Infosys Springboard Virtual Internship Program

Submited by:

Sumit Ingle

Under the guidance of Mentor **Praveen**

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 includes face detection using Haar Cascades, custom preprocessing and data

augmentation, and classification with percentage predictions. A web-based frontend will enable

users to upload images and visualize aging signs with annotated bounding boxes and labels.

Outcomes:

● Detect and localize facial features indicating aging.

● Classify detected features into categories like wrinkles, dark spots, puffy eyes, and clear skin

using a trained CNN model.

● Train and evaluate an EfficientNetB0 model for robust classification.

● Build a web-based frontend for uploading facial images and viewing annotated outputs.

● Integrate a backend pipeline that processes input images and returns annotated

results.

● Export annotated outputs and logs for documentation or analysis.

Modules to be implemented:

● Dataset Setup and Image Labeling

● Image preprocessing, augmentation, and one-hot encoding

● EfficientNetB0-based image classification using TensorFlow/Keras

● Frontend interface for image upload and result display

● Backend pipeline for processing and model inference

● Testing, Evaluation & Optimization

● Final Presentation & Documentation

Milestone 1: Dataset Setup and Image Labeling

Module 1: Dataset Preparation

As a student, I began by setting up and inspecting the facial image dataset. The dataset was obtained from Kaggle

The dataset contains facial images categorized based on different skin conditions. The labeling process involved assigning each image to one of the following classes:

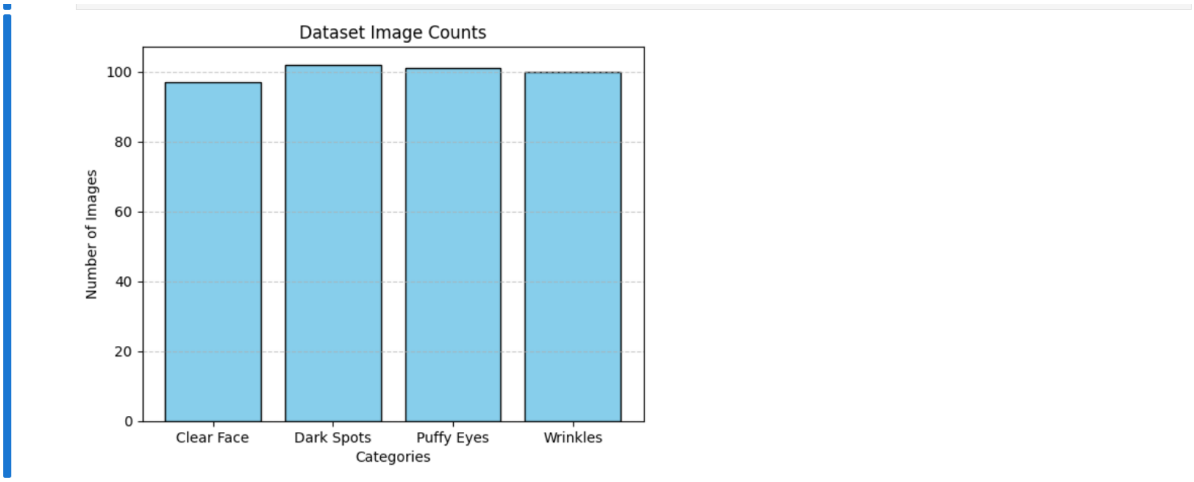
- Wrinkles
- Dark Spots
- Puffy Eyes
- Clear Skin

During the initial inspection, I observed that the dataset contained several noisy and duplicate samples. These were carefully removed to maintain data quality. Additionally, I ensured that the dataset maintained a balanced distribution of samples across all four classes to avoid model bias during training.

To verify this, I generated a class distribution plot, which confirmed that each category contained an approximately equal number of images.

Through this module, I learned the importance of data cleaning, accurate labeling, and balanced class distribution in building a reliable computer vision model.

Output:



Module 2: Image Preprocessing

In this module, I focused on preparing the dataset for model training through a series of preprocessing and augmentation steps. The primary goal was to enhance the dataset's quality, improve generalization, and ensure uniformity across all samples.

