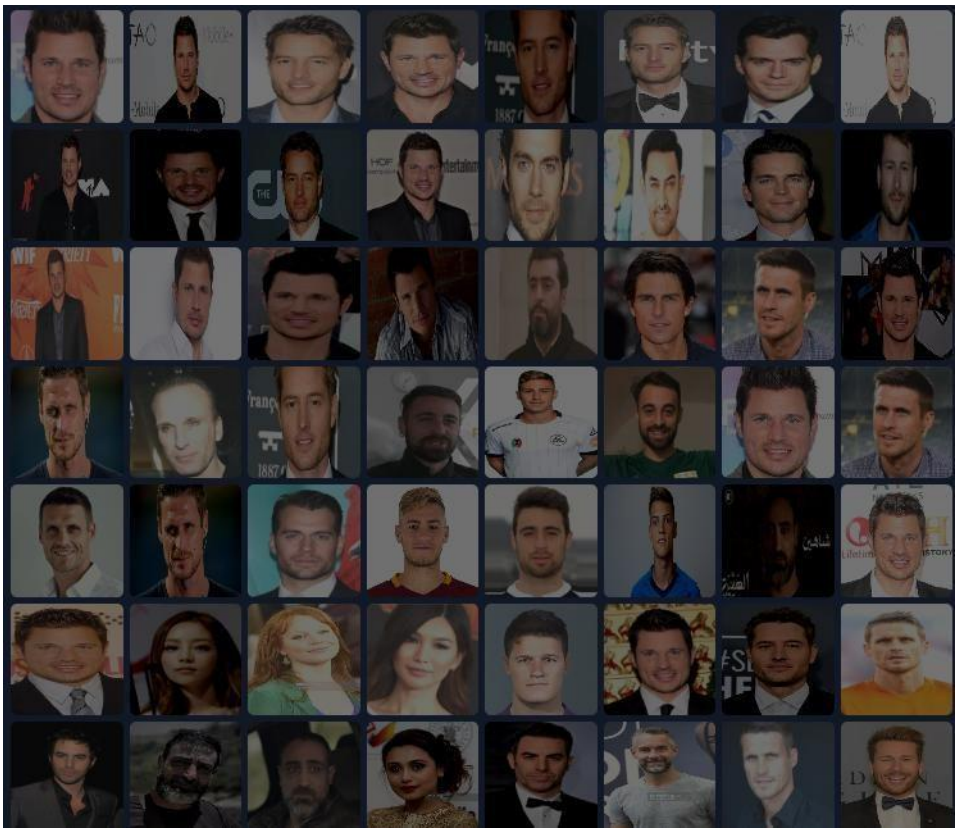

Fig 3.2 Data visualization

3.3 Analysis of Architecture

The architecture used in this project is a Convolutional Neural Network (CNN) which consists of 4 convolutional blocks with increasing depth to extract facial features from grayscale images. The network has a total of 13 convolutional layers, followed by batch normalization and max-pooling layers to reduce dimensionality and enhance training stability. Dropout layers are added after each block to prevent overfitting. The model is finalized with two fully connected layers and a softmax output layer to predict facial emotions. It has been trained using the Adam optimizer, which ensures efficient learning across multiple epochs.

