DermalScan_AI
Facial Skin Aging Detection App

Infosys springboard virtual internship 6.0

Submitted By:

Gargi Kulkarni

Under the guidance of

Mr. Praveen Sir

Problem Statement:

There is currently no tool that can automatically evaluate the presence of age-related facial characteristics on a person's face without having to go to a dermatologist for an examination or wait until after an experimental study has been completed. Researchers have created a system, based upon deep learning, to provide users with real-time analysis of their facial images and receive immediate recommendations based upon these results.

Objectives:

- Detecting and locating facial characteristics which indicate the presence of age.
- Categorise the characteristics into 4 specific categories (i.e. Wrinkled, Dark Spot, Puffy Eyes and Clear Skin) using a Convolutional Neural Network (CNN).
- Create a user-friendly web based front end by allowing users to upload images with the expected annotated outcomes and percentage predictions.
- Connect the pipeline with a backend to process uploaded images, make inferences and return annotated results.

Technologies & Libraries Used:

- **Python (v3.10.1):** The primary language used for all data processing and model building.
- **Pandas(2.3.3):** Used for creating DataFrames, managing the dataset, and handling CSV files (import pandas as pd).
- **Matplotlib(3.10.7):** It is the foundational plotting and visualization library for Python.
- **Seaborn(0.13.2):** It simplifies complex statistical plots (like heatmaps or count plots) into single lines of code and makes them look better automatically.
- **Scikit-learn(1.7.2):** Used for splitting the data into training and testing sets (train_test_split).
- **Numpy(2.2.6):** It is a fundamental dependency for Pandas, Scikit-Learn, and TensorFlow, and is required for numerical operations.

Development Tools:

- **Visual Studio Code (VS Code):** The Integrated Development Environment (IDE) used to write and run your code.
- **Jupyter Notebook:** Used for interactive coding and ensuring the code runs step by step.

Methodology:

Milestone 1: Dataset Preparation and Preprocessing-

➢ Module 1: Dataset Setup and Image Labeling

- Aggregated facial images into four distinct classes: Wrinkles, Dark Spots, Puffy Eyes, and Clear Skin.
- Manually filtered out poor-quality samples and ensured equal representation across all classes to prevent model bias.
- Produced a finalized, labeled dataset directory and a class distribution plot confirming data balance.

➢ Output:

```
Starting to scan folders...
Found 300 images in: wrinkles
Found 303 images in: dark_spots
Found 300 images in: puffy_eyes
Found 300 images in: clear_skin
-----------------------------
Total images loaded: 1203
```