The preprocessing stage involved resizing all facial images to a standard resolution of 224×224 pixels and normalizing the pixel values to maintain consistency and facilitate efficient learning during training.

To improve model robustness and prevent overfitting, I applied various image augmentation techniques, including:

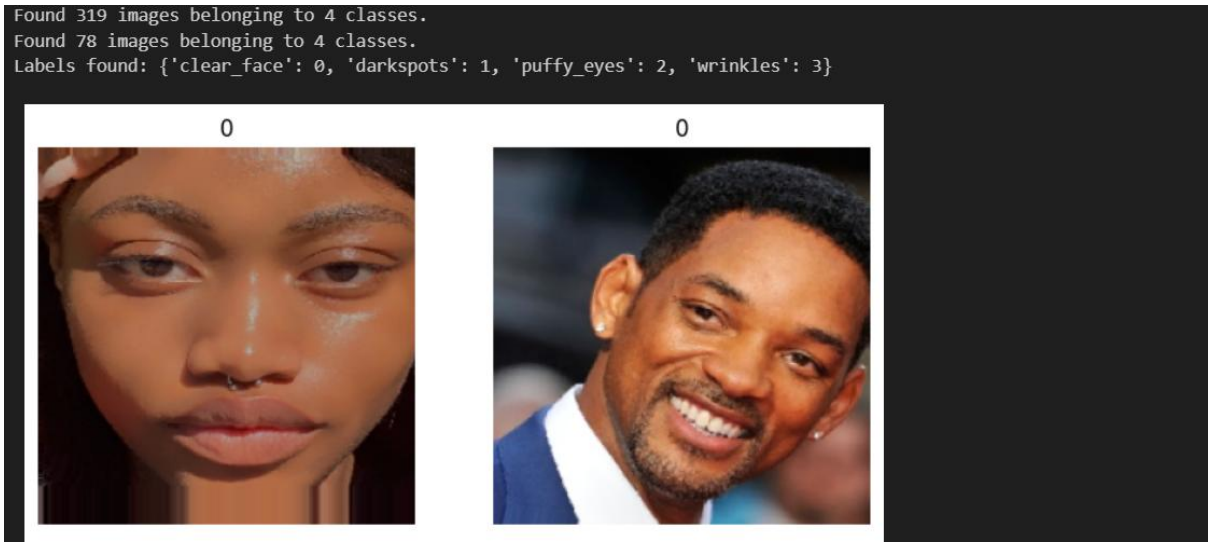
- Horizontal Flipping
- Random Rotation
- Zoom Transformation

Each of these transformations helped create variations in the dataset, simulating real-world conditions and increasing the diversity of training samples.

Additionally, I performed one-hot encoding on the class labels to convert them into a machine-readable numerical format, ensuring compatibility with deep learning models.

A visualization of augmented samples was generated to verify the effectiveness and quality of the applied augmentation

Output:



Module 3: Model Training with DenseNet121

---

1. Purpose & Overview

This module describes the process for training a deep convolutional neural network to classify facial skin conditions (clear face, dark spots, puffy eyes, wrinkles). Originally specified for EfficientNetB0, the implementation uses DenseNet121 with transfer learning, leveraging ImageNet weights for faster convergence and better generalization on limited facial image data.

Core objectives:

- Achieve high, robust accuracy in facial skin condition classification
- Minimize overfitting via augmentation, regularization, and validation monitoring
- Deliver a trained .h5 model and visual training metrics

---

2. Data Preparation & Augmentation

Dataset:

- Images sorted in the directory by class (dataset_path)
- Each category (clear_face, darkspots, puffy_eyes, wrinkles) in its own subfolder

Preprocessing & Augmentation:

- Implemented with ImageDataGenerator:
    - Rescale pixel values to

    - Random Rotation: Up to 30°
    - Zoom: Up to 30%
    - Width/Height Shifts: Up to 20%
    - Shear: Up to 15%
    - Brightness Adjustment: Range 0.7–1.3
    - Horizontal Flipping: For invariance
    - Fill Mode: Nearest neighbor
    - Validation Split: 20%
- Augmentation ensures the model generalizes across lighting, pose, and feature variation

Data Loading:

- Two generator objects:
    - train_gen: Loads training images with augmentations
    - val_gen: Loads validation images without shuffling

