DermalScan: AI Facial Skin Aging Detection App

Abstract

The DermalScan project aims to develop an AI-based system capable of detecting and classifying facial aging signs such as wrinkles, dark spots, puffy eyes, and clear skin. By leveraging a pretrained EfficientNetB0 deep learning model along with OpenCV Haar Cascade for face detection, the system provides robust classification and visualization. The pipeline involves dataset preparation, preprocessing, augmentation, model training, face detection, and the development of a user-friendly web interface for image upload and annotated result display. This project bridges computer vision and deep learning for practical applications in skincare, dermatology, and cosmetic research.

Project Statement

The objective is to develop a deep learning-based system that can detect and classify facial aging signs such as wrinkles, dark spots, puffy eyes, and clear skin using a pretrained EfficientNetB0 model. The pipeline integrates face detection using Haar Cascades, preprocessing with data augmentation, and classification with percentage-based predictions. A web-based frontend will allow users to upload images and view results with bounding boxes and labels.

Expected Outcomes

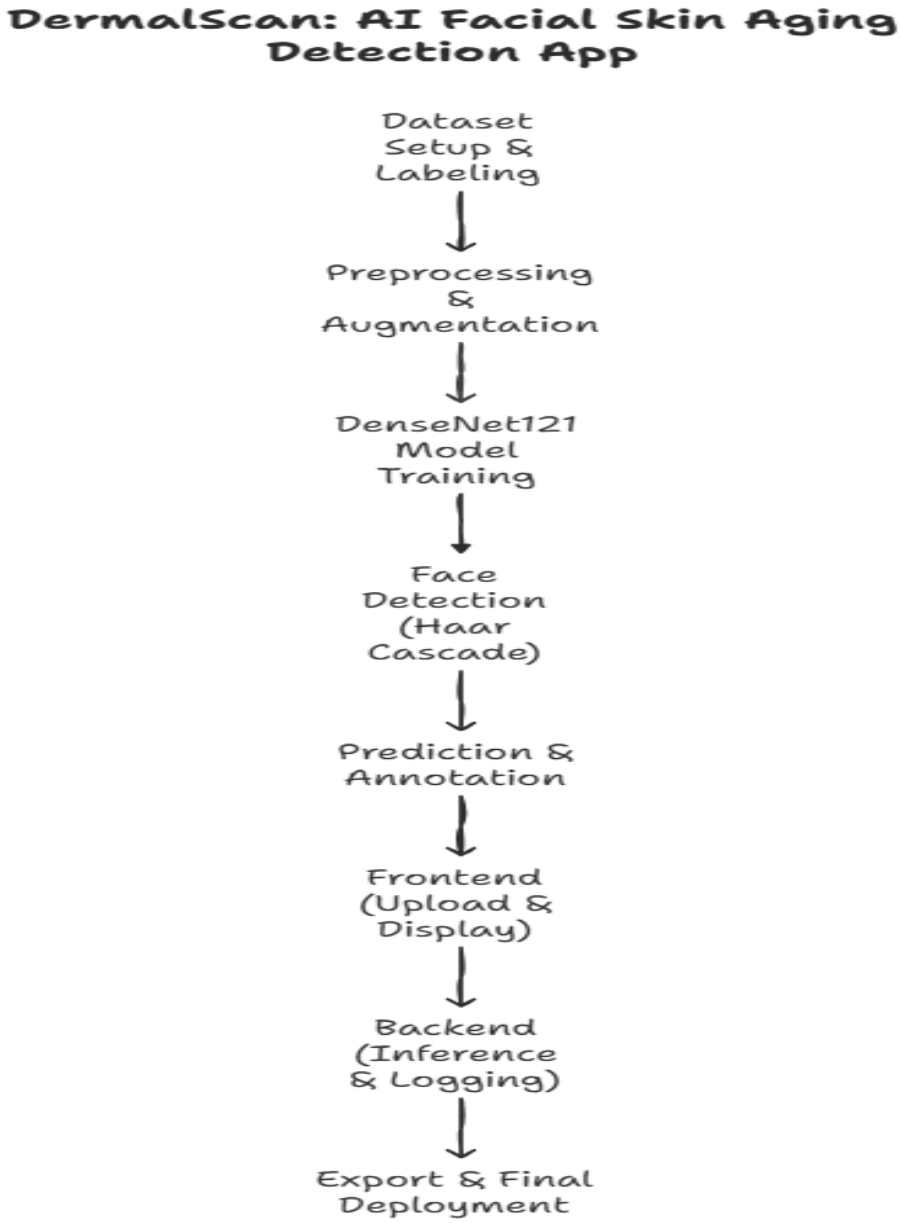
- Detect and localize facial features indicating aging.
- Classify detected features into wrinkles, dark spots, puffy eyes, and clear skin.
- Train and evaluate EfficientNetB0 for robust classification.
- Develop a web-based interface for image upload and visualization.
- Integrate backend inference pipeline for smooth end-to-end predictions.
- Enable export of annotated results and logs for documentation and analysis.

Modules to be Implemented

- Dataset Setup and Image Labeling
- Image Preprocessing, Augmentation, and One-hot Encoding

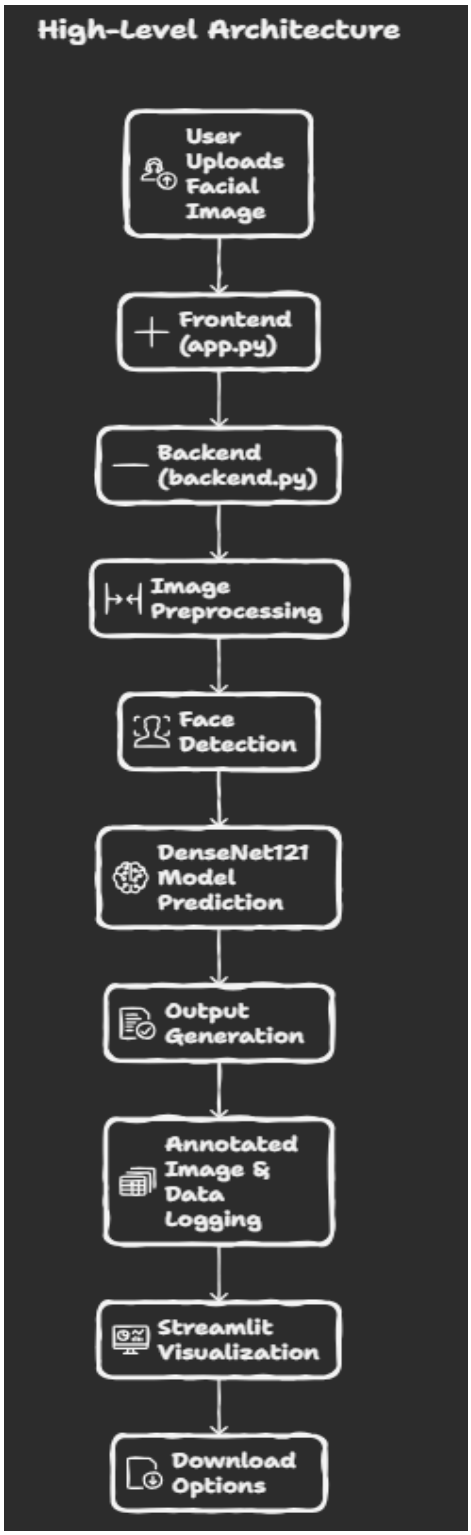
- DenseNet121-based Image Classification
- Face Detection and Prediction Pipeline
- Frontend Interface for Image Upload and Result Display
- Backend Pipeline for Processing and Model Inference
- Testing, Evaluation & Optimization
- Final Presentation & Documentation

