Project Title: DermalScan: AI_Facial Skin Aging Detection App

Infosys SpringBoard Virtual Internship Program

Subimtted by:

Ashritha Ambati to Mr.Praveen sir

Problem Statement

Skin-related issues such as dark spots, wrinkles, and puffy eyes are common concerns affecting people of different ages. Accurate and early identification of these conditions can support better skincare decisions and help detect underlying health problems.

Manual diagnosis is often subjective, time-consuming, and requires expert knowledge, making automated computer-vision solutions highly beneficial.

To build an intelligent skin-analysis model, the first and most important requirement is to ensure that the image dataset is well-organized and properly represented. Without a correctly structured dataset, the performance of machine learning algorithms becomes unreliable. Additionally, skin-image datasets are often limited or imbalanced, which can negatively affect model accuracy — therefore, data augmentation is necessary to increase diversity and prevent bias.

Outcomes

- Collected and organized skin images into four categories
- Performed image validation and dataset statistics
- Applied data augmentation to improve dataset size and balance
- Preprocessed images for model training
- Built and evaluated a CNN model for skin-condition classification
- System can classify wrinkles, dark spots, puffy eyes, and clear skin automatically

Introduction

Skin-related conditions such as wrinkles, puffy eyes, and dark spots are common cosmetic and dermatological concerns. Early detection helps individuals make informed skincare decisions and can reflect underlying health conditions. Manual examination by dermatologists can be time-consuming, subjective, and inaccessible to many people. Artificial Intelligence, particularly Deep Learning, enables automated and accurate skin-condition assessment from facial images. This project focuses on identifying specific signs of facial skin aging using image classification techniques.

Milestone 1: Dataset Preparation and Preprocessing

Module 1: Dataset Setup and Image Labeling

Collected and organized images across categories: clear skin, acne, dark spots, puffy eyes, wrinkles. Loaded images using TensorFlow/Keras and converted them into array format. Ensured consistent dimensions and validated dataset quality through visualization. Prepared a clean, structured dataset ready for preprocessing and model development. Plotted data distribution for each category using a bar graph to verify class balance

Code:

```
import os import matplotlib.pyplot as plt folder_path =
r"C:\Users\ambat\Downloads\DATASET-20251203T133241Z-1-
001\DATASET" print(os.listdir(folder_path)) image_exts =
(".jpg", ".jpeg", ".png", ".bmp",
".gif") classes = ["clear skin","Dark spots","puffy
eyes","wrinkles"] counts = [] #image count for c in classes:
folder = os.path.join(folder_path, c)     count = sum(
        1 for f in os.listdir(folder)
if os.path.isfile(os.path.join(folder,
f))        and
f.lower().endswith(image_exts)     )
counts.append(count)     print(f"{c}:
{count} images")
plt.figure(figsize=(8,5)) colors =
['blue',
'green', 'yellow', 'violet'] # bar chart plt.bar(classes,
counts, color=colors) plt.title("Number of Images in Each
Class") plt.xlabel("Classes") plt.ylabel("Number of Images")
for i, count in enumerate(counts):     plt.text(i, count + 1,
str(count), ha='center', fontsize=12)  plt.show()
```

Output:

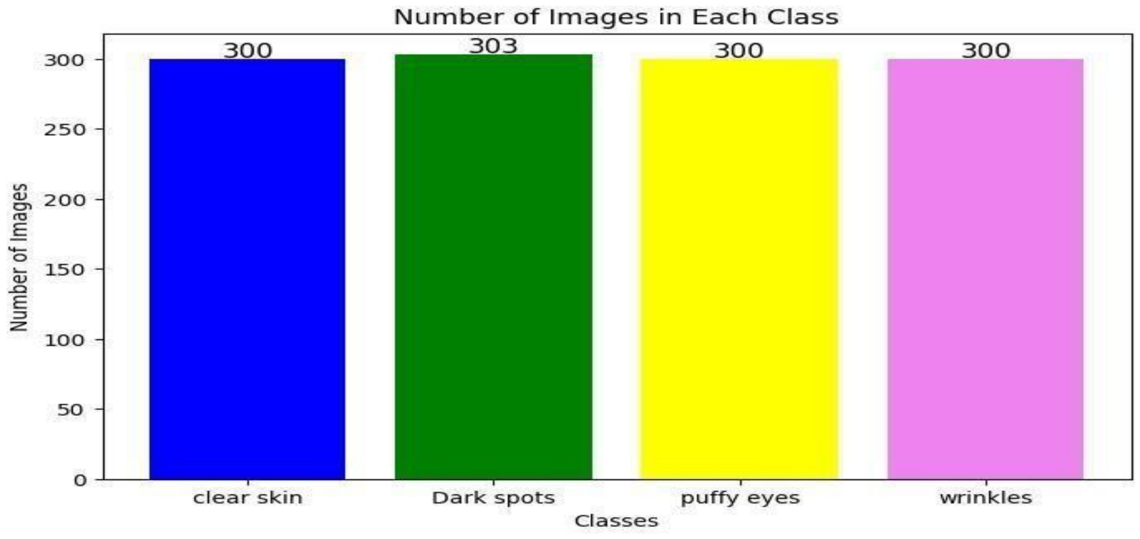


FIG1: Distributed Classes in Bar Graph

Module 2: Image Prepocessing and Augumentation

Implemented an augmentation pipeline using Keras ImageDataGenerator. Applied key transformations: rotation, zoom, shifts, shear, brightness adjustment, and horizontal flip. Generated multiple augmented samples for each image to increase dataset diversity. Visualized original vs. augmented images to validate augmentation behavior.. Produced a more robust and varied dataset suitable for deep-learning model training.

Code:

```
import matplotlib.pyplot
as plt
from tensorflow.keras.preprocessing.image import
ImageDataGenerator, load_img,  img_to_array import numpy as np


image_list = [
   r"C:\Users\ambat\Downloads\DATASET-20251203T133241Z-1-
001\DATASET\clear skin\clear_skin_061.jpg",
   r"C:\Users\ambat\Downloads\DATASET-20251203T133241Z-1-
001\DATASET\dark spots\2de295c0-801d-429e-8f74-
fce181cc87cc.jpg",
   r"C:\Users\ambat\Downloads\DATASET-20251203T133241Z-1-
001\DATASET\puffy eyes\14.jpg",
   r"C:\Users\ambat\Downloads\DATASET-
20251203T133241Z1001\DATASET\wrinkles\36.jpg"
]

# Augmentation Configuration

aug = ImageDataGenerator(
rotation_range=30,
zoom_range=0.25,
width_shift_range=0.1,
height_shift_range=0.1,
brightness_range=[0.7, 1.3],
horizontal_flip=True,
vertical_flip=True
)

#  Displaying

rows = 5
cols = 5
plt.figure(figsize=(12,
6))   for row,
img_list in
enumerate(image_li
st):

    # Load image

original=load_img(img_list,target_size=(224, 224))
arr=img_to_array(original)
arr=np.expand_dims(arr, axis=0)

 # Show Original Image

plt.subplot(rows, cols, row * cols + 1)
plt.imshow(original)
plt.title(f"Original {row+1}")
plt.axis("off")
```

```
    # Show Augmented images
for      j      in      range(4):augmented=next(aug.flow(arr,
batch_size=1))[0].astype("uint8")
 plt.subplot(rows, cols, row * cols + (j + 2))
plt.imshow(augmented)
 plt.title(f"Aug {j+1}")
 plt.axis("off")
 plt.tight_layout()
 plt.show()
```

Output:

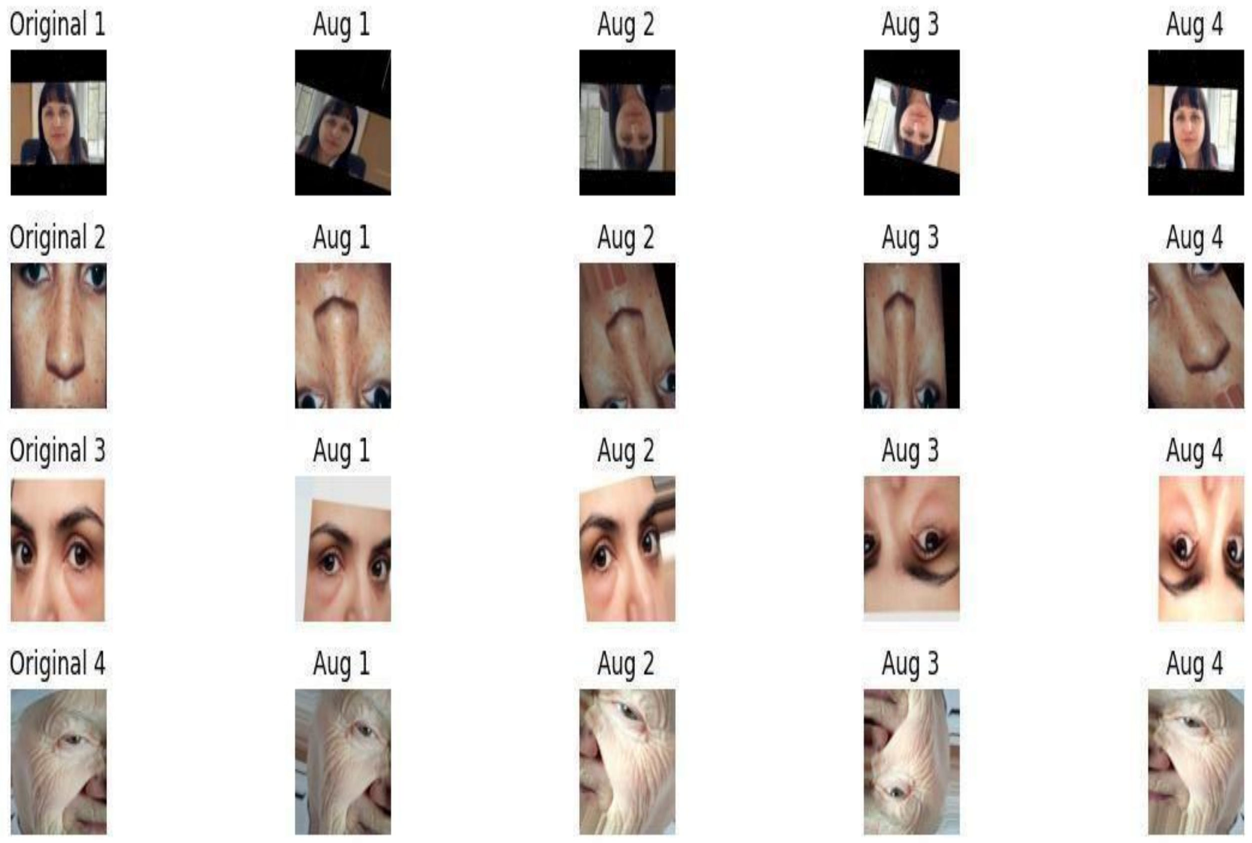


Fig2: visualization of Augmented Images

Milestone 2: Model Training and Evaluation

Overview

• Milestone 2 focuses on training a deep learning model for skin condition classification and integrating it with a face detection and prediction pipeline. This milestone ensures that the trained model achieves high accuracy and can be applied to real-world face images for prediction along with age estimation.

Module 3: Model Training with EfficientNetB0

Objective

• To train a robust Convolutional Neural Network (CNN) using transfer learning for accurate skin condition classification.

Tasks Performed

• Used pretrained EfficientNetB0, ResNet50, MobileNetV2 as the base model for transfer learning

• Applied categorical cross-entropy as the loss function. • Used the Adam optimizer for efficient weight updates.

• Trained the model on labeled skin condition images.

• Performed validation during training to monitor performance.

• Plotted training and validation accuracy and loss curves.

Deliverables:

• Trained EfficientNetB0 , ResNet50 and MobileNetV2 models saved as an .h5 file.

• Accuracy and loss graphs for training and validation phases.

Evaluation Results:

• Achieved classification accuracy greater than 90%.

• Validation accuracy remained stable, indicating good generalization and minimal overfitting.

