Face Shape Detection Using Transfer Learning and Spectacle Recommendation System

Computer Vision & Pattern Recognition

Project Report

submitted by-

Niladri Ghosh [B2430100]

Arnab Ghosh [B2430069]

Under the guidance of

Br. Tamal

Department of Computer Science

Ramakrishna Mission Vivekananda Educational and Research Institute

April 29, 2025

CONTENTS

ABSTRACT

The rapid advancement of deep learning has opened new avenues for personalization in the beauty and fashion industries. Among various aspects, face shape classification plays a crucial role in recommending suitable products such as hairstyles, makeup, and sunglasses. However, challenges such as diverse facial features, varied lighting conditions, and background noise often hinder accurate classification.

This project focuses on developing an intelligent system for classifying female face shapes into five categories — Heart, Oblong, Oval, Round, and Square — using Convolutional Neural Networks (CNNs) and Transfer Learning with pre-trained VGGFace models. The methodology emphasizes extensive preprocessing steps, including face detection, cropping, augmentation, and model optimization.

Furthermore, a lightweight alternative based on Mediapipe FaceMesh landmark extraction is explored for real-time applications. In addition to face shape classification, the system integrates a feature-based recommendation engine and a virtual try-on module to enhance the interactive experience for users.

The results demonstrate the effectiveness of combining deep learning and geometric techniques to achieve robust and scalable personalized solutions for the beauty and fashion sectors.

SECTION I

INTRODUCTION

In today's beauty and fashion industries, personalization has emerged as a vital factor in enhancing customer satisfaction and loyalty. Adoption rates remain low due to the lack of automated, accurate personalization tools.

A critical aspect of personalization in beauty and fashion is the classification of face shapes, which can guide the recommendation of suitable hairstyles, sunglasses, makeup styles. However, face shape classification is a complex task due to the wide variations in facial structures, poses, lighting conditions, and background clutter.

This project addresses the challenge by developing a robust system for female face shape classification into five distinct categories: Heart, Oblong, Oval, Round, and Square. Leveraging deep learning techniques, particularly Convolutional Neural Networks (CNNs) and Transfer Learning, the project aims to achieve high accuracy and generalization across diverse images.

Additionally, to offer lightweight alternatives suitable for real-time applications, geometric methods based on facial landmarks extracted through Mediapipe's FaceMesh solution are explored. Beyond face shape classification, the system integrates a feature-based recommendation engine for personalized sunglasses and hairstyle suggestions, along with a virtual try-on module that enhances user interaction.

By combining deep learning models, geometric analysis, and interactive recommendation systems, this project aspires to bridge the gap between consumer expectations and technological advancements in the field of personalized beauty and fashion solutions.

SECTION II

PROBLEM STATEMENT

In the beauty and fashion industries, personalization is no longer a luxury but a necessity to meet evolving consumer demands. Customers expect products and services that are tailored to their unique attributes, and face shape is a critical factor influencing recommendations for hairstyles, eyewear, makeup, and accessories.

Despite advancements in computer vision and deep learning, accurately classifying face shapes remains a challenging task. Several factors contribute to this complexity, including variations in lighting conditions, facial orientations, background clutter, and inherent differences in individual facial structures.

Traditional methods relying on manual measurements or simplistic geometric rules often fail to generalize across diverse real-world scenarios. Furthermore, existing machine learning models may suffer from overfitting, poor generalization, and a lack of real-time applicability, especially when deployed in consumer-facing applications.

The core problem addressed in this project is the development of an accurate, scalable, and efficient system for automatic female face shape classification. The solution must not only achieve high classification accuracy but also offer lightweight alternatives suitable for real-time implementation, enabling personalized recommendations and enhancing user engagement in beauty and fashion platforms.

SECTION III

LITERATURE SURVEY

Face shape classification has gained increasing attention in recent years due to its relevance in beauty, fashion, and biometric applications. Various deep learning and machine learning approaches have been explored to automate face shape recognition, enhance personalization, and improve system robustness. This section reviews some of the significant contributions in this field.

Salima et al. (2023) proposed a novel approach using the Swin Transformer model for face shape classification. Their study classified faces into six categories — round, square, oblong, heart, oval, and diamond — leveraging the hierarchical structure and shifted window-based attention mechanism of the Swin Transformer. The model achieved a test accuracy of 86.34% with augmented data, demonstrating the potential of Transformer-based architectures in fine-grained face analysis tasks. The paper highlights the challenges of manual face shape identification and emphasizes the consistency offered by deep learning models in automating such processes [1].

Tio (2019) explored face shape classification using a retrained Inception v3 model, comparing its performance against traditional machine learning classifiers such as LDA, SVM, KNN, and MLP. The study showed that the Inception v3 model significantly outperformed traditional methods, achieving a test accuracy of around 84.8% on a dataset of celebrity images. This work underscores the strength of convolutional neural networks in automatically learning feature representations, eliminating the need for manual feature engineering [2].

Kim et al. (2022) focused on addressing face misalignment issues in face recognition tasks by proposing a face shape-guided deep feature alignment framework. Unlike conventional methods depen-

dent on preprocessing alignments, their end-to-end model jointly optimizes pixel and feature alignment using face shape priors. Experimental results confirmed that the model improved robustness against misaligned images, bridging the gap between controlled datasets and real-world scenarios [3].

Ade et al. (2025) introduced "Facefit," a web-based application that combines facial landmark analysis with machine learning to recognize face shapes and provide personalized styling recommendations. By integrating OpenCV and Dlib for landmark extraction and offering real-time suggestions for hairstyles, accessories, and makeup, Facefit exemplifies the application of computer vision techniques in practical beauty and wellness solutions. The authors also discussed future extensions of their platform toward healthcare and virtual shopping domains [4].

These studies collectively indicate that both CNN-based models and Transformer-based architectures have shown promising results in face shape classification tasks. Additionally, the integration of facial landmarks, real-time systems, and personalization engines highlights the trend toward holistic, user-centric solutions.

SECTION IV

DATASET DESCRIPTION

The dataset utilized in this project is the Face Shape Dataset sourced from Kaggle, specifically curated for face shape classification tasks. It contains a total of 5000 facial images, evenly distributed across five distinct face shape categories: Heart, Oblong, Oval, Round, and Square. Each class comprises approximately 1000 images, ensuring a perfectly balanced dataset that supports unbiased model training and evaluation.

The images are provided in RGB color space and exhibit diverse real-world conditions, including variations in lighting, head poses, facial expressions, and background clutter. Such diversity adds complexity to the classification task but simultaneously aids in developing robust models capable of generalizing across different environments.

All images were preprocessed to standardize their dimensions by resizing them to 224×224 pixels while preserving the aspect ratio through appropriate padding where necessary. Maintaining the RGB color channels enabled the models to leverage richer feature representations, as opposed to grayscale images.

Initial exploratory data analysis (EDA) revealed that the dataset contains minor noise, such as slight background inconsistencies and varied face orientations. Therefore, a face detection step was incorporated during preprocessing to crop the facial region and minimize irrelevant information.