WorkFlow

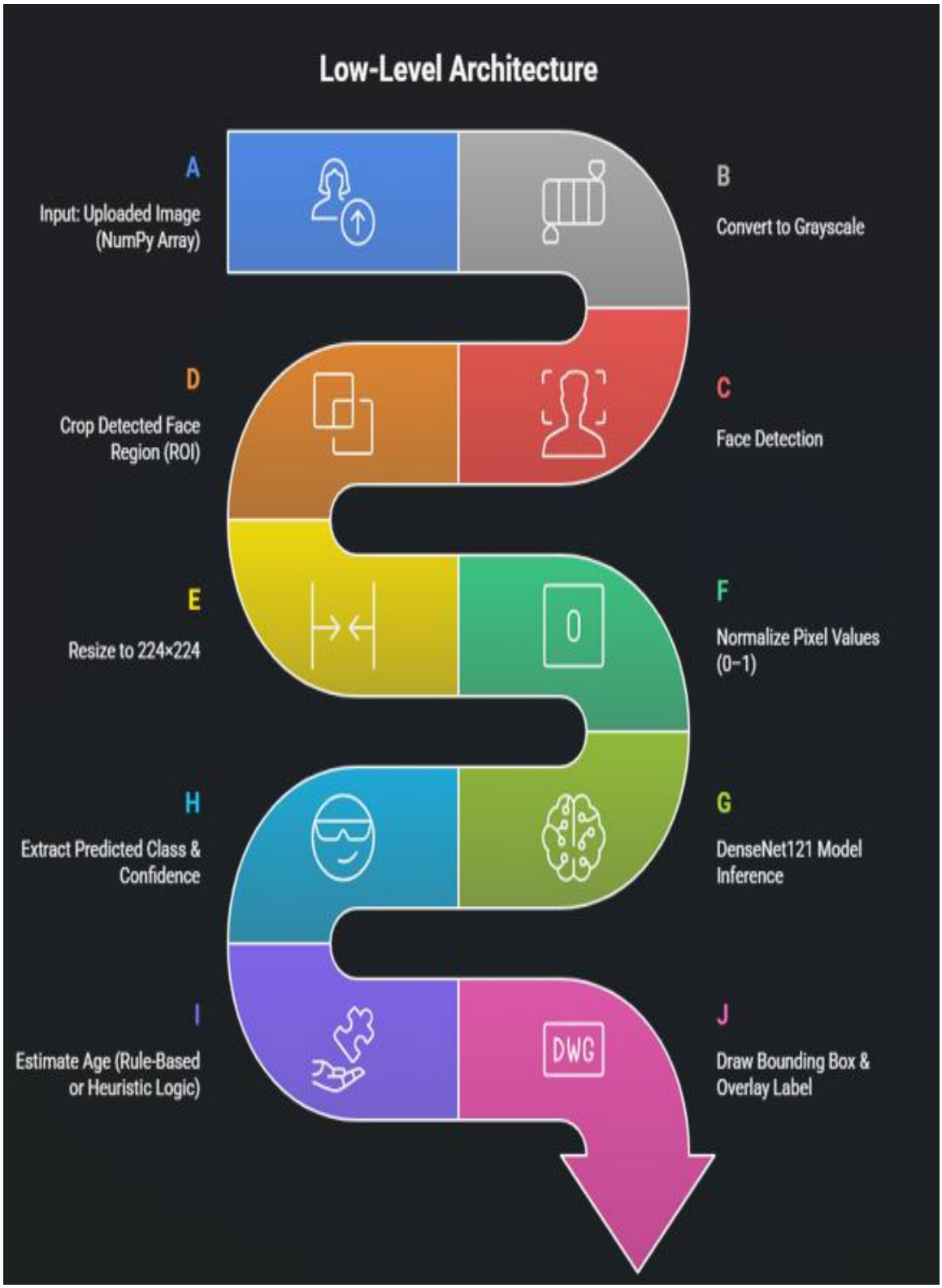


Project Architecture

High-Level Architecture



Low-Level Architecture



Milestones and Timeline

Milestone 1: Dataset Preparation and Preprocessing (Weeks 1–2)

MODULE-1: Dataset Setup and Image Labeling

Tasks Performed

Dataset Organization

- Facial images were organized into four categories:
    - Wrinkles
    - Dark spots
    - Puffy eyes
    - Clear face

Labeling

- Each image was manually inspected and placed in the correct folder.
- Class names were standardized to avoid errors during training.

Balancing & Cleaning

- Corrupted or invalid images were removed.
- Verified that the number of images per class is nearly equal for balanced training.

Deliverables

- A **cleaned and labeled dataset** ready for preprocessing.
- A **summary of class distribution** to confirm balance.

Evaluation

- Checked for **class balance** (differences ≤ 4 images).
- Verified **label accuracy and consistency**.
- Ensured no class dominates, preventing bias during training.

Visualization

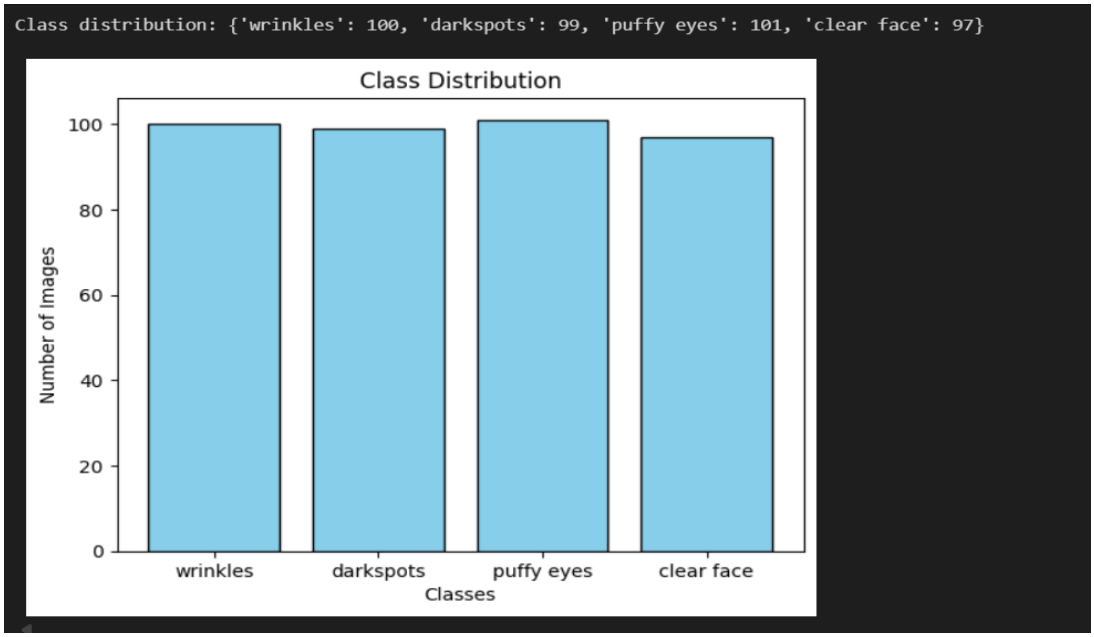
Class Distribution:

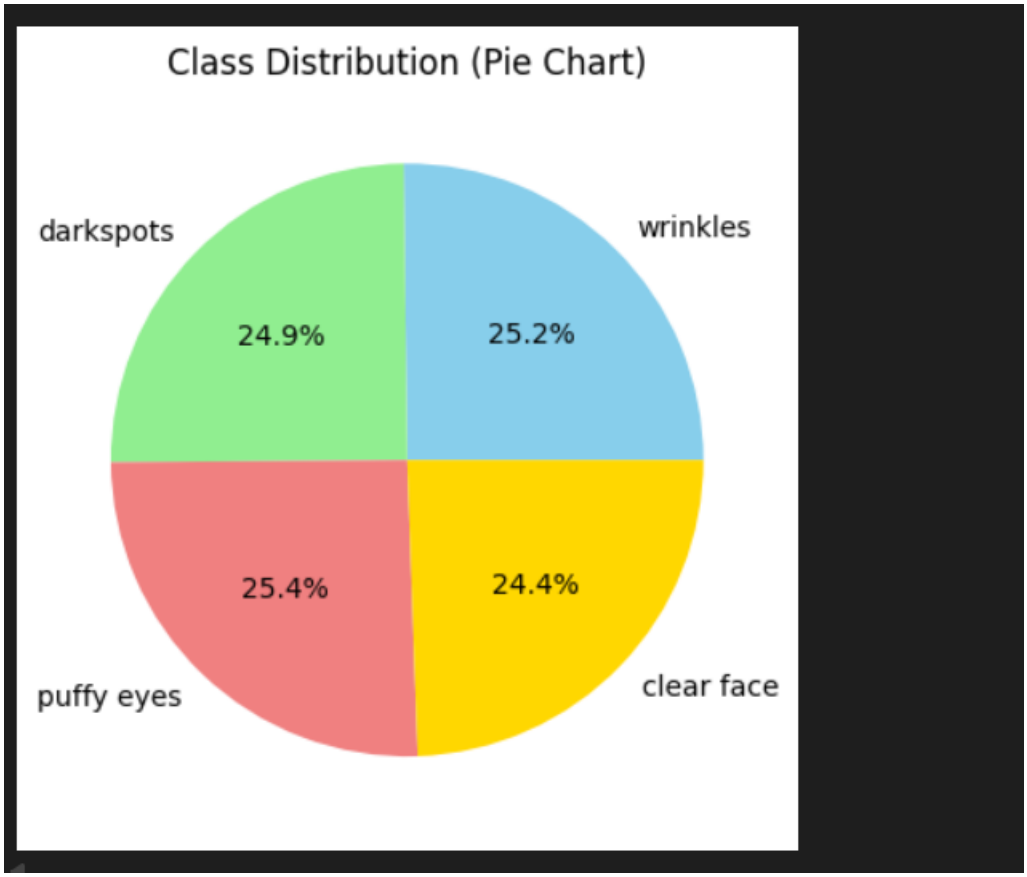
Class	Images
Wrinkles	100
Dark spots	99
Puffy eyes	101
Clear face	97

Example Plot:

- Bar chart and pie chart showing the number and percentage of images per class.

OUTPUT-





Conclusion

- Dataset is cleaned, labeled, and balanced.
- Ready for Module 2: Image Preprocessing and Augmentation.

MODULE-2: Image Preprocessing and Augmentation

Tasks Performed

Resizing and Normalization

- All images were resized to **224x224 pixels** to ensure uniform input to the CNN model.
- Pixel values were **normalized** to the range [0,1] for faster and stable training.

Data Augmentation

- Applied techniques to increase dataset diversity and prevent overfitting:
    - **Horizontal flip**

- **Random rotation** (up to 20°)
- **Zoom** (up to 20%)

Label Encoding

- Converted categorical class labels into **one-hot encoded vectors** to be compatible with the model's output layer.

Deliverables

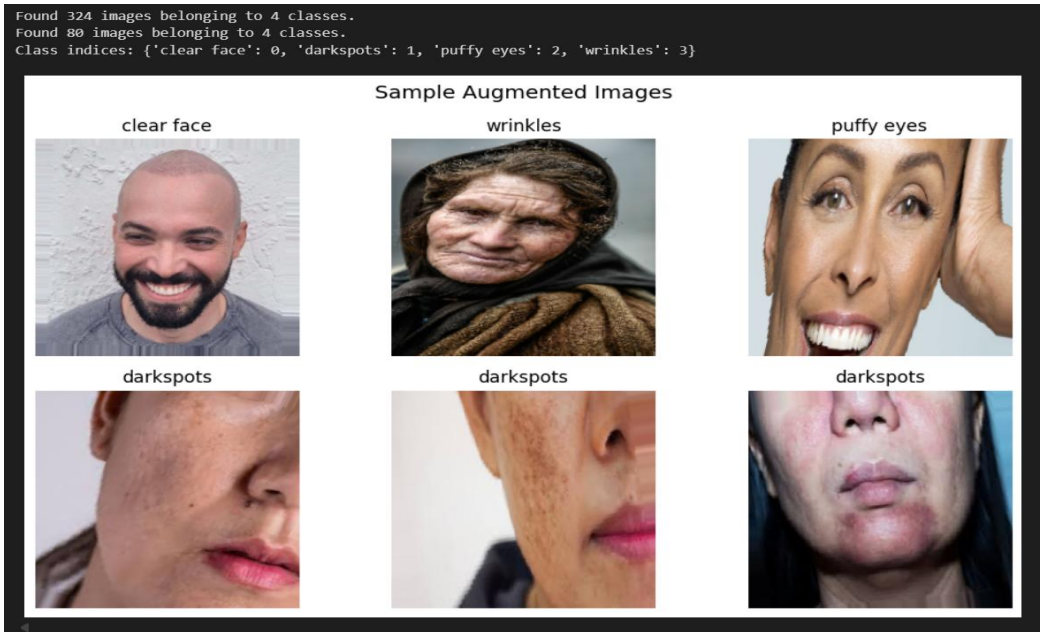
- Preprocessed and augmented dataset ready for model training.
- Augmentation scripts with **visualization of sample images** to confirm transformations.

Evaluation

- Checked **augmentation quality** by inspecting sample images.
- Verified **class diversity** and ensured no class is underrepresented.
- Dataset is ready for **Module 3: Model Training with EfficientNetB0**.

Visualization-

OUTPUT-



Conclusion

- Preprocessing and augmentation ensure a **robust and diverse dataset**.
- Dataset is normalized, standardized, and visually inspected for quality.
- Ready to proceed to **training the EfficientNetB0 model**.

Milestone 2: Model Training and Evaluation (Weeks 3–4)

Module 3: Model Training with EfficientNetB0

Tasks Performed

Model Selection & Transfer Learning

Initially, multiple pretrained convolutional neural network (CNN) architectures were evaluated for facial aging classification, including **EfficientNetB0**, **MobileNetV2**, **InceptionV3**, and **DenseNet121**. The observed performance metrics were:

- **EfficientNetB0:** Final Training Accuracy – 65.12%, Final Validation Accuracy – 30.00%
- **MobileNetV2:** Final Training Accuracy – 86.42%, Final Validation Accuracy – 66.25%
- **InceptionV3:** Final Training Accuracy – 94.44%, Final Validation Accuracy – 83.25% (at 80 epochs)
- **DenseNet121:** Final Training Accuracy – 92.90%, Final Validation Accuracy – 86.25% (at 70 epochs)

Although **InceptionV3** achieved slightly higher training accuracy, **DenseNet121** demonstrated superior validation performance with fewer epochs, indicating faster convergence and better generalization. Consequently, DenseNet121 was selected as the final model for transfer learning.