FINAL MODEL PERFORMANCE COMPARISON

	Model	Training Accuracy (%)	Validation Accuracy (%)
1	MobileNetV2	90.342677	87.866110
2	ResNet50	95.430946	89.121342

3	EfficientNetB0	87.123573	90.794981

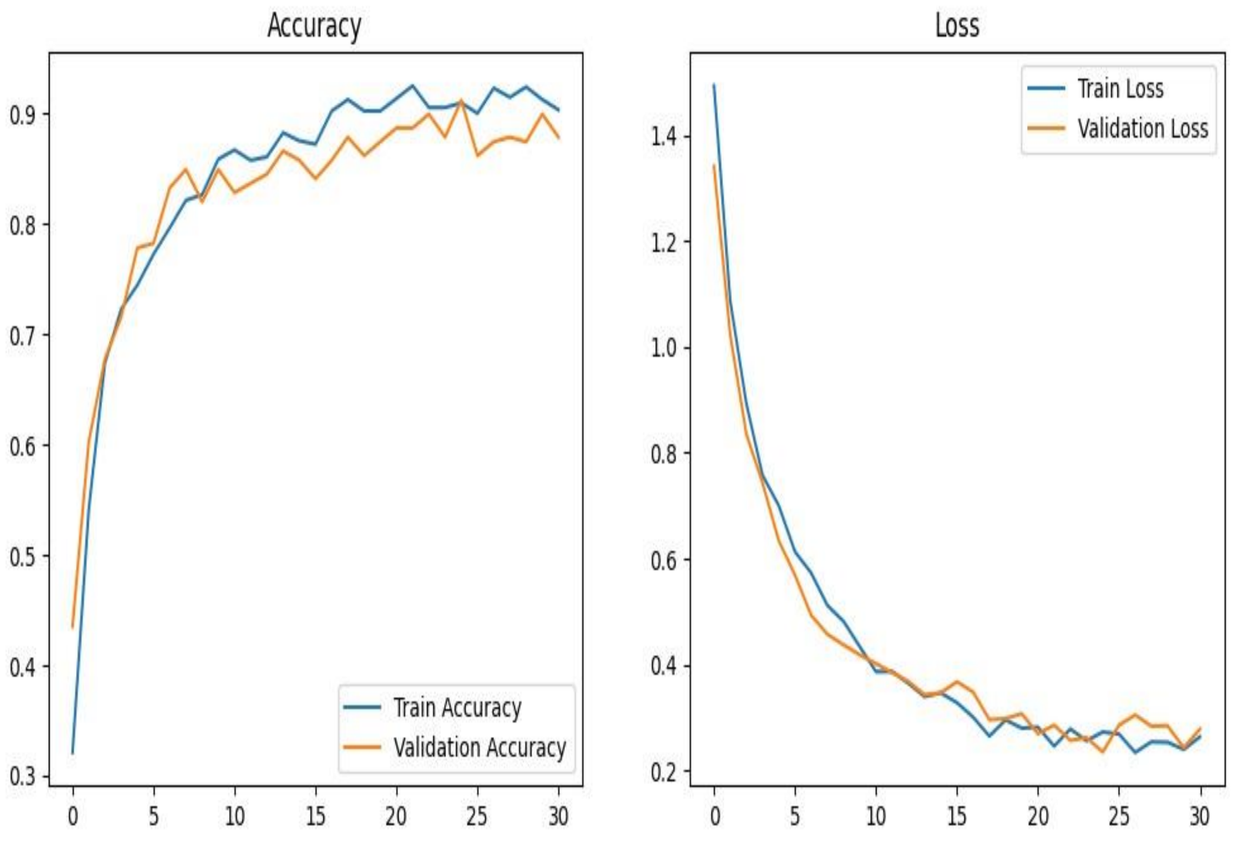


Fig3:Training and Validation Accuracy and Loss Graphs of MobileNetv2

Module 4: Face Detection and Prediction Pipeline

Objective

The primary objective of this module is to detect faces from input images and apply a trained skin classification model to predict skin conditions and age. This pipeline integrates face detection, preprocessing, and prediction into a unified workflow.

Tasks Performed

- **Face Detection:** Implemented using OpenCV and Haar Cascade classifier for robust detection of facial regions.
- **Preprocessing:** Detected faces were cropped and resized to match the input requirements of the prediction model.
- **Model Application:** Applied the trained EfficientNetB0 model to cropped face regions for skin condition classification and age prediction.

- **Visualization:** Displayed results by overlaying bounding boxes and prediction outputs on the original images.

Displayed Outputs

- **Predicted Skin Condition:** Classification of skin condition based on trained model.
- **Confidence Percentage:** Confidence scores associated with each prediction.
- **Predicted Age:** Age estimation integrated into the pipeline.
- **Bounding Boxes:** Visual markers drawn around detected faces with overlaid prediction results.