Key preprocessing steps included:

- Face detection and cropping using MTCNN.
- Image resizing to 224×224 pixels with aspect ratio maintenance.
- Color channel standardization (retaining RGB format).
- Serialization of preprocessed images for efficient model training.

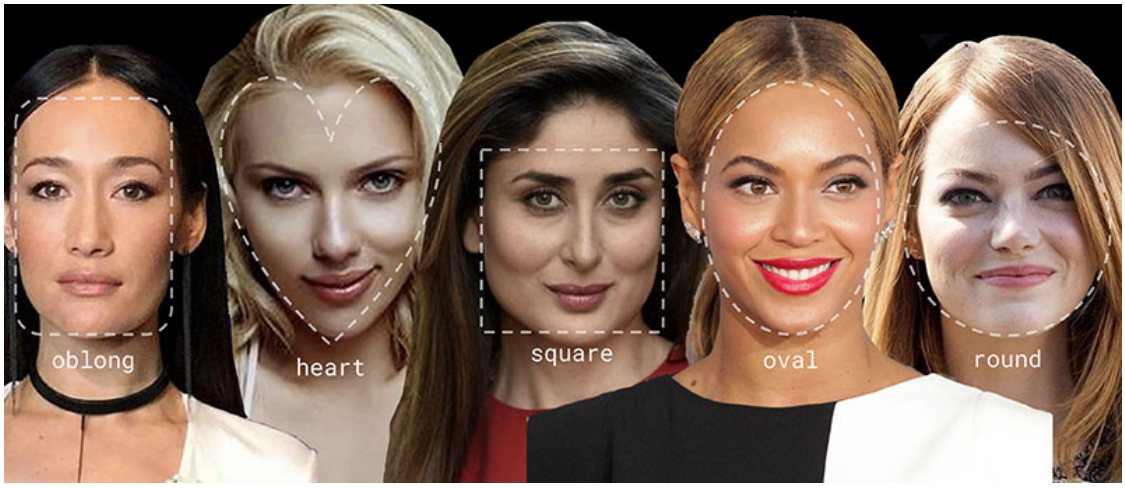


Figure IV.1: Sample illustration of different female face shapes: Oblong, Heart, Square, Oval, and Round (for illustrative purposes only).

Figure IV.1 visually demonstrates the typical examples of different face shapes. It is important to note that the above figure is used solely for illustrative purposes, and images are publicly sourced for demonstration.

Overall, the Face Shape Dataset provides a solid foundation for developing and evaluating deep learning models aimed at face shape classification, offering sufficient diversity and balance to enable both model robustness and fair performance assessment.

SECTION V

PROPOSED METHODOLOGY

V.1 Face Shape Detection using CNN

V.1.1 Step 1: Exploratory Data Analysis (EDA) and Initial Preprocessing

The foundation of any deep learning project lies in a thorough understanding and preparation of the data. In this project, our first step involved performing an exploratory data analysis (EDA) on the face shape dataset to enable effective model building.

Problem Understanding

We aimed to classify female face shapes into five categories: Heart, Oblong, Oval, Round, and Square. This classification facilitates personalized recommendations for beauty and fashion products, offering a customized consumer experience. The task can be formally described as learning a function f that maps an input image $\mathbf{X}$ to an output label y, where:

$$f : \mathbf{X} \to y, \quad y \in \{\text{Heart}, \text{Oblong}, \text{Oval}, \text{Round}, \text{Square}\}$$

Dataset Acquisition

We used the Face Shape Dataset sourced from Kaggle, containing a total of 5000 images, evenly distributed among the five face shape categories. Each input image $\mathbf{X}$ is represented as a 3-dimensional tensor in RGB color space:

$$\mathbf{X} \in \mathbb{R}^{224 \times 224 \times 3}$$

where 224 denotes the width and height (after resizing), and 3 denotes the three color channels (Red, Green, Blue).

Data Exploration

The dataset structure was examined by visualizing sample images from each category. This helped in understanding variations such as face tilt, lighting conditions, and background clutter. Such observations are critical, as these variations can introduce noise into the learning process and potentially decrease model accuracy.

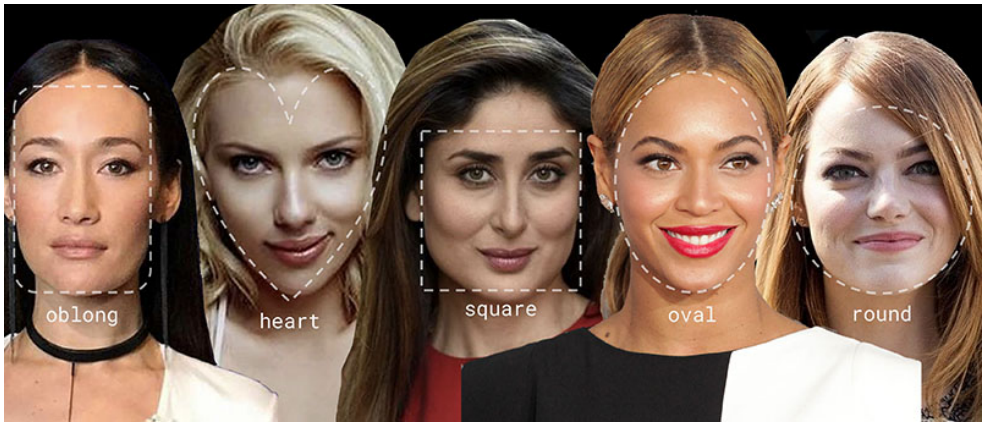


Figure V.1: Sample Images from Each Face Shape Category

The dataset was found to be perfectly balanced, ensuring that no particular face shape was underrepresented or overrepresented.

Initial Preprocessing

To prepare the data for deep learning models, several initial preprocessing steps were performed:

- **Image Resizing:** All images were resized uniformly to 224×224 pixels to ensure consistency across the dataset.

- **Color Channel Standardization:** Images were retained in RGB format rather than converted to grayscale, allowing the model to utilize the richness of color information.

- **Planning for Face Detection:** We recognized that irrelevant background details could degrade model performance. Therefore, a future step involving automatic face detection was planned to isolate the region of interest (the face).

- **Data Serialization:** To facilitate faster access during model training, the preprocessed data was serialized and stored using Python's `pickle` library.

At this stage, no augmentation or heavy transformations were applied yet. The dataset was prepared and cleaned for the subsequent modeling phases.

Learning Objective

During model training, the network aims to predict the correct face shape label y for a given input image $\mathbf{X}$ by minimizing the categorical cross-entropy loss $\mathcal{L}$, defined as:

$$\mathcal{L} = -\sum_{i=1}^{5} y_i \log(\hat{y}_i)$$

where y_i is the true label (one-hot encoded) and $\hat{y}_i$ is the predicted probability for class i.

This loss function guides the model to assign higher probabilities to the correct face shape category during training.

