International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 12 Issue: 04 | Apr 2025 www.irjet.net p-ISSN: 2395-0072

AI-Powered Face Shape Detection for Personalized Hairstyle Recommendations Using Deep Learning Models.

Nikita Shinde[1], Sanjeevani Tagade[2], Saloni Varma[3]

[123]Bachelor of Technology in Department of Data Science, Usha Mittal Institute of Technology, Mumbai, India

---

Abstract - This research introduces a novel method to face shape detection in order to better recommend hairstyles. Precise recognition of the shape of the face– whether oval, round, heart, square or oblong is essential in recommending hair styles that match as nearly as possible an individual's personal characteristics. Our method combines advanced image processing and machine learning algorithms with AI– integrating deep learning models that have been pre-trained, such as a CNN-based architecture like VGG16 or MobileNet. It identified facial landmarks using deep learning models trained on a comprehensive data set of face images. In addition, to combat variations in facial expressions and lighting conditions, we have extended the system with artificial intelligence techniques that help it better handle these different circumstances. Empirical results demonstrate that when our face shape detection model is integrated with a hair do counseling system, the system offers hair suggestions tailored according to face shape as determined by the headband. Methodology efficiency and accuracy demonstrated by experiment will directly indicate the potential application prospects for personal grooming mode plus Virtual Styling Tools.

Key Words*: *Face shape detection, hairstyle recommendation, deep learning, convolutional neural networks (CNN), VGG16, MobileNet, image processing, facial landmarks, machine learning, virtual styling, personal grooming.

1.INTRODUCTION

Choosing an appropriate hairstyle in the current era of personal care and style can be daunting, given the multitude of choices available. Many people find it challenging to identify a look that suits their particular facial structure, often resorting to experimentation or seeking expert guidance, which can be both time-consuming and potentially unsuccessful. This research investigates the creation of an artificial intelligence-driven system that identifies face shapes and suggests tailored hairstyles to address this issue. Conventional approaches to hairstyle selection often lack accuracy, as they rely on subjective judgments or generic recommendations. Although some digital applications claim to offer hairstyle advice, many fail to precisely evaluate face shapes or provide truly personalized suggestions. This underscores the necessity for a smart, user-friendly solution that employs cutting-edge technology to accurately analyze facial characteristics. Our study examines the potential of utilizing deep learning models, such as CNN-based architectures like VGG16 and MobileNet, to enhance face shape classification. By training the system on a varied collection of facial images, we aim to develop a model capable of consistently recognizing different face shapes. Furthermore, we concentrate on creating an easy-to-use interface that enables users to effortlessly upload their photos and receive hairstyle recommendations tailored to their facial features. Ultimately, this research aims to connect technology with personal styling, simplifying the process of hairstyle selection.

2. LITERATURE SURVEY

In this it involves segmenting the head and identifying the face plane to minimize shadows. It uses an algorithm that treats 3D body points as evaluation features, employing Eigenvectors for refinement and techniques like Ellipsoid Fitting and Mahalanobis distance for segmentation. The HaarCascade Classifier and dlib functions are used for real-data detection, improving classification accuracy[1]. Face recognition has advanced over 30 years, widely used in commercial and law enforcement sectors. Despite progress, challenges like lighting and angle variations remain. Face shape detection benefits industries like fashion and e-commerce. CNNs are highly effective for face classification, automatically identifying features with minimal training. Key challenges include adapting to varied conditions, improving real-time processing, and enhancing personalization. Future improvements focus on robustness, real-time capabilities, and integrating multimodal data for more accurate recognition[2]. Recent advances in face shape classification, mainly using CNNs, have improved accuracy in fashion and healthcare. Challenges like image quality and expressions remain, requiring stronger algorithms and real-time improvements[3]. Advancements in hairstyle recommendation use CNNs to analyze face shapes and suggest suitable styles. Real-time simulations improve user experience, but challenges like lighting, hair diversity, and processing speed remain, requiring better datasets and personalization [4]. Hybrid models combining CNNs with other techniques improve hairstyle recommendations. Challenges like lighting, expressions, and hair types persist. Augmented reality and real-time simulations enhance user experience, driving personalized, data-driven advancements [5]. The CelebA dataset, known for its diverse celebrity facial images with annotations, has been adapted for hairstyle recommendation. Researchers use it to train CNN models,

improving accuracy in matching hairstyles to facial features. Techniques like transfer learning and data augmentation help address challenges like lighting, expressions, and hair variations. This shift towards large-scale, high-quality data enhances personalized hairstyle suggestions [6].

