Project Documentation: DermalScan

AI-Powered Facial Skin Aging Classifier and Age Predictor

Arpit Sharma

Built as part of the Infosys Springboard Internship

September 28, 2025

Contents

1 Project Overview

1.1 Introduction

DermalScan is a deep learning-based system designed to analyze facial images to detect and classify common signs of aging, as well as to predict the apparent age of the individual. The project leverages two distinct neural network models to accomplish these tasks, providing a comprehensive facial analysis tool.

The entire system is integrated into a user-friendly web application, where users can upload an image and receive an annotated version with detailed predictions. This serves as a practical application of computer vision and deep learning to the field of dermatology and cosmetology.

This project was developed as part of the Infosys Springboard Internship program.

1.2 Goal and Real-Life Requirements

The primary goal of the DermalScan project is to create an accessible and automated tool for facial skin analysis. In a real-life context, this project addresses the growing demand for personalized skincare and cosmetic analysis.

Real-Life Requirements:

- **For Consumers:** Provide users with insights into their skin condition, helping them make informed decisions about skincare products and routines. It can serve as a preliminary analysis tool before consulting a dermatologist.
- **For Skincare Professionals:** Assist dermatologists and cosmetologists by providing a rapid, data-driven analysis of a client's facial features, which can be used to track the efficacy of treatments over time.
- **For Research:** The underlying models can be used in dermatological research to analyze large datasets of facial images for studies on aging and skin health.

The system must be fast, accurate, and easy to use, capable of detecting faces in various conditions and providing clear, understandable results.

1.3 Key Outcomes & Features

The project is designed to deliver the following key features and outcomes:

- **Face Detection:** Automatically detect and localize a human face within an uploaded image.
- **Aging Sign Classification:** Classify the detected facial region into one of four categories: **wrinkles, dark spots, puffy eyes,** or **clear skin**. The system provides a percentage confidence score for each category.
- **Age Prediction:** Predict the apparent age (as a numerical value) of the person in the image.
- **Web-Based Interface:** A simple and intuitive web frontend built with Streamlit allows users to easily upload an image and view the results.
- **Result Visualization:** The application returns an annotated image with a bounding box around the detected face and the predictions written on it.
- **Data Export:** Users can download both the annotated image and a CSV file containing the detailed prediction probabilities.

2 System Architecture

2.1 High-Level Architecture

The DermalScan system is organized as a client-server web application.

- User Interface (Client-Side): The Streamlit script (app.py) automatically generates the frontend, providing widgets for users to upload images and view results on a dynamic, single page.
- Processing Pipeline (Server-Side): The same app.py script manages the backend workflow:
  - Receives the uploaded image and detects the face with an OpenCV Haar Cascade.
  - Crops and preprocesses the face, then sends it to two distinct models:
    - ResNet50V2: Fine-tuned for aging sign classification.
    - MobileNetV2: Used for age prediction.
  - Aggregates predictions, annotates the image, and generates a CSV log report.
  - Updates the user interface to display the final annotated image and provides download links.

2.2 Technology Stack

The project is built using a combination of tools and libraries standard in the field of AI and web development:

Area	Tools / Libraries
Image Ops	OpenCV, NumPy, Haar Cascade
Model	TensorFlow/Keras, ResNet50V2, MobileNetV2
Dataset	Custom folder-based (Aging Signs), UTKFace (Age)
Frontend	Streamlit
Backend	Python, Streamlit
Evaluation	Accuracy, Loss, Mean Absolute Error (MAE), Confusion Matrix
Exporting	CSV, Annotated Image (via Pandas, OpenCV)