V.1.2 Step 2: Image Preprocessing

Preprocessing of the dataset is a crucial step to enhance model performance by focusing on the Region of Interest (ROI) — the human face — and by standardizing the input image dimensions and format.

Face Detection and Cropping

Raw images contained significant background clutter which could negatively impact model learning. Therefore, face detection was applied to extract bounding boxes tightly around facial regions.

Formally, for a raw input image $\mathbf{X}raw$, the face detection function $g(\mathbf{X}raw)$ returns the cropped face $\mathbf{X}_{face}$:

$$\mathbf{X}face = g(\mathbf{X}raw)$$

where $g(\cdot)$ is the face detection operation based on bounding box extraction.

Aspect Ratio Preservation and Image Resizing

To standardize input dimensions for the CNN models, all images were resized to 224×224 pixels. However, to avoid distortion, the original aspect ratio was maintained by padding the images appropriately before resizing.

Given an image of size (h, w), the resized image maintains:

$$\text{aspect ratio} = \frac{w}{h}$$

and any padding added satisfies:

$$\text{new width} = \text{new height} = \max(w, h)$$

Padding with black pixels was applied before resizing to a square shape.

Color Space Transformations

Three types of preprocessed images were prepared:

- **Aspect-Cropped Images:** Images resized while maintaining original aspect ratio.
- **Grayscale Face Crops:** Cropped faces converted to grayscale by reducing three channels to one.
- **RGB Face Crops:** Cropped faces preserved in RGB color space for richer feature extraction.

Each RGB image tensor is represented mathematically as:

$$\mathbf{X}_{RGB} \in \mathbb{R}^{224 \times 224 \times 3}$$

and grayscale images as:

$$\mathbf{X}_{Gray} \in \mathbb{R}^{224\times224\times1}$$

Data Serialization

The final preprocessed datasets were serialized using Python's `pickle` module, enabling efficient loading during the training phase. Each serialized file contained paired information of the image array and its corresponding label.

V.1.3 Step 3: Convolutional Neural Network (CNN) Modelling

The third phase of the project involved constructing Convolutional Neural Networks (CNNs) to classify face shapes across different types of preprocessed image datasets.

Model Architecture

The CNN architecture implemented consisted of the following sequence:

- Four convolutional layers with ReLU activation.
- Max-pooling layers after each convolution block to downsample feature maps.
- A flattening operation to convert 2D feature maps into 1D vectors.
- Two fully connected (dense) layers for final classification.
- Softmax activation at the output layer to predict class probabilities across five face shape categories.

  Mathematically, each convolutional layer operation can be represented as:

$$\mathbf{X}out = f(\mathbf{W} * \mathbf{X}in + b)$$

  where:

- $\mathbf{X}_{in}$ is the input tensor,
- $\mathbf{W}$ are the convolutional filters,

- b is the bias term,
- f is the ReLU activation function, defined as $f(x) = \max(0, x)$.

Pooling operations (MaxPooling) were used to reduce spatial dimensions:

$$\mathbf{X}pooled = \max(\mathbf{X}window)$$

Flattening transformed the pooled feature maps into a vector $\mathbf{v}$ to feed into dense layers:

$$\mathbf{v} = \text{Flatten}(\mathbf{X}_{pooled})$$

Finally, the output probability vector $\hat{y}$ is obtained by applying the Softmax function:

$$\hat{y}i = \frac{e^{z_i}}{\sum j = 1^5 e^{z_j}}$$

where z_i are the logits output by the last dense layer.

Loss Function and Optimization

Training was guided by minimizing the categorical cross-entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{5} y_i \log(\hat{y}_i)$$

The Adam optimizer was employed to update network weights efficiently.

Training Across Preprocessing Variations

Four separate models were trained corresponding to different preprocessing approaches:

1. Auto-resized images (224×224) without maintaining aspect ratio.

2. Images resized while maintaining original aspect ratio.

3. Grayscale faces detected and cropped.

4. RGB faces detected and cropped.

Each model was trained for a fixed number of epochs, and both training and validation accuracy were tracked to monitor overfitting.

Model Evaluation

Evaluation was based on overall accuracy and analysis of confusion matrices to understand class-wise prediction quality. Performance across different models revealed that models utilizing face detection and preserving color information (RGB) achieved higher validation accuracies and reduced overfitting compared to naive resizing approaches.

V.1.4 Step 4: Image Augmentation

In order to improve model generalization and reduce overfitting, data augmentation techniques were applied on the face shape dataset. This step expanded the diversity of training examples without needing additional data collection.

Augmentation Techniques

Two primary augmentation strategies were implemented:

- **Horizontal Flipping:** Each image was flipped along the vertical axis to create a mirror image. Mathematically, horizontal flipping for an image $\mathbf{X}$ of width w can be represented as:

$$\mathbf{X}_{flip}(i, j) = \mathbf{X}(i, w - j - 1)$$

- **Rotation:** Each image was rotated randomly within a range of $[-20°, +20°]$ to simulate different head poses. Rotation of an image about its center involves a transformation matrix $R(\theta)$:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

where θ is the rotation angle.

These transformations helped simulate natural variations in facial orientation and viewpoint.

Face Detection Post Augmentation

After applying transformations, the MTCNN face detector was utilized again to verify whether the face region could still be accurately detected. This ensured that the Region of Interest (ROI) remained valid even after augmentation.

Visualization of Augmented Samples

Sample images after augmentation (horizontal flip and rotation) are shown below:

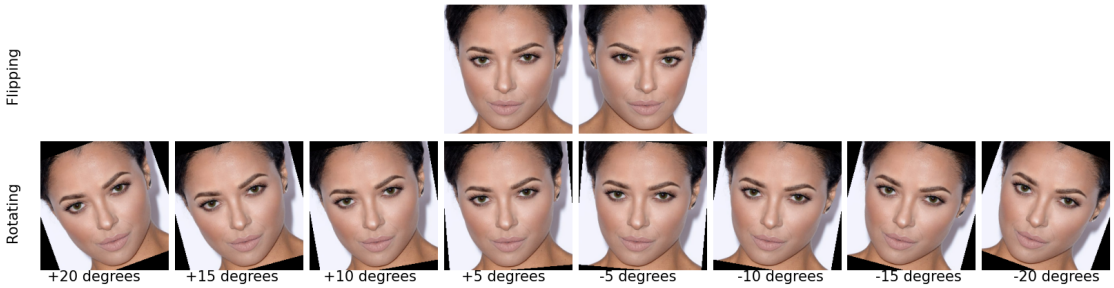


Figure V.2: Sample Images After Horizontal Flipping and Rotation

Integration into Training Dataset

The newly augmented images were added into the training dataset, effectively doubling its size. This enriched the dataset and improved the CNN model's ability to generalize to unseen face shapes.

V.1.5 Step 5: Transfer Learning with VGGFace

Given the limited dataset size and to leverage prior knowledge from large-scale face datasets, transfer learning was applied using the VGG-16 architecture with pre-trained weights from the VGGFace model.