Deliverables

- Bounding boxes around detected faces.
- Skin condition labels with confidence percentages.
- Age prediction results integrated into the detection pipeline.

Evaluation

- **Face Detection Accuracy:** Achieved satisfactory results with correct localization of facial regions.
- **Skin Condition Prediction:** Predictions were displayed clearly with confidence scores, ensuring interpretability.
- **Age Prediction Integration:** Age estimation was successfully incorporated into the pipeline, complementing skin condition classification.

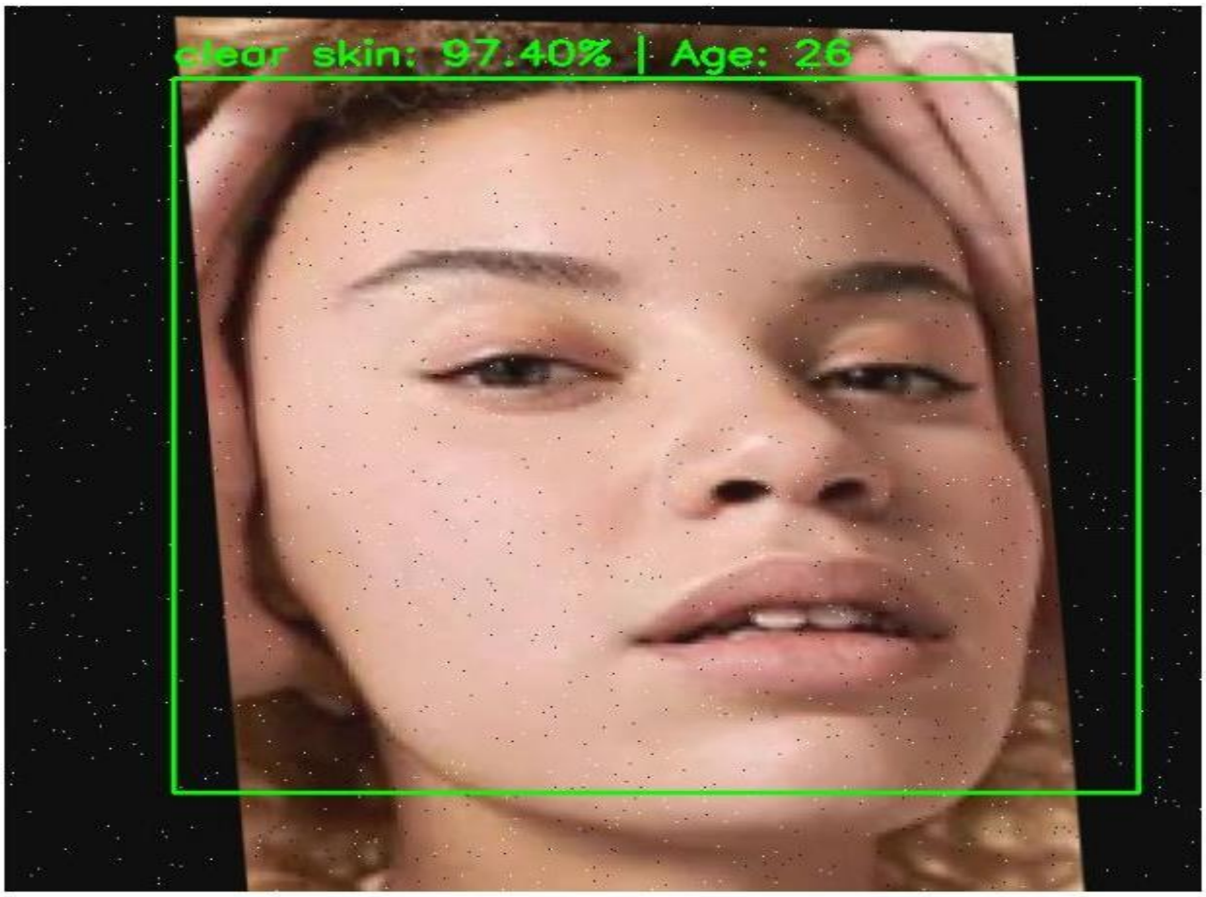


Fig4:Face Detection and Age Prediction

Milestone 3: Frontend and Backend Integration

Module 5: Web UI for Image Upload and Visualization

• Objective

The objective of Module 5 is to design and implement a responsive web-based user interface that allows users to upload facial images and visually analyze skin aging conditions. The interface acts as a bridge between the user and the backend inference pipeline, enabling seamless image upload, result visualization, and data export.