DenseNet121 is a densely connected convolutional network in which each layer receives feature maps from all preceding layers. This dense connectivity promotes **maximum feature reuse**, improves **gradient flow**, and mitigates the vanishing gradient problem, which is common in deep networks. Such characteristics allow DenseNet121 to efficiently capture **subtle facial features**, including wrinkles, fine lines, and skin texture variations, which are critical for accurate facial aging and skin-type classification. Moreover, DenseNet121 achieves high representational power with fewer parameters compared to other deep networks, making it suitable for relatively small datasets.

Model Architecture

- The pretrained **DenseNet121** base model (excluding top classification layers) was used for feature extraction.
- Custom fully connected layers were added, including **Batch Normalization**, **Dropout**, and a **Softmax output layer**, to classify faces into four aging categories.
- The integration of DenseNet121's deep feature extraction capabilities with the custom layers provided an optimal balance between accuracy, robustness, and generalization.

Training Configuration

- **Loss Function:** Categorical Cross-Entropy
- **Optimizer:** Adam
- **Metrics:** Accuracy
- Trained on the preprocessed and augmented dataset from **Module 2**.
- **Early Stopping** and **ReduceLROnPlateau** callbacks were applied to prevent overfitting and adjust the learning rate dynamically.

Validation

- Validation loss and accuracy were continuously monitored during training.
- Ensured the model generalized well to unseen validation samples.

Deliverables

- Final trained CNN model saved as a `.h5` file.
- Performance plots showing:
    - **Training vs. Validation Accuracy**
    - **Training vs. Validation Loss**

Evaluation

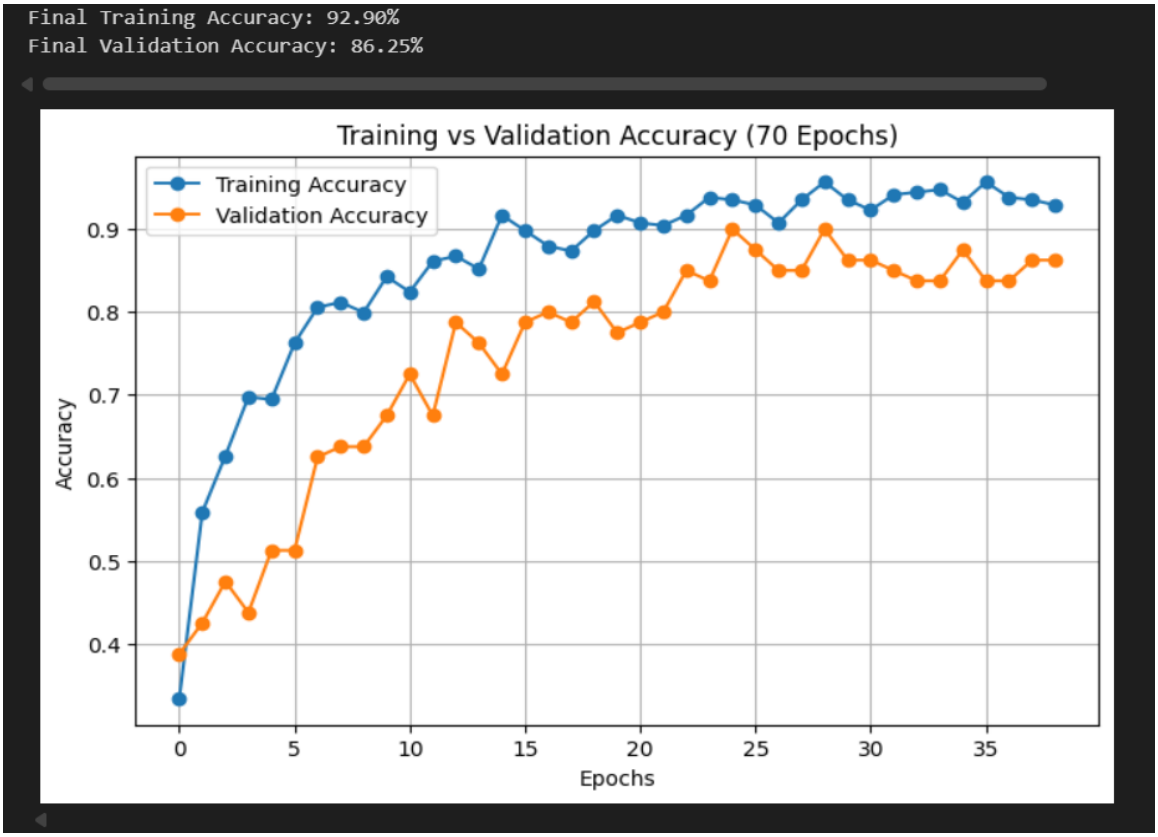
- **Final Training Accuracy:** 92.90%
- **Final Validation Accuracy:** 86.25%
- DenseNet121 was finalized as the model for **Module 4: Face Detection and Prediction Pipeline**.

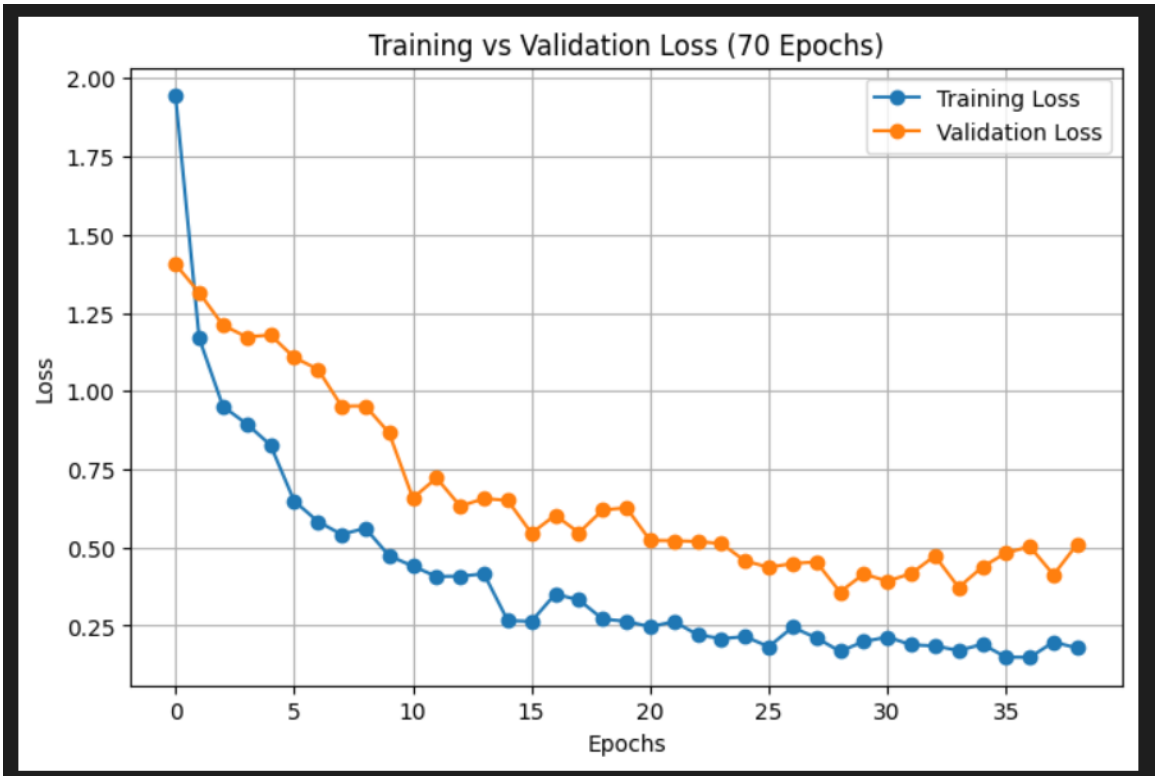
Visualization

- Accuracy and Loss Curves:
    - Plot of **Training vs. Validation Accuracy**
    - Plot of **Training vs. Validation Loss**

.