3. METHODOLOGY

3.1 Overview of System/Project

• **Frontend:**
Conceptual Overview
The front page (homepage) in this project serves several purposes:
Introduction: It introduces the app's purpose and functionality. Image Upload: It provides a way for users to upload their profile picture. Face Detection (Initiation): It initiates the face detection and hairstyle recommendation process.

1.Setting up the Basic React App
Create React App: Create a new React application using the following command in the root directory of your project: npx create-react-app frontend Navigate: Move into the frontend directory cd frontend Install Dependencies: Install the following dependencies which you will need to build the frontend for your application: npm install axios react-router-dom axios is used for making http requests to your backend. react-router-dom for creating routes and navigation in the frontend.

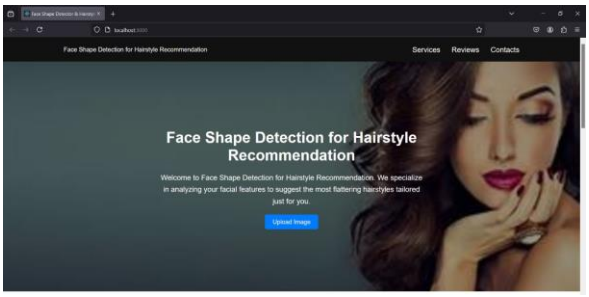

Fig 3.1 Upload form

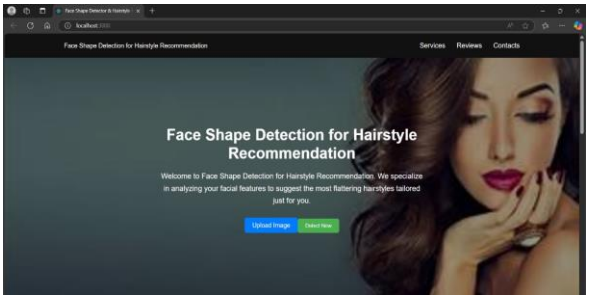

Fig 3.2 Detect now button

Building the Header
Create Component: Create a new file named Header.js in the src/components folder.

Building the Hero Section
Create Component: Create a file named HeroSection.js inside the src/components folder.
Import Components:
Import the upload form to upload image.
Import Result Display component to display image with landmarks and face shape information.
Import use Navigate hook for routing.
Java script import React, { useState } from 'react';
import Upload Form from './Upload Form';
import Result Display from './Result Display';
import{useNavigate}from'react-router-dom';
This component implements a carousel that contains introductory text, image upload and detection buttons, and the results if the image is uploaded.
It uses a use Navigate hook to perform routing and redirect to the next page.

Building the Upload Form
Create Component: Create a new file named UploadForm.js in the src/components folder.
This component provides an image upload element and a "Detect Now" button.
It makes an API call to the Flask server using the /detect_face_shape endpoint.
It displays the errors if the upload fails.

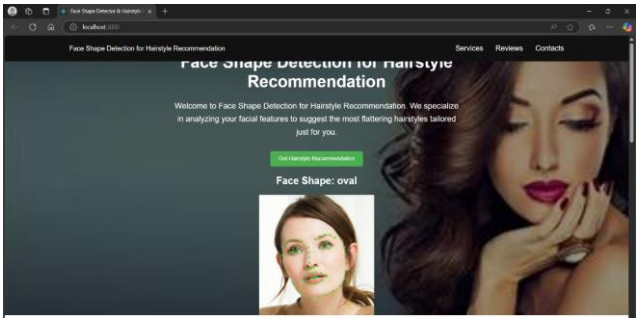

Fig 3.3 Face shape name detected

Face shape name detected Section
By examining the user's face, the system identifies its shape--in this particular instance it is "oval".