• Overview

This module is implemented using **Streamlit**, a lightweight Python framework for building data-driven web applications. The UI enables users to:

- Upload facial images (JPG/PNG formats)
- View original and annotated images side by side
- Display detected facial bounding boxes with predicted skin condition labels

- Show confidence scores and estimated age
- Maintain a prediction history table
- Download annotated images and prediction logs in CSV format

The frontend communicates with the backend inference module (run_inference) to process images and display results in real time.

• User Interface Components

1. Page Configuration

The application uses a wide layout for better visualization of images and results:

st.set_page_config(page_title="DermalScan", layout="wide")

2. Header and Styling

A centered header is created using custom CSS to enhance visual appeal and maintain a professional layout.

- Title: **DermalScan – AI Skin Analysis**
- Subtitle: *Upload a face image to analyze skin condition*

3. Image Upload Section

Users can upload facial images using Streamlit's file uploader component:

- Supported formats: JPG, JPEG, PNG
- Prevents unnecessary reprocessing using file hashing

```
uploaded_file = st.file_uploader("Upload face image", type=["jpg",
"jpeg", "png"])
```

4. Backend Inference Trigger

Once a new image is uploaded:

- The image is temporarily saved Passed to the backend inference

  function
- Results are returned as:

  o Annotated image o Prediction table containing bounding box coordinates,

  condition, confidence, estimated age, and face detector confidence

This ensures **only new images are processed**, improving performance and avoiding UI lag.

5. Image Visualization The UI displays:

- **Original Image** (left column)
- **Annotated Image** with bounding boxes and labels (right column)

This side-by-side layout allows users to clearly compare results.

6. Download Options

Users can:

- Download the annotated image as PNG
- Export prediction history as a CSV file

This supports result documentation and further analysis.

7. Prediction History Table

All predictions are stored using Streamlit session state and displayed in a dynamic table containing:

- Filename
- Bounding box coordinates (X1, Y1, X2, Y2)
- Detected condition
- Confidence percentage
- Estimated age
- Face detector confidence

This table updates automatically after each successful inference.

Module 6: Backend Pipeline for Model Inference

• Objective

To design a modular and efficient backend pipeline that processes uploaded facial images, performs face detection, skin condition classification, and age estimation, and returns structured results to the frontend. The pipeline must be reliable, scalable, and optimized for real-time inference in the DermalScan system.

• Overview

This module implements the core inference logic of DermalScan. A pretrained MobileNetV2 deep learning model is loaded to classify facial skin conditions. The backend receives an image from the frontend, detects the face region using MediaPipe Face Detection, preprocesses the cropped face, and

generates predictions with confidence scores and estimated age. The output is returned as both an annotated image and a structured results table.
Additional considerations include:

- Error handling for invalid uploads or missing faces.
- Performance optimization using GPU acceleration and caching.
- Scalability for multiple concurrent requests.