---

3. Model Architecture: DenseNet121 with Transfer Learning

Base Model:

- DenseNet121 imported via Keras, loaded with pretrained ImageNet weights, top layers removed
- Input size set to (224, 224, 3)

Layer Freezing:

- Early layers frozen: Only the last 40 layers unfrozen for initial training to preserve learned features and prevent overfitting

Custom Classification Head:

- GlobalAveragePooling: Reduces last layer's spatial dimension
- BatchNormalization: For training stability
- Dense Layer (256 units, ReLU): Captures nonlinear features
- Dropout (0.4): Helps prevent overfitting
- Output Layer: Softmax, units = number of classes

Model Compilation:

- Optimizer: Adam, learning rate 0.0005 for initial training
- Loss: Categorical cross-entropy for multi-class classification

- Metric: Accuracy tracked

---

4. Regularization & Training Strategy

Callbacks:

- ModelCheckpoint: Saves the model only when validation accuracy improves (filename: best_densenet_model.h5)
- ReduceLROnPlateau: Reduces learning rate by a factor of 0.4 if validation loss doesn't improve for 3 epochs (min LR 1e-6)
- EarlyStopping: Stops training if validation loss stagnates for 6 epochs, restores the best weights

---

5. Training & Fine-Tuning

Initial Training:

- Train on the unfrozen (last) 40 layers for 25 epochs
- Monitors both training and validation sets

Fine-Tuning:

- Unfreezes the last 60 layers for deeper adaptation (after initial convergence)
- Lowers LR to 1e-5 for fine gradients
- Train for up to 15 additional epochs with all callbacks in effect