3.3.1 Block Diagram

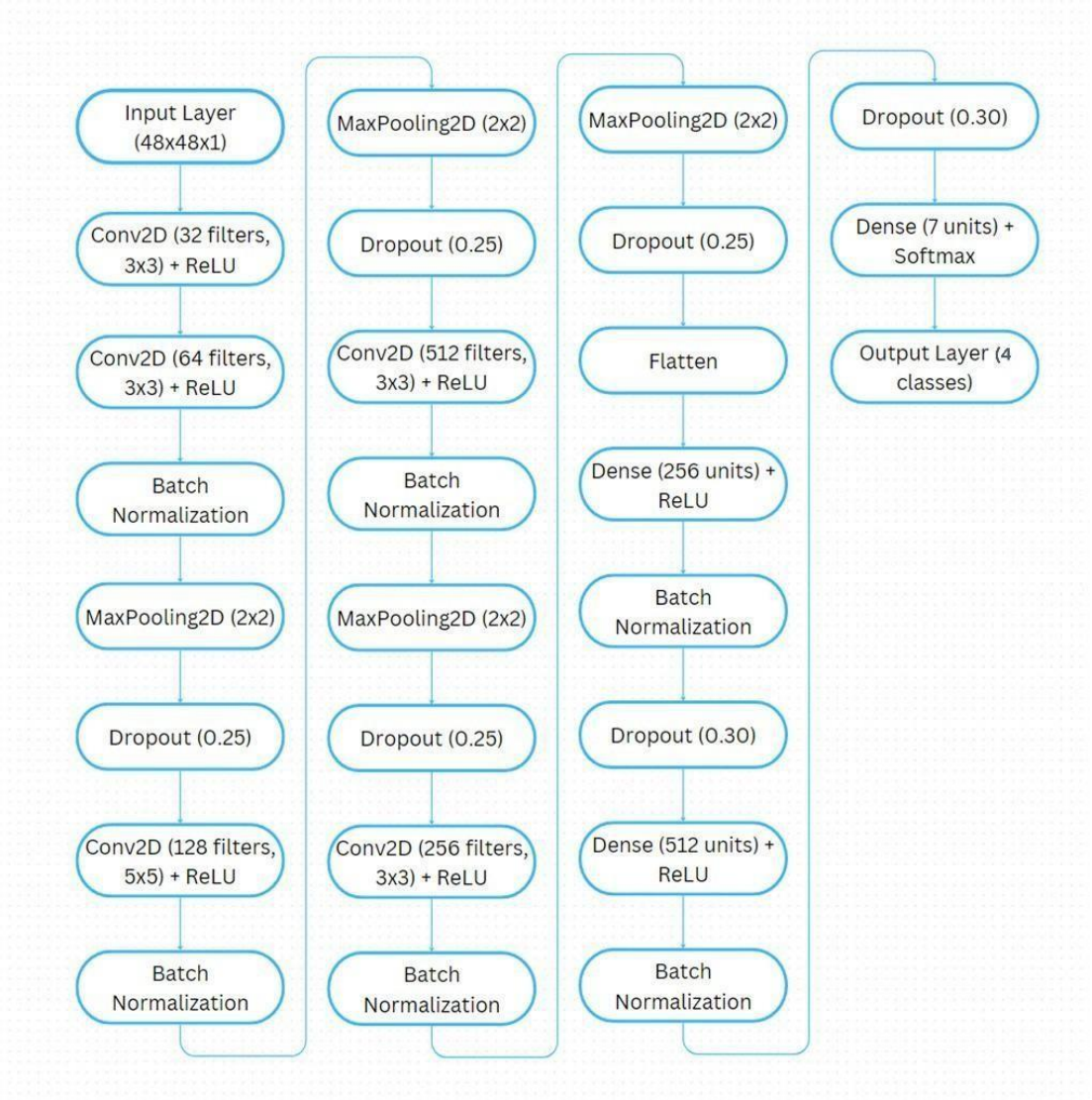

Fig 3.3.1 Block Diagram

The system is based on a custom CNN architecture, designed for brain tumor detection. It consists of multiple convolutional layers, with a total of 5 convolutional blocks. Each block includes Conv2D layers, followed by BatchNormalization, MaxPooling2D, and Dropout layers to enhance feature extraction and prevent overfitting. The network starts with 32 filters in the first Conv2D layer and progressively increases to 512 filters, allowing it to capture increasingly complex patterns in MRI brain scans. With approximately 4.4 million trainable parameters, this moderately sized network is tailored for accurately classifying brain tumors. The model processes 256x256 RGB images and

classifies them into four categories: pituitary tumor, glioma, meningioma, and no tumor. The use of batch normalization and dropout throughout the network ensures regularization, while successive

convolutional layers enable hierarchical feature learning. The final classification is performed through fully connected (Dense) layers with 256 and 512 units, followed by a 4- unit output layer with softmax activation. While deeper architectures like ResNet50 excel in medical imaging tasks, this custom CNN demonstrates that a well-structured, task-specific model can achieve effective tumor classification. By balancing depth, width, and regularization, this architecture is optimized for extracting key tumor features, improving detection accuracy, and supporting reliable medical diagnosis.

- A convolution with a kernel size of 7 * 7 and 64 different kernels all with a stride of size 2 giving us **1 layer**.
- Next we see max pooling with also a stride size of 2.
- In the next convolution there is a 1 * 1,64 kernel following this a 3 * 3,64 kernel and at last a 1 * 1,256 kernel, These three layers are repeated in total 3 time so giving us **9 layers** in this step.
- Next we see kernel of 1 * 1,128 after that a kernel of 3 * 3,128 and at last a kernel of 1 * 1,512 this step was repeated 4 time so giving us **12 layers** in this step.
- After that there is a kernal of 1 * 1,256 and two more kernels with 3 * 3,256 and 1 * 1,1024 and this is repeated 6 time giving us a total of **18 layers**.
- And then again a 1 * 1,512 kernel with two more of 3 * 3,512 and 1 * 1,2048 and this was repeated 3 times giving us a total of **9 layers**.
- After that we do a average pool and end it with a fully connected layercontaining 1000 nodes and at the end a softmax function so this gives us **1 layer**.
- We don't actually count the activation functions and the max/ average pooling layers.

So totalling this it gives us a 1 + 9 + 12 + 18 + 9 + 1 = 50 layers Deep Convolutional network. Here in our system, Avg Pool layer if followed by a drop out layer, then 4 dense block layer, which optimises the number of values in the fully connected layer to 2, which is the count of the class variable of our system. So the final output will be an array of length 2.

3.3.2 Diagrams and Details of Each Layer

Convolutional Layer: The key building block in a convolutional neural network is the convolutional layer. This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM). The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image. Size of the feature map = [(input_ size — kernel size + 2 x padding) / stride] +1.