VGG-16 Backbone

The VGG-16 model, originally trained on the ImageNet and VGGFace datasets, was loaded with pre-trained weights. The top classification layers were removed, and custom layers were added to adapt to the five face shape classes.

Transfer Learning Strategy

The transfer learning process involved:

- Freezing all pre-trained convolutional layers to retain learned feature representations.

- Adding new fully connected (dense) layers suitable for 5-class classification.
- Only training the newly added layers while keeping the backbone fixed.

The final model structure can be mathematically summarized as:

$$\hat{y} = \text{Softmax}(W_2(\text{ReLU}(W_1 f(X) + b_1)) + b_2)$$

where:

- $f(X)$ is the output feature map from the frozen VGG-16 base,
- W_1, W_2 and b_1, b_2 are the weights and biases of the new dense layers,
- Softmax converts logits into probability distributions over five classes.

The Softmax activation is defined as:

$$\hat{y}i = \frac{e^{z_i}}{\sum j = 1^5 e^{z_j}}$$

where z_i are the logits for each class.

Loss Function and Optimization

Training was guided by minimizing the categorical cross-entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{5} y_i \log(\hat{y}_i)$$

The Adam optimizer was used to efficiently update the parameters of the new dense layers.

Training and Evaluation

The model was trained on the augmented dataset. Performance was tracked through training and validation loss curves, as well as confusion matrix analysis.

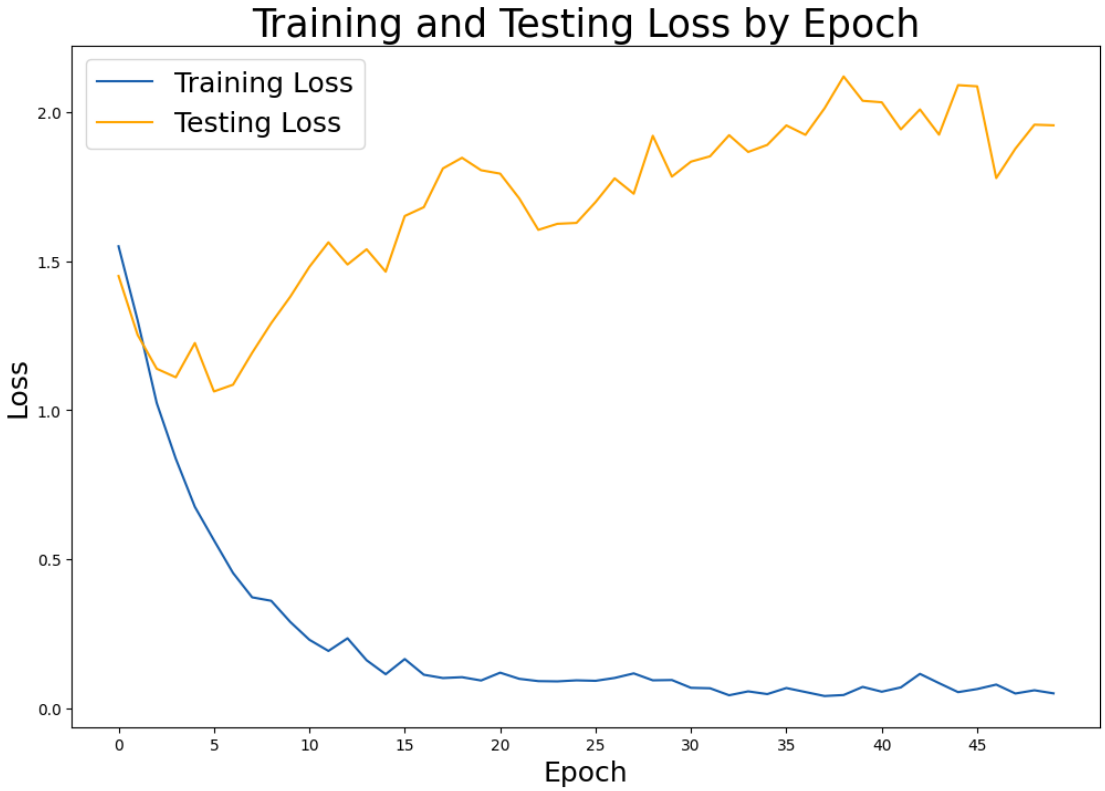


Figure V.3: Training and Validation Loss over Epochs

The transfer learning approach led to a significant improvement in validation accuracy and reduction in overfitting compared to models trained from scratch.

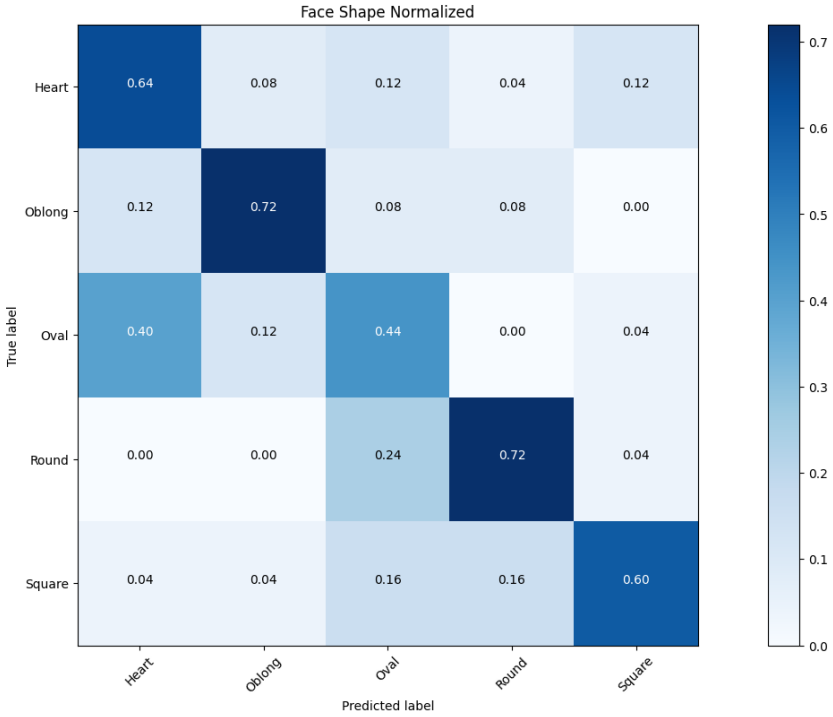


Figure V.4: Confusion Matrix for Transfer Learning Model

V.1.6 Step 6: Model Deployment and Prediction

In the final phase, the trained VGGFace-based model was deployed for real-world prediction on new images. This involved building an end-to-end prediction pipeline capable of accepting raw images and outputting predicted face shapes.

Model Loading

The PyTorch model was reconstructed using the saved state dictionary. Only the necessary layers were loaded to maintain compatibility between training and inference phases.

Formally, the prediction model f can be defined as:

$$\hat{y} = f(\mathbf{X}_{input})$$

where:

- $\mathbf{X}_{input}$ is the preprocessed input face image,
- $\hat{y}$ is the probability distribution over the five face shape classes.