Fig 1. Dataset Class Distribution Summary

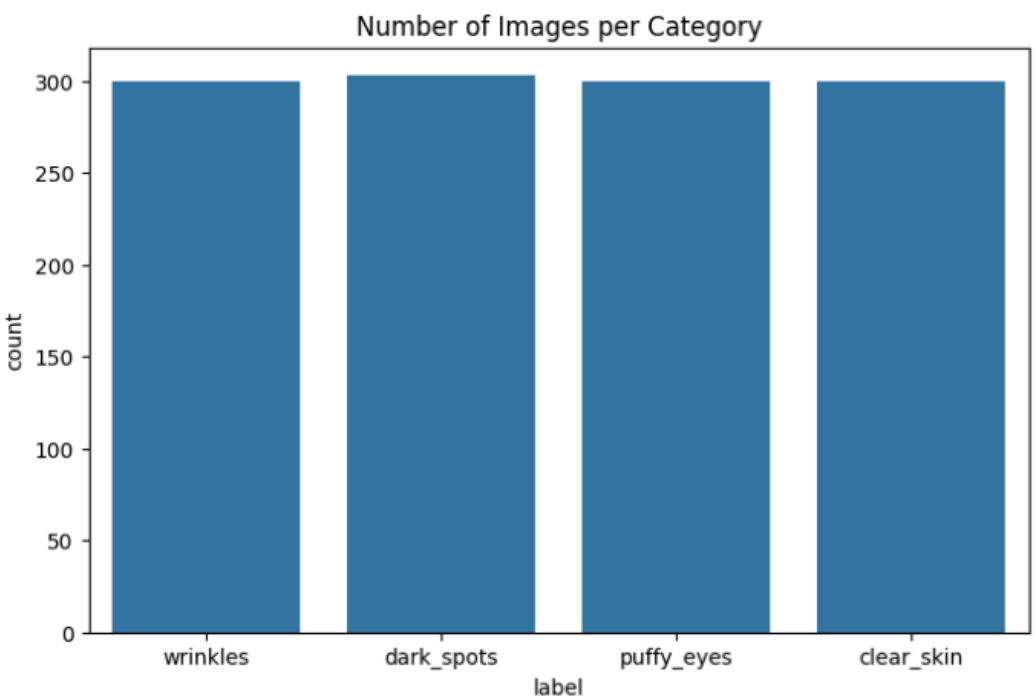


Fig 2. Distribution of Images Across Skin Condition Categories

- **Module 2: Image Preprocessing and Augmentation**
  - We have successfully transformed raw images into a clean format ready for model training.
  - Organized the facial images into four distinct classes: Wrinkles, Dark Spots, Puffy Eyes, and Clear Skin.
  - Resized all images to 224x224 pixels and normalized pixel values so the model receives consistent input.
  - Applied transformations like rotation, flips, and zoom to artificially increase the dataset size and help the model generalize better.

- **Image Preprocessing:** Preprocessing refers to the operations applied to images to standardize and format them before they are fed into a model. The goal is to transform the raw data into a consistent state that the algorithm can process efficiently.

- **Image Augmentation:** Augmentation involves creating new, modified versions of images from your existing training dataset. This artificially expands the size of the dataset without actually collecting more data.