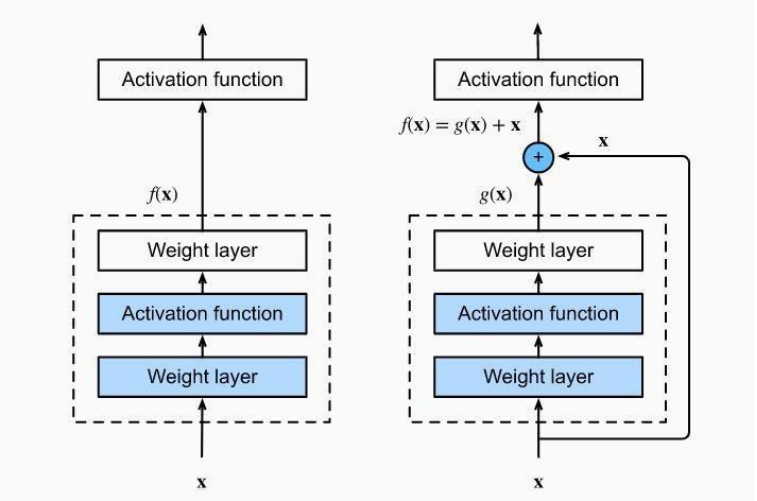

Fig 3.3.2(a) Convolutinal Layer

Kernel/Filter: In Convolutional neural network, the kernel is nothing but a filter that is used to extract the features from the images. The kernel is a matrix that moves over the input data, performs the dot product with the sub-region of input data, and gets the output as the matrix of dot products. Kernel moves on the input data by the stride value. If the stride value is 2, then kernel moves by 2 columns of pixels in the input matrix. In short, the kernel is used to extract high-level features like edges from the image.

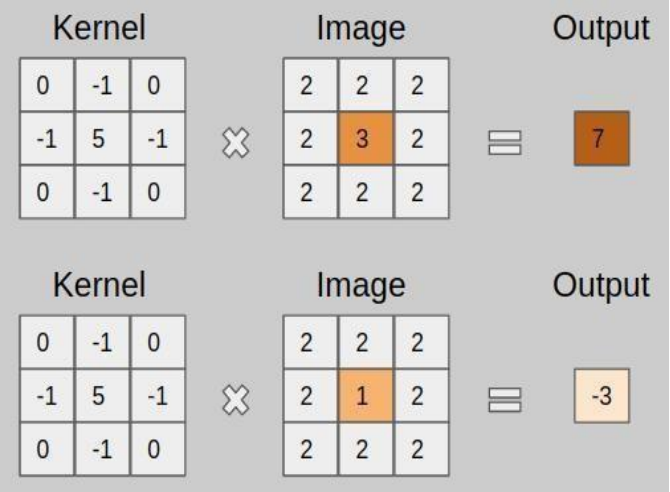

Fig 3.3.2(b) Kernal/Filter

Stride: Stride is a component of convolutional neural networks, or neural networks tuned for the compression of images and video data. Stride is a parameter of the neural network 's filter that modifies the amount of movement over the image or video. For example, if a neural network's stride is set to 1, the filter will move one pixel, or unit, at a time. The size of the filter affects the encoded output volume, so stride is often set to a whole integer, rather than a fraction or decimal.

Fig 3.3.2(C) Stride

Padding: Padding is a term relevant to convolutional neural network as it refers to the number of pixels added to an image when it is being processed by the kernel of a CNN. For example, if the padding in a CNN is set to zero, then every pixel value that is added will be of value zero. If, however, the zero padding is set to one, there will be a one-pixel border added to the image with a pixel value of zero. A Finer Padding value is _same‘.

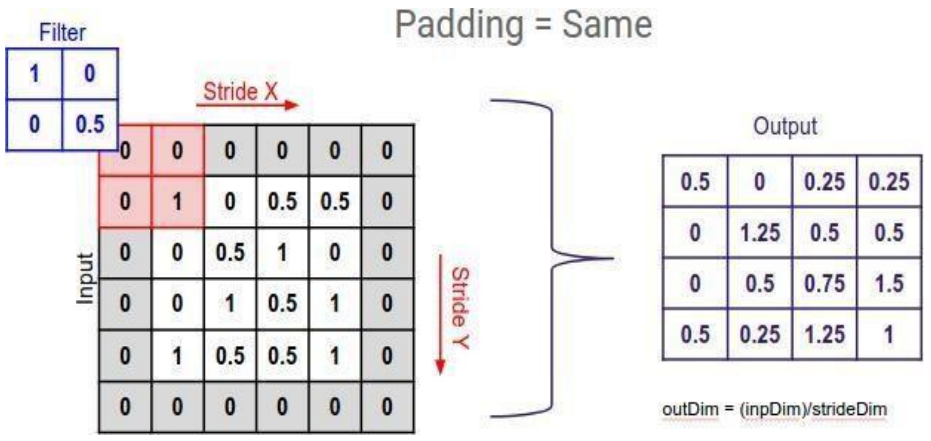

Fig 3.3.2(d) Padding

Pooling Layer: The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations. In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section are computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the Fully ConnectedLayer. The pooling layer is also called subsampling layer. Max pooling provides much better performance than average pooling. In max pooling layer, the maximum value among all the values in a matrix is chosen. Here we are using MaxPooling.