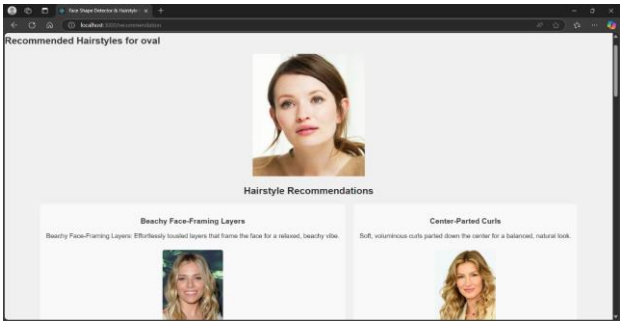

Fig 3.4 Get hairstyle recommendation

Get hairstyle recommendation Section
It will gives you hairstyle ideas that suit your face shape. After detecting your face shape

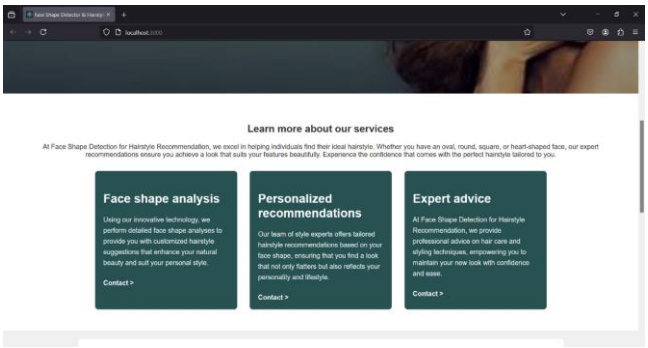

Fig 3.5 About Services

ServicesSection.js (Services Section)
Purpose: Implements the services section.
Explanation: Displays the services that this application offers in a clear manner using cards. The cards have headings, descriptions and the call to actions.

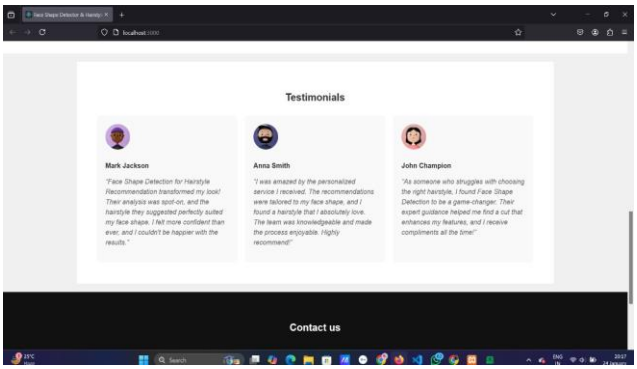

Fig 3.6 Testimonials

Testimonials.js (Testimonials Section)
Purpose: Displays the testimonials from different users.
Explanation: Displays a list of testimonials with the person's name, avatar, and text.

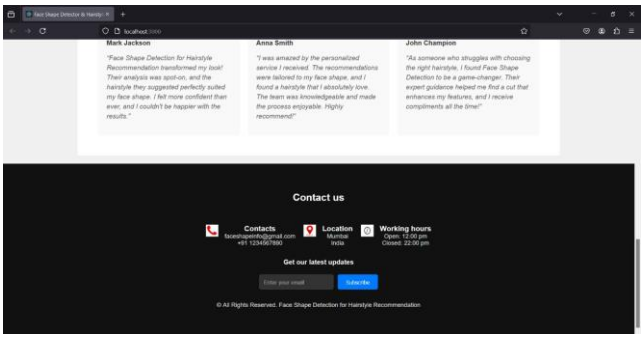

Fig 3.7 Contact us

ContactSection.js (Contact Section)
Purpose: Displays the contact information and the newsletter signup section.
Explanation: Displays contact information and a simple newsletter subscription form.
Includes the copyright information for the website.

2. Building the Result Display
Create Component: Create a new file named ResultDisplay.js in the src/components folder. This component receives the result from Hero Section and renders the face shape and the image with landmarks.

3. Implementing the Main App Component
Import Components: Import the Header and Hero Section components along with all other relevant components in App.js.
This sets up the routing and renders all the components. Header renders at the top, below the header, a router renders either Hero Section, Recommendation Page or Try On Page based on the path. The other components like Services Section, Testimonials and and Contact Section are rendered below.