- **Output:**

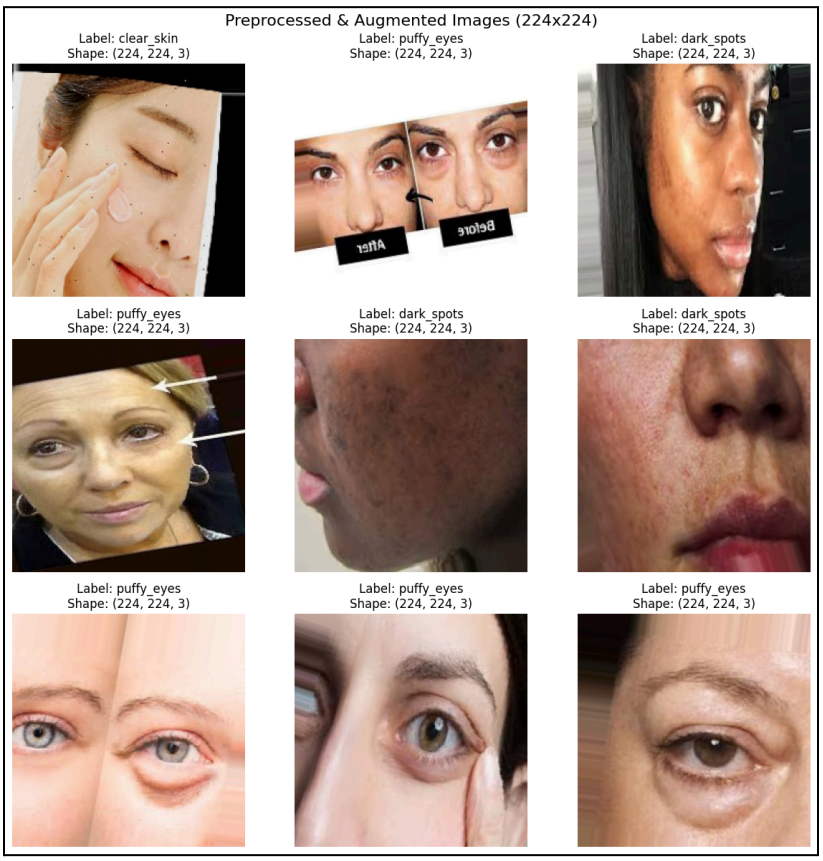


Fig 3. Preprocessed & Augmented Images Combined

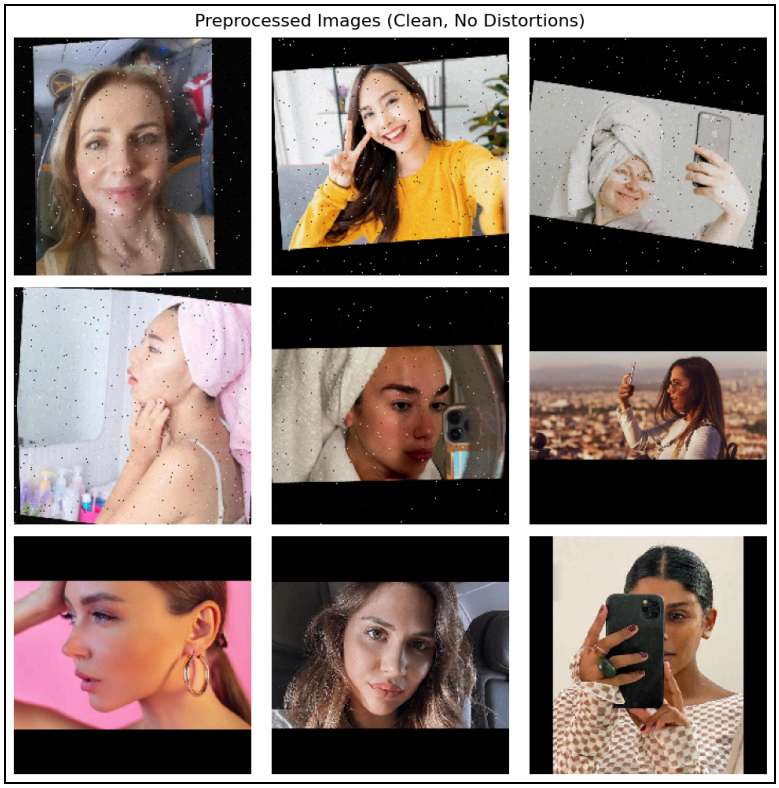


Fig 4. Preprocessed Image

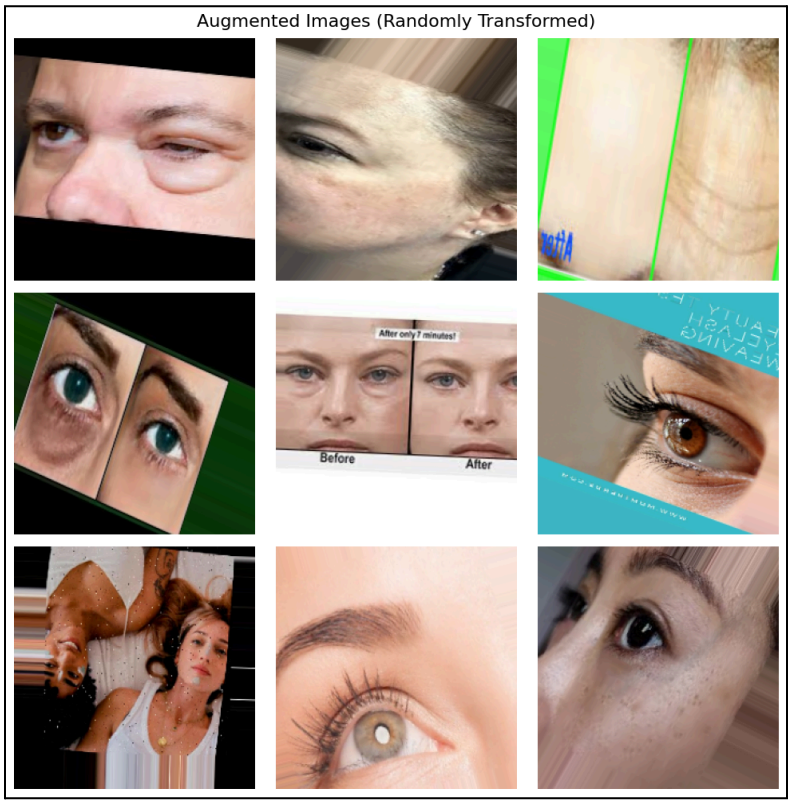


Fig 5. Augmented Images

Milestone 2: Model Training and Evaluation -

➢ Module 3: Model Training with EfficientNetB0

- Leverages a pre-trained EfficientNetB0 model for transfer learning, utilizing the Adam optimizer and Categorical Cross-Entropy to achieve high-performance classification.
- Focuses on reaching a minimum 90% accuracy threshold, evidenced by stable training/validation loss and accuracy curves saved in the final .h5 model file.
- Ensures the model generalizes well to unseen data by monitoring for stability and preventing overfitting during the training cycles.