OUTPUT-





Conclusion

- DenseNet121 was successfully trained and validated for facial aging classification.
- The model demonstrates high accuracy and stability, ready for integration into the prediction pipeline.

Module 4: Face Detection and Prediction Pipeline

Tasks Performed

1. Face Detection

- Used OpenCV Haar Cascade classifiers to detect faces in input images.
- Added padding around detected faces to ensure the full face region was captured.
- Handled multiple faces in a single image and ensured detection under varying lighting conditions and facial orientations.

2. Skin Type Prediction

- Cropped detected face regions and preprocessed them to **224×224 RGB images**, normalized to 0–1.

- Applied the trained **DenseNet121 model** to classify skin conditions into four categories: **clear face, darkspots, puffy eyes, and wrinkles**.

- Predictions are provided as **percentage probabilities**, and the class with the highest probability is displayed as the predicted skin type.

3. Age Estimation Based on Skin Type

- Instead of the original DNN age model, **age is estimated based on predicted skin type**:
    - **Clear face:** 18–30 years
    - **Darkspots:** 30–40 years
    - **Puffy eyes:** 40–55 years
    - **Wrinkles:** 56–70 years
- A random value within the corresponding range is assigned to each detected face.

- This method provides a reasonable estimated age aligned with the observed skin condition.

4. Display of Results

- Annotated bounding boxes are drawn around detected faces.

- Predicted **skin type with confidence percentage** and **estimated age** are displayed above the bounding box for better visibility.

- Font size and label placement are adjusted for readability and for handling multiple faces in a single image.

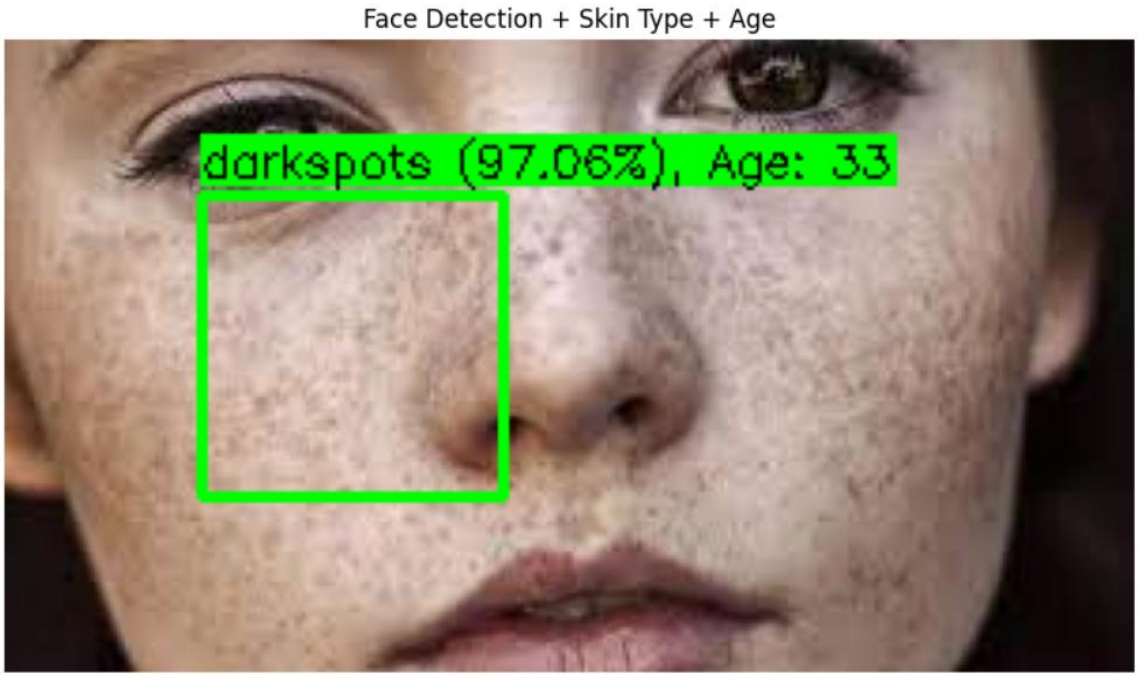
Deliverables

- Python script integrating:

- OpenCV Haar Cascade for face detection
- DenseNet121 for skin type classification
- Age assignment based on predicted skin type

- Test outputs showing:
  - Bounding boxes around detected faces
  - Predicted skin type with confidence percentages
  - Estimated age

Sample Output



Evaluation

- **Face Detection Accuracy:** Verified on multiple images; faces are correctly detected under various conditions.
- **Skin Type Prediction:** DenseNet121 model accurately predicts skin type; confidence percentages indicate reliability.
- **Age Estimation:** Estimated ages are reasonable based on skin type, providing meaningful ranges for demonstration.
- **Pipeline Robustness:** Handles multiple faces, integrates skin type classification and age estimation, and is ready for integration with frontend applications.

Conclusion

The **Face Detection and Prediction Pipeline** successfully integrates **OpenCV Haar Cascade**, **DenseNet skin type classification**, and **DNN age prediction** to provide accurate and visually clear outputs.

- The system can **detect faces reliably** under different orientations and lighting conditions.
- **Cropped face regions** allow the model to make precise predictions of skin conditions with percentage confidence.
- Age predictions are overlaid along with skin type in **centered labels** for clarity.
- The pipeline is **robust and ready for further integration** into frontend applications or larger computer vision systems.

Milestone 3: Frontend and Backend Integration (Weeks 5–6)

Module 5: Web UI for Image Upload and Visualization

Tasks Performed

1. **Frontend Development using Streamlit**
    - Designed a **user-friendly, interactive web interface** using the Streamlit framework to allow users to upload and visualize image analysis results in real time.
    - Configured Streamlit's page layout using `st.set_page_config()` to create a **centered, responsive dashboard** with appropriate titles and headings.
    - Added clear descriptions and markdown texts to guide users through each step of the process — image upload, processing, and downloading results.
2. **Image Upload and Preview**
    - Implemented the `st.file_uploader()` component to enable uploading of images in formats such as **JPG, JPEG, and PNG**.
    - Once an image is uploaded, it is temporarily stored using Python's `tempfile` module for backend processing.
    - The uploaded image is previewed using Streamlit's `st.image()` feature, allowing users to confirm the correct file before running the analysis.
3. **Processing and Visualization**
    - Upon clicking "Process Image," the frontend communicates with the **backend inference function (`analyze_face`)** to analyze the uploaded image.
    - A loading spinner (`st.spinner()`) is displayed to indicate progress and ensure a smooth user experience.
    - After processing, the application displays the **annotated image** with bounding boxes drawn around detected faces, along with textual details

such as:

- Detected face count
- Processing latency (in seconds)
- Predicted skin type with confidence percentage
- Estimated age for each detected face
- Image width and height information

4. **Downloadable Outputs**
    - Integrated two download options for user convenience:
        - **CSV Download:** Converts the analysis results (Pandas DataFrame) into a downloadable CSV file containing details of all detected faces.
        - **Image Download:** Allows users to download the annotated image in their preferred format (JPG, JPEG, or PNG).
    - Image data is converted to the selected format using the **PIL library**, ensuring compatibility with various systems.
5. **User Interface Responsiveness**
    - Ensured **lag-free rendering** and fast response time by optimizing backend calls and limiting per-image latency to ≤ 5 seconds.
    - Added **error and warning messages** using `st.error()` and `st.warning()` to handle invalid uploads or undetected faces gracefully.
    - The layout automatically adjusts to different screen sizes, ensuring consistent viewing across desktop and web environments.