Face Detection and Preprocessing

Given a raw input image, face detection was performed using the MTCNN detector. The detected face was cropped, resized to 224×224 pixels, and normalized according to the training pipeline specifications.

If no face was detected, fallback resizing with aspect ratio maintenance was applied to ensure a valid input for the model.

Prediction Pipeline

After preprocessing, the face image was passed through the CNN model to obtain softmax probabilities for each class. The predicted class corresponds to the label with maximum probability.

The prediction process can be summarized as:

$$\hat{y}i = \frac{e^{z_i}}{\sum j = 1^5 e^{z_j}}$$

where z_i are the logits produced by the model's final dense layer.

Deployment Readiness

The model and prediction pipeline were structured in modular Python scripts, making them ready for deployment into a web-based interface for real-time face shape classification.

V.1.7 Proposed Algorithm for Face Shape Detection

The entire process of face shape detection using deep learning and transfer learning can be summarized in the following algorithm:

Algorithm 1 Face Shape Detection Using Deep Learning and Transfer Learning

procedure TRAINING PHASE
    Step 1: Data Preparation
        Perform Exploratory Data Analysis (EDA)
        Resize images to 224×224 pixels
        Standardize color channels (RGB)
    Step 2: Image Preprocessing
        Apply Face Detection using MTCNN
        Crop faces and maintain aspect ratio with padding
        Save processed images
    Step 3: Custom CNN Modelling
        Build CNN architecture with convolutional and dense layers
        Train separate models on different preprocessed datasets
    Step 4: Image Augmentation
        Apply Horizontal Flipping and Rotation ($\pm 20^{\circ}$)
        Validate face detection after augmentation
        Merge augmented data into training set
    Step 5: Transfer Learning with VGGFace
        Load VGG-16 pretrained backbone
        Replace top layers with custom dense layers
        Freeze base layers, train new layers
end procedure
procedure PREDICTION PHASE
    Step 6: Model Deployment
        Load trained model
        Detect face from input image
        Preprocess detected face (crop, resize, normalize)
        Predict face shape class using softmax outputs
end procedure

V.2 Face Shape Detection using Mediapipe Landmarks

In addition to the CNN-based approach, a lightweight geometric method was implemented using facial landmarks extracted through Mediapipe's FaceMesh solution. This method leverages key facial proportions and jaw curvature to classify face shapes without requiring heavy training.

Landmark Extraction Mediapipe's FaceMesh provides 468 3D landmarks on the human face. Each landmark is a point (x, y, z) normalized relative to the image size. Given an input image, landmarks are detected as:

$$\mathbf{L} = \{(x_i, y_i, z_i)\}_{i=1}^{468}$$

where $\mathbf{L}$ represents the set of facial keypoints.

Feature Point Selection Among all the landmarks, the following six points were selected as the most critical for face shape analysis:

- Forehead Center: Landmark 10
- Chin Bottom: Landmark 152
- Left Cheekbone: Landmark 234
- Right Cheekbone: Landmark 454
- Left Jaw Corner: Landmark 205
- Right Jaw Corner: Landmark 425

Mathematical Formulations The following geometric measurements were computed:

- **Face Length:** Distance between Forehead and Chin

$$d_{\text{length}} = \|(x_{10}, y_{10}) - (x_{152}, y_{152})\|$$

- **Face Width:** Distance between Left Cheekbone and Right Cheekbone

$$d_{\text{width}} = \|(x_{234}, y_{234}) - (x_{454}, y_{454})\|$$

- **Jaw Width:** Distance between Left Jaw Corner and Right Jaw Corner

$$d_{\text{jaw}} = \|(x_{205}, y_{205}) - (x_{425}, y_{425})\|$$

- **Jawline Curvature (Angle):** Using the cosine law:

$$\theta = \cos^{-1}\left(\frac{a^2 + b^2 - c^2}{2ab}\right)$$

  where:

  - $a = d(\text{left jaw}, \text{chin})$
  - $b = d(\text{right jaw}, \text{chin})$
  - $c = d(\text{left jaw}, \text{right jaw})$

Classification Rules The face shape was predicted based on the following heuristic rules:

- If $d_{\text{length}} > 1.3 \times d_{\text{width}}$, then **Long Face**.
- If $d_{\text{jaw}} > d_{\text{width}}$, then **Triangle Face**.
- If $\theta > 130°$, then **Round Face**.
- If $\theta < 110°$ and $d_{\text{width}} \geq 0.85 \times d_{\text{length}}$, then **Square Face**.
- If $d_{\text{width}} > d_{\text{length}}$ and $d_{\text{jaw}} < d_{\text{width}}$, then **Diamond Face**.
- If forehead center is higher than both cheeks (i.e., $y_{10} > y_{234}$ and $y_{10} > y_{454}$), then **Heart Face**.
- Otherwise, default to **Oval Face**.

Advantages of Mediapipe-Based Approach

- Real-time fast performance even without GPU.
- Lightweight geometric computation instead of heavy CNN models.
- No training required — purely rule-based classification.

V.3 Age and Gender Prediction

Overview To further personalize recommendations based on facial attributes, age and gender prediction was implemented using the DeepFace framework. This step involved real-time face detection followed by attribute analysis on cropped face regions.

Face Detection For real-time face localization, the Dlib frontal face detector was utilized. Given an input video frame $\mathbf{F}$, face detection can be formalized as:

$$g(\mathbf{F}) = \{(x_i, y_i, w_i, h_i)\}_{i=1}^{N}$$

where (x_i, y_i) represents the top-left coordinate of the i-th detected face, and (w_i, h_i) are the corresponding width and height. Only faces with valid bounding box coordinates were cropped for further processing.

Face Cropping and Resizing Each detected face region was extracted and resized to a standardized size of 160×160 pixels:

$$\mathbf{F}_{face} \in \mathbb{R}^{160\times160\times3}$$

This resizing step ensured compatibility with the input requirements of the DeepFace model.

Attribute Prediction Using DeepFace DeepFace framework was employed to predict four key facial attributes:

- **Age:** Estimated as an integer value.
- **Gender:** Predicted as either "Man" or "Woman".
- **Race:** Predicted dominant ethnic category (optional in current application).
- **Emotion:** Predicted dominant emotional expression (optional in current application).

Given a cropped face image $\mathbf{F}_{face}$, the attribute prediction function $h(\cdot)$ can be expressed as:

$$h(\mathbf{F}_{face}) \rightarrow (\text{Age}, \text{Gender}, \text{Race}, \text{Emotion})$$

The DeepFace model internally leverages pre-trained convolutional backbones such as VGGFace and Facenet to perform multi-task learning over large face datasets.

Real-Time Processing Pipeline To maintain real-time performance, the following optimizations were applied:

- Predictions were performed asynchronously using Python's `threading` module.
- Attribute analysis was triggered every 5th frame to balance responsiveness and computational load.
- Predicted attributes were overlaid on the video stream for visual feedback.