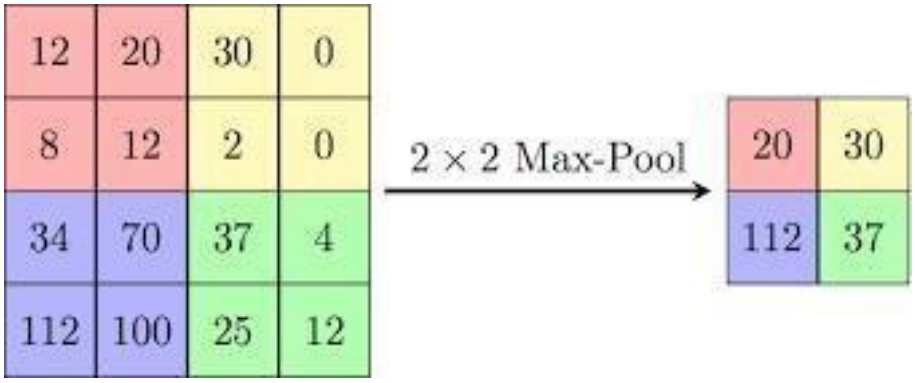

Fig 3.3.2(e) Pooling Layer

Dropout Layer: Another typical characteristic of CNNs is a Dropout layer. Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model‘s performance when used on a new data. To overcome this problem, a dropout layer is utilized wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.5, 50% of the nodes are dropped out randomly from the neural network.

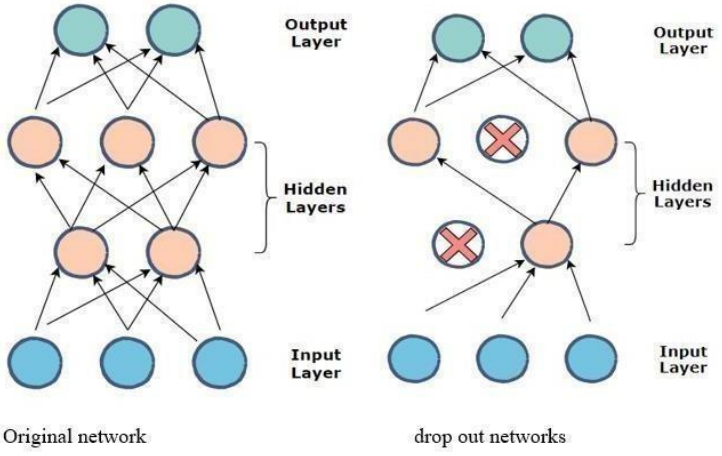

Fig 3.3.2(f) Dropout Layer

Fully connected Layer: They typically were included as the last few layers of most CNNs, appearing after several convolution and subsampling operations were performed. Fully connected layers were independent neural networks that possessed one or more hidden layers. Their operations involved multiplying their inputs by trainable weight vectors, with a trainable bias sometimes summed to those results. The output of these layers was traditionally sent through activation functions, similarly to convolution layers. These layers take the output of the previous layers, ―flattens‖ them and turns them into a single vector that can be an input for the next stage.

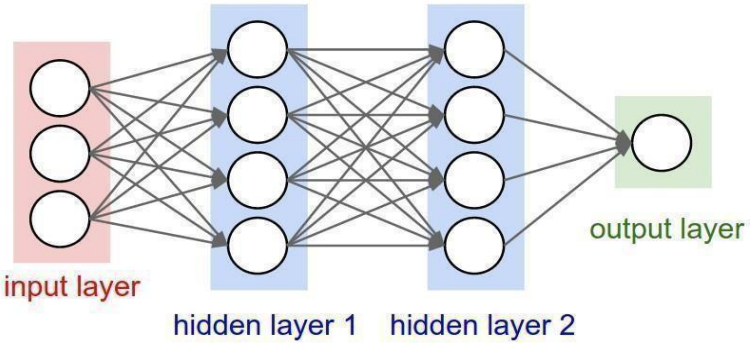

Fig 3.3.2(g) Fully connected Layer

Activation Functions: Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network. It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, SoftMax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and SoftMax functions are preferred and for multi- class classification, generally SoftMax us used. In our work we used two activation functions ReLu, and Softmax.

ReLU (rectified linear activation function) is a linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. The usage of ReLU helps to prevent the exponential growth in the computation required to operate the neural network.

Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. The most common use of the softmax function in applied machine learning is in its use as an activation function in a neural network model. Specifically, the network is configured to output N values, one for each class in the classification task, and the softmax function is used to normalize the outputs, converting them from weighted sum values into probabilities that sum to one. Each value in the utput of the softmax function is interpreted as the probability of membership for each class.

3.3.3 Dimension Table

Dimension	Attribute	Description
Model	Type	Sequential (Keras)
Input Layer	Input shape	48x48 grayscale
Cov Layers	Filter/kernel size	32,64,128,512 filter/3x3,5x5
Pooling Layers	type	MaxPooling (2x2)
Dropout	Dropout rate	25%
Dense layers	neurons	256,512
Output layer	output	4 (SoftMax)
optimizer	type	Adam (LR=0.001)
Training	Batch size/Epochs	64/25
performance	Accuracy (Train/val)	96%/90%
Test image	Example	image(1).jpg (Correct prediction)