➢ Output:

```
Found 1203 files belonging to 4 classes.
Using 963 files for training.
Found 1203 files belonging to 4 classes.
Using 240 files for validation.
Epoch 1/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 65s 2s/step - accuracy: 0.5659 - loss: 1.0531 - val_accuracy: 0.7583 - val_loss: 0.7807
Epoch 2/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 42s 1s/step - accuracy: 0.8017 - loss: 0.6219 - val_accuracy: 0.7917 - val_loss: 0.6278
Epoch 3/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 39s 1s/step - accuracy: 0.8307 - loss: 0.5143 - val_accuracy: 0.8083 - val_loss: 0.5663
Epoch 4/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 37s 1s/step - accuracy: 0.8401 - loss: 0.4661 - val_accuracy: 0.8208 - val_loss: 0.5294
Epoch 5/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 36s 1s/step - accuracy: 0.8567 - loss: 0.4167 - val_accuracy: 0.8208 - val_loss: 0.5095
Epoch 6/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 43s 1s/step - accuracy: 0.8775 - loss: 0.3927 - val_accuracy: 0.8292 - val_loss: 0.4967
Epoch 7/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 40s 1s/step - accuracy: 0.8827 - loss: 0.3653 - val_accuracy: 0.8333 - val_loss: 0.4821
Epoch 8/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 34s 1s/step - accuracy: 0.8920 - loss: 0.3375 - val_accuracy: 0.8292 - val_loss: 0.4704
Epoch 9/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 35s 1s/step - accuracy: 0.8982 - loss: 0.3294 - val_accuracy: 0.8250 - val_loss: 0.4668
Epoch 10/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 36s 1s/step - accuracy: 0.9076 - loss: 0.3036 - val_accuracy: 0.8375 - val_loss: 0.4605
```

Fig 6. Training and Validation Metrics for EfficientNetB0

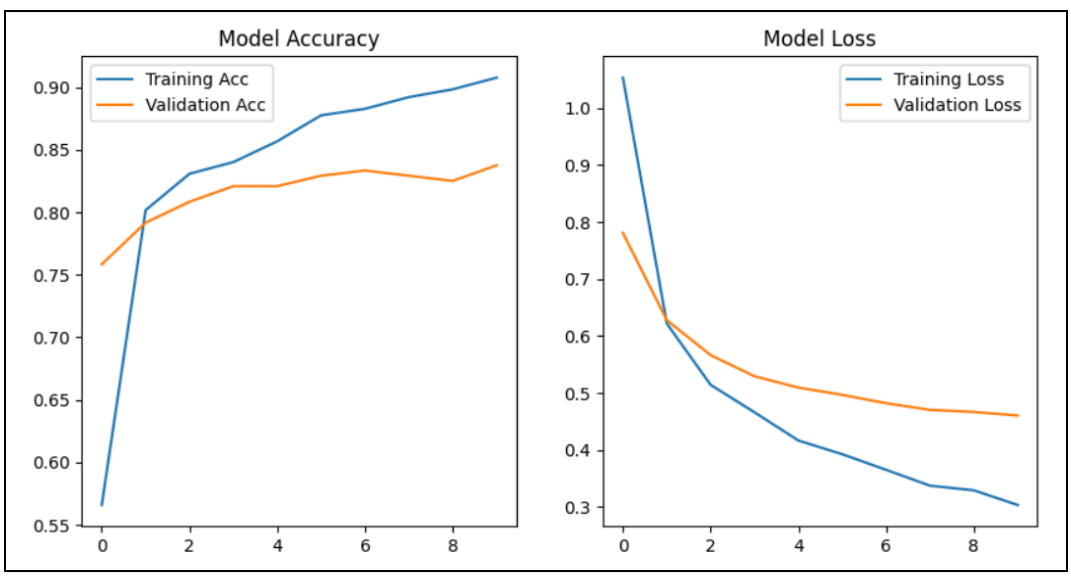
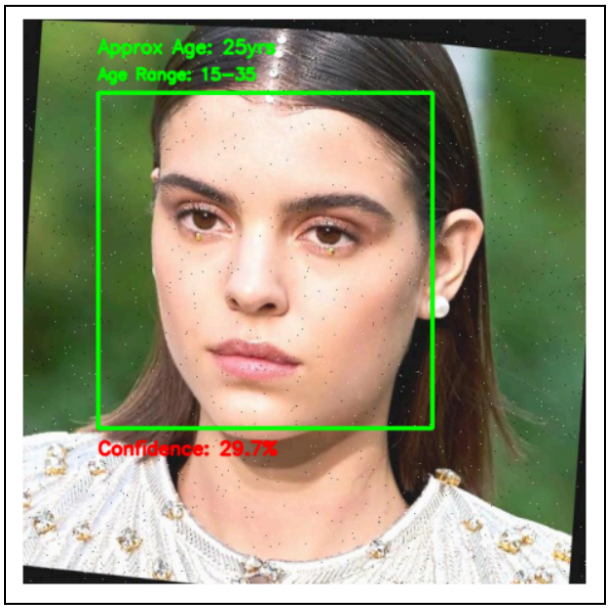