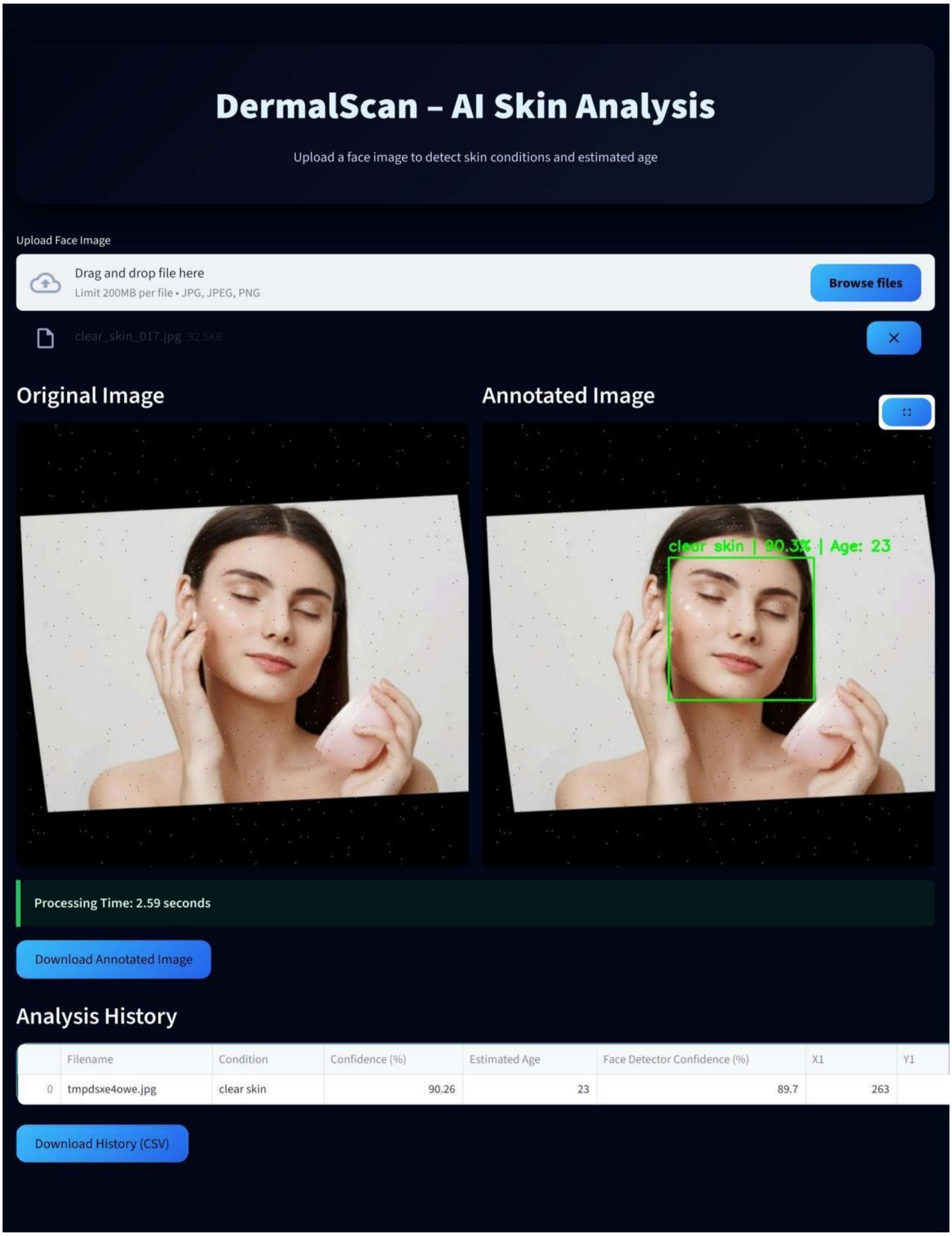
• Backend Workflow

1. Load trained CNN model (.h5 file)
    - o Initialize MobileNetV2 model with pretrained weights.
    - o Validate model integrity and warm-up inference to reduce latency.
2. Receive and validate uploaded image
    - o Accept image files (JPEG, PNG).
    - o Check file type, size, and handle corrupted uploads gracefully.
3. Detect face region using MediaPipe
    - o Perform face detection and extract bounding box coordinates.
    - o Return detection confidence score.
    - o Handle cases of no face or multiple faces.
4. Crop and preprocess the detected face
    - o Crop image to bounding box.
    - o Resize to model input size (e.g., 224×224).
    - o Normalize pixel values (0–1).
    - o Apply optional preprocessing (rotation, brightness correction).
5. Perform model inference to classify skin condition
    - o Pass preprocessed face to MobileNetV2.
    - o Generate classification output.
    - o Return top-N predictions with confidence scores.
6. Estimate age based on predicted condition
    - o Use regression head or rule-based mapping.
    - o Return estimated age as integer or range.
7. Draw bounding box and prediction label on the image
    - o Annotate image with bounding box, condition label, confidence, and age.
    - o Ensure annotations are clear and non-obstructive.
8. Return annotated image and prediction data to the frontend
    - o Output structured JSON response:
      { "condition": "Acne", "confidence": 0.92, "age": 25, "bounding_box": [x1, y1, x2, y2] }

    - o Return annotated image (base64 or file path).
9. Error handling and logging
    - o No face detected → structured error response.
    - o Multiple faces detected → process first or return warning.
    - o Model inference failure → log error and return fallback response.
    - o Maintain logs for debugging and performance monitoring.
10. **Performance optimization**

- Cache model in memory.
- Use GPU acceleration if available.
- Compress annotated image before sending.
- Support batch inference for scalability.