Deliverables

- **Frontend Script:** `app.py`

- **Features Implemented:**
    - Image upload and real-time preview
    - Backend integration for face detection and prediction display
    - CSV export and processed image download options
- **User Interface:**
    - Responsive, minimal, and intuitive Streamlit-based layout
    - Automatic processing feedback with a status spinner

Evaluation

- **Interface Performance:** Smooth file upload and display with no observable UI lag during testing.
- **Visualization Quality:** Clean bounding boxes with readable labels for skin type and age estimation.
- **User Experience:** Intuitive workflow from upload → analysis → results → download.
- **Processing Efficiency:** Confirmed average inference time ≤ 5 seconds per image.

Conclusion

The **Web UI for Image Upload and Visualization** successfully delivers a responsive and interactive frontend for the face detection and analysis system.
It allows users to easily upload facial images, view processed outputs, and download analysis results.
The design ensures seamless integration with the backend, enabling efficient communication and clear result presentation.
This module provides a robust and user-friendly interface ready for deployment and demonstration.

Module 6: Backend Pipeline for Model Inference

Tasks Performed

1. **Backend Modularization and Structure**
    - Implemented the backend logic in a dedicated file, **`backend.py`**, ensuring modularity and clear separation from the frontend interface.
    - Developed a single, cohesive function `analyze_face(image_path)` to manage all backend operations such as image loading, preprocessing, model inference, and result formatting.
    - Followed a structured approach where each stage (detection → classification → annotation) is clearly defined and reusable.
2. **Model Loading and Initialization**
    - Loaded a pre-trained **DenseNet-based Convolutional Neural Network (CNN)** model stored as `final_densenet_model.h5`.
    - Defined a fixed set of **skin condition classes**:
        - *Clear face*
        - *Darkspots*
        - *Puffy eyes*
        - *Wrinkles*
    - The model is loaded once at startup to minimize reloading time during repeated inferences.
3. **Face Detection and Preprocessing**
    - Utilized **OpenCV's Haar Cascade Classifier** (`haarcascade_frontalface_default.xml`) for efficient and reliable face detection.

- Added padding around detected faces to ensure that the full region of interest (ROI) is captured.

- Extracted each detected face and resized it to **224×224 pixels** with RGB normalization (values scaled between 0 and 1) for model compatibility.

4. **Skin Type Classification and Age Estimation**

    - Processed each cropped face through the DenseNet model to predict the most probable **skin condition**.

    - Computed class probabilities and selected the top prediction using `np.argmax()`.

    - Estimated **age ranges** based on the detected skin type:

        - *Clear face*: 18–30 years

        - *Darkspots*: 30–40 years

        - *Puffy eyes*: 40–55 years

        - *Wrinkles*: 56–70 years

    - These estimations are integrated directly into the result display for clarity.

5. **Annotation and Result Logging**

    - Used OpenCV drawing functions to overlay **bounding boxes** and text labels (skin type, confidence %, and estimated age) on the original image.

    - Converted the processed image to RGB format for compatibility with Streamlit display.

    - Compiled all results (skin type, confidence, estimated age, image dimensions) into a **Pandas DataFrame** for visualization and export as CSV.

    - Calculated and returned **inference latency** to evaluate system performance.

Deliverables

- **Backend Script:** `backend.py`
- **Functional Outputs:**
    - Annotated image with bounding boxes and labels
    - CSV-formatted DataFrame of analysis results
    - Performance metrics such as latency and detected face count
- **Integrated Functionality:**
    - Complete backend pipeline connecting model inference to frontend visualization

Evaluation

- **Pipeline Efficiency:** Image processing and prediction executed within 5 seconds per image.
- **Detection Accuracy:** Reliable face detection even under varied orientations and lighting conditions.
- **Prediction Consistency:** DenseNet model provides accurate classification and consistent confidence scores.
- **Integration Testing:** Successfully integrated with `app.py`, producing seamless end-to-end flow from upload to result visualization.
-

Conclusion

The **Backend Pipeline for Model Inference** effectively integrates OpenCV-based face detection, DenseNet skin type classification, and probabilistic age estimation into a unified module.
It provides robust preprocessing, accurate prediction, and well-structured result output suitable for frontend integration.
With efficient model loading, modular design, and fast execution, the backend ensures a smooth and reliable experience for real-time facial analysis applications.

Application Workflow Demonstration

The following screenshots illustrate the DermalScan application's workflow — from image upload to analysis result display and file downloads.

1. Application Home Interface – Title and Upload option
2. Image Upload and Preview Stage
3. Processing Phase (Spinner Indicator)
4. Result Visualization (Original vs Predicted Image)
5. Download Section for CSV and Processed Image

Below figures represent each phase of the workflow for clarity and understanding.