Fig 7. Model Performance Metrics (Accuracy and Loss)

➢ Module 4: Face Detection and Prediction Pipeline

- Combines OpenCV Haar Cascades for initial face detection with the trained CNN model to perform inference on cropped face regions.
- Generates a processed output display featuring bounding boxes around detected faces, labeled with predicted age and confidence percentages.
- Evaluates the end-to-end success based on the precision of the face locator and the correctness of the final class predictions.

➢ Output:



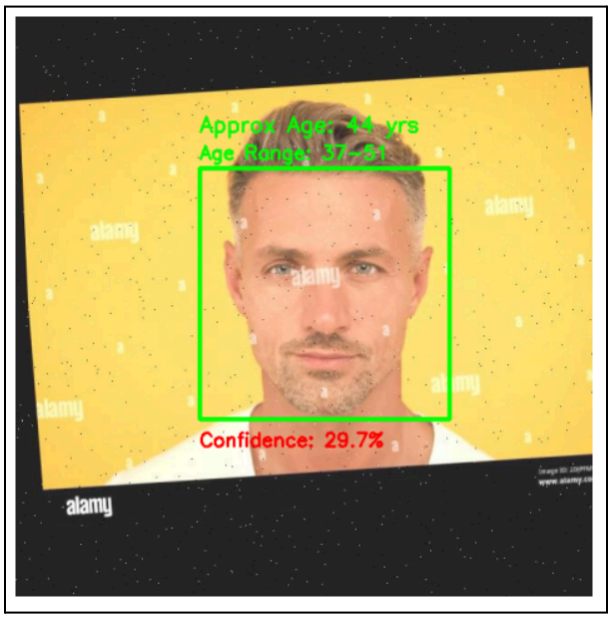


Fig 8. Face Detection and Age Prediction Inference

Milestone 3: Frontend and Backend Integration-

➢ Module 5: Web UI for Image Upload and Visualization

- Designed and developed a responsive web-based user interface using Streamlit
- Implemented an image upload component to allow users to submit images for analysis.
- Displayed the uploaded image as a preview before processing to ensure correct input.
- Integrated visualization of model predictions by drawing bounding boxes on detected regions.
- Annotated each bounding box with:
  - Predicted class label
  - Corresponding confidence percentage
- Ensured clean and readable annotation overlays for better interpretability.
- Optimized UI rendering to avoid lag during image upload and prediction display.

➢ Output:

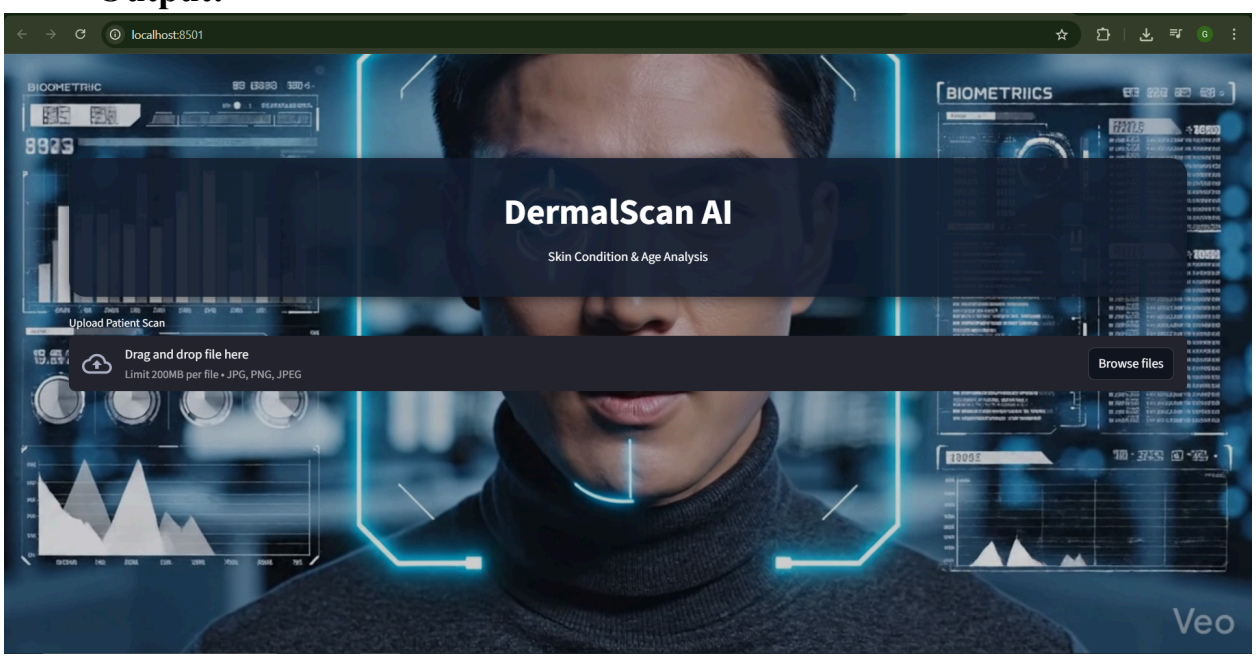


Fig 9. DermalScan AI Web Interface for Skin Condition & Age Analysis

➢ Module 6: Backend Pipeline for Model Inference

- Modularized image preprocessing and inference logic for improved code maintainability.
- Loaded the trained EfficientNet model for performing image inference.
- Processed uploaded images through preprocessing, detection, and prediction stages.
- Generated prediction results including:
- Class labels
- Confidence scores
- Age range & Estimated age
- Bounding box coordinates
- Returned prediction results seamlessly to the frontend interface.
- Logged prediction outputs and bounding box data for debugging and evaluation purposes.
- Conducted end-to-end testing to validate complete frontend–backend integration.

➢ Output:

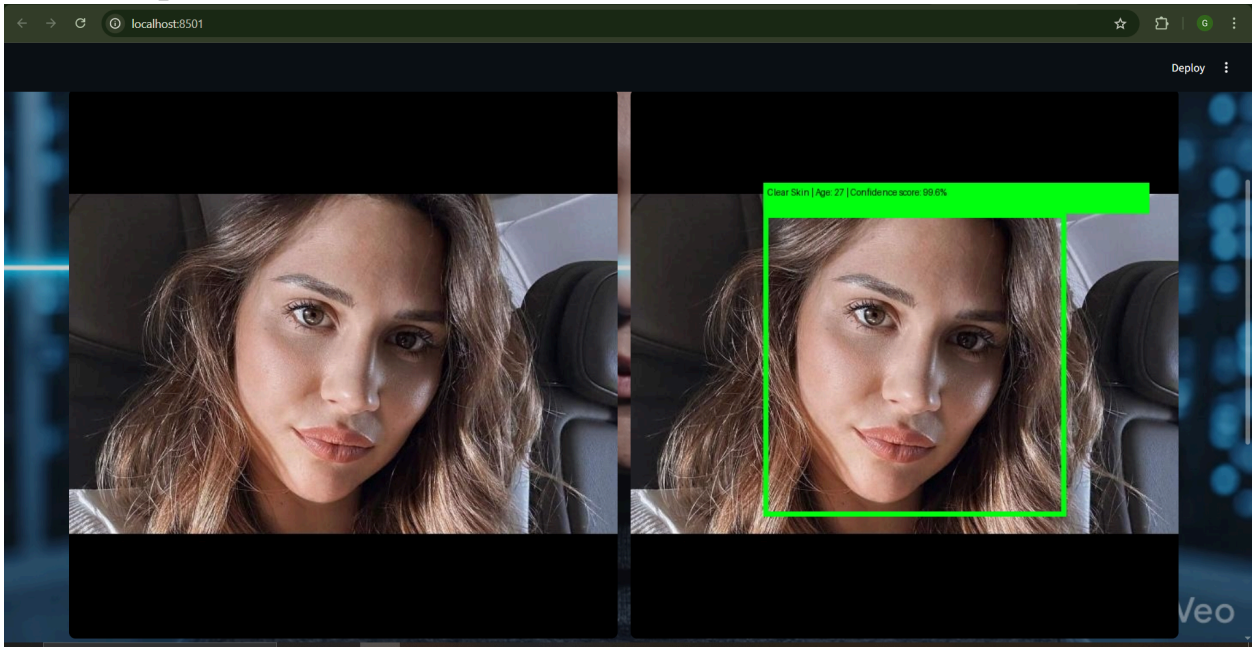


Fig 10. Inference Results Visualization: Comparison of Original Image and Annotated Output with Detected Bounding Box

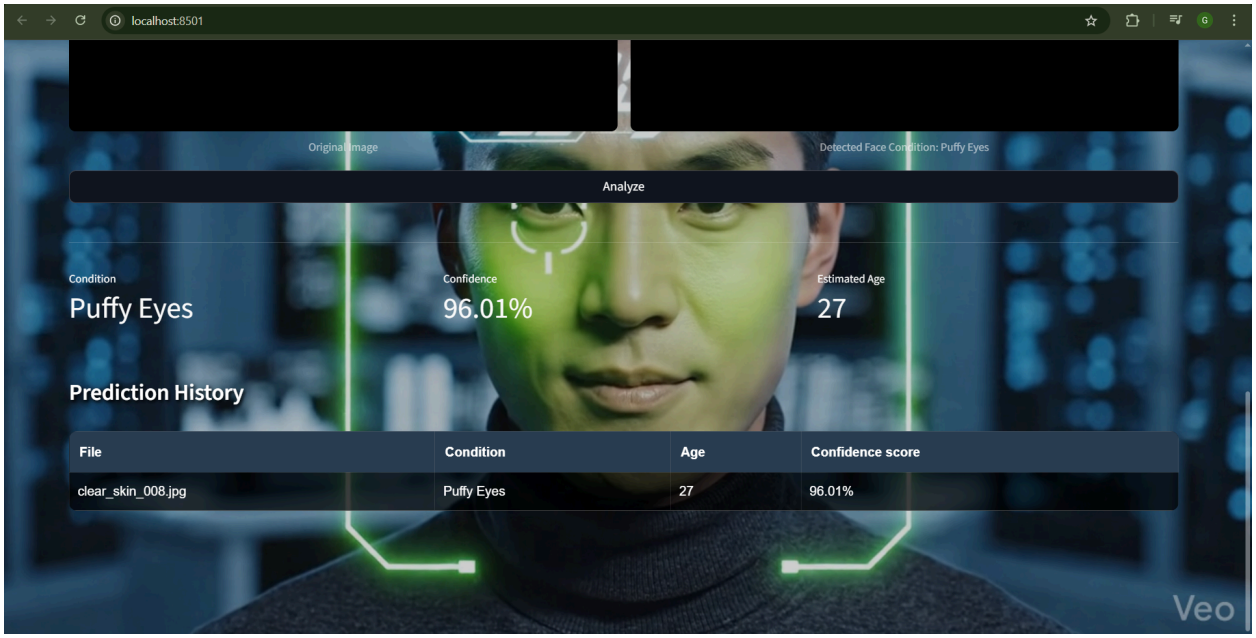


Fig 11. Integrated Results Dashboard: Condition Analysis and Prediction History

Milestone 4: Finalization and Delivery -

➢ Module 7: Export and Logging

- Implemented data export functionality in the frontend to allow users to download both the annotated analysis image and the prediction history.
- Developed a CSV logging system to record session data, including predicted conditions, age estimates, and confidence scores for future reference.
- Conducted rigorous testing on a diverse set of images to validate the consistency of the export logs and the accuracy of the final outputs.
- Finalized the results display to ensure that downloaded files maintain the same visual quality and data integrity as the live preview.