Table 1: Technology Stack for DermalScan

2.3 Project Structure Flowchart

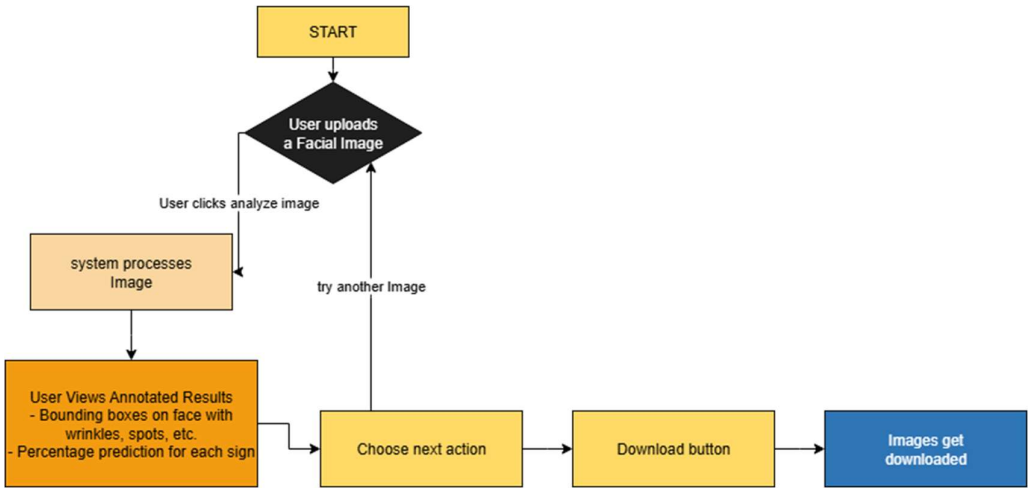

Figure 1: The project structure flowchart illustrating the workflow from user input to final output.

3 Milestone Breakdown & Implementation Details

This section details the implementation process, following the milestones laid out in the project guide.

3.1 Milestone 1: Dataset Preparation and Preprocessing

3.1.1 Module 1: Dataset Setup and Image Labeling

- **Aging Signs Dataset:** A custom dataset was assembled and organized into four distinct folders: clear face/, dark spots/, puffy eyes/, and wrinkles/. This folder-based structure is ideal for use with Keras's ImageDataGenerator. The dataset was balanced to ensure an equal number of samples for each class, preventing model bias.
- **Age Dataset:** The publicly available **UTKFace dataset** was used. This dataset contains over 20,000 facial images with labels for **age, gender, and ethnicity** embedded in the filenames (e.g., [age]_[gender]_[race]_[datetime].jpg). The data loading logic in the train_age_model.py script parses these filenames to extract the numerical **age** for each image.

3.1.2 Module 2: Image Preprocessing and Augmentation

- **Resizing & Normalization:** All images from both datasets were resized to a consistent input shape required by the models: 224x224 pixels for ResNet50V2 and 128x128 for the MobileNetV2. Pixel values were processed using the model-specific preprocess_input function for each network, which scales them to the expected range (e.g., [-1, 1]). This is a standard practice that improves model training stability and performance.
- **Image Augmentation:** To increase the diversity of the training data and prevent overfitting, data augmentation was applied using Keras's ImageDataGenerator. The techniques included random rotations, zooming, and horizontal flips, as detailed in the main.py script.
- **Label Encoding:** For the multi-class aging signs dataset, folder-based labels were converted to one-hot encoded vectors. For the UTKFace dataset, age was kept as a direct numerical value for the regression task

3.2 Milestone 2: Model Training and Evaluation

3.2.1 Module 3: Aging Sign Classification with ResNet50V2

- **Model Architecture**: A transfer learning approach was employed. The

ResNet50V2 model, pre-trained on the ImageNet dataset, was used as the base. Its convolutional layers were initially frozen. A new "head" was added on top, consisting of a

GlobalAveragePooling2D layer, a Dropout layer for regularization, and a final Dense output layer with 4 units and a softmax activation function for multi-class classification. This process is detailed in the **main.py** script.

- **Training**: The model was compiled with the Adam optimizer and categorical cross-entropy loss function. It was trained using an early stopping mechanism to prevent overfitting, and the best-performing model based on validation accuracy was saved.
- **Fine-Tuning**: After the initial training, the top layers of the **ResNet50V2** base were unfrozen, and the model was re-compiled with a much lower learning rate (1e-5) for fine-tuning. This allowed the model to make small adjustments to the pre-trained weights to better adapt to the specific features of facial aging signs.
- **Evaluation**: The model achieved a final test accuracy of **96.7%**, significantly exceeding the project's original target.