OUTPUT:



DermalScan – AI Skin Analysis

Upload a face image to detect skin conditions and estimated age

Upload Face Image

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

clear_skin_017.jpg 32.5KB

Original Image

Annotated Image

Processing Time: 2.59 seconds

Analysis History

	Filename	Condition	Confidence (%)	Estimated Age	Face Detector Confidence (%)	X1	Y1
0	tmpdsxe4owe.jpg	clear skin	90.26	23	89.7	263	

Fig 5: User Interface

Milestone 4: Finalization and Delivery (Weeks 7–8)

• Overview

Milestone 4 represents the final stage of the AI DermalScan project, where the complete system is stabilized, validated, and prepared for final submission and demonstration. The focus of this milestone is to ensure that all components of the application work seamlessly together and that the outputs produced by the system are accurate, reliable, and well-structured for end users and evaluators.

Module 7: Export and Logging

• Objective:

To enable systematic export of results and maintain structured logs for evaluation, reporting, and future analysis.

• Description:

In this module, the AI DermalScan application is enhanced with export and logging capabilities. Users are provided with options to download annotated facial images displaying detected aging features along with bounding boxes and confidence percentages. In parallel, the system generates structured CSV logs capturing prediction details such as image name, detected class, and probability score. Extensive testing is conducted on diverse facial images to validate the robustness, consistency, and correctness of the outputs.

• Key Activities:

- Implement functionality to export annotated facial images.

- Generate downloadable CSV files containing prediction results and confidence scores.

- Perform final validation using multiple and diverse test images.

- Verify correctness, consistency, and usability of exported outputs.

•Deliverables:

- Annotated facial images with detected aging features.

- CSV files containing prediction logs and confidence percentages.

- Final validated outputs demonstrating system stability.

Evaluation Criteria:

- Accurate and error-free export functionality.

- Proper CSV structure with consistent and readable data.

- Reliable alignment between annotated images and logged predictions.

Overall Deliverables

- End-to-end AI DermalScan application with complete functionality.

- Trained and validated MobileNetV2-based facial skin aging classification model.

- Integrated face detection and prediction pipeline using OpenCV and deep learning.

- Web-based interface for image upload, processing, and result visualization.

- Exportable annotated images and structured CSV prediction logs.

- Modular, readable, and well-organized source code suitable for extension and maintenance.

Final Conclusion:

The AI DermalScan project successfully demonstrates the practical application of deep learning and computer vision techniques for facial skin aging detection. By combining robust face detection, efficient image preprocessing, and an MobileNetV2-based classification model, the system accurately identifies key aging-related facial features such as wrinkles, dark spots, puffy eyes, and clear skin. The integration of a web-based interface ensures ease of use, while export and logging capabilities enhance transparency, reproducibility, and documentation.

Through comprehensive testing and final validation, the application proves to be stable, reliable, and suitable for real-world usage or academic evaluation. The modular architecture and clean implementation allow for future enhancements, such as incorporating additional aging indicators, improving model accuracy with larger datasets, or extending the system for age estimation and dermatological analysis. Overall, the project meets its defined objectives and serves as a strong foundation for further research and development in AI-driven facial analysis systems.

Libraries :

1. TensorFlow / Keras

Modules Used: load_img, img_to_array, ImageDataGenerator

Purpose: Loading input images into a workable format. Converting images to numerical arrays for processing. Applying augmentation techniques (rotation, zoom, shift, brightness, flip, shear), Managing preprocessing pipelines for deep-learning workflows

2. NumPy

Purpose: Numerical operations on image arrays, Array reshaping, normalization, and manipulation, Supporting backend computations for augmentation

3. Matplotlib (matplotlib.pyplot)

Purpose: Visualizing original images, Displaying augmented images in grid format. Validating dataset structure and augmentation results

4. OS Library (import os)

Purpose: Folder traversal , Dataset loading import os

5. OPEN CV

Purpose: Face detection using pre-trained Haar Cascade

6.MediaPipe

Purpose: Face Detection using pretrained models

7.Streamlit

Purpose: Building an inter-active web based user interface