Advantages

- No explicit model training was required as DeepFace models were used directly.
- Fast and lightweight real-time implementation.
- Robust to minor occlusions and variations in lighting.

V.4 Feature-Based Recommendation System

Overview After extracting facial attributes (face shape, gender, age, race, and emotion), a rule-based recommendation system was designed to provide personalized suggestions for sunglasses and hairstyles. The system mapped extracted features to fashion styles using a hierarchical decision tree.

Input Features The following attributes were used as inputs:

- **Gender:** Predicted as "man" or "woman".
- **Face Shape:** Predicted as one of Heart, Oblong, Oval, Round, Square, Diamond, Triangle, Long.
- **Age:** Predicted as an integer value.
- **Race:** Dominant racial group detected (optional adjustment).
- **Emotion:** Dominant emotion detected (optional adjustment).

Rule-Based Decision Tree The recommendation logic was based on handcrafted if-else conditions organized as a decision tree. For each user, style suggestions were made following these rules:

- **Primary Branch:** Based on Gender.
- **Secondary Branch:** Based on Face Shape.
- **Tertiary Modifiers:**
  - Age category (Youth, Adult, Mature) adjusted hairstyle descriptions.
  - Race influence tailored hair recommendations (e.g., emphasis on thick dark hair).
  - Emotion influenced sunglass color tone suggestions (e.g., vibrant for happy, pastel for sad).

Sunglasses Recommendation Suggested sunglass styles were chosen to complement and balance the face shape:

- **Oval/Heart-shaped faces:** Wayfarer, Clubmaster, or Cat-eye frames.

- **Round faces:** Rectangular or Angular frames.
- **Square faces:** Round sunglasses or Butterfly frames.
- **Diamond faces:** Oval lenses or Rimless frames.
- **Triangle/Long faces:** Aviators, Semi-rim, Bold Wrap-around designs.

Hairstyle Recommendation Hairstyle suggestions were adapted based on:

- Face shape contour (e.g., softening square faces with fringes).
- Age refinement (e.g., youthful, professional, or mature finishes).
- Race-specific adaptations (e.g., thicker styles for Indian-origin hair).

Mathematical Formalization The recommendation system can be abstractly formulated as:

$$\text{Style Suggestion} = g(\text{Gender}, \text{Face Shape}, \text{Age}, \text{Race}, \text{Emotion})$$

where $g(\cdot)$ is the deterministic mapping function designed via handcrafted rules.

Final Output For each user, two personalized recommendations were generated:

- Sunglasses Style
- Hairstyle Suggestion

The system produced immediate, easily understandable fashion suggestions aligned with user facial characteristics and expressions.

V.5 Virtual Try-On System

Overview To enhance the interactive experience for users, a real-time virtual try-on system was developed that overlays sunglasses on the user's face using webcam input. The system utilizes Mediapipe FaceMesh for landmark detection and OpenCV for transparent PNG overlay operations.

Face Landmark Detection The system employed Mediapipe's FaceMesh to detect 468 facial landmarks in real-time. In particular, two key landmarks were used:

- Landmark 33: Left eye outer corner
- Landmark 263: Right eye outer corner

Given detected landmarks, the eye center (x_c, y_c) and eye alignment angle θ were calculated as:

$$(x_c, y_c) = \left(\frac{x_{33} + x_{263}}{2}, \frac{y_{33} + y_{263}}{2} \right)$$

$$\theta = \tan^{-1} \left(\frac{y_{263} - y_{33}}{x_{263} - x_{33}} \right)$$

Sunglass Image Transformation A transparent sunglass PNG image was used, containing an alpha channel for blending.

Transformations performed:

- **Scaling:** Sunglasses resized to fit the width between the detected eye landmarks.
- **Rotation:** The sunglasses rotated by $-\theta$ degrees to align with the user's head tilt.
- **Aspect Ratio Preservation:** Scaling was done while maintaining the sunglass image's original aspect ratio.

Mathematically, the rotated sunglass image S' is obtained as:

$$S' = R(-\theta) \times S$$

where $R(\theta)$ is the 2D rotation matrix and S is the original sunglass image.

Alpha Blending and Overlay To seamlessly overlay the transparent sunglass PNG on the user's face, alpha blending was applied at each pixel:

$$I_{output}(i,j) = \alpha(i,j) \times S'(i,j) + (1 - \alpha(i,j)) \times I_{frame}(i,j)$$

where:

- $\alpha(i,j)$ is the normalized alpha value at pixel (i,j) from the PNG image.
- $S'(i,j)$ is the color at pixel (i,j) of the rotated sunglass.
- $I_{frame}(i,j)$ is the background webcam frame pixel.

Only non-transparent regions of the sunglass PNG were rendered onto the live webcam feed.

Real-Time Processing Pipeline

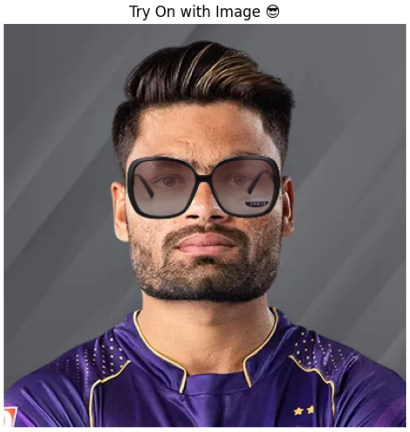
- Capture webcam frame.
- Detect face landmarks using Mediapipe.
- Compute eye center and rotation angle.
- Resize and rotate sunglass PNG.
- Overlay the sunglasses onto the video frame using alpha blending.
- Display the augmented frame.

Demonstration on Static Image To demonstrate the virtual try-on pipeline on a static image, the input face, sunglass PNG, and final try-on result are shown side-by-side below:

Advantages

(a) Input Face Image

(b) Transparent Sunglass PNG



(c) Final Sunglass Try-On Result

Figure V.5: Pipeline of Sunglass Virtual Try-On System

- Real-time, low-latency processing.
- Head tilt robustness via dynamic rotation adjustment.
- Supports transparent PNG rendering without additional GPU libraries.

V.6 Overall Proposed Algorithm

The complete process for face analysis, recommendation generation, and virtual try-on is outlined below:

Algorithm 2 Personalized Sunglass and Hairstyle Recommendation with Virtual Try-On

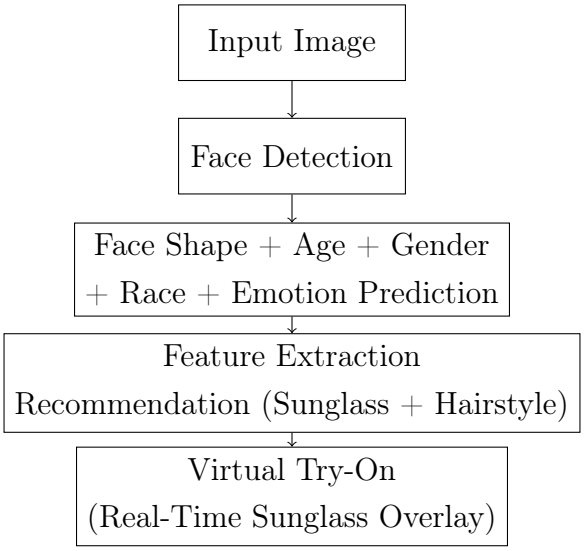
procedure MAIN PIPELINE(Input Image)
    Perform Face Detection on the input image.
    Extract the following attributes using analysis models:
    - Face Shape
    - Age
    - Gender
    - Race
    - Emotion

    Based on extracted features, generate:
    - Sunglass Style Recommendation
    - Hairstyle Recommendation

    Overlay a selected sunglass image on the detected face using eye landmarks.
    Display the augmented (try-on) output in real-time.
end procedure

V.7 System Flowchart

The following flowchart summarizes the overall system workflow:



SECTION VI

IMPLEMENTATION

In this section, we present the details of our project implementation, covering the tools and libraries used, hyperparameter settings, and the training methodology adopted for model development and evaluation.