➢ Output:

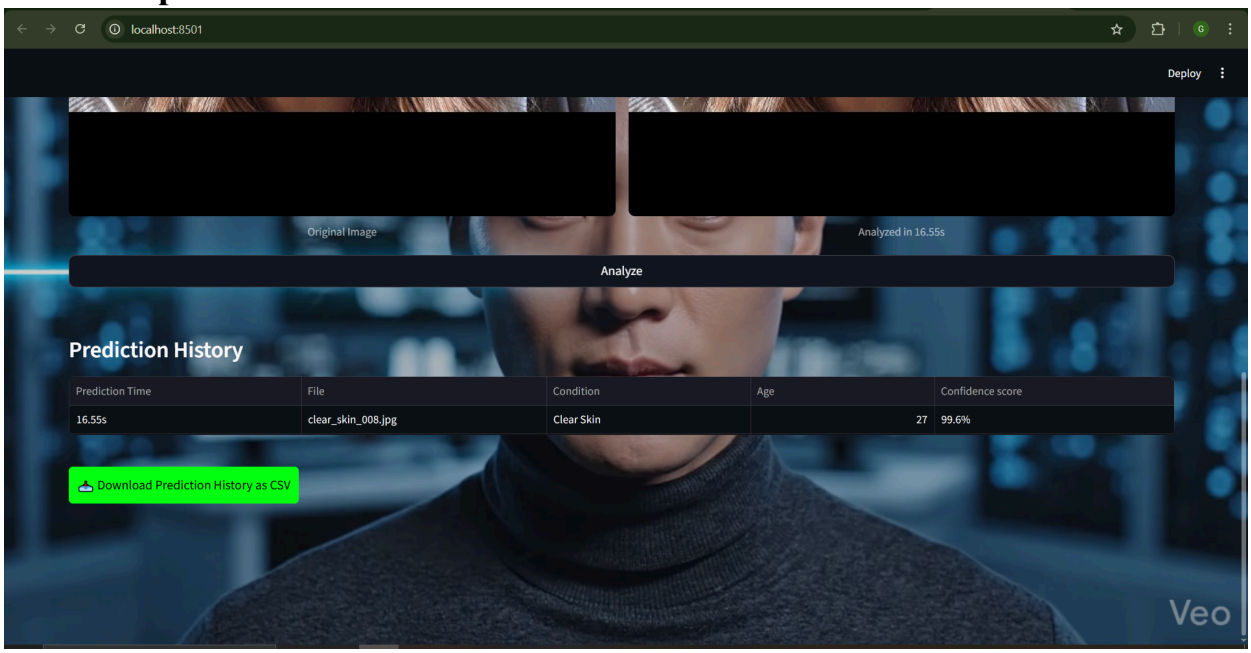


Fig 12. Export and Logging Interface: Prediction History Log with CSV Download Functionality

➢ Module 8: Documentation and Final Presentation

- Authored comprehensive User and Developer Guides (README.md) detailing installation steps, dependencies, and usage instructions.
- Structured the GitHub repository to include all source code, trained models (.h5 files), and dataset samples for reproducibility.
- Designed a professional presentation slide deck summarizing the Problem Statement, Methodology, Technical Architecture, and Future Scope.
- Recorded and edited a final walkthrough video demonstrating the full end-to-end workflow, from image upload to result interpretation and export.

Conclusion:

The **DermalScan: AI Facial Skin Aging Detection App** was successfully developed and deployed to provide an automated, accessible solution for analyzing facial skin conditions. By leveraging deep learning techniques, specifically the **EfficientNetB0** architecture, the model achieved a robust training **accuracy of approximately 90.76%** in classifying skin features across four **distinct categories: Wrinkles, Dark Spots, Puffy Eyes, and Clear Skin.**

The project effectively addressed the problem statement by eliminating the dependency on immediate clinical examinations for preliminary skin analysis. The integration of **OpenCV** for precise face detection and **Streamlit** for a responsive web interface ensured a seamless and user-friendly experience. Additionally, the successful implementation of export and logging modules allows users to maintain a historical record of their analysis, adding significant practical value to the application.

In summary, DermalScan demonstrates the powerful application of Computer Vision in the healthcare and cosmetic domains, offering a reliable, non-invasive tool for real-time age estimation and skin condition monitoring.