4. Adding Styles
Create File: Create a styles.css file in the src folder.

5. Rendering the Main App
Import App: Import the main app component in index.js

- **Backend:**

The main duties of the backend are:
Handle image uploads from the frontend: handles and uploads images to the server from the front end.
Use the Dlib library to identify faces in the image and extract facial landmarks. Face Detection and Landmark Prediction: Face Shape Classification: Use the pre-trained TensorFlow / Keras CNN model to classify the user's face shape.
With the face shape determined, give back some hairstyles that are recommended by matching it to user preferences. Q: Recommendation for Hairstyles Written as a function underneath the classes that calls itself depending on state changes
API Endpoints: Expose API endpoints for these functions so that the front end can interact with them.

- **Backend Code Files:**

app.py: Master Flask application that sets the API end points and handles loading models and requests.
utils.py: Containing helper functions to load the Dlib face detector, perform landmark detection and generate hairstyle recommendations.
face_shape_classifier.py: This file defines the Face Shape Classifier class for loading the trained CNN model and making face shape predictions.
requirements.txt: The required Python packages for the backend are ones specified here to run it right. Make Your Back-end Collection Step-by-Step

• **Python Environment Configuration:**
Create a virtual environment for the backend. Sample Python Environment Variable File
pip install Usual PKG name
Model Loading and API Start (app.py):
In the Flask app, the trained component of face shape classification model face_shape_classifier.h5 is started up.
Again, Dlib face detector and facial landmark predictor must be loaded.
All models are loaded during initialization by calling load_dlib_model function. The loading is done in app.py and related files to avoid cluttering up utils.py with app-specific code that is not reusable.
Specifically, error handling techniques should be used for model loading. Of course if you do not know these techniques then feel free to ask questions here about them./ detect_face_shape API Endpoints (app.py):
Image upload: image data uploaded to /detect_face_shape, a POST request route in the Flask app, should contain an image file.

• **Image Processing:**
A read is made from the subsequent drop down window requesting an image.
The image is put into OpenCV format.
Dlib face detector and landmark predictor are used to find the face and extract landmark points from it (detect_face_and_landmarks function from utils.py).

• **Face Shape Prediction:**
To predict the face shape, the extracted face is being passed to Face Shape Classifier.
Face Shape Classifier pre-processes the image, and then uses its trained CNN model for face shape prediction.
Response Generation: Then this API returns to front end JSON response, and such data detects the face shape, landmarks and face co-ordinates, at the same time also the base-64 image.
/ recommend_hairstyle API Endpoint (app.py) :
Face Shape Reception: This API route receives a JSON POST request containing the detected face shape from front end.
recommend the get hairstyle: The face shape api recommends hairstyles for the given face shape.
Response Generation: The API sends a JSON response (back) containing a list of recommended hair-styles to the front end.
utils.py - Helper Functions
load_dlib_model: Loads the Dlib face detector and landmark predictor.
detect_face_and_landmarks: this function takes an image the face detector, and landmark predictor as an input, returns the image with landmarks and bounding box coordinates.
get_hairstyle_recommendations: this function takes in a face shape and returns a list of hairstyles from our data dictionary. Also handled in here, are any exception caused by loading hairstyle or recommendation data.
face_shape_classifier.py - FaceShapeClassifier Class
This class is responsible for loading the model of facial classification and making predictions.
init: Loads the trained CNN model and prepares the label encoder.
preprocess_image: The method processes an input image to conform with expected input of the CNN model. Generally this just means resizing, normalization and perhaps some other helpfull transformation as well.
predict: The face shape that's being pre-processed is (already) predicted by the original image.

• **Running the Backend:**
Open a terminal and navigate to the back-end directory then run the command python app.py.
The Flask development server is now running.