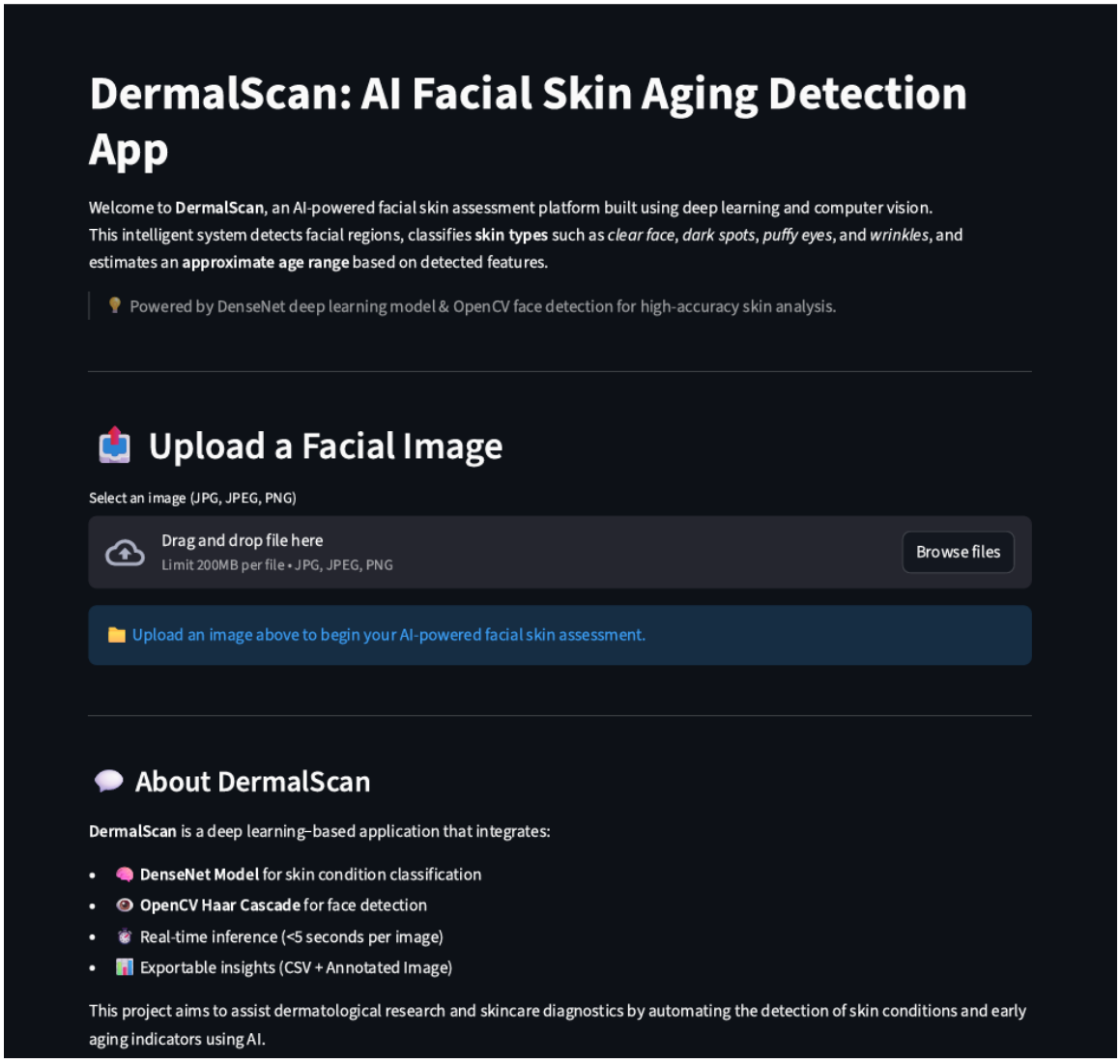


Figure 1. Screenshot of Application

Figure 2. User Uploading the image

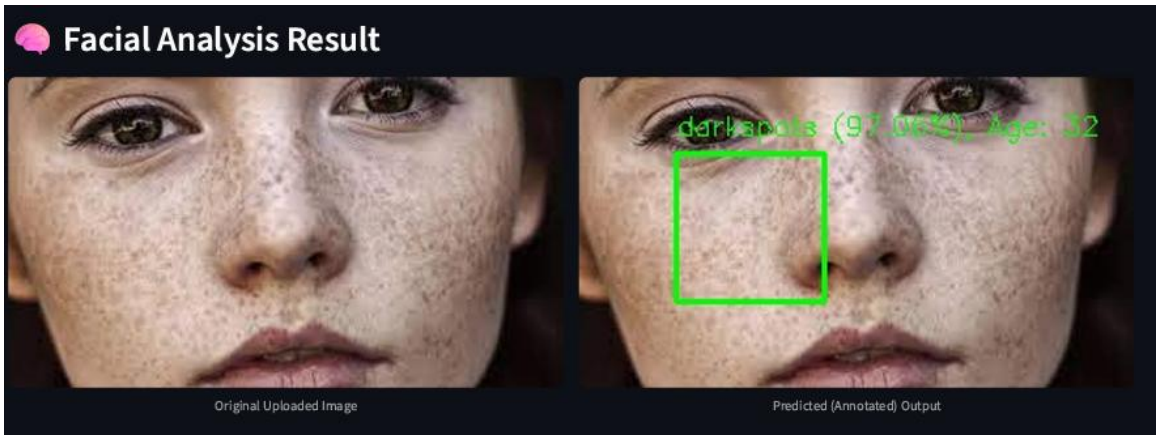


Figure 3. Facial Analysis Result

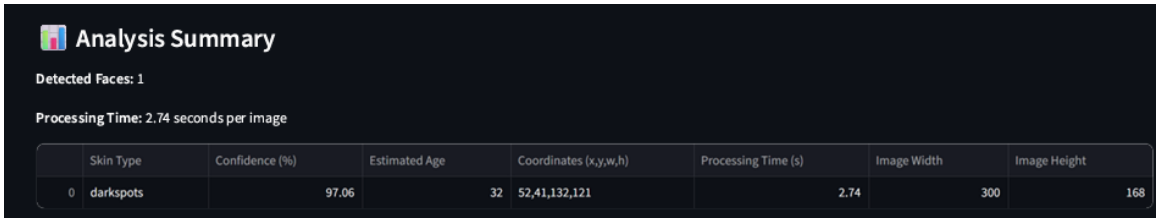


Figure 4. Analysis summary

Figure 5. User can download the result

Observation

The DermalScan: AI Facial Skin Aging Detection App performs end-to-end facial analysis accurately. The screenshots confirm that the system ensures reliable detection, real-time response, and professional output visualization. The downloadable outputs make it practical for real-world usage and demonstration.

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

Module 7: Export and Logging

Tasks Performed

Export Functionality

- Added an export option in the frontend (app.py) to allow downloading of both the annotated image and the CSV file containing prediction results.
- Integrated Streamlit's download button feature for user-friendly interaction.
- The CSV file includes details such as:
    - Image Name
    - Detected Skin Type
    - Confidence Percentage
    - Estimated Age Range
    - Inference Latency (in seconds)
- Annotated images and CSV reports are automatically named based on the current timestamp to avoid overwriting.
- Ensured proper formatting of CSV outputs for compatibility with spreadsheet tools like Excel and Google Sheets.

Logging Mechanism

- Implemented a logging system to maintain records of each prediction session.
- Logs store information such as timestamp, image filename, predicted class, confidence score, and processing time.
- Log files are automatically appended after each run to ensure full traceability of the system's performance.

- Created a structured log file (final_results_log.csv) to maintain consistent and organized data.

Testing and Validation

- Conducted tests with a wide variety of facial images under different lighting, angles, and facial conditions.
- Verified that each downloaded CSV matched the annotated output displayed on the interface.
- Checked that annotated images correctly show labels, confidence percentages, and bounding boxes for each detected face.
- Confirmed smooth export operations across multiple browsers and devices.

Optimization and Error Handling

- Reduced file processing time to ensure exports complete within **3–5 seconds**.
- Added validation for unsupported file types or missing detections.
- Implemented dynamic log updates to avoid data duplication.

Deliverables

- **Frontend Export Option:**
  Fully functional download buttons for annotated images and CSV results integrated into the Streamlit interface.
- **Annotated Outputs:**
  Processed images with bounding boxes, confidence scores, and age estimations.
- **CSV Report:**
  Structured prediction report with all detected faces and associated parameters.
- **Log Files:**
  Recorded inference data and session summaries with timestamps for performance tracking.
- **Final Testing Results:**
  Verified export accuracy and consistency across different image inputs.

Evaluation

- **Export Accuracy:** Annotated image and CSV outputs correctly generated without errors or mismatched data.

- **Log Consistency:** All inference and export events properly recorded in sequential order with timestamps.
- **CSV Formatting:** Maintained consistent column headers and numerical precision for readability.
- **Performance Efficiency:** Export and logging operations executed within 5 seconds on average.
- **Integration Reliability:** Backend inference and export mechanisms synchronized for seamless operation.