---

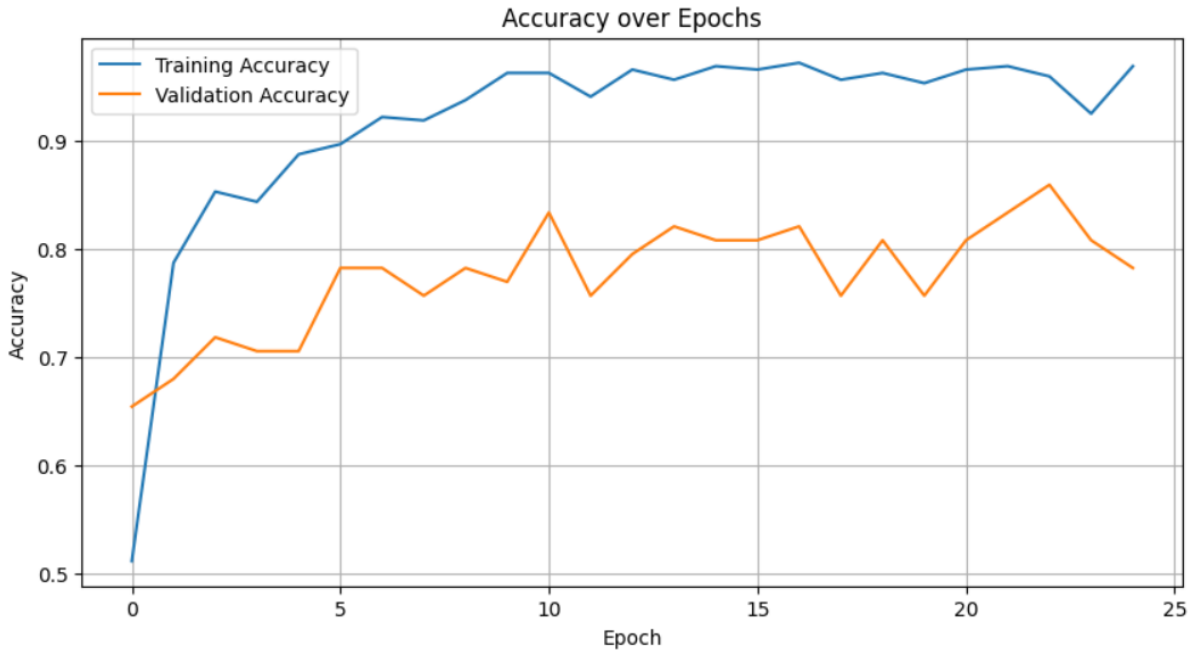
6. Monitoring & Visualization

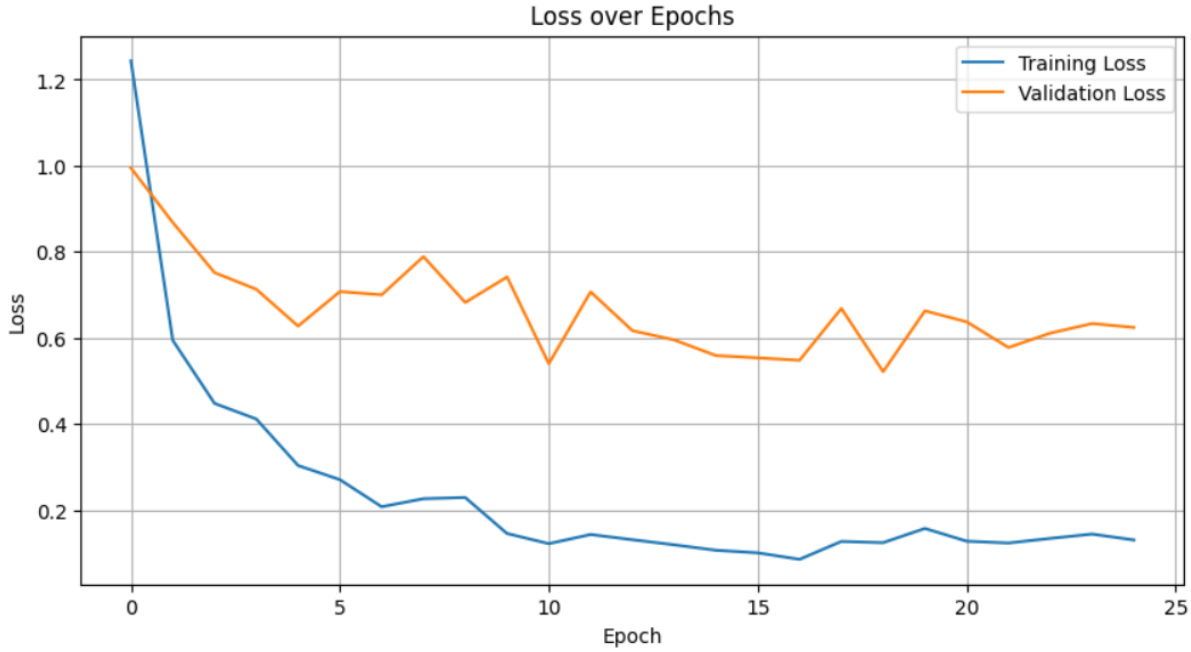
Metrics Tracked:

- Training Accuracy & Loss
- Validation Accuracy & Loss

Visualization:

- Plots accuracy and loss over all epochs (initial + fine-tune) using matplotlib
- Useful for diagnosing convergence, generalization, and overfitting


Accuracy over Epochs
Training Accuracy
Validation Accuracy
Accuracy
Epoch
0.5
0.6
0.7
0.8
0.9
0
5
10
15
20
25


Loss over Epochs
Training Loss
Validation Loss
Loss
Epoch
0.2
0.4
0.6
0.8
1.0
1.2
0
5
10
15
20
25

Module 4: Face Detection and Prediction Pipeline

---

1. Purpose

This module introduces an automated pipeline for facial region detection and classification using OpenCV and a trained DenseNet121 deep learning model.

Objectives:

- Detect faces in input images using Haar Cascade classifiers
- Crop detected face regions for focused prediction
- Predict skin condition classes (clear face, dark spots, puffy eyes, wrinkles)
- Estimate biological age based on predicted class
- Visualize results by overlaying bounding boxes and labels

---

2. Workflow and Data Flow

Step-by-step process:

1. Load the Trained Model:
    - Imports TensorFlow/Keras, loads a DenseNet121 model from disk (best_densenet_model.h5), trained per previous module.
2. Face Detection:
    - Utilizes OpenCV's Haar Cascade classifier (haarcascade_frontalface_default.xml) for robust frontal face detection.
    - Converts the image to grayscale and applies detection (detectMultiScale).
    - Able to process images with multiple faces simultaneously.
3. Face Cropping & Preprocessing:
    - Extracts the bounding box of each detected face.
    - Crops the region, resizes to model's input size (224×224).
    - Normalizes pixel values and converts to a shape compatible for deep learning inference.
4. Prediction with DenseNet Model:
    - Inputs cropped faces into DenseNet for classification.
    - Predicts the likelihood for each class: clear face, darkspots, puffy eyes, wrinkles.