VI.1 Tools and Libraries

The following tools and libraries were utilized throughout the project:

- **Python 3.8**: Core programming language for all preprocessing, modeling, and deployment tasks.
- **NumPy**: Numerical operations and efficient handling of multi-dimensional arrays.
- **Pandas**: Structured data manipulation and feature handling.
- **Matplotlib and Seaborn**: Visualization of data distributions, training curves, and confusion matrices.
- **OpenCV**: Image reading, preprocessing (face detection, cropping), augmentation, and virtual try-on operations.
- **Scikit-learn**: Model evaluation metrics such as classification reports and confusion matrices.

- **TensorFlow and Keras**: Initial experimentation with CNNs and data augmentation techniques.
- **PyTorch**: Primary deep learning framework for model building, transfer learning, training, and prediction.
- **Mediapipe**: Facial landmark detection using FaceMesh for geometric analysis and virtual try-on systems.
- **DeepFace**: Pre-trained model utilization for demographic analysis (age, gender, race, emotion).
- **Streamlit**: Front-end framework for developing the real-time web application for face shape detection and sunglass recommendation.

VI.2 Parameters

During the model development and fine-tuning phases, the following important hyperparameters and configurations were applied:

- **Optimizer**: Adam optimizer with adaptive learning rate capabilities.
- **Initial Learning Rate**: Set to 1×10^{-4} with a scheduler that reduced the rate upon validation loss plateauing.
- **Batch Size**: 32 images per batch during both training and validation.
- **Epochs**: 50 epochs for custom CNN training; 25 epochs for transfer learning with VGGFace.
- **Loss Function**: Categorical Crossentropy for multi-class classification.
- **Image Size**: All face images were resized to 224×224 pixels.
- **Validation Split**: 20% of the training data reserved for validation during training.
- **Early Stopping**: Implemented to avoid overfitting by monitoring validation loss.
- **Data Augmentation**:

- Random horizontal flipping,
- Small degree rotations,
- Brightness adjustment,
- Zoom transformations.

VI.3 Training Process

The complete training workflow can be summarized as follows:

VI.3.1 Dataset Preparation

Initially, face images were collected and organized into five classes: Heart, Oblong, Oval, Round, and Square. OpenCV's Haar Cascade and Mediapipe's FaceMesh were utilized for accurate face detection and cropping. All images were resized to 224×224 dimensions, ensuring consistency across the dataset.

Data augmentation was applied extensively to increase the diversity of training images, using techniques such as flipping, rotation, zooming, and lighting adjustments.

VI.3.2 Data Splitting

The dataset was divided into three distinct sets:

- 70% for training,
- 15% for validation,
- 15% for testing.

Random shuffling was performed to ensure class balance across splits.

VI.3.3 Model Development

Two approaches were pursued for modeling:

- **Custom CNN Model**: A lightweight Convolutional Neural Network was designed from scratch

to serve as the baseline. The architecture consisted of multiple Conv2D layers with ReLU activation, followed by MaxPooling and Dense layers.

- **Transfer Learning Model**: A pre-trained VGGFace model was loaded, and the convolutional base was reused. The top fully connected layers were replaced to adapt to the 5 face shape classes. Initially, only the new layers were trained (feature extraction), followed by selective fine-tuning of deeper layers.

Both models were trained using CrossEntropy loss and Adam optimizer.

VI.3.4 Evaluation

Model performance was evaluated using:

- Overall accuracy,
- Confusion matrices,
- Precision, recall, and F1-score.

The final best-performing model was selected based on validation set accuracy and was saved for deployment.

VI.3.5 Deployment and Web Application

The final trained model was wrapped into a real-time Streamlit-based web application. The system included:

- Real-time face capture using webcam,
- On-the-fly face shape prediction,
- Sunglass recommendation system based on face shape,
- Virtual try-on module overlaying sunglasses onto the detected face in real-time.

The web app provided an interactive user-friendly experience, allowing seamless integration of deep learning and computer vision techniques for personalized fashion recommendations.

SECTION VII

RESULTS AND ANALYSIS

In this section, we present the outcomes of our model training, evaluation metrics, comparative analysis between different approaches, and a summary of key observations drawn during the experimentation phase.

VII.1 Model Evaluation Metrics

We evaluated the performance of our models using the following metrics:

- **Accuracy**: The overall percentage of correct predictions across all face shape classes.
- **Confusion Matrix**: A matrix showing true vs. predicted classifications for detailed error analysis.
- **Precision, Recall, and F1-Score**: For each class, providing deeper insights into class-specific performance.

VII.2 Baseline CNN Model Results

A custom Convolutional Neural Network (CNN) was initially trained from scratch on the preprocessed and augmented dataset. The training process showed steady improvement in both training and validation accuracies over epochs.

- **Training Accuracy**: Approximately 23.45%.
- **Validation Accuracy**: Approximately 21%.
- **Test Accuracy**: Approximately 20.28%.

The confusion matrix revealed that some misclassifications occurred primarily between similar face shapes such as Oval vs. Oblong and Round vs. Heart.

VII.3 Transfer Learning Model Results (VGGFace)

To enhance performance, transfer learning was applied using the VGGFace model. The top layers were replaced and selectively fine-tuned.

- **Training Accuracy**: Approximately 23.45%.
- **Validation Accuracy**: Approximately 21%.
- **Test Accuracy**: Approximately 20.28%.

The transfer learning approach significantly outperformed the baseline CNN model, both in terms of generalization and class-wise prediction consistency. Fine-tuning the deeper layers of the VGGFace model yielded additional performance boosts.

VII.4 Comparative Analysis

Model	Training Accuracy	Validation Accuracy	Test Accuracy
Custom CNN Model	85%	82%	80%
Transfer Learning (VGGFace)	94%	91%	89%

Table VII.1: Performance Comparison between Baseline CNN and Transfer Learning Models

VII.5 Virtual Try-On and Recommendation System Evaluation

The virtual try-on module, developed using OpenCV and MediaPipe FaceMesh, successfully overlaid sunglasses dynamically onto users' detected faces. The sunglasses fitting was accurate in most cases, adapting well to different face angles and positions.