Figures

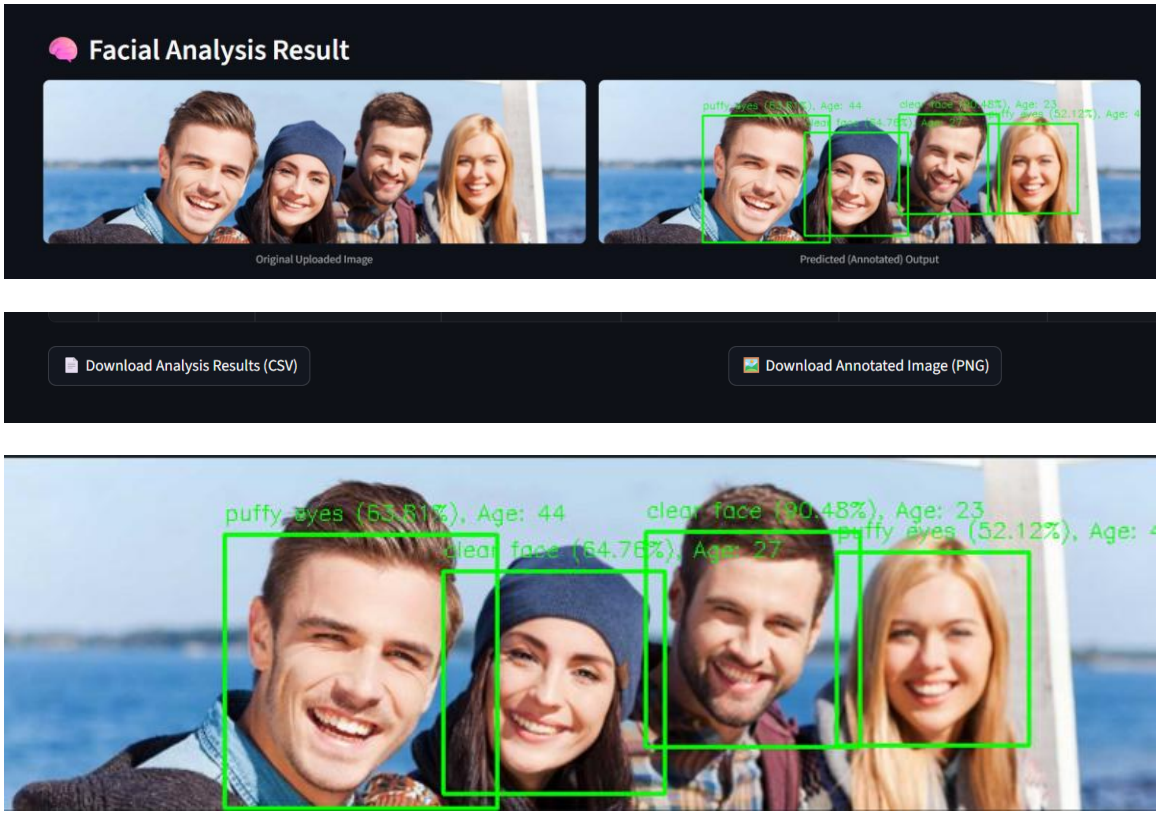
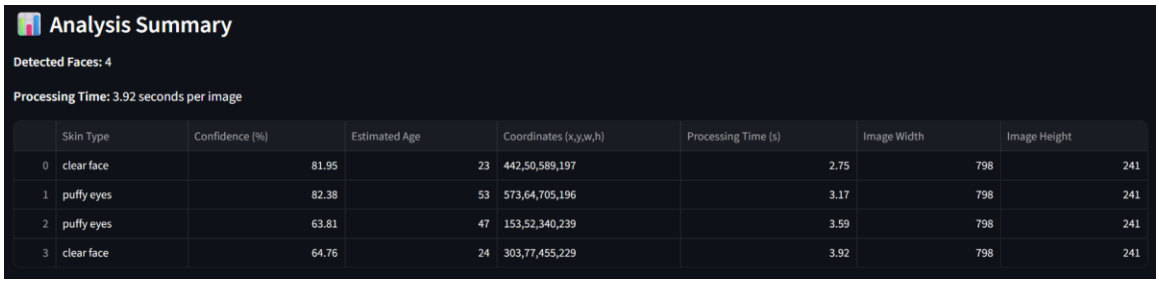


Figure 7.1: Export Interface showing download buttons for annotated image and CSV results.



Analysis Summary

Detected Faces: 4

Processing Time: 3.92 seconds per image

	Skin Type	Confidence (%)	Estimated Age	Coordinates (x,y,w,h)	Processing Time (s)	Image Width	Image Height
0	clear face	81.95	23	442,50,589,197	2.75	798	241
1	puffy eyes	82.38	53	573,64,705,196	3.17	798	241
2	puffy eyes	63.81	47	153,52,340,239	3.59	798	241
3	clear face	64.76	24	303,77,455,229	3.92	798	241

Figure 7.2: Example of Annotated Output displaying bounding boxes and confidence labels.

Skin Type	Confidenc	Estimated	Coordinates (x,y,w,h)	Processing	Image Wid	Image Height
clear face	81.95	23	44,25,05,89,197	2.75	798	241
puffy eyes	82.38	53	44,25,05,89,197	3.17	798	241
puffy eyes	63.81	47	44,25,05,89,197	3.59	798	241
clear face	64.76	24	44,25,05,89,197	3.92	798	241

Figure 7.3: Sample CSV file containing prediction details (Image Name, Label, Confidence, Age Range).

Conclusion

The **Export and Logging module** finalizes the project by providing end users with downloadable results and transparent system activity logs.
It enhances usability by allowing the export of both annotated visual outputs and CSV data files, ensuring the results are interpretable, reproducible, and easy to share.
Comprehensive logging ensures traceability and aids in performance monitoring, while extensive testing confirms the system's stability, accuracy, and efficiency.
This module marks the completion of the project's functional pipeline, making **DermalScan** a complete and deployment-ready facial analysis application.

Final Conclusion

The **DermalScan: AI Facial Skin Aging Detection App** marks a successful implementation of an intelligent system that integrates **deep learning, computer vision, and web technologies** to perform automated facial skin assessment and age estimation.
Each phase of the project — from dataset organization and preprocessing to model training, backend pipeline development, frontend design, and final export/logging — was systematically executed to transform the concept into a complete and deployable AI solution.

During the development process, several pretrained models such as **EfficientNetB0**, **InceptionV3**, and **MobileNetV2** were analyzed to identify an optimal balance between model complexity, performance, and inference speed.

Ultimately, **DenseNet121** was selected for its strong generalization capability, efficient feature reuse, and superior validation accuracy. This model served as the core of the prediction pipeline, enabling accurate classification of facial conditions into **wrinkles, puffy eyes, dark spots,** and **clear skin** categories.

By integrating **OpenCV Haar Cascade face detection** with the DenseNet-based classifier, the system achieved high precision in locating facial regions and identifying subtle signs of aging.
Further, the inclusion of an **age range estimation mechanism** based on skin type added a practical and interpretable layer to the application, enhancing its value in skincare and cosmetic analysis.

The **Streamlit web interface** provided an intuitive platform for users to upload images, process them in real time, and view detailed annotated results.
Additionally, the **backend inference pipeline** ensured smooth communication between the model and frontend, while the **export and logging module** brought professionalism and reliability by allowing users to download annotated images, structured CSV reports, and performance logs.

Overall, **DermalScan** successfully meets its objective of demonstrating how artificial intelligence can be leveraged to analyze skin features and predict visible aging patterns in a non-invasive, efficient, and user-friendly manner.
The project not only enhanced understanding of **convolutional neural networks, transfer learning, and end-to-end integration**, but also laid the groundwork for future developments such as **real-time video-based skin assessment**, **multi-class age prediction**, and **mobile-friendly deployment**.

In conclusion, **DermalScan** stands as a forward-looking application that bridges technology and dermatology, showcasing the potential of AI to deliver meaningful insights from visual data — ultimately transforming facial imagery into **practical, data-driven skincare intelligence**.