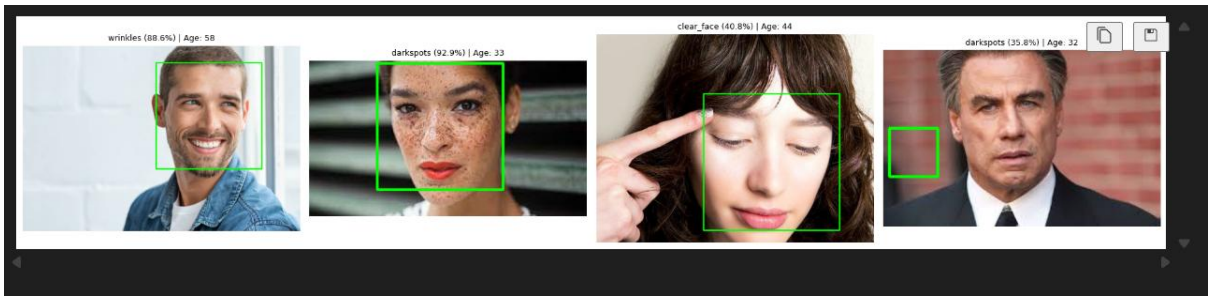
- Returns top prediction, confidence percentage, and estimated age via predict_age (randomly sampled within class-specific range for interpretability).

5. Result Visualization:
    - Draws bounding boxes on detected face regions in green.
    - Returns an annotated image (RGB for display in matplotlib).
    - Prepends predicted label, confidence, and age as subplot titles for clarity.
6. Batch Testing & Display:
    - Iterates over sample images in each class-labeled folder.
    - Randomly selects and tests one image per class.
    - Displays all results as a grid of annotated images using matplotlib.

---

3. Code Features & Highlights

- Robust error handling:
  Prints appropriate messages for missing images, unread files, or absent detected faces.
- Multi-face capability:
  Detects and predicts for every face found, enabling future batch analysis for group photos.
- Class & Age Estimation:
    - Predefined age ranges assigned per skin condition for more interpretable results.
- Visualization:
    - Uses matplotlib for easy-to-understand comparison across classes.
    - Displays bounding boxes on images and titles with prediction info (label, confidence, age) for direct validation.
- Silencing TensorFlow warnings:
  Ensures clean output and logs for the user/developer by suppressing unnecessary messages.
- Directory flexibility:
  Each class label maps to its own directory under a main dataset folder (dataset_dir), facilitating scalable testing.



Module 5 & 6

1. Purpose and Overview

DermalScan AI is a full-stack application built to detect facial skin conditions (e.g., wrinkles, dark spots) and estimate biological age from uploaded images.
It uses a pre-trained deep learning classification model, and provides a responsive, real-time user interface for analysis visualization and result download.

Core objectives:

- Simple upload and fast feedback for dermatology analysis
- Visual explanation: annotated images, labeled predictions, bounding boxes
- Persistent history logging for cumulative review

---

2. Architecture & Data Flow

Component Breakdown

Component	**Role**	**Tech Details**
Streamlit Frontend	User interaction, file upload, output display	Streamlit, HTML/CSS
Preprocessing Module	Face detection, cropping, resizing, normalization	OpenCV, Numpy, Keras
Model Inference	Condition prediction, confidence, age estimation	TensorFlow/Keras
Visualization	Bounding boxes, annotated overlays	OpenCV
Logging & Session	Cumulative history, downloadable CSVs	Pandas, Streamlit