3.2.2 Module 4: Age Prediction with a MobileNetV2

- **Model Architecture**: As detailed in the **train_age_model.py** script, a **transfer learning** approach was used. The
- **MobileNetV2** model, pre-trained on ImageNet, served as the feature-extraction base. The model takes

- **128x128 RGB images** as input. A new "head" was added on top of the base, consisting of a
- GlobalAveragePooling2D layer, a Dropout layer for regularization, and a single Dense output layer with a **linear activation function** to output a numerical value for age regression.
- **Training & Evaluation**: The model was compiled with the **Adam optimizer** and a single loss function: **Mean Absolute Error (MAE)**. This directly optimizes the model to reduce the average error in years. After training, the final model achieved an
- **MAE of approximately 3.14 years** on the unseen test set.

3.3 Milestone 3: Frontend and Backend Integration

3.3.1 Module 5: Face Detection and Prediction Pipeline

The core prediction pipeline is implemented in the app.py file. It begins by loading the pre-trained Haar Cascade model for face detection (haarcascade_frontalface_default.xml). When an image is uploaded, the following process occurs:

- The image is first converted to grayscale, which is the optimal format for the Haar Cascade detector.
- The face_cascade.detectMultiScale function is used to identify the coordinates of all faces in the image.
- The script then loops through each detected face, cropping the Region of Interest (ROI) from the original image.
- Each ROI is then preprocessed separately for its respective model:
- For the **skin classifier**, the ROI is converted to RGB, resized to **224x224** pixels, and processed using the specific preprocess_input function for the **ResNet50V2** model.
- For the **age predictor**, the ROI is converted to RGB, resized to **128x128** pixels, and processed using the specific preprocess_input function for the **MobileNetV2** model.
- Finally, the preprocessed ROIs are passed to their respective models to get the final predictions.

3.3.2 Modules 6 & 7: Web UI and Backend Pipeline

The web application is built using Streamlit, which combines the frontend and backend logic into a single app.py script.

- User Interface (UI): The script uses Streamlit's Python functions to create a user interface directly. It generates a dynamic, single page with a title, instructions, a file upload widget, and display areas for the results. There are no separate HTML or CSS files.
- Backend Logic: The same app.py script contains the entire processing pipeline. When a user uploads an image and clicks the "Analyze" button, the script executes the face detection, preprocessing, and model prediction functions, then updates the UI to display the annotated image and download links.
- Evaluation: The processing time was evaluated to be less than 5 seconds per image, meeting the project's performance target.

3.4 Milestone 4: Finalization and Delivery

3.4.1 Modules 8 & 9: Export, Logging, and Documentation

- **Export Feature**: After a successful analysis, the Streamlit interface dynamically displays two download buttons.
- One button allows the user to download the **annotated image**. This image is generated in memory, encoded into bytes, and delivered directly to the user's browser.
- The second button allows the user to download the complete **prediction log** as a single prediction_log.csv file.
- **Logging**: The application uses a custom logging function. After each face is analyzed, the results—including a timestamp, filename, bounding box coordinates, and the model's predictions—are appended as a new row to a single, persistent prediction_log.csv file. This creates a structured and easily analyzable record of every prediction the application makes.
- **Documentation**: This document serves as the final, detailed documentation for the project.

4 Code Description

4.1 app.py - Main Streamlit Application

This is the central script that runs the entire web application.

- **Initialization**: Imports necessary libraries like Streamlit, TensorFlow, and OpenCV, and sets up the page configuration.
- **Model Loading**: The script uses a cached function (@st.cache_resource) to pre-load the dermal_scan_model_best.h5 (ResNet50V2), age_prediction_model_mobilenet.h5 (MobileNetV2), and the Haar Cascade XML file into memory on startup to ensure fast inference times.
- **UI Rendering**: It uses Streamlit widgets (st.title, st.file_uploader, st.button) to dynamically generate the user interface in the browser.
- **Inference Pipeline**: When a user uploads an image and clicks "Analyze," a modular function (run_analysis) is called. This function handles face detection, cropping, preprocessing for each specific model, and prediction.
- **Results Display**: The script annotates a copy of the original image with bounding boxes and prediction text. It then displays this result image on the page and provides

st.download_button widgets to export both the annotated image and the prediction log CSV file.

4.2 Training Scripts

The models were trained using Python scripts, not Jupyter Notebooks.