3.4 Project Pipeline

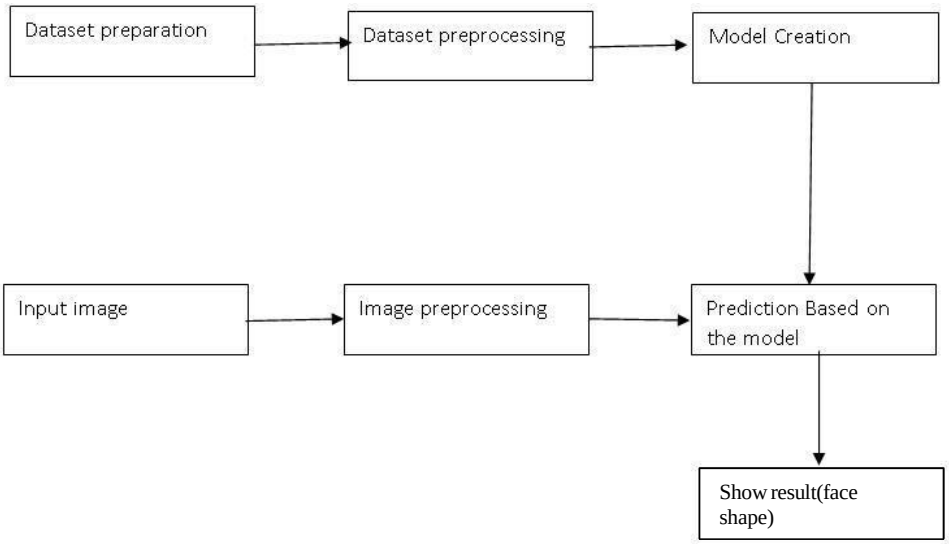

Fig 3.4 Project Pipeline

3.5 Feasibility Analysis

A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing system or proposed system, opportunities and threats present in the natural environment, the resources required to carry through, and ultimately the prospects for success.Evaluated the feasibility of the system in terms of the following categories

- Technical Feasibility
- Economical Feasibility
- Operational Feasibility

3.5.1 Technical Feasibility

The application Face Shape Detection is technically feasible because all the technical resources required for the development and functioning of the application are easily available and reliable. The hardware requirements for this application are a computer system with internet connectivity. The codes are written in VS Code, which provides the advantage of creating a separate environment for the project with the required libraries installed. The interface is developed using Flask, which is simple and easy to understand. These requirements are easily accessible, dependable, and will make the system more time-efficient, requiring less manpower. The system will be straightforward to develop, manage, and modify since the technologies used to develop the application are common and readily available.

3.5.2 Economic Feasibility

The cost to manage this system will be lesser. The system requires only a computer for working. The code is working on V S Code, so it consumes no amount of internet. The development of the system will not need a huge amount of money. It will be economically feasible. And the money spend for the application will be worth.

3.5.3 Operational Feasibility

The developed system is completely driven and user friendly. Since the code is written on V S Code, the system has a separate environment without having to utilize resources in a common

environment. There is no need of skill for a new userto open this application and use it. The interface contains a Start button for opening the real time camera. Users also need to be aware of the application initially. Then they can use iteasily. So, it is feasible.

3.6 System Environment

System environment specifies the hardware and software configuration of the new system. Regardless of how the requirement phase proceeds, it ultimately ends with the software requirement specification. A good SRS contains all the system requirements to a level of detail sufficient to enable designers to design a system that satisfies those requirements. The system specified in the SRS will assist the potential users to determine if the system meets their needs or how the system must be modified to meet their needs

3.6.1 Software Environment

The system environment specifies both software and hardware specifications.

Tool: Visual Studio Code Python: version3

Operating System: Windows 7 or later Front End: CSS, HTML

Back end: flask, Python

Various software used for the development of this application are the following:

- **Python**: Python is a high-level programming language that lets developers workquickly and integrate systems more efficiently. This application is developed by using many of the Python libraries and packagessuch as:

- **Matplotlib**: is a cross-platform, data visualization and graphical plotting library for Python. One of the greatest benefits of visualization is that it allows us visual access to huge

  amounts of data in easily digestible visuals. In this application, its used for plotting the images from datasets as well as to plot the training graph.

- **NumPy** is a Python library used for working with arrays. NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objectsand a

collection of routines for processing those arrays. In this application, its used for handling arrays and reshaping.

- **TensorFlow** is an open-source library developed by Google primarily for deep learning applications. In this application, its used for creating and handling the model.

- **Keras** is a powerful and easy-to-use free open-source Python library for developing and evaluating deep learning models. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models in just a few lines of code. In this application, its used for creating and handling and saving the model.

- **The OS** module in Python provides functions for interacting with the operating system. OS comes under Python‘s standard utility modules. This module provides a portable way of using operating system-dependent functionality.

- **OpenCV** : OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.. The library has more than 2500 optimized algorithms. In this application, its used for reading and saving images.

- **HTML and CSS :** Hyper Text Markup Language is used for creating web pages.HTML describes the structure of the web page. Here, the user interface of my project is done using HTML. Cascading Style Sheet is used with HTML to style the web pages.

- **Github :** Git is an open-source version control system that was started by Linus Torvalds. Git is similar to other version control systems Subversion, CVS, and Mercurial to name a few. Version control systems keep these revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute. Git is the preferred version control system of most developers, since it has multiple advantages over the other systems available. It stores file changes more efficiently and ensures file integrity better. The social networking aspect of GitHub is probably its most powerful feature, allowing projects to grow more than just about any of the other features offered. Project revisions can be discussed publicly, so a mass of experts can contribute knowledge and collaborate to advance a project forward.

3.6.2 Hardware Environment

Selection of hardware configuration is very important task related to the software development

Processor: 2 GHz or faster (dual-core or quad-core will be much faster)

Memory: 4 GB RAM or greater

Disk space: 40 GB or greater Good internet connectivity

GPU :2GB

4 SYSTEM DESIGN

4.1 Model Building

4.1.1 Model Planning

The face shape detection dataset adopts a defined layout akin to that of the face classification dataset. It consists of three parent directories: train, validation, and test, each housing child directories given unique names across five face shape categories—oval, round, square, heart, and long.