End-to-End Data Flow

1. Image Upload:
    - User browses and uploads a face photo via sidebar (st.file_uploader).
    - Image saved to uploads/.
2. Preprocessing & Face Detection:
    - Converts uploaded image to grayscale for detection.
    - Applies Haar Cascade for robust face region identification.
    - Optional padding added to bounding box for context.
    - If no face is found, heuristic fallback box is drawn.
3. Image Cropping and Model Preprocessing:
    - Crops detected face region(s) using bounding box coordinates.
    - Resizes cropped image to 224x224 pixels (model expected input).
    - Converts cropped region to array and applies normalization (preprocess_input).
4. Inference:
    - Model invoked with preprocessed face crop.
    - Returns probability vector for defined classes (clear face, darkspots, puffy eyes, wrinkles).
    - Maps highest value to class label (CLASS_NAMES dict).
    - Confidence (%) is calculated as maximum prediction probability.
5. Biological Age Estimation:
    - Based on predicted class, assigns a realistic random age within predefined ranges.
6. Annotation & Visualization:
    - Draws bounding box over the detected area with label, confidence and age overlay (OpenCV).
    - Annotated image saved to results/annotated_output.jpg.
7. Display to User:
    - Original image shown on left, processed annotated output on right (st.columns).
    - Progress bar simulates real-time feedback.
8. History and Download:
    - Compiles detection results with bounding box details, condition, confidence, age, file name, prediction duration.
    - Appends new results to persistent session DataFrame (st.session_state.history).

- Full results displayed as a table; CSV available to download.
- Annotated output image download provided.

9. Error Handling:
    - If no face is found, fallback bounding box is used and a warning shown.
    - For empty uploads, prompts user with info to upload an image.
10. Performance Measurement:
    - Captures total analysis time per upload.
    - Ensures workflow target of ≤5 seconds per image.