3.2 Algorithm

We used different machine learning techniques to boost our current model
Random Forest is a fine machine-learning technique that compiles the judgment of many trees overturned for the sake of obtaining more accurate results, from first dataset we got random forest 47% accuracy. From second dataset we got 46%. SVM (Support Vector Machine) finds good ways to erect barriers between different categories where none existed before. from first dataset we got svm 36% accuracy. From Second dataset we got 47%. Logistic Regression is used to solve unknowns in term of how much one class best fits them, from first dataset logistic regression accuracy is 49%. with Second dataset we got accuracy 52%. Unlike the others, XGBoost works in conjunction with past mistakes to make sure models better represent reality. from first dataset xgboost accuracy is 39%. From Second dataset we got accuracy 42%. Vgg16 In addition, VGG16 Is A deep learning model for extracting image features that can help analyze the actual operating state of a system as accurately as possible. from first dataset accuracy of vgg16 is 47%, from Second dataset accuracy we got 69%. MobileNet MobileNet is a lightweight Convolutional Neural Network (CNN) that is good for efficient image classification, object detection and recognition. from first dataset MobileNet accuracy is 46%, from Second dataset accuracy we got 47%. Using Dlib to detect facial landmarks, harmonize faces and identify facial features such as distance from jaw width or face length. By pre-processing followed by K-means to get rid of outliers, the data is balanced using SMOTE. Also through Optuna to find an optimized set of tuning hyperparameters (though XGBoost seemed just right). A great many machine learning algorithms are explored. For example Random Forests are trained; Support vector machines (SVM) are trained in addition to Logistic Regression (LR) and XGBoost which is using deep learning architecture VGG16 Shown 47% Accuracy, and MobileNet with augmented data 46% test accuracy. The results are validated by confusion matrices and classification reports with standard accuracies.

4. HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements: Processor: Intel i5 (min) or i7
RAM: 8GB (min) or 16GB,
Storage: 10GB free space
Software Requirements: Python - Libraries: OpenCV, dlib, scikit-learn, numpy dlib installation, IDE: Colab.

5. RESULTS

We tried other machine learning models on both of our data sets to enhance our existing model. On the first set Random Forest achieves 47%, in the second 46%. SVM reached 36% and 47% while Logistic Regression did better with 49% and 52%. XG Boost scored 39% and 42%. VGG16 showed a big jump from 47% to 69% while MobileNet got to 46% in the end. To validate our findings, we used confusion matrix and classification report. These reports ensure that the results are both accurate and reliable. The accuracy of various models from the viewpoint which model will have better performance later on. By contrast, VGG16 model achieved highest accuracy among all the tested models we calculated and considered. It's mostly better than the other models such as VGG, which is the highest among all models now. This result means that in our study, VGG16 was the most effective model. With its higher accuracy, VGG 16 will be able to make better predictions than any of the other models we tested.

6. CONCLUSION

In this project, we successfully implemented a facial landmark detection and label classifier with machine learning techniques which included steps such as: detection of the facial features, facial alignment operation, extraction of all necessary measurements for creating models to classify face shapes. Data preprocessing steps resizing and eliminating outliers improved dataset quality. A variety of models, among which we may note MobileNet and VGG16 pushed the limits, were tested. The most successful model reached 69% accuracy. In order to improve accuracy, we have concentrate on narrowing the focus of feature extraction techniques, and employing models. Finite hyperparameter optimization and delving into ensemble learning should also augment performance. This will be particularly helpful when installed as a real-time face shape classification flow incoming beauty, and virtual try-on solutions can also be created with it.

7. REFERENCE

[1] Kamble, Y. M., & Kulkarni, R. B. (2024). Recommendation System for Hairstyle Based on Face Recognition Using AI and Machine Learning. International Journal of Software Innovation (IJSI), 12(1), 1-10.

[2] Rohan S, Suhas R Vittal, Neeraj H Gowda, Dr. Priya R Sankapal.(2022). Face Shape Classifier Using Deep Learning. International Research Journal of Engineering and Technology (IRJET).

[3] Mehta, A., & Mahmoud, T. (2022). Human Face Shape Classification with Machine Learning. Mahmoud, Human Face Shape Classification with Machine Learning.

[4] Rajapaksha, S. V., & Kumara, B. T. G. S. (2018). Hairstyle Recommendation Based on Face Shape Using Image Processing.

[5] Chen, Y., Zhang, Y., Huang, Z., Luo, Z., & Chen, J. (2021, August). CelebHair: A new large-scale dataset for hairstyle recommendation based on CelebA. In International Conference on Knowledge Science, Engineering and Management (pp. 323-336). Cham: Springer International Publishing.

[6] Wu, Y., & Ji, Q. (2015). Discriminative deep face shape model for facial point detection. International Journal of Computer Vision, 113.