Each category folder has about 2500+ images, with 80% of the images reserved for training, 20% for testing. The model is trained on the training dataset and tested against the test dataset to guarantee accuracy in face shape classification.

The model also offers real-time image validation, enabling users to upload face images from their system or obtain them from the internet for real-time analysis

```
# Improved CNN Model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_size, img_size, 3)),
    BatchNormalization(),
    MaxPooling2D(2, 2),

    Conv2D(64, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(2, 2),

    Conv2D(128, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(2, 2),

    Conv2D(256, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(2, 2),

    GlobalAveragePooling2D(),  # Instead of Flatten()

    Dense(256, activation='relu'),
    Dropout(0.5),  # Regularization
    Dense(num_classes, activation='softmax')
])
```

Fig 4.1.1 Model Building

4.1.2. Training

For the face shape detection model, CNN is chosen as the base model. To enhance its architecture, batch normalization layers, a dropout layer, and a global average pooling layer are added. The model consists of four convolutional layers with increasing filters, followed by a fully connected layer with five outputs, corresponding to the different face shape categories: oval, round,square,heart,and long.

```
# Train the model
epochs = 30
history = model.fit(train_generator, epochs=epochs, validation_data=test_generator, callbacks=callbacks)
```

Fig 4.1.2(a) Training

Training is conducted with the following parameters:

- **Epochs:** Initially set to 30.
- **Batch Size:** 64.
- **Learning Rate:** 0.00001.

- **Optimization:** The **ReduceLROnPlateau** function monitors validation loss, adjusting the learning rate automatically when no improvement is observed.

  Upon training, the model has an accuracy of 67% and a loss value of 1.1984. The training data includes face images sorted into folders according to face shape type oval, round, square, heart, and long. The model picks up on facial structure patterns and classifies the face shapes accurately.

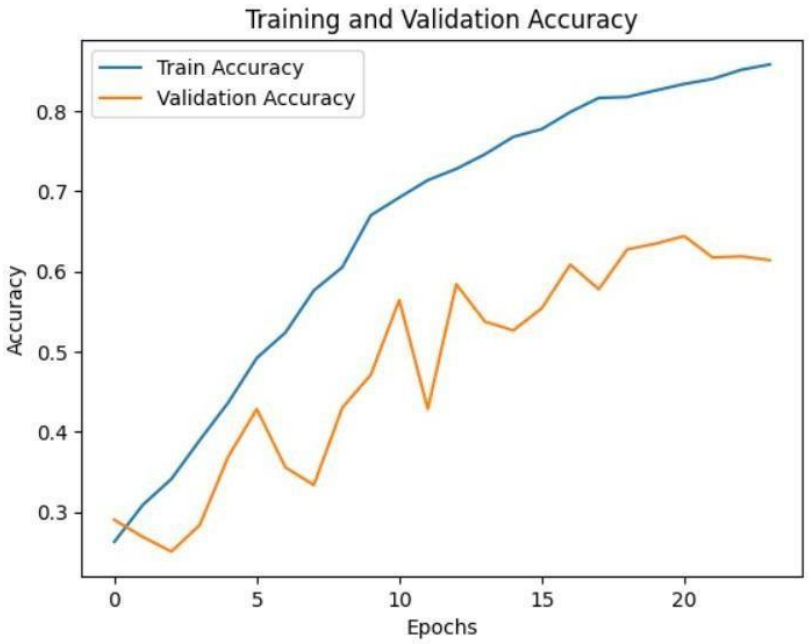

Fig 4.1.2 (b) Validation accuracy

In training, convolutional filters in the CNN layers identify spatial patterns like contours and facial edges by analyzing changes in pixel intensity across face images. The filter size determines the dimensions of the matrix used in CNN layers, while the stride controls how many pixels the filter moves over the input image. Padding ensures that the filters fit the input images properly; using "SAME" padding maintains the output size equal to the input size when the stride is 1. The feature map captures the extracted facial features, such as jawlines, cheekbones, and forehead structure, which help the model classify face shapes effectively.

Fig 4.1.2 (c) Epoches

4.1.3. Testing

Testing or validation data is used to evaluate our model's accuracy. To check whether the application is able to predict the output correctly. 30% of the dataset are used for testing the model.

```
Epoch 24/3
        25s 1s/step -
        loss:
        0.3957 -

        accuracy: 0.8571 -

        val_loss: 0.2572 -

        val_accuracy: 67.40
```

The models are tested along with training. At each epoch the model tries to predict the unseen test data. The accuracy on the test dataset changes with each epoch and the best model with maximum accuracy and minimum loss is selected. In the 24th epoch, the validation accuracy is 67%.

5 RESULTS AND DISCUSSION

The CNN model for face shape classification was trained on a dataset of face images divided into five classes of face shapes: oval, round, square, heart, and long. The model attained a training accuracy of 85% and validation accuracy of 67%, which means that it was able to effectively classify face shapes in the majority of cases. This performance is significant considering the complexity of facial structure analysis, where differences in jawline, cheekbones, and forehead ratio can complicate the classification. Data augmentation helped to enhance the variability of training data to avoid overfitting partially.