---

3. Key Features and Code Details

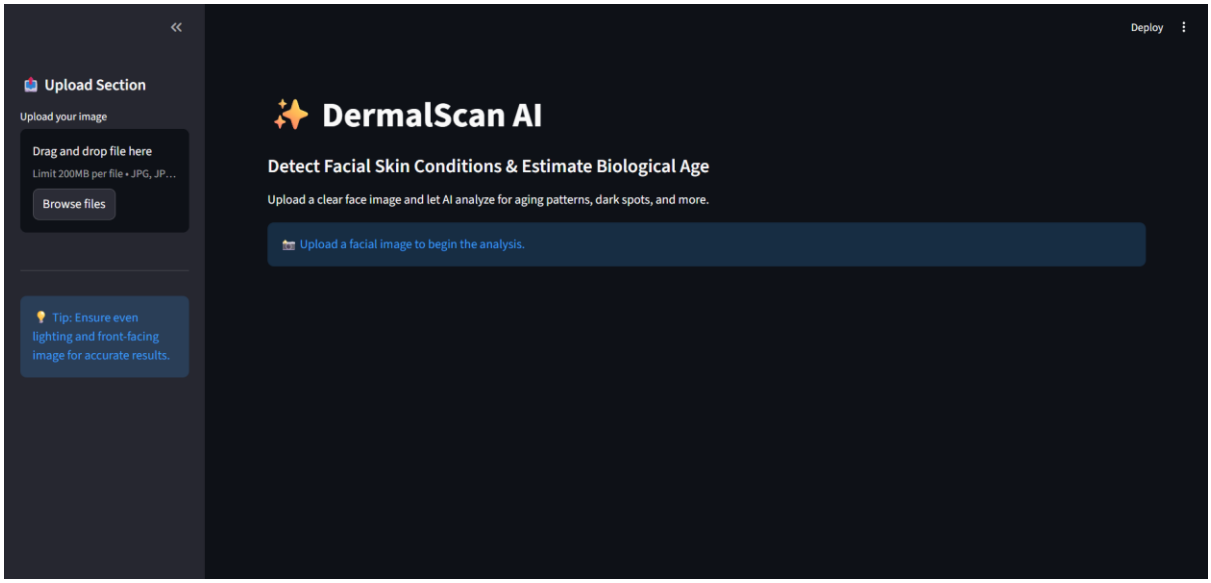
Frontend (Web UI) – What's Done

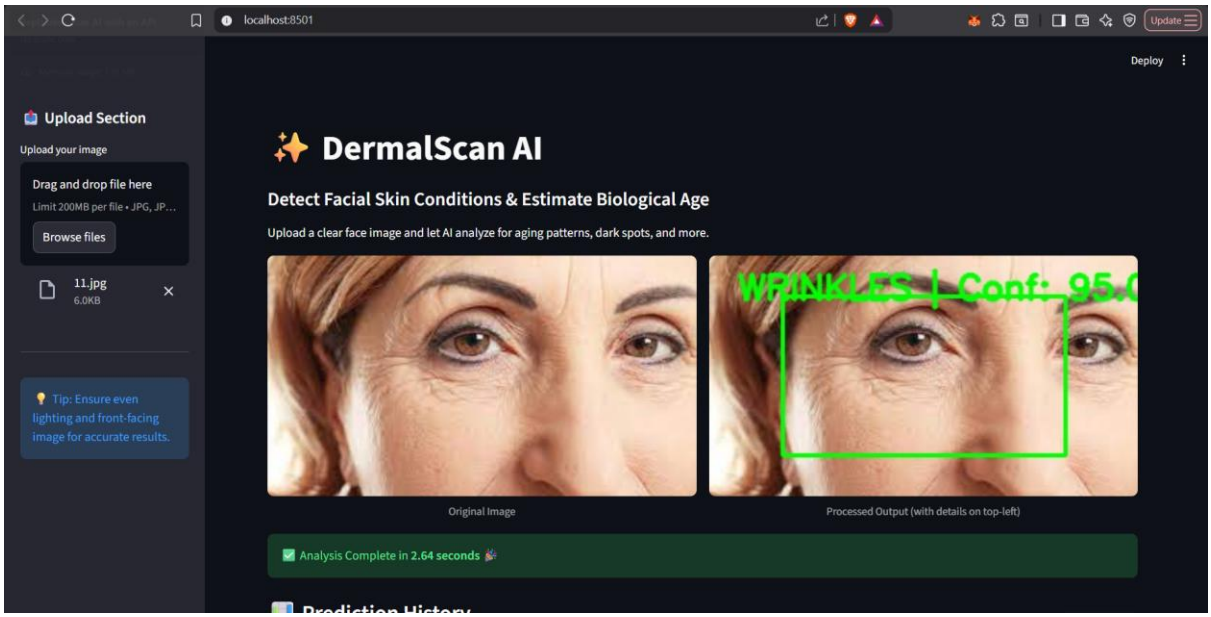
- Streamlit page setup: Wide layout, branded title and favicon
- Sidebar:
    - File upload widget for supported formats
    - User instructions and tips for input quality
- Image Display:
    - Shows both original and annotated (result) images side-by-side
    - Automatically scales images for optimal viewing
- Progress Feedback:
    - Animated analysis progress bar for improved user perception during inference
    - Spinner with "Analyzing image..." message
- Results Table:
    - Shows prediction history with details for every batch analyzed in current session
    - Organized columns: File name, Class ID, condition, confidence, estimated age, bounding box, prediction time
- Download Buttons:
    - Direct export of processed image (JPEG) and full prediction history (CSV)
    - All files stored in dedicated results directory
- Condition-based Age Estimation:
    - Uses pre-set biological age ranges per class for realistic output
- Responsive UI:

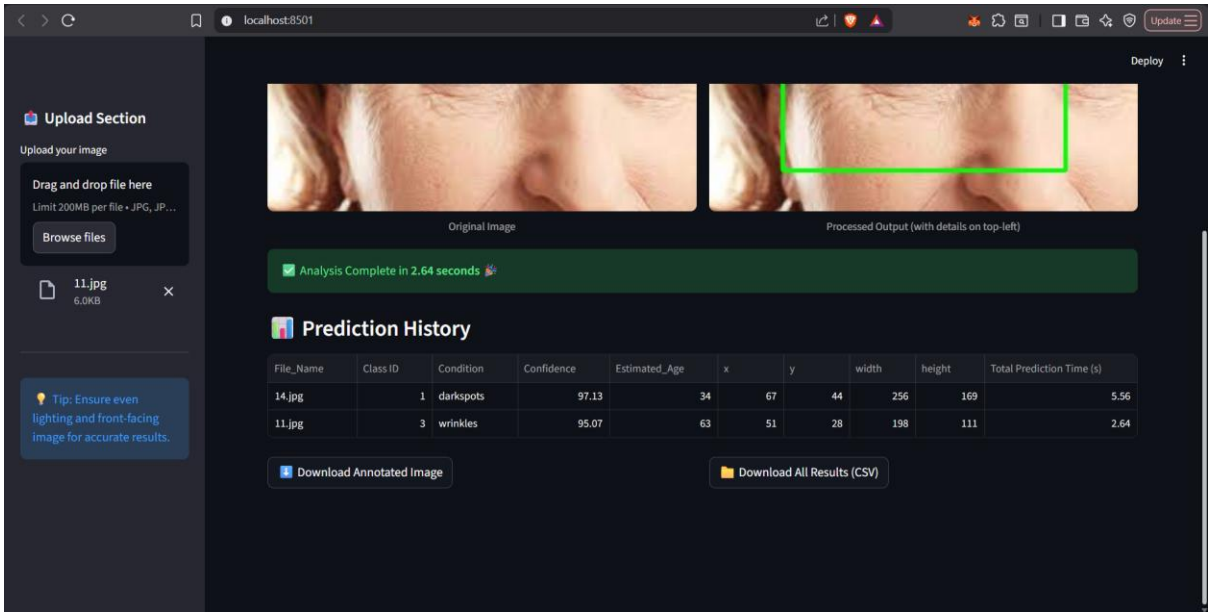
- All columns, progress, and buttons adapt to window size for usability

Backend (Inference Pipeline) – What's Done

- Model Loading:
  - Loads EfficientNet/DenseNet once per session with caching to maximize performance and minimize wait
  - Enables fast re-analysis for multiple user uploads
- Face Detection & Fallback:
  - Robust detection via Haar Cascade (OpenCV)
  - Implements fallback to ensure output even if detection fails
- Image Preprocessing:
  - Converts input to grayscale
  - Crops/pads region, resizes to 224x224, normalizes with mobilenet utility
- Classification & Confidence:
  - Returns detectable class, confidence (%), numerical ID, and mapped labels
- Annotation:
  - Draws bounding boxes and overlays info directly on image
  - Saves result to file for both UI display and download
- Logging & Persistent History:
  - Appends session results to DataFrame
  - Persists results across multiple uploads for review and export
- Performance:
  - Tracks and displays prediction latency per image batch
- Error Handling:
  - Warns user if input is invalid or no face is detected


Deploy
Upload Section
Upload your image
Drag and drop file here
Limit 200MB per file • JPG, JP...
Browse files
Tip: Ensure even lighting and front-facing image for accurate results.
DermalScan AI
Detect Facial Skin Conditions & Estimate Biological Age
Upload a clear face image and let AI analyze for aging patterns, dark spots, and more.
Upload a facial image to begin the analysis.


localhost:8501
Update
Deploy
Upload Section
Upload your image
Drag and drop file here
Limit 200MB per file • JPG, JP...
Browse files
11.jpg
6.0KB
Tip: Ensure even lighting and front-facing image for accurate results.
DermalScan AI
Detect Facial Skin Conditions & Estimate Biological Age
Upload a clear face image and let AI analyze for aging patterns, dark spots, and more.
WRINKLES | Conf:
Original Image
Processed Output (with details on top-left)
Analysis Complete in 2.64 seconds


localhost:8501
Update
Deploy
Upload Section
Upload your image
Drag and drop file here
Limit 200MB per file • JPG, JP...
Browse files
11.jpg
6.0KB
Tip: Ensure even lighting and front-facing image for accurate results.
Original Image
Processed Output (with details on top-left)
Analysis Complete in 2.64 seconds
Prediction History
File_Name | Class ID | Condition | Confidence | Estimated_Age | x | y | width | height | Total Prediction Time (s)
14.jpg | 1 | darkspots | 97.13 | 34 | 67 | 44 | 256 | 169 | 5.56
11.jpg | 3 | wrinkles | 95.07 | 63 | 51 | 28 | 198 | 111 | 2.64
Download Annotated Image
Download All Results (CSV)