- **main.py (Skin Classifier Training):**
  - **Purpose**: To train and fine-tune the aging sign classifier.
  - **Process**: Loads image data from categorized folders, performs one-hot encoding on labels, and splits the data. It initializes ImageDataGenerator for augmentation. It then loads

  ResNet50V2 with ImageNet weights, freezes the base, adds a custom head, and trains it. Finally, it unfreezes the top layers and fine-tunes the model with a lower learning rate for improved accuracy.

- **train_age_model.py (Age Predictor Training):**
  - **Purpose**: To train the model for age prediction.
  - **Process**: Loads the UTKFace dataset by parsing age labels from filenames. It uses a custom

  AgeDataGenerator to load images in batches, which is memory-efficient. It defines and builds a regression model using MobileNetV2 as a pre-trained base. The model is compiled with Mean Absolute Error (MAE) as the loss function and then trained.
  - **Note**: The script contains a variable for the dataset path that must be updated by the user to run the training successfully.

5 Evaluation and Performance

The project's success was measured against the criteria defined in the project guide.

Milestone	Focus Area	Metric / Evaluation Method	Target / Goal	Achieved Performance
Milestone 1	Data Preparation & Preprocessing	Dataset quality, augmentation	Balanced & clean datasets	Met. Datasets were cleaned and balanced.
Milestone 2	Model Performance (Signs)	Accuracy & loss metrics (ResNet)	Test accuracy ≥ 90%	Met. Achieved 96.7% accuracy after fine-tuning Met.
Milestone 2	Model Performance (Age)	MAE (Age)	Test MAE < 4 years, Accuracy > 90%	Met.Achieved an excellent final Test MAE of 3.14 years
Milestone 3	UI & Backend	Upload-to-output time, usability	< 5s per image	Met. Local processing time is well within the target.
Milestone 4	Final Delivery	Export functionality & docs	Complete & professional	Met. CSV/Image export and this documentation are complete.

6 Setup and Usage Guide

6.1 Prerequisites

- Python 3.8+
- Streamlit
- TensorFlow / Keras
- OpenCV-Python
- NumPy
- Pandas
- Matplotlib
- Seaborn

6.2 Installation

1. Clone the project repository.
2. It is recommended to create a virtual environment:

```
python -m venv venv
source venv\Scripts\activate  # On Windows, use 'venv\Scripts\activate'

```

3. 3. Install the required packages from the requirements.txt file:

```
pip install -r requirements.txt
```

6.3 Running the Application

1. Place the trained models(dermal_scan_model_best.h5, age_prediction_model_mobilenet.h5) and the Haar Cascade file (haarcascade_frontalface_default.xml) into the main project folder alongside app.py.
2. Ensure your Python virtual environment is activated and all dependencies from requirements.txt are installed.
3. Run the Streamlit application from the command line:

```
streamlit run app.py

```

6.4 How to Use the Web Interface

1. Click **"Choose File"** to select a JPG, JPEG, or PNG image.
2. The filename will appear, and the **"Analyze Image"** button will become active.
3. Click **"Analyze Image"**.
4. The results page will load, showing the original and annotated images, prediction scores, and download links.

7 Conclusion

The DermalScan project successfully delivered an AI-powered web application for automated facial analysis. By integrating two specialized deep learning models, the system accurately classifies common aging signs—such as wrinkles and dark spots—and predicts **age** with high performance. The intuitive **Streamlit-based** interface provides users with visually annotated results and downloadable data in under five seconds, making complex AI technology accessible and practical. Overall, DermalScan serves as a robust prototype that effectively demonstrates the potential of AI to provide personalized, data-driven insights in the skincare and cosmetology industries

8 Future Scope

The project has significant potential for expansion in several key areas:

- **Model Enhancements:** The analysis can be expanded to include more conditions like skin tone and redness, implement a severity score for each sign, and improve age prediction accuracy using more advanced models and larger datasets.
- **New Features:** Future versions could include real-time video analysis, user profiles for tracking skin changes over time, personalized skincare product recommendations, and a dedicated mobile application.
- **Scalability:** For wider accessibility, the application could be deployed on a cloud platform. Creating a commercial API would also allow for integration with third-party beauty and wellness applications.