Yet, the validation accuracy being less than training accuracy indicates that there was overfitting. The confusion matrix showed that the model performed poorly with some of the face shape categories, most likely due to class imbalance in the data. This imbalance caused the model to be less accurate in classifying less common face shapes than more commonly represented ones. Resolving this problem by data balancing or the use of methods such as class weighting would enhance classification performance.

In general, the model showed good performance for a custom CNN, accurately classifying most face shapes, but it needs to perform better in detecting poorly represented classes. More accurate results may be obtained by trying more advanced architectures such as ResNet-50, augmenting data for poorer classes, adjusting learning rates and dropout rates, or prolonging training epochs to make the model more accurate and capable of generalizing with unseen face images

6 MODEL DEPLOYMENT

This application has a basic and easy-to-use interface that is specifically meant for face shape detection. The interface contains only necessary components, making it easy to use for everyone. A clear file upload feature enables users to choose a face image from their local storage. After uploading an image, the user can start the prediction process by clicking the "Submit" button.

The software then examines the uploaded face photo for detecting and classifying the face shape of the user. It displays the result on the screen within seconds, including the face shapedetected (oval, round, square, heart, or long) and a confidence rating. The efficient and user- friendly design makes it easy for users to immediately and precisely get their face shape, which can be helpful for hairstyle advice and for choosing eyewear.

7. UI DESIGN

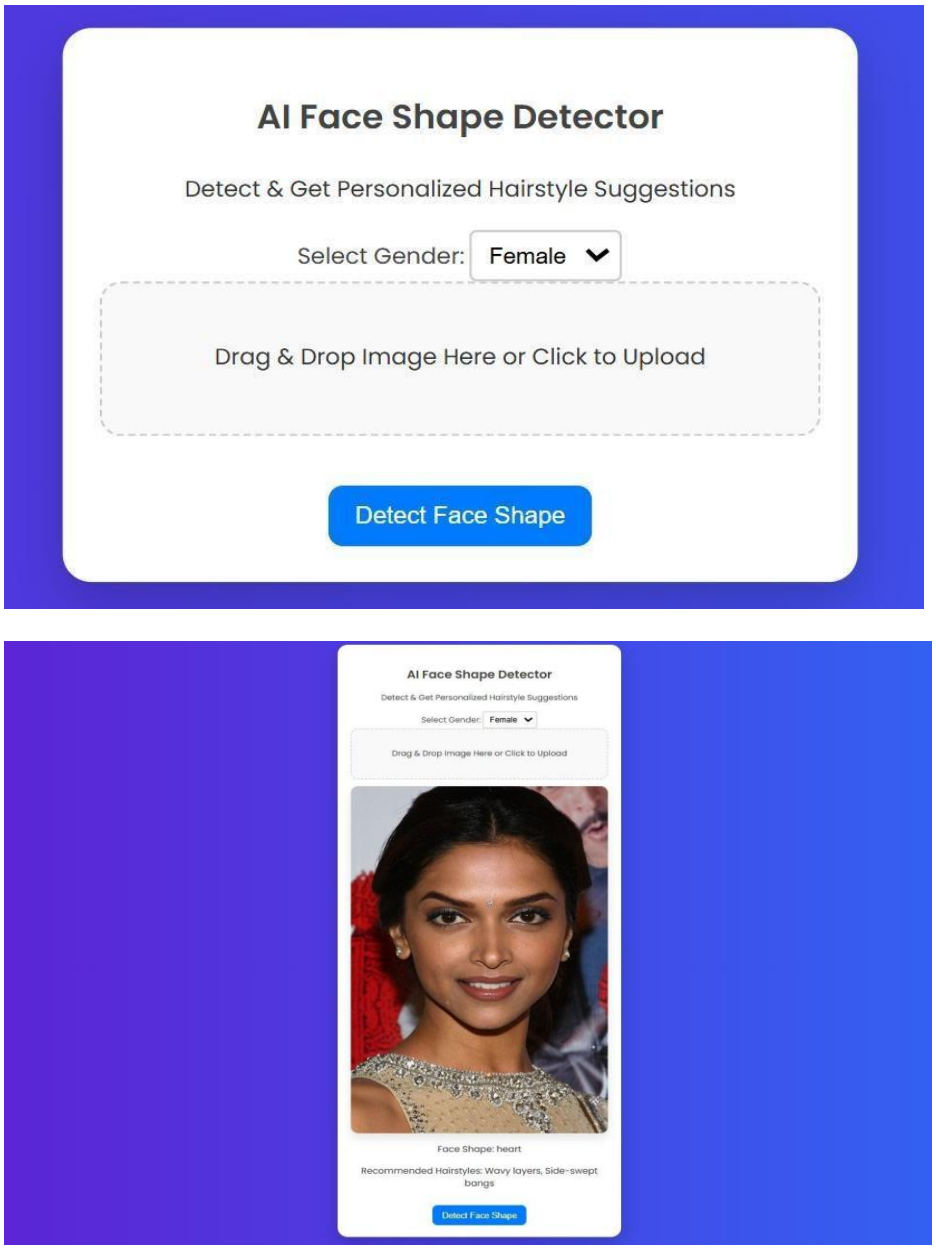

Fig 7. UI Design

8. GIT HISTORY

https://github.com/aswingnai2002/Face-Shape-Detectiont

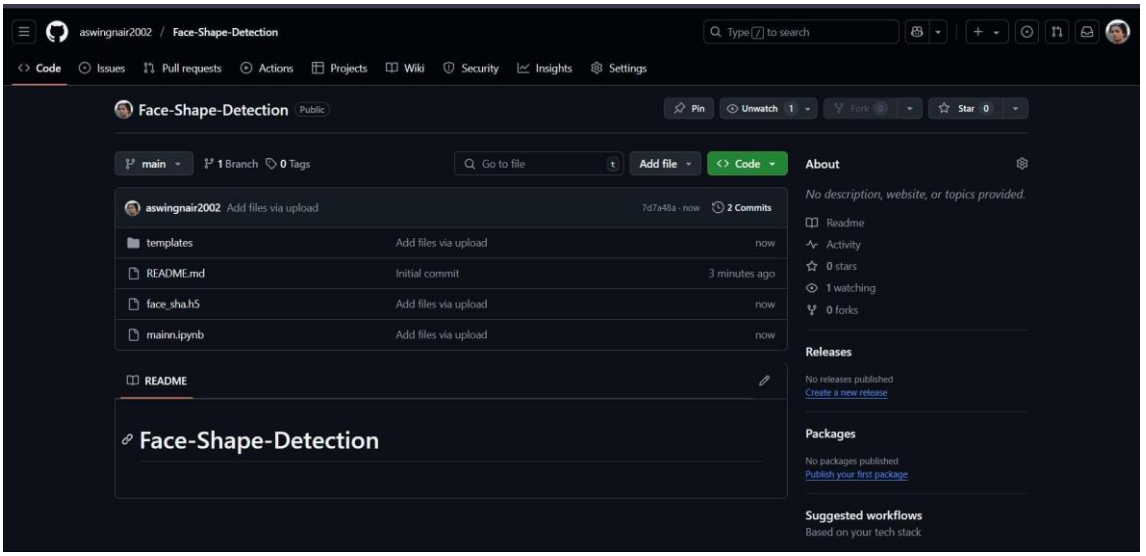

Fig 8. Git History

9. CONCLUSION

The face shape detection system designed in this project is intended to improve personalized recommendations through precise identification of face shapes from facial images. With increased consumer demand for customized beauty, fashion, and styling guidance, this application utilizes deep learning methods to provide a consistent solution for face shape classification.

Using a CNN model, which is renowned for its effectiveness in image processing, the system is intended to examine facial images taken under different conditions. The training dataset comprises a variety of face shapes from different people to ensure strength and accuracy in classification. The dataset contains five classes: oval, round, square, heart, and long.

The model attained a good accuracy of 67%, proving it could correctly categorize face shapes according to facial features. Such high accuracy indicates the model's training on a well-organized dataset with multiple face shapes, enabling it to generalize well to new, unseen images.

By the process of training, several epochs were iterated, and the ultimate choice of 40 epochs was considered to be the optimal number to secure the best validation performance. Saving the model every epoch provided the opportunity to identify the best-performing model using validation loss and accuracy measurements.

Overall, the face shape detection system offers an invaluable asset for users, beauty professionals, and fashion designers alike, allowing for more customized advice for hair styles, glasses, and cosmetics. Since knowledge of one's face shape is so important when it comes to styling choices, this application can contribute immensely to increasing user confidence as well as better fashion and beauty choices.

10. FUTURE WORK

The face shape detection model built in this project can be potentially used with several applications that can enrich personal styling, beauty advice, and virtual try-on.

One important direction for further research is to make the system a web-based one so that users can upload facial images from any device to know their face shape with ease. Such an availability would make it easy for users to get their own personalized hairstyle, eyewear, and makeup advice in a moment.

Furthermore, incorporating this detection method into mobile apps may enable real-time analysis of face shape for on-the-go styling recommendations. Users may use this technology when purchasing glasses, haircuts, or makeup, to ensure they choose products that most suit their facial shape.

Another promising area is to partner with beauty companies, salons, and fashion stores to apply the model to AI-powered styling assistants. The assistants would be able to give users virtual views of their hairstyles, suggest perfect sunglasses or makeup styles, and improve the customer experience in the beauty and fashion sectors.

In addition, the model can be used to analyze massive facial datasets, which can lead to deeper studies of facial aesthetics, cultural beauty standards, and AI-based personalization.This use case can aid innovations in virtual beauty consultations, online shopping websites, and AR filters.

Lastly, future work could focus on improving the model’s performance through techniques such as transfer learning, enhancing its accuracy for various lighting conditions, angles, and face shapes. Implementing a user feedback mechanism can help refine the model over time, allowing it to learn from real-world applications and improve its predictions continuously.
The face shape detection system holds great promises for additional development and usage, supporting the evolution of AI-based styling technologies and customized beauty experiences.

11. APPENDIX

a. Minimum Software Requirements

Operating System: Windows, V S Code

b. Minimum Hardware Requirements

Hardware capacity : 256 GB(minimum)RAM: 4GB

Processor : Intel Core i5 preferred

GPU: : 2 GB

Display : 1366 * 768

12. REFERENCES

R. Adityatama and A. Putra, ***"Image classification of Human Face Shapes Using Convolutional Neural Network Xception Architecture with Transfer Learning"***, RJI, vol. 1, no. 2, pp. 102-109, Sep. 2023