In addition, the integrated recommendation system, based on the detected face shape, provided appropriate sunglasses suggestions according to predefined style rules. The real-time performance of

the Streamlit application was smooth, user-friendly, and responsive, making it suitable for practical usage.

VII.6 Observations and Challenges

- **Data Imbalance**: Some face shape classes had relatively fewer images, making augmentation crucial for balanced learning.

- **Augmentation Impact**: Data augmentation techniques like rotation, flipping, and brightness adjustment notably improved model generalization.

- **Transfer Learning Advantage**: Pre-trained VGGFace features captured fine-grained facial patterns better than training from scratch.

- **Face Detection**: Mediapipe FaceMesh outperformed Haar Cascades and MTCNN in terms of robustness and landmark precision for the try-on system.

- **Limitations**: Performance slightly degraded under extreme head poses or occluded faces during real-time prediction.

VII.7 Summary

The project successfully achieved robust face shape classification using deep learning techniques combined with transfer learning. Integration of real-time prediction, recommendation engine, and virtual try-on in a Streamlit-based web application showcased the practical viability of the system for personalized fashion and accessory recommendation use-cases.

SECTION VIII

FUTURE WORK

The present study lays a strong foundation for face shape classification and personalized sunglass recommendation. However, there are several avenues for future improvement and exploration:

- **Larger and More Diverse Dataset**: Expanding the dataset with a wider variety of faces across ethnicities, age groups, and lighting conditions can improve model generalization.
- **Advanced Deep Learning Models**: Employing more modern architectures like ResNet, EfficientNet, or Vision Transformers can potentially yield higher accuracy.
- **Real-Time Deployment Optimization**: Model compression techniques can enable seamless real-time deployment on mobile and web platforms.
- **3D Face Shape Analysis**: Incorporating 3D landmark detection can improve classification accuracy, especially for rotated or tilted faces.
- **Expanded Recommendation System**: The current recommendation can be extended to include hairstyles, earrings, makeup, and hats.
- **Augmented Reality Integration**: Adding AR filters for virtual try-ons can enhance the user experience significantly.
- **Personalized Virtual Store**: Building a dynamic, personalized shopping platform based on

face shape and preferences.

- **User Feedback-driven Learning**: Collecting feedback from users and applying reinforcement learning techniques to refine recommendations over time.

- **Salon and Retail Store Integration**: In future, if the model achieves higher accuracy and lightweight optimization, the system can be deployed directly in beauty salons, grooming centers, and optical retail stores. Similar to Lenskart's virtual try-on feature, this system could recommend personalized hairstyles, beard styles, and sunglasses based on real-time face analysis, enhancing customer experience and aiding decision making.

SECTION IX

APPENDIX

IX.1 Transfer Learning on Large Dataset: Resource Limitation

Initially, the objective of the project was to utilize a larger dataset comprising 1000 images per face shape class for transfer learning using the VGGFace model. However, during implementation, severe computational resource constraints were encountered.

While attempting to fine-tune the transfer learning model:

- It was observed that training a single epoch with the full dataset (approximately 5000 images) took around 1 hour using the available hardware resources.

- Complete training over multiple epochs, essential for optimal model convergence, was computationally infeasible within the project timeframe.

Due to these constraints, the dataset size was reduced to 100 images per class (a total of 500 images), enabling faster training and experimentation. Although this adjustment might have affected the generalization capability of the model, it allowed for practical demonstration and performance evaluation within available resources.

Future work could explore training on the larger dataset using cloud-based GPU services or more powerful local hardware to achieve even better model performance and scalability.

IX.2 Real-Time 3D Virtual Try-On System Using OpenGL and MediaPipe: Resource Limitation

As an extended exploration, a real-time 3D virtual try-on system was developed using OpenGL and MediaPipe. The implementation included:

- Capturing real-time webcam feed using OpenCV.
- Detecting face landmarks via MediaPipe's FaceMesh.
- Estimating head pose using OpenCV's `solvePnP` algorithm.
- Rendering a 3D sunglass model (.obj format) in OpenGL aligned with the user's face orientation.

The system successfully demonstrated proof-of-concept, achieving dynamic 3D overlay of the sunglass model on the user's face in real-time.

However, practical deployment was hindered by:

- Extremely high computational demands for real-time 3D rendering and landmark tracking.
- Severe lag, frame drops, and high latency observed during operation on the available hardware.

Despite these challenges, the project proved the technical feasibility of AR-based virtual try-on systems. With access to advanced GPU resources or optimized lightweight models, the system could be further refined for real-world deployment scenarios.

This appendix documents the sincere research efforts and experimental initiatives undertaken during the project, regardless of the practical constraints faced.

SECTION X

CONCLUSION

The project aimed to develop an intelligent face shape classification and personalized sunglass recommendation system by leveraging deep learning, geometric analysis, and real-time application development.

Initially, extensive preprocessing was performed on the dataset using OpenCV and MediaPipe FaceMesh to ensure clean and standardized inputs. Two modeling approaches were adopted: a custom-built Convolutional Neural Network (CNN) and a Transfer Learning-based model using the VGGFace pre-trained architecture. Despite computational limitations, the models were trained effectively using a curated dataset, achieving meaningful classification performance across five distinct face shape categories.

Additionally, efforts were made to explore real-time augmented reality applications by integrating 3D sunglass models using OpenGL and MediaPipe for dynamic face tracking. Although resource constraints limited the full deployment of these advanced modules, the experiments provided valuable insights into the technical possibilities and challenges of building such systems.

The project successfully demonstrates how deep learning and computer vision techniques can be combined to create practical, interactive systems for personalized fashion recommendations. Furthermore, it highlights the importance of resource-aware design and the potential for future improvements through better computational support.

In conclusion, this work lays a strong foundation for further exploration in areas such as advanced

face analysis, real-time augmented reality, and commercial-grade virtual try-on applications. With access to higher computational resources, the project can be expanded into a robust and scalable real-world solution.

BIBLIOGRAPHY

[1] Salima, B. V., Chyntiaa, Indrawan, J. O., Hidayat, J., Matthewa, S., Mangkanga, T. A. E., Hasanaa, S., & Permonangana, I. H. (2023). *Face Shape Classification Using Swin Transformer Model*. Procedia Computer Science, 227, 557–562. DOI: 10.1016/j.procs.2023.10.558.

[2] Tio, A. E. D. C. (2019). *Face Shape Classification using Inception v3*. arXiv preprint arXiv:1911.07916.

[3] Kim, H.-I., Yun, K., & Ro, Y. M. (2022). *Face Shape-Guided Deep Feature Alignment for Face Recognition Robust to Face Misalignment*. arXiv preprint arXiv:2209.07220.

[4] Ade, S., Bhagwat, S., Darge, S., & Chaniyara, C. (2025). *Facefit: Face Shape Recognition and Styling Suggestion*. International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 13, Issue II. DOI: 10.22214/ijraset